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 datalogi tisdagen den 8 november 2022 klockan 9.00 i Sal F3, Lindstedtsvägen 26, Kungliga Tekniska Högskolan, Stockholm.

Tryck: Universitetsservice US AB

Abstract

In recent years, computer vision-based assistive technologies have enabled visually impaired people to use automatic visual recognition on their mobile phones. These systems should be capable of recognizing objects on fine-grained levels to provide the user with accurate predictions. Additionally, the user should have the option to update the system continuously to recognize new objects of interest. However, there are several challenges that need to be tackled to enable such features with assistive vision systems in real and highly-varying environments. For instance, fine-grained image recognition usually requires large amounts of labeled data to be robust. Moreover, image classifiers struggle with retaining performance of previously learned abilities when they are adapted to new tasks. This thesis is divided into two parts where we address these challenges. First, we focus on the application of using assistive vision systems for grocery shopping, where items are naturally structured based on fine-grained details. We demonstrate how image classifiers can be trained with a combination of natural images and web-scraped information about the groceries to obtain more accurate classification performance compared to only using natural images for training. Thereafter, we bring forward a new approach for continual learning called replay scheduling, where we select which tasks to replay at different times to improve memory retention. Furthermore, we propose a novel framework for learning replay scheduling policies that can generalize to new continual learning scenarios for mitigating the catastrophic forgetting effect in image classifiers. This thesis provides insights on practical challenges that need to be addressed to enhance the usefulness of computer vision for assisting the visually impaired in real-world scenarios.

Sammanfattning

De senaste åren har teknologiska hjälpmedel baserade på datorseende möjliggjort för synskadade personer att använda sig av automatisk visuell igenkänning på deras mobiltelefoner. Dessa system bör kunna känna igen objekt på finfördelade nivåer för att förse användaren med noggranna prediktioner. Användaren bör även ha möjligheten att uppdatera systemet kontinuerligt till att känna igen nya objekt av intresse. Dock finns det flera utmaningar som behöver avklaras för att aktivera dessa funktioner i synhjälpmedelssystem i reella och mycket varierande miljöer. Exempelvis behöver finfördelad bildigenkänning vanligtvis stora mängder märkt data för att vara robust. Dessutom har bildklassificerare besvär med att behålla sin prestanda av tidigare inlärda förmågor när de anpassas till nya uppgifter. Denna avhandling är uppdelad i två delar, där vi tar oss an dessa utmaningar. Först fokuserar vi på tillämpningen av att använda synhjälpmedelssystem för att handla matvaror, där varorna är naturligt strukturerade enligt finfördelade detaljer. Vi påvisar hur bildklassificerare kan tränas med en kombination av naturliga bilder och webbskrapad information om matvarorna för att erhålla mer träffsäker klassificeringsförmåga jämfört med att enbart använda naturliga bilder för träning. Därefter lägger vi fram ett nytt tillvägagångssätt för kontinuerlig inlärning som kallas replay scheduling (reprisschemaläggning), där vi väljer vilka uppgifter som ska repeteras vid olika tidpunkter för att förbättra bibehållande av minnen. Vi föreslår även ett nytt ramverk för inlärning av policyer för replay scheduling som kan generalisera till nya scenarion för kontinuerlig inlärning för att mildra effekten av katastrofal glömska i bildklassificerare. Denna avhandling ger insyn till praktiska utmaningar som behöver lösas för att förbättra användbarheten hos datorseende till att hjälpa synskadade personer i verkliga scenarier.

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
Unpublished manuscript

D ***Policy Learning for Replay Scheduling in Continual Learning***
Marcus Klasson, Hedvig Kjellström, Cheng Zhang
Unpublished manuscript

Acronyms

List of commonly used acronyms:

A2C	Advantage Actor-Critic
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	Visually Impaired

Contents

List of Papers iii

Acronyms v

Contents 1

I Overview 3

1 Introduction 5
1.1 Visual Impairments . . . 6
1.2 Assistive Vision Technologies . . . 6
1.3 Scope of Thesis . . . 7
1.4 Thesis Contributions . . . 9
1.5 Thesis Outline . . . 12

2 Background 13
2.1 Problem Settings in Machine Learning . . . 13
2.2 Deep Learning . . . 14

3 Fine-Grained Image Recognition 21
3.1 Related Work . . . 21
3.2 Dataset Collection . . . 23
3.3 Fine-Grained Image Recognition of Grocery Items . . . 24
3.4 Experiments . . . 26
3.5 Discussion . . . 29

4 Continual Learning 31
4.1 Related Work . . . 31
4.2 Replay Scheduling in Continual Learning . . . 33
4.3 Policy Learning for Replay Scheduling . . . 36
4.4 Experiments . . . 37
4.5 Discussion . . . 41

5 Conclusions and Future Directions 43
5.1 Conclusions . . . 43
5.2 Future Directions . . . 45

Bibliography **49**

II Included Papers **75**

A A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels **77**

B Using Variational Multi-View Learning for Classification of Grocery Items **93**

C Learn the Time to Learn: Replay Scheduling in Continual Learning **137**

D Policy Learning for Replay Scheduling in Continual Learning **163**

Part I

Overview

Chapter 1

Introduction

Vision is probably the most important of all senses that humans possess. Our society is built on having the ability to see. For example, if we want to cross a street, thick colored stripes on the road and signs above our heads indicate where the cross walk is located in order for us to cross the street appropriately. Another example is how we use text to communicate with each other, where messages that we read and write consist of sequences of symbols that are structured into words of some specific language. Furthermore, visuals are used frequently in education to help students comprehend new information and concepts. Basically, many everyday tasks such as traveling, communication, and learning become more convenient with normal vision.

Millions of people have visual impairments that cannot be treated with correctable eyewear [1]. To enhance the mobility of visually impaired (VI) people, there exist various kinds of assistive tools, such as screen readers and Braille typewriting machines, for supporting them with receiving information and communicating through text. More recently, several assistive vision technologies have emerged in the form of mobile phone applications [2–7] and wearable devices [8–10]. Furthermore, several works in computer vision research has been specifically focused developing methods for assisting VI people with e.g. object recognition [11–18] and wayfinding [19–24].

Despite the recent advances in deep learning for computer vision [25–29], several challenges have been demonstrated when deploying computer vision models in real-world scenarios. For example, the models may surprisingly fail to recognize known objects appearing in new poses [30], lighting conditions [31], and geographical locations [32], models trained on images from one set of hospitals may perform poorly on image conditions from another [33], and models can suffer from biases that disadvantage certain users based on their gender and ethnicity [34]. Part of the reason for these challenges is that specifying a model that can be injected with the rich complexity of the real world is difficult [35]. To enhance their applicability in the real-world, researchers are striving for developing computer vision models that can recognize objects in detail [36], learn new abilities continuously [37, 38], and adapt fast to new scenarios from few examples [39–41]. Furthermore, such tasks should be possible to execute in a time-efficient and robust manner in assistive vision devices to improve the user experience.

In this thesis, we address challenges on computer vision models for recognizing objects and their fine-grained details, as well as continually adapting the models to

new concepts while retaining its previously learned knowledge. We will continue this chapter by briefly describing visual impairments in Section 1.1, followed by a summary of assistive vision technologies in Section 1.2. Section 1.3 describes the scope of the thesis, and we summarize the contributions of the included papers in Section 1.4. In Section 1.5, we present the overall outline of this thesis.

1.1 Visual Impairments

A visual impairment is defined as the decrease of an individual's ability to see from various distances [42]. The conditions varies from having issues with seeing from far or near distances to blindness, i.e., complete or nearly-complete vision loss. Visual capabilities are in general assessed by measuring the visual acuity (sharpness) of seeing from some fixed distance, e.g., letters of different size on an eye chart. The visual acuity is measured differently based on whether near- or far-sighted VI people are being assessed. For far-sighted VI people, visual acuity is calculated by the ratio between the distance that the subject and a normal-sighted person can recognize the target object. For near-sighted VI people, visual acuity is measured using a standardized point system defining the font size of letters that the subject can recognize [43].

In 2020, it was estimated that 338 million people possess a moderate to severe visual impairment globally, including 43 million people that are blind [1]. Furthermore, the World Health Organization (WHO) have estimated that at least 2.2 billion people live with a near or distance visual impairment, where at least 1 billion cases could have been prevented or yet has to be addressed [43]. The number of untreated cases is projected to be 1.7 billion people by 2050, mainly due to global population growth as well as increased aging among the populations [1]. The main causes for having vision loss are uncorrected refractive errors, untreated cataracts, age-related macular degeneration, glaucoma, diabetic retinopathy, where 90% of such cases are preventable and treatable [44]. These causes also varies between different countries and regions with different socioeconomic status.

There exists many kinds of tools for assisting VI people with everyday tasks. The white cane and guiding dogs are probably the most common tools for enhancing their mobility by preparing the user for what is present in their near surroundings. Furthermore, there are different tools for various recognition tasks, such as currency markers for identifying different bills in wallets, color indicators for notifying the user of the color of clothes, and labeling apparatus with Braille writing for distinguishing between similar items [45]. There also exist Braille keyboards and screen readers compatible with both computers and mobile phones for helping VI people with e.g. sending text messages and emails, surfing the web, and using calendar apps [46]. In the next section, we introduce a selection of assistive vision technologies aimed for assisting VI people with recognition of objects in their environment.

1.2 Assistive Vision Technologies

Cameras are used by blind people to record events and memories similarly as people with normal vision [47]. Moreover, there is an urge among VI people to use their

cameras for daily tasks other than recording events, such as object recognition [48], face recognition [49], and currency identification [50]. In this thesis, we focus on object recognition which can be challenging in many ways without visual information. For instance, VI people may need assistance from non-visual cues indicating how to capture the whole item in the camera view [51], and also may need to point the camera towards some specific side of the object to identify details of interest to the user. These challenges have encouraged the development of technical aids that use computer vision for assisting VI people which we will cover next.

In the last decade, we have seen several variants of assistive vision technologies emerging on the market. The advances of such technologies have made it possible for VI people to reduce the need for human assistance for performing daily tasks where vision is a necessity. Currently, these devices have taken the form of both mobile phone applications [2–5] and wearable devices [8–10] where computer vision methods are used for visual tasks, such as object and face recognition, barcode scanning, color and currency identification, and text recognition. An alternative to the computer vision-based technologies are other mobile phone apps where VI users have video calls with sighted volunteers that help them with any kind of task requiring visual capabilities [6, 7].

Despite the increased availability of assistive vision technologies, there are still several challenges remaining to be tackled in order to enhance the user experience. Although machine learning techniques have been successfully applied in computer vision applications such as object recognition [25, 26, 52], generating scene descriptions [28, 29, 53], and visual question answering [54–56], these methods require immense amounts of human-annotated data for providing accurate predictions. The annotation becomes even more labor-intensive in scenarios where the assistive vision system must distinguish between objects with subtle differences, making it challenging to provide users with more detailed information than the general object class. Another challenge is how to continuously adapt the system to recognize new objects while ensuring that the system is still capable of classifying previous known items correctly. Furthermore, assistive vision technologies need system requirements to provide better usability, where the system should process images in real-time, be robust when applied in new environments, as well as ensuring privacy for the user.

1.3 Scope of Thesis

This thesis focuses on two topics for machine learning and computer vision-based assistive technologies, namely fine-grained image recognition (FGIR) [36] and continual learning (CL) [37, 38]. The FGIR task involves identifying visually similar subcategories of some general object classes. An example is when one has to distinguish between two kinds of visually similar juice packages with different ingredients. In Paper A and B, we study FGIR from the real-world application of recognizing groceries from mobile phone with an assistive vision app. The CL setting targets the scenarios where a classifier receives data from new object classes to learn from a continuous stream, while retaining its capability of recognizing previously learned objects. In Paper C and D, we introduce a novel approach for improving the retention of previously learned abilities of the classifier. The common denominator of these topics is image classification, However, both have challenges of their own that have to be

addressed for enabling such adding such features into assistive vision technologies. Next, we describe the challenges that we have focused on in this thesis.

Challenges in Fine-Grained Image Recognition

One main challenge for FGIR is the required data collection procedure. Firstly, annotating the specific details about the objects in the collected data becomes more time-consuming for the annotators, and may also require expert knowledge of the domain. Secondly, as some fine-grained classes could be rare, the dataset may be unbalanced making it difficult for the classifier classify the rare classes accurately from only a few training samples. A real-world application for helping VI people using assistive vision technologies that contains both of these challenges is grocery shopping [12, 57–62]. Grocery items usually require visual information to distinguish between them, e.g., when distinguishing between two juice packages of the same shape but with different ingredients. Similarly for raw groceries, it can be difficult to tell the difference between two different kinds of apples without the ability to see, unless the customer knows how the apples smell or how the texture of their peel differs when touching them. Furthermore, the conditions of the grocery store environment can disturb the recognition performance for computer vision-based assistive devices, e.g., when multiple and misplaced items appear in the camera view as well as poor lighting conditions in particular areas. Collecting training data that covers all possible scenarios that can occur in the store would be a nearly impossible task. Our goal is to reduce the need for large amounts of training data in the grocery stores by collecting web-scraped information about the items to utilize when training the image classifier.

Challenges in Continual Learning

The main challenge in CL is catastrophic forgetting [63] where the classifier overwrites previously learned knowledge with information about recently learned objects. Replay-based CL [37, 38] is a simple yet efficient approach for mitigating catastrophic forgetting where samples from old classes are stored in a small memory which can be re-trained on when learning new classes. In contrast to the small memory requirement, many machine learning systems in real-world applications are limited by constraints on processing times rather than data storage capacity [64–66]. In such settings, retraining on all seen data is often prohibited, the system would need a method for selecting what data to replay from a huge amount of stored data to mitigate catastrophic forgetting. Motivated by human learning techniques for improving memory retention [67–71], we propose a novel approach called replay scheduling where we have policy for selecting which tasks to replay at different times. Our goal is to demonstrate that scheduling over which tasks to replay can be crucial for CL systems in settings where immense amounts of historical data is accessible. To enable our approach in real-world CL scenarios, we need to develop efficient methods that can propose replay schedules to mitigate catastrophic forgetting in new CL scenarios without additional computations.

Table 1.1: Two examples of grocery item classes in the dataset presented in Paper A. We display the class label (coarse-grained class in parenthesis), two natural images taken with a mobile phone inside grocery stores, and the web-scraped iconic image and text description of the items.

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.**"*

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 classifiers 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 convolutional networks for image classification.

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 and wrote most of the text. All authors took part in 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 benefits from combining the natural images and web-scraped information in the Grocery Store dataset from Paper A. We employ a deep generative model called Variational Canonical Correlation Analysis (VCCA) [72] for learning joint representations of the different data views, i.e., natural images, iconic images, and text descriptions, that are used for training the classifiers. We performed a thorough ablation study over all data views to demonstrate how they contribute individually to enhancing the classification performance. 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 learned by a variational autoencoder (VAE) [73] and VCCA. 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$_{\boldsymbol{xiwy}}$

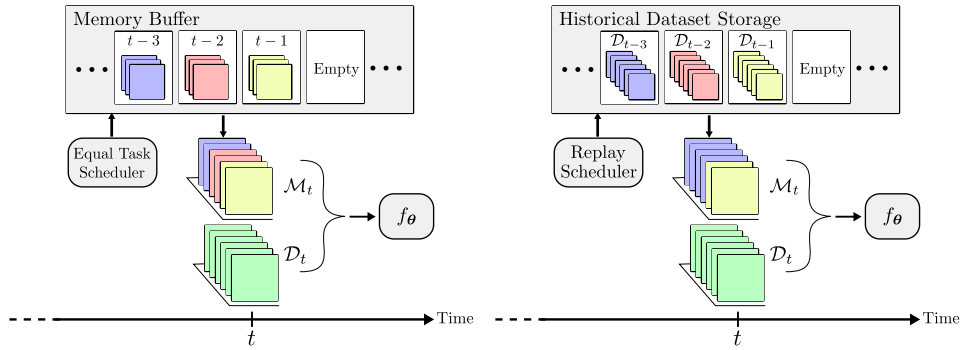
Figure 1.1: Visualizations of the latent representations projected in 2D space with PCA from models VAE$_{\boldsymbol{x}}$ and VCCA$_{\boldsymbol{xiwy}}$, where we plot the corresponding iconic image for each latent representation. We observe that VCCA$_{\boldsymbol{xiwy}}$ structures the items based 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, created the visualizations, and wrote most of the text. All authors took part in discussing the results and contributed to writing the manuscript.

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

Marcus Klasson, Hedvig Kjellström, Cheng Zhang. *Unpublished manuscript.*

Summary: We present a slightly altered CL setting where the historical data is accessible at any time to mitigate catastrophic forgetting. However, retraining the CL system from all historical data is prohibitive due to constraints on processing times. 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. Figure 1.2 illustrates the differences between our replay scheduling approach and the traditional replay approach in CL. We study the benefits of replay scheduling by allowing multiple episodes in the CL environment to enable searches for efficient replay schedules using Monte Carlo tree search. We show that the replay schedules from MCTS can



(a) The traditional replay approach.

(b) Our replay scheduling approach.

Figure 1.2: Illustrations of the (a) traditional replay approach where an equal amount of old samples in a small memory buffer are replayed at every time, and (b) our replay scheduling approach from Paper C where the scheduler selects which tasks to replay by selecting samples from the stored previously seen datasets.

significantly improve the performance compared to baselines with scheduling policies. We conclude that learning the time to replay different tasks can be critical for the performance of replay-based methods in our new proposed CL setting.

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

Paper D: Policy Learning for Replay Scheduling in Continual Learning

Marcus Klasson, Hedvig Kjellström, Cheng Zhang. *Unpublished manuscript.*

Summary: To enable replay scheduling in real-world CL scenarios, we propose a reinforcement learning-based framework for learning replay scheduling policies. Figure 1.3 illustrates the pipeline of our framework. Our intuition is that there may exist general patterns regarding the replay scheduling, e.g., tasks that are harder or have been forgotten should be replayed more often. Therefore, we define the state as the validation performance of the classifier, which is passed to the policy that selects actions for how to compose the replay memory to efficiently mitigate catastrophic forgetting. Our framework enables learning policies that can generalize to new CL scenarios without additional computational cost or training in the test environment. In the experiments, we show that the learned policies can generalize to CL scenarios with new 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, created the visualizations, and wrote most of the text. All authors took part in discussing the results and contributed to writing the manuscript.

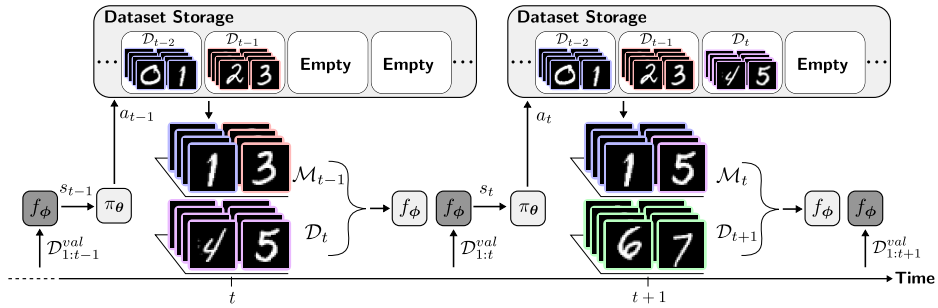


Figure 1.3: Illustration over our policy learning framework for replay scheduling prposed in Paper D. The policy $\pi_{\boldsymbol{\theta}}$ receives state s_t of the classifier $f_{\boldsymbol{\phi}}$ to select action a_t determining how the replay memory $\mathcal{M}_t$ should be constructed. Dark gray-shaded box means that classifier is in evaluation mode.

1.5 Thesis Outline

The rest of the thesis is organized as follows: Chapter 2 provides some preliminaries on deep learning methods in computer vision that we used in this thesis. Chapter 3 is focused on our contributions in FGIR where we summarize Paper A and B on grocery item classification with an assistive vision app. Similarly, in Chapter 4, we focus on our contributions in CL and summarize the content in Paper C and D on our proposed replay scheduling approach in CL. Finally, in Chapter 5, we provide our conclusions of the presented works and present some potential research directions that we believe could be interesting to look deeper into in future research on computer vision-based assistive technologies.

Chapter 2

Background

In this chapter, we provide an overview of the preliminaries for comprehending the methodology in the included papers in this thesis. We assume that the reader has some basic knowledge in calculus, linear algebra, and probability theory. Section 2.1 gives an introduction to machine learning, including different problem settings, some terminology, as well as the notation that will be used throughout this thesis. In Section 2.2, we give an overview of deep learning [74], where we cover the different types of network architectures that has been used in the thesis, as well as a brief introduction to deep reinforcement learning.

2.1 Problem Settings in Machine Learning

Machine learning tasks can be divided into three main fields, namely, *supervised learning*, *unsupervised learning*, and *reinforcement learning* (RL). In this section, we will give a brief introduction to these topics to provide some context on the challenges that we are addressing in this thesis.

We will begin by introducing some algebraic notation that will be used throughout this thesis. For representing data, we use the vector $\boldsymbol{x}$ which is a column vector

$$\boldsymbol{x} = [x_1, \ldots, x_D]^T,$$

where x_i for $1 \leq i \leq D$ is the i-th scalar, real-valued feature of the data vector, and the superscript T transposes the row vector into a column vector. In some cases, each data vector $\boldsymbol{x}$ have an assigned class label y which is an integer number. Scalars can hence both represent real values and integers. The data vector $\boldsymbol{x}$ will often represent a 2-dimensional image, where each feature x_i is a pixel. In this thesis, we still however denote images as column vectors in our equations for simplicity.

We assume that the data follows some underlying distribution, $p_{data}(\boldsymbol{x})$, usually referred to as the data generating distribution. In machine learning, we are often interested in approximating p_{data} with a distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ parameterized by $\boldsymbol{\theta}$ given a finite dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ of N samples where $\boldsymbol{x}^{(i)} \sim p_{data}(\boldsymbol{x})$. Estimating the parameters $\boldsymbol{\theta}$ from the dataset $\mathcal{D}$ is called the *training phase*. A central goal for most applications in machine learning is to make predictions on new data which is called *generalization*. We measure the generalization capability by evaluating the task of interest on a held-out dataset during the *test phase*.

Supervised Learning. In this setting, we are given a dataset $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ where each data point $\boldsymbol{x}^{(i)}$ is accompanied with a target $y^{(i)}$. In classification problems, the target $y \in \{1, \ldots, C\}$ belongs to one of C discrete categories, while in regression problems, the targets $y \in \mathbb{R}$ are continuous and real-valued. The goal is to estimate $p_{\boldsymbol{\theta}}(y|\boldsymbol{x})$ which is the probability of assigning the target y given data $\boldsymbol{x}$. We represent $p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = f_{\boldsymbol{\theta}}(x)$ with the parameterized function $f_{\boldsymbol{\theta}}(x)$ which maps data $\boldsymbol{x}$ to target variables y.

Unsupervised Learning. The goal in the unsupervised setting is to reveal hidden structures in the given dataset $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ without access to target variables. For example, we might be interested in estimating the data generating distribution p_{data} with $p_{\boldsymbol{\theta}}$ (as mentioned above), discovering groups of similar data points using clustering techniques, or projecting high-dimensional data into two or three dimensions for visualization purposes.

Reinforcement Learning. For these problems, we have a *learning agent* that performs actions based on perceived states of the environment to reach some specific goal represented by a reward signal r. An example of such a task is the Mountain Car problem [75] where the agent tries to drive a car over the top of a hill. The action selection is modeled by the policy $\pi_{\boldsymbol{\theta}}(a|\boldsymbol{s})$ which is a mapping from states $\boldsymbol{s}$ to actions a. The goal for the agent is to learn a policy that maximizes the reward within a specified time limit.

2.2 Deep Learning

In this section, we give an introduction to deep learning [74] which is a family of machine learning methods used in this thesis. Deep learning methods are capable of learning tasks from large and high-dimensional datasets thanks to larger volumes of available training data, more efficient machine learning algorithms, and advances in computer hardware in the last decades. A deep learning architecture is constructed by stacking layers of non-linear mapping functions that give intermediate representations of the input data, where the last layer usually outputs a predicted target answer from the queried input. This approach has been successfully applied in various number of fields such as computer vision [25, 26], natural language processing [76], and reinforcement learning [77, 78].

Deep networks are most commonly optimized using the Stochastic Gradient Descent (SGD) algorithm. This procedure uses gradients of a loss function $\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y})$ with respect to every parameter to perform the update step

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}), \tag{2.1}$$

where η is the learning rate which determines how much the weights should be updated with the computed gradient. Common loss functions $\mathcal{L}$ are the cross-entropy loss for classification and the mean-squared error for regression tasks. The gradients are computed using the backpropagation algorithm [79] which estimates the gradients iteratively one layer at a time, going back and forth. To improve the computational efficiency, the mapping function parameters are estimated from the average of the gradients from a randomly sampled mini-batch of data.

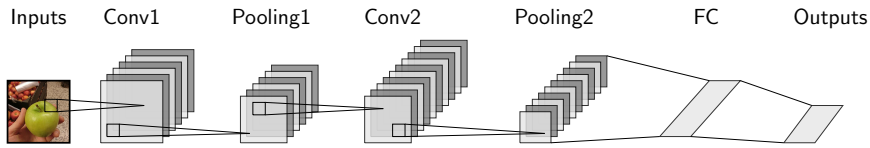


Figure 2.2: Simple CNN architecture with two convolutional layers (Conv) for extracting feature maps, two subsampling layers performing a pooling operation (Pooling), and one fully-connected layer (FC) to produce the network output.

Many innovations have been designed in recent years to stabilize the training of deep neural network architectures. For instance, dropout [80] is used for mitigating overfitting to the training data by randomly making some layer outputs passive to encourage the parameters to take more (or less) responsibility for the inputs. Batch normalization [81, 82] normalizes the layer inputs to enable faster and more stable learning. Residual connections [26] eases training networks with hundreds of layers by mitigating the degradation problem, where adding more layers resulted in higher training error. Furthermore, various adaptive learning rate optimizers [83–85] for the SGD updates, learning rate schedulers [86], and network initializations [87, 88] have been proposed to enable more efficient training. We refer the reader to the provided references for the details on these techniques. Next, we will describe three popular types of neural networks that have been applied in the experiments of this thesis, namely, multilayer perceptrons (MLPs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs).

Multilayer Perceptrons. The simplest form of feedforward neural networks is the MLP based on the Perceptron [89]. Basically, MLPs consist of multiple hidden layers performing matrix multiplication to extract intermediate representations of the input. Each matrix multiplication is followed by an activation function, such as the Rectified Linear Unit (ReLU) activation, which is an essential step for enabling neural networks to learn non-linear functions. Figure 2.1 shows an illustration of an MLP with two hidden layers. The edges between two columns of units represent the matrix multiplication where each edge corresponds to a parameter in the network. The hidden layers are often called fully-connected layers in MLPs.

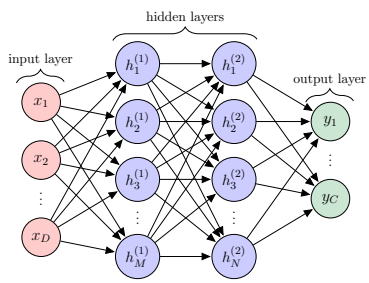


Figure 2.1: Illustration of MLP with two hidden layers.

Convolutional Neural Networks. Convolutional layers consist of a set of filters defined by adaptable parameters to capture relationships between neighboring features in the data [90]. Extracting the feature representations from one layer corresponds to convolving the input data with all filters, where the parameters in each filter are shared across every spatial location of the input. This design choice enables each filter to extract important features at any position in the data. Furthermore, parameter sharing also reduces the amount of parameters in the network and lowers the number of operations for computing the outputs. Figure 2.2 shows an illustration of a CNN architecture. Each filter slides across the width and height of the

input to compute dot products between the filter weights and the local input region to produce a 2-dimensional feature map. The feature maps from all filters are then stacked depth-wise to obtain the output volume, which are often subsampled along their spatial dimensions with a pooling operation after a non-linear activation function. The feature maps in the last subsampling layer are flattened and fed into an arbitrary number of fully-connected layers (an MLP) to produce the final output.

Recurrent Neural Networks. RNNs are a family of neural network models specialized for processing sequences of data. These networks predicts the next output given the past sequence and uses an extension of backpropagation to compute gradients across different time steps. Similar to CNNs, RNNs apply parameter sharing across time steps to enable extracting relevant information that can appear at different positions in the sequence. RNNs makes an assumption that all relevant information about the past sequence $\boldsymbol{x}_{1:t-1}$ can be summarized in a hidden state $\boldsymbol{h}_t$. The state evolves over time through the updated equation $\boldsymbol{h}_t = f_{\boldsymbol{\theta}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_{t-1})$ where $f_{\boldsymbol{\theta}}$ is a neural network for capturing the long-term dependencies. Common choices for this network are the Long Short-Term Memory (LSTM) [91] and Gated Recurrent Unit (GRU) [92] which use gating functions for determining what information is relevant for the future time steps. Figure 2.3 shows a graphical representation of an RNN. The initial hidden state $\boldsymbol{h}_0$ is usually set to zero or learned.

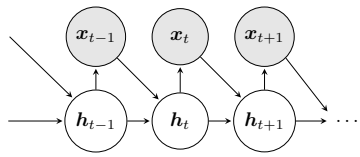


Figure 2.3: Graphical representation of RNN.

Attention-based network architectures, such as Transformers [52,93] and Graph Neural Networks [94,95], are not covered in this thesis; we refer the reader to the provided references for information on these networks.

Deep Autoencoders

A common network architecture for learning latent representations of high-dimensional data in unsupervised fashions with deep learning are autoencoders. These architectures are typically built using two networks called decoder $f_{\boldsymbol{\theta}}$ and encoder $g_{\boldsymbol{\phi}}$ with a bottleneck layer between the networks that extracts a low-dimensional representation $\boldsymbol{z}$ of the data $\boldsymbol{x}$. The idea is that the learned representation $\boldsymbol{z}$ should contain enough relevant information for reconstructing the input data. Figure 2.4 shows an illustration of an autoencoder network. The encoder extracts the latent representation with $\boldsymbol{z} = g_{\boldsymbol{\phi}}(\boldsymbol{x})$, while the decoder aims to reconstruct the original input, such that $\hat{\boldsymbol{x}} = f_{\boldsymbol{\theta}}(\boldsymbol{z})$ where $\hat{\boldsymbol{x}} \approx \boldsymbol{x}$. There exist various autoencoder approaches for improving the quality of the learned representations. For example, denoising autoencoders [96] adds noise to the input data to avoid overfitting, and sparse autoencoders [97] induces adds sparsity constraints on the activation functions to improve robustness.

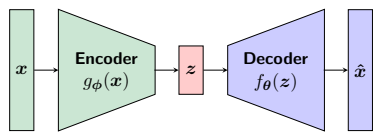


Figure 2.4: Illustration of an autoencoder network.

Multimodal Autoencoders. Autoencoders can easily be extended to handling multiple types of data. The goal here is usually to learn one joint latent representation for capturing correspondences between the data types [98]. Consider the multimodal dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ originate from two different data views (real-world images, web images) or modalities (visual signals, text descriptions) that share some high-level information. For example, $\boldsymbol{x}$ could be an image of a living room and $\boldsymbol{y}$ is a text sentence describing the appearance of the room, where objects are located etc. Figure 2.5 shows an illustration of the network architecture. The input $\boldsymbol{x}$ and $\boldsymbol{y}$ have two separate encoders, $g_{\boldsymbol{\phi_x}}(\boldsymbol{x})$ and $g_{\boldsymbol{\phi_y}}(\boldsymbol{y})$, that extracts the intermediate latent representations $\boldsymbol{h_x}$ and $\boldsymbol{h_y}$ respectively. The intermediate representations are then combined into the joint latent representation $\boldsymbol{z}$, often through some element-wise operation (addition, multiplication) or concatenation. The joint multimodal representation $\boldsymbol{z}$ is then passed through two separate decoder networks, $f_{\boldsymbol{\theta_x}}(\boldsymbol{z})$ and $f_{\boldsymbol{\theta_y}}(\boldsymbol{z})$, for predicting the original input data separately. Combining different modalities, such as images, natural language, and audio signals, with autoencoders for learning more rich representations of data has been studied frequently the last decade [98–103] One advantage is that the network architecture can be trained in an trained end-to-end fashion to be used for both learning representations and making predictions of the used data types. However, a major challenge is how to handle scenarios where some data types could be missing. An option here is to learn a joint representation by using a single encoder for the data type that is available at both training and test phases that is used for predicting each data type [99].

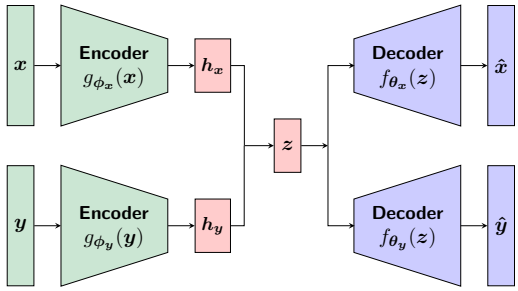


Figure 2.5: Illustration of a multimodal autoencoder network for learning a joint representation $\boldsymbol{z}$ from the different but related data types $\boldsymbol{x}$ and $\boldsymbol{y}$. The joint representation $\boldsymbol{z}$ is obtained by combining the intermediate representations $\boldsymbol{h_x}$ and $\boldsymbol{h_y}$.

Variational Autoencoders

In this section, we give an overview of the variational autoencoder (VAE) [73,104] that we apply for representation learning in Paper A and B. The VAE is a deep generative model that enable learning of data representations by combining ideas from deep learning and probabilistic graphical models [105]. As commonly done in unsupervised learning, VAEs aim to learn a parametric distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ that approximates the data distribution p_{data} given a dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^N$ where $\boldsymbol{x}^{(i)} \sim p_{data}$. We frame the problem of learning $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ by expressing the distribution using latent variables $\boldsymbol{z}$, such that

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z} = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z})\, d\boldsymbol{z}, \tag{2.2}$$

where the joint distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ is the model which factorizes into the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ and the prior distribution $p(\boldsymbol{z})$ for the latent variables. The latent variables $\boldsymbol{z}$

are introduced for representing underlying structures and relationships in $\boldsymbol{x}$ that can be difficult to observe from the data itself. We can view $\boldsymbol{z}$ as containing meaningful information for the data generation process by assuming that the data was generated with the following steps:

1. sample the latent vector $\boldsymbol{z} \sim p(\boldsymbol{z})$ from the prior.
2. generate data point $\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ given the sampled $\boldsymbol{z}$.

The Evidence Lower Bound. The problem with learning the parameters $\boldsymbol{\theta}$ that maximize $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is that the integral in Equation 2.2 is almost always intractable due to the prohibitively large number of values for $\boldsymbol{z}$ that need to be evaluated. Instead, we can reformulate the problem using Bayes' rule

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

where $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ is the posterior of $\boldsymbol{z}$ given $\boldsymbol{x}$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is the likelihood of $\boldsymbol{x}$ given $\boldsymbol{z}$. The posterior needs to be approximated using variational inference [106, 107], where we define an approximate posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ parameterized by $\boldsymbol{\phi}$ to infer the latents $\boldsymbol{z}$. This allows us to form a lower bound on the marginal log-likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ given by

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

which is called the evidence lower bound (ELBO). The expected value over $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ can be estimated with Monte Carlo sampling using $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, and the Kullback-Leibler (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. Commonly, the approximate posterior is 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 parameters $\boldsymbol{\phi}$ are used for computing the mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})$ and standard deviation $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})$ of the approximate posterior given $\boldsymbol{x}$, and $\mathrm{diag}(\cdot)$ is for constraining the covariance matrix to be a diagonal matrix. The prior distribution is also selected to be a zero-mean, standard Gaussian distribution $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{I})$ where $\mathbf{I}$ is the identity matrix.

Network Architecture. The VAE architecture resembles a standard autoencoder where the encoder and decoder networks are probabilistic in the sense that they output parameters of probability distributions. In contrast to autoencoders that learn latent representations $\boldsymbol{z}$ that minimizes the reconstruction error on the data, VAEs learns an approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ that maximizes the likelihood of $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. Similar to autoencoders, we can extract latent representations of the data with the probabilistic encoder. However, VAEs can also estimate the likelihood of unseen data points under the learned model, as well as generate new data by decoding latents sampled from the prior. Figure 2.6 shows an illustration of a VAE architecture. The encoder represents the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ which outputs the mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})$ and standard deviation $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})$ of a Gaussian distribution over $\boldsymbol{z}$, which is the standard choice for $q_{\boldsymbol{\phi}}$ [73]. The latent vector is sampled using the

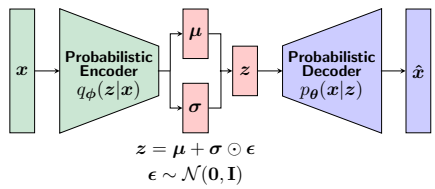


Figure 2.6: Illustration of a variational autoencoder with the reparameterization trick equation.

"reparametrization trick" [73, 108] by computing

$$\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2.5}$$

where $\odot$ denotes element-wise multiplication. The reparametrization enables computing gradients of the expectation $\mathbb{E}_{z\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$ in Equation 2.4 with the backpropagation algorithm. The decoder represents the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ that can be used to generate data $\boldsymbol{x}$ from sampled latents $\boldsymbol{z}$, where the distributional form depends on the data type of $\boldsymbol{x}$.

VAEs have been used in various applications for modeling images, text, and audio data [109–115]. Furthermore, they have been extended to modeling data from different modalities due to their capability of handling missing modalities to enable cross-modal data generation [72, 101, 102, 116, 117]. In Paper A and B, we employ Variational Canonical Correlation Analysis (VCCA) [72] 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 RL to provide some preliminaries for Paper D. The RL setup considers a learning agent that interacts with an environment E over a number of discrete time steps [75]. The environment is modeled with a Markov Decision Process (MDP) [118] represented as a tuple $E = (\mathcal{S}, \mathcal{A}, P, R, \mu, \gamma)$ consisting of the state space $\mathcal{S}$, action space $\mathcal{A}$, state transition probability $P(s'|s, a)$, reward function $R(s, a)$, initial state distribution $\mu(s_1)$, and discount factor γ. 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 for the agent is to learns policy that maximizes the expected return.

The value function $V^\pi(s) = \mathbb{E}[G_t|s_t = s]$ defines the expected return following policy π from state s. Value functions are essential for learning as they can be used for comparing policies, such that $\pi \geq \pi'$ if and only if $V^\pi(s) \geq V^{\pi'}(s)$ [75]. An optimal policy π^* is defined as a policy that is better than or equal to all other policies. There may exist several optimal policies π^* that share the same optimal value function given by $V^*(s) = \max_\pi V^\pi(s)$ for all states $s \in \mathcal{S}$. Similar to the value function, we also have the action value function $Q^\pi(s, a) = \mathbb{E}[G_t|s_t = s, a_t = a]$ which is the expected return for selecting action a in state s and following policy π. Optimal policis also share the same action value function given by $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ for all states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$. The optimal action value function Q^* can be written in terms of the optimal value function V^* as

$$Q^*(s, a) = \mathbb{E}[r_{t+1} + \gamma V^*(s)|s_t = s, a_t = a]. \tag{2.6}$$

This means that if we have access to Q^*, we can directly obtain the optimal action $a*$ in state s by using $a^* = \arg\max_a Q^*(s, a)$ to maximize the expected return.

Deep Q-Networks. The Deep Q-Network (DQN) [77, 119] is a value-based RL algorithm which aims to approximate the optimal action value function as $Q^*(s,a) \approx Q_{\boldsymbol{\theta}}(s,a)$ with a neural network $Q_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$. For learning the function $Q_{\boldsymbol{\theta}}$, we collect data from the environment with by using an epsilon-greedy policy, i.e., $a_t = \arg\max_a Q_{\boldsymbol{\theta}}(s_t, a)$, in state s_t. For estimating $Q_{\boldsymbol{\theta}}$, we minimize the following loss based on the Bellman equation:

$$\mathcal{L}_{\text{DQN}}(\boldsymbol{\theta}) = (r_t + \gamma \max_{a'} Q_{\boldsymbol{\theta}^-}(s_{t+1}, a') - Q_{\boldsymbol{\theta}}(s_t, a_t))^2, \tag{2.7}$$

where $\boldsymbol{\theta}^-$ is a previous copy of the network $\boldsymbol{\theta}$ referred to as the target network. In addition to the introduction of target networks, several methods have been proposed for stabilizing the learning process, such as using experience replay [120] to sample training data stored in a replay buffer, applying L1-smoothing to the loss, correcting the action value estimates [121], and improving generalization across actions [122].

Policy Gradient Methods. An alternative to value-based RL is using policy gradient methods for learning the policy. Here, the optimal policy is estimated directly with a parameterized form $\pi_{\boldsymbol{\theta}}(a|s)$ where $\boldsymbol{\theta}$ represents the parameters of a neural network. The policy network takes the state s_t as input and outputs a distribution over the possible actions a_t. Policy gradient methods often use an actor-critic approach [75, 123], where the policy is known as the actor that selects actions, and the critic evaluates the actions made by the actor by estimating the value function $V_{\boldsymbol{\lambda}}(s_t)$ parameterized by $\boldsymbol{\lambda}$. For updating the policy, the agent collects experiences from the environment with the current policy and aims to minimize the loss

$$\mathcal{L}_{\text{PG}}(\boldsymbol{\theta}) = \mathbb{E}_{s_t \sim E, a_t \sim \pi_{\boldsymbol{\theta}}}[\log \pi_{\boldsymbol{\theta}}(a_t|s_t) \hat{A}(s_t, a_t)], \tag{2.8}$$

where $\hat{A}(s,a)$ is an estimate of the advantage function, which describes how much better or worse a taken action is on average, and is given by

$$\hat{A}(s_t, a_t) = \sum_{i=0}^{k-1} = \gamma^i r_i + \gamma^k V_{\boldsymbol{\lambda}}(s_{t+k}) - V_{\boldsymbol{\lambda}}(s_t), \tag{2.9}$$

where k can vary depending on if we reached a terminal state from s_t and is upper-bounded by the number of steps between updates t_{max} [124]. A popular actor-critic method is Proximal Policy Optimization [125] 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 Image Recognition

In this chapter, we provide an overview for Paper A and B that focuses on fine-grained image recognition (FGIR) [36] of grocery items. Object recognition in grocery shopping scenarios is a challenging task for visually impaired (VI) people [22, 48, 126] due to the necessity of visual capabilities to navigate and identify the products. Regarding identifying products, groceries are often structured in fine-grained manners where details on appearance and shape help us to distinguish between different items. However, since distinguishing between packaged items often requires reading text and raw food items can have similar shapes, recognizing fine-grained details can be difficult without the ability to see. In recent years, there has been an emerging interest of development of computer vision-based assistive technologies for assisting VIs with grocery shopping [12, 57–62]. Nevertheless, fine-grained recognition of groceries still remain challenging due to the need for real-world training data for such systems to be robust in grocery stores. We aimed to enhance the robustness and predictive performance of image classifiers by combining natural images with external web information with the items. In Paper A, we have collected a dataset with natural images of groceries with a mobile phone camera, and web-scraped images and text descriptions about the items, for evaluating image classifiers in real-world supermarket scenarios. We perform an ablation study over the different data types in the data set in Paper B, where we show that utilizing all data types improves the fine-grained classification performance over image classifiers trained with only the natural images. We cover some related work (Section 3.1), describe the dataset collection process (Section 3.2) and our used approach for utilizing multi-view data (Section 3.3), as well as summarize the experimental results from the ablation study (Section 3.4).

3.1 Related Work

In this section, we will briefly discuss the related work on FGIR, datasets for FGIR, and multi-view learning.

Fine-Grained Image Recognition. The goal with FGIR [36] is to distinguish between a set of visually similar object classes that belong to some super-class. The main challenge is to recognize the fine-grained visual details that are required for

discriminating between visually similar objects. In recent years, the successes with deep learning in computer vision have encouraged researchers to explore various approaches for FGIR in different domains with the goal of discriminating between sub-classes of different animals [127], cars [128], fruits [129], and retail products [130]. These approaches can broadly be divided into three directions where the deep networks utilize (i) localization-classification subnetworks, (ii) end-to-end feature encodings, and (iii) external information. In (i), the goal is to find parts that are shared across the sub-classes of some general object and then discover how these localized parts differ in appearance. This can be achieved by utilizing feature maps from convolutional layers as local descriptors [131,132], employing detection and segmentation techniques for localizing object parts [133–135], or leveraging attention mechanisms in scenarios where common object parts are difficult to represent [136, 137]. With (ii), the goal has been to learn features that are better at capturing local differences by computing second-order feature interactions [138, 139], as well as designing novel loss functions [140, 141]. In (iii), data from external domains, such as the web or different modalities (e.g., text or audio), is utilized to influence the learning of fine-grained details in the objects. These additional data types can be relatively cheap to collect and serves as extra guidance for learning useful representations to the FGIR task. The goal with learning from noisy web data is to reduce the need for human-annotated data [142–144], while learning from multimodal data aims to obtain a joint representation that captures the correspondences between the images and the other modalities [145–147]. In addition to FGIR, both web-scraped and multimodal data has been used for zero-shot learning to transfer knowledge from densely annotated categories to learn new fine-grained classes [148–150]. Our work in Paper A and B is mostly related to (iii) FGIR with external information, where we use web-scraped images and text descriptions in combination with natural images to improve the classification performance of fine-grained grocery recognition.

Datasets for Fine-Grained Image Recognition There exist many image datasets for benchmarking computer vision models [15, 127, 151–156]. These datasets ranges from large-scale datasets annotated by groups of classes, such as [151,157], to smaller but densely-annotated datasets [155,158–160]. There also exist domain-specific datasets for FGIR tasks of animals and species [127, 161–164], fashion [165, 166], airplanes and cars [167–170], faces [171–173], and foods [129, 174]. Several of the mentioned datasets have also been extended to benchmarking in zero/few-shot image classification [148,175–177]. In computer vision for visually impaired people, we have recently seen an emergence of datasets collected by people who are blind/low-vision [13–15, 59, 178]. 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 [130, 179–183]. Similarly, there exist other kinds of food datasets with images of various food dishes [174,184,185], cooking videos [186,187], recipes [188–190], and also restaurant-oriented information [191, 192]. Our grocery item dataset (see Section 3.2) shares several 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.

Multi-view Learning. Machine learning with multiple types of data is an active research field [98, 193]. In particular, the intersection of computer vision and natural language processing has been of great interest and has led to applications such as image captioning [28, 194, 195], visual question answering [54, 56, 196]. In this thesis, we have focused on learning joint representations across different data types by applying the subspace learning approach from multi-view learning [193]. A view can be defined as data that has been recorded from a specific sensor [197]. For example, we consider natural and web images to be from different views as the images have been recorded using different cameras. Subspace learning approaches aims to learn joint representations from multiple views by assuming that each view have been generated from a latent space that is shared among the views. Most methods originate from Canonical Correlation Analysis [198] (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 [199] and Deep Canonically Correlated Autoencoders [100]. Inspired from the Deep CCA variants, Variational CCA [72] (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. An advantage of VCCA is the straight-forward extension to modeling both a shared latent space and private latent spaces which capture the view-specific variations to focus the learning of shared variations into the shared latent space [72, 197, 200–203]. In Paper B, we employ VCCA, and its extention to private latent spaces, for learning joint representations of the different data views in our grocery item dataset to obtain more robust image classifiers.

3.2 Dataset Collection

In this section, we describe our procedure for collecting the image dataset of grocery items in Paper A. We visited several supermarkets in Stockholm, Sweden, to collect natural images of groceries with a mobile phone camera to imitate a shopping scenario using an assistive vision device. The collected images captures situations that can be challenging for assistive vision devices, such as misplaced items, hand occlusions, multiple instances and classes present, and various lighting conditions (see Figure 3.1). 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, e.g., apples and milk packages, we also provided the general class of the items to establish a hierarchical labeling structure of the data. Furthermore, we complemented the image dataset with web-scraped information of each grocery item. More specifically, we visited the online shopping website of a Swedish supermarket chain and downloaded 1) an iconic image of the item on a white background, 2) a text description that describes the flavor and ingredients of the item,

Figure 3.1: Example images of challenging scenarios.

Table 3.1: Examples of grocery item classes in the Grocery Store dataset. We display the class label (coarse-grained class in parenthesis), followed by two natural images taken with a mobile phone inside grocery stores, and the 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 descriptions 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**."*

and 3) the nutrition values for items where it was applicable. 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 that helps discriminating between visually similar items. We have established training, validation, and test sets of the natural images where the split was made according to the grocery store location to avoid mixing images from different stores. Finally, we made the dataset publicly available under the MIT license[1].

3.3 Fine-Grained Image Recognition of Grocery Items

In this section, we describe the problem setting of the grocery item classification task from the dataset collected in Paper A. Moreover, we describe our approach from Paper B for learning joint representations of the available data views to improve the classification performance.

Problem Setting

We focus on the application of training an image recognition app in mobile phones for assisting VIs with grocery shopping. Training such image classifiers typically requires immense amounts of annotated images of the groceries. Additionally, the natural images should be collected in scenarios where the app would be used, such as grocery stores or home environments, which further complicates the collection procedure. To

[1]Grocery Store Dataset URL: https://github.com/marcusklasson/GroceryStoreDataset

reduce the need for collecting and annotating natural images from mobile phones, we aim to use web-scraped information about the groceries as additional supervision when training the image classifiers. The web-scraped information should help the classifier to focus on learning the learn fine-grained details of the items to improve the classification performance and robustness in real environments.

The following data views are available in the dataset for training the image classifiers:

- $\boldsymbol{I}$: Natural images of the grocery items taken inside grocery stores.
- $\boldsymbol{x}$: Feature vectors of the natural images $\boldsymbol{I}$ extracted from some CNN $f_{\boldsymbol{\varphi}}$ parameterized by $\boldsymbol{\varphi}$.
- $\boldsymbol{y}$: Class labels of the grocery items in the 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 $\boldsymbol{i}$.

See Table 3.1 for examples of the data views. The straightforward approach is to train a CNN with pairs of natural images $\boldsymbol{I}$ and class labels $\boldsymbol{y}$ in a supervised setting. However, the number of samples may be insufficient for training a CNN from scratch, which can lead to overfitting and low generalization capability to new images. One alternative is to employ transfer learning [204] where parameters from a CNN pretrained on some large dataset, such as Imagenet [151], are used for initialization and we adapt the network to the grocery item recognition task by either 1) fine-tune some of the final layers, or 2) train a linear classifier from extracted features [205]. We take the latter approach and use extracted features for training to mitigate overfitting. Furthermore, we use the web-scraped data views for learning more rich representations of the grocery items that can be used for training more accurate and robust image classifiers, which we present in the next section.

Multi-View Representation Learning of Grocery Items

We describe the multi-view learning approach we applied for learning joint representations of grocery items to use for training the image classifiers. More specifically, we employed a deep latent variable model called Variational Canonical Correlation Analysis [72] (VCCA) for learning the joint representations. In the VCCA approach, we assume that the different data views have been generated from the same, shared latent space before they are observed in their original data type. The goal is to obtain joint representations by learning the shared latent space that captures the correspondences across the available views. The joint representations can then be used for training more accurate and robust image classifiers.

Figure 3.2 shows the network architecture of VCCA used in Paper B for learning the joint representations from all available data views. In this example, we have an input image $\boldsymbol{I}$ of a juice package with the fine-grained class label $\boldsymbol{y}$ *God Morgon Orange and Red Grapefruit Juice*. We obtain the latent representation $\boldsymbol{z}$ by encoding the feature vector $\boldsymbol{x} = f_{\boldsymbol{\varphi}}(\boldsymbol{I})$ of the input $\boldsymbol{I}$ extracted from a CNN $f_{\boldsymbol{\varphi}}$ pretrained on Imagenet. In Paper B, we use a DenseNet [206] architecture as $f_{\boldsymbol{\varphi}}$. We assume that the latent representation $\boldsymbol{z}$ is shared by all available views to establish a more rich representation of the grocery item images by using the each available view as additional supervision to the natural images. We capture the correspondences between

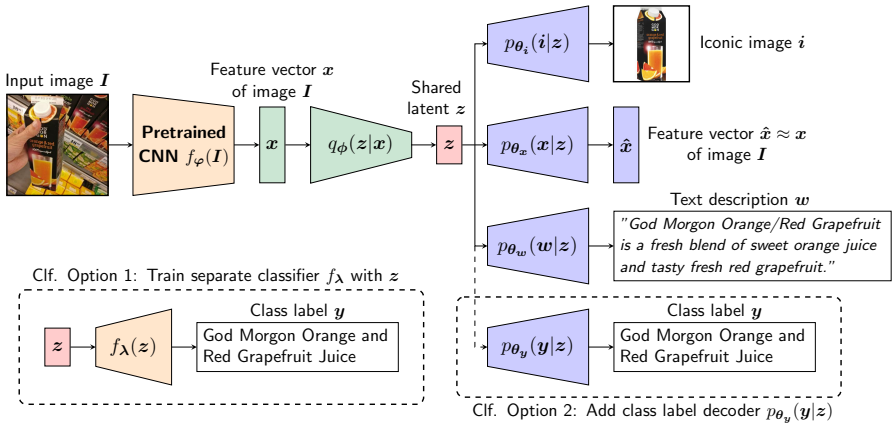


Figure 3.2: Illustration of the VCCA architecture used in Paper B for fine-grained classification.

each view by predicting each view separately using the latent $\boldsymbol{z}$ as input to decoder networks that are specific for each view. The decoder $p_{\boldsymbol{\theta_x}}(\boldsymbol{x}|\boldsymbol{z})$ is an MLP similar to the encoder and outputs a reconstruction $\hat{\boldsymbol{x}}$ of the feature vector $\boldsymbol{x}$. The decoder $p_{\boldsymbol{\theta_i}}(\boldsymbol{i}|\boldsymbol{z})$ is a CNN that predicts the iconic image $\boldsymbol{i}$, while the decoder $p_{\boldsymbol{\theta_w}}(\boldsymbol{w}|\boldsymbol{z})$ is an RNN predicting the next word in the text description $\boldsymbol{w}$. We have two options for training the image classifier. The first option is a two-step procedure where we 1) train the VCCA model from all available views except the class labels $\boldsymbol{y}$, then 2) train a separate classifier $f_{\boldsymbol{\lambda}}(\boldsymbol{z})$ on the learned latent representations $\boldsymbol{z}$ and the class labels $\boldsymbol{y}$. The second option is to learn the classifier simultaneously as the latent space by adding a decoder $p_{\boldsymbol{\theta_y}}(\boldsymbol{y}|\boldsymbol{z})$ that predicts the class label $\boldsymbol{y}$ from $\boldsymbol{z}$. Note that we only use $\boldsymbol{x}$ in the encoder part since the natural images is the only view that is available at test time when predicting the item class of new images taken in the grocery store. In the next section, we perform an ablation study over the available views to analyze how each view affects the fine-grained classification performance.

3.4 Experiments

In this section, we summarize the main experimental results from Paper B. The experiments involved an ablation study using VCCA with the possible combinations of available data views to investigate how each data view contributes to the classification performance. We present results on fine-grained classification performance, visualize the learned latent space to provide insights in how the web-scraped information affect the performance, and demonstrate how the iconic image decoder can be used for explaining the misclassifications by generating iconic images from natural images. We use subscripts to denote which data views that are utilized by VCCA. Adding $\boldsymbol{y}$ to the subscript means that the VCCA model uses a class label decoder $p_{\boldsymbol{\theta_y}}(\boldsymbol{y}|\boldsymbol{z})$ for classification, otherwise we have trained a separate linear classifier $f_{\boldsymbol{\lambda}}(\boldsymbol{z})$ with softmax activation on the latent representation after training VCCA.

Fine-Grained Classification Results. Adding the web-scraped views in VCCA improves the classification performance over approaches that only utilize the natural

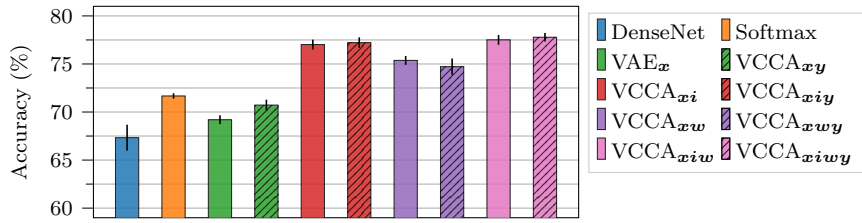


Figure 3.3: Fine-grained accuracies on the Grocery Store dataset for all classification methods. We show the means and standard deviations averaged over 5 seeds. Adding the iconic image $\boldsymbol{i}$ and text description $\boldsymbol{w}$ for learning joint representations with VCCA improves the classification performance over approaches that only utilize the natural images and class labels.

images and class labels. We performed an ablation study over the available views that can be used by VCCA. The fine-grained classification performance was measured with a separate linear classifier trained on the learned latent representations and a class label decoder. We also compared the performance against training a DenseNet from scratch on the dataset (DenseNet) and a linear classifier with softmax activation trained on extracted features from a pretrained DenseNet (Softmax). Figure 3.3 shows a bar plot over the fine-grained accuracies achieved by all classification methods. For the methods only using natural images and class labels, i.e., DenseNet, Softmax, VAE$_{\boldsymbol{x}}$, and VCCA$_{\boldsymbol{xy}}$, we see that Softmax performs best which could be due to loss of information in VAE$_{\boldsymbol{x}}$ and VCCA$_{\boldsymbol{xy}}$ when the feature vectors are compressed into lower-dimensional latent representations. However, the performance of all VCCA models utilizing the iconic image $\boldsymbol{i}$ and/or the text description $\boldsymbol{w}$ outperforms Softmax significantly, which shows that both web-scraped views are useful for enhancing the fine-grained classification performance. Only utilizing $\boldsymbol{i}$ in VCCA performs on par with combining both $\boldsymbol{i}$ and $\boldsymbol{w}$ which could potentially be improved by processing the text descriptions with some word filtering or pretrained word embeddings [76, 207]. Finally, we observe that both classification options for VCCA performs similar, which shows that both options can be practical for classification using the learned latent representation.

Iconic Image Generation. The iconic image decoder can be used for interpreting the outcomes of predicted classes. Figure 3.4 shows two examples of iconic images decoded from VCCA$_{\boldsymbol{xiwy}}$ from two natural images of the classes *Orange Bell Pepper* and *Anjou Pear* from the test set. On the first row, the model has generated an iconic image of an orange-green colored bell pepper with the shape of the green bell pepper in its iconic image. On the second row, the decoded iconic image is a *Granny Smith* apple instead of the true class *Anjou Pear*. Hence, the iconic image decoder can be used as a tool for providing insights in why the classifier makes prediction errors.

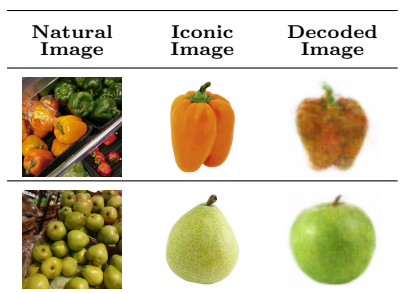
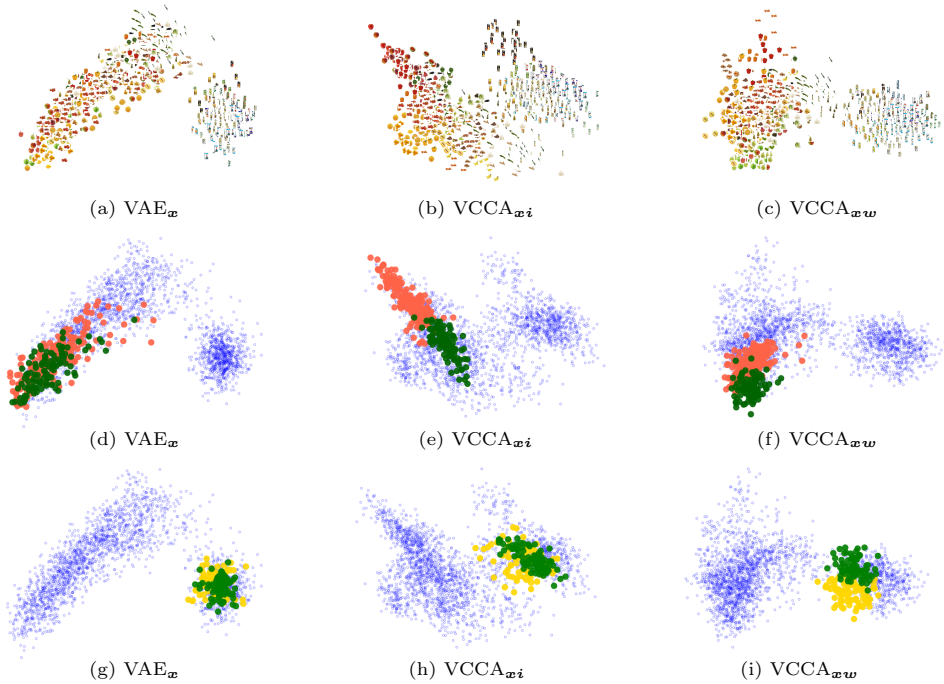


Figure 3.4: Examples of decoded iconic images from VCCA$_{\boldsymbol{xiwy}}$ with their corresponding natural image and true iconic image.



(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 3.5: 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.

Latent Space Visualizations. 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. Figure 3.5 demonstrates how adding either the iconic image $\boldsymbol{i}$ or the text description $\boldsymbol{w}$ can change the structure of the latent space. We used PCA for projecting the learned latent representations of the test set into a 2-dimensional space. We have plotted the corresponding iconic image for the latent representations of VAE$_{\boldsymbol{x}}$, VCCA$_{\boldsymbol{xi}}$, VCCA$_{\boldsymbol{xw}}$ for visualization purposes in Figure 3.5(a-c). The VAE$_{\boldsymbol{x}}$ model in Figure 3.5(a) separates raw and packaged grocery items into two separate clusters in the latent space. When adding the iconic image in in Figure 3.5(b), we observe that the latent space becomes structured according to the color of the items. For VCCA$_{\boldsymbol{xw}}$ in in Figure 3.5(c), the latent space becomes structured according to the ingredients of the items, where we see in the raw grocery item 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 gain more insights in how the web-scraped views affect the latent space. In Figure 3.5(d-f), we inspect how items with different colors are separated in the latent space by utilizing $\boldsymbol{i}$ or $\boldsymbol{w}$ in VCCA by plotting the green and red apple classes as green and red points respectively. All

other classes are plotted as smaller points in blue. We observe that both VCCA$_{\boldsymbol{xi}}$ and VCCA$_{\boldsymbol{xw}}$ manages to separate the apple classes better than VAE$_{\boldsymbol{x}}$ where VCCA$_{\boldsymbol{xi}}$ obtains the most clear separation. We also want to inspect how the views separates visually similar items with different ingredients and flavors. Hence, in Figure 3.5(g-i), we have plotted white-colored yoghurt and juice package classes in the colors green and yellow respectively. We see that VCCA$_{\boldsymbol{xw}}$ manages to separate these items better than VCCA$_{\boldsymbol{xw}}$ due to different ingredient information about yoghurt and juice that is mentioned in the text descriptions.

3.5 Discussion

In this chapter, we summarized the contributions in Paper A and B on FGIR for assisting VI people with grocery shopping using an assistive vision app on their mobile phone. In Paper A, we collected a publicly available dataset with mobile phone images of groceries taken in grocery stores as well as web-scraped information about the items from a supermarket website. In Paper B, we show how the web-scraped information can be multi-view learning can be used for use the web-scraped information to train more accurate and robust image classifiers. More specifically, the iconic images structures the items after visual similarities such as colors and shapes in the learned latent space, while the text descriptions pushes items with similar ingredients and flavors closer to each other. This shows that the cheap-to-collect web-scraped views can serve as a good alternative for improving the fine-grained classification performance instead of collecting more natural images in the grocery stores.

Regarding the dataset collection in Paper A, we have some belated suggestions on how to make the collection procedure easier and also how to enrich the dataset. The images should be collected by recording videos of the grocery items rather than taking still images in the grocery stores. This would potentially enhance the dataset size and could also mitigate the class imbalance in the training and test splits. Furthermore, we could evaluate the classifiers on video sequences which would be a more realistic and user-friendly setting for how an assistive vision app would be used by VI people. Another suggestion to enrich the dataset would be to download more instances of iconic images and text descriptions from supermarket websites, which could potentially allow the VCCA model to capture the view-specific variations into the learned latent representations. Finally, it would have been valuable to have natural images collected by VI people to benchmark the classifiers on scenarios with the actual target users [15, 47].

In Paper B and Section 3.4, we observed that utilizing the iconic images in VCCA affects the classification performance significantly better than the text descriptions. We believe that this could be due to the noise in the text descriptions where often there are only a few words that are relevant to the image recognition task. In the future, we could pre-process the text descriptions by removing stop words, e.g., 'the', 'it', 'and', etc., and other words that are irrelevant for recognizing the fine-grained details of the items. Moreover, we could use pretrained word embeddings [207–209] for considering word similarities, add attention mechanisms [93, 210] in the model for focusing on relevant words, and use data augmentation techniques for text [211] to learn more robust joint latent representations.

Finally, we believe the Grocery Store dataset could be used for benchmarking image classifiers on zero/few-shot learning [39, 40] and continual learning [37, 38] settings. This would be an important extension of the dataset as these applications are crucial to enhance the usability of assistive vision systems in real-world scenarios.

Chapter 4

Continual Learning

In this chapter, we give an overview of Paper C and D where we focus on continual learning (CL) [37, 38]. A CL system receives samples of classes to learn continuously over time while maintaining its overall performance on every seen class. Such capability is critical for assistive vision systems as the system may need to adapt to unseen items in new environments. Furthermore, CL on mobile devices would enable personalizing the devices to specific items that the VI user wants to locate and recognize. The main challenge with the CL setup is catastrophic forgetting [63] where previously learned abilities are overwritten by recent acquired knowledge (see Figure 4.1). However, retraining on all previously seen data may be prohibited due to lack of computational budget or accessibility to the seen datasets. Replay-based CL methods efficiently mitigate catastrophic forgetting by revisiting old samples stored in a small memory buffer [212–214].

Most replay-based methods in CL use simple strategies for retrieving replay samples and ignore the time to learn which is important for memory retention in human learning [67, 69, 70]. Furthermore, most works rely on tiny memory buffers although historical data is almost always available in many real-world machine learning applications [64–66]. Nevertheless, the requirement of small replay memories remains as due to limitations on computational budget and data transmission times. To this end, we propose in Paper C a new CL setting where historical data is available while the processing time is limited. For this new setting, we propose *replay scheduling* where we select which tasks to replay at different times, and demonstrate the advantages of considering time to replay in CL. In Paper D, we present a framework based on reinforcement learning (RL) [75] for learning replay scheduling policies that can be transferred to new CL scenarios for mitigating catastrophic forgetting without added computational cost. Our proposed replay scheduling approach opens up for new research directions that can bring current CL research closer to real-world needs.

4.1 Related Work

In this section, we give an overview of different approaches in CL, especially replay-based methods which our proposed replay scheduling approach in Paper C and D is based on.

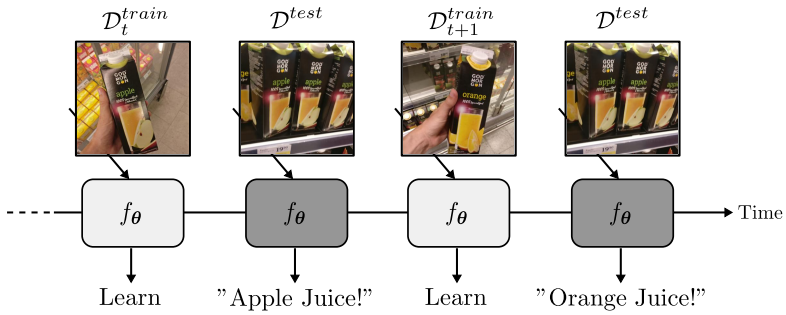


Figure 4.1: Illustration of catastrophic forgetting in a continual learning scenario. The network $f_{\boldsymbol{\theta}}$ receives a task dataset of an Apple Juice class in $\mathcal{D}_t^{train}$ to learn, and correctly predicts the Apple Juice class in the unseen test image in $\mathcal{D}^{test}$. The next task dataset involves learning an Orange Juice class from $\mathcal{D}_{t+1}^{train}$ which $f_{\boldsymbol{\theta}}$ adapts to without retaining knowledge about the Apple Juice class, such that $f_{\boldsymbol{\theta}}$ makes the wrong prediction during evaluation on $\mathcal{D}^{test}$ again.

Continual Learning. The existing CL approaches for mitigating catastrophic forgetting in neural networks can broadly be divided into three main areas, namely, regularization-based, architecture-based, and replay-based methods. Regularization-based methods mainly focus on preventing the parameters important for previous tasks from wide changes and fit the remaining parameters to new tasks [215–217]. Knowledge distillation methods [218] also belong to these methods where stored classification logits are used for regularizing the output units for previous tasks in the network [219, 220]. More recently, projection-based approaches have been proposed for constraining the parameter updates to subspaces which avoid interference with previous tasks [221, 222]. Architecture-based methods focus on adding task-specific network modules for every seen task [223–226], or isolating parameters for predicting specific task in fixed-size networks [227–229]. Replay-based methods revisits samples from old tasks when learning new tasks. The old samples are either stored in an external memory [212–214,230–232], or synthesized with a generative model [116,233,234]. Both regularization- and architecture-based methods can be combined with replay for improving the models capability of remembering tasks [217, 225, 235–239]. Our replay scheduling idea is originated from replay-based methods which we will cover more in detail next.

Replay-based Continual Learning. Much research effort in replay-based CL has focused on selecting higher quality samples to store in memory [212, 217, 240–245]. In [212], several selection strategies are reviewed in scenarios with tiny memory capacity, such as reservoir sampling [246], first-in first-out buffer [213], k-Means, and Mean-of-Features [244]. However, elaborate selection strategies have been shown to give little benefit over random selection for image classification problems [214, 247]. More recent work has been focused on enhancing the memory capacity by storing compressed features rather than raw data [214, 248], evolving the memory samples using data augmentation to avoid overfitting [231,249], and using contrastive learning to improve memory retention [250,251]. Selecting the time to replay old tasks has mostly been ignored in the literature, with an exception in [252] which replays memory samples that would most interfere with a foreseen parameter update. Our replay scheduling approach differs from the above mentioned works since we focus on

learning to select which tasks to replay. Nevertheless, our scheduling can be combined with any selection strategy and replay-based method.

4.2 Replay Scheduling in Continual Learning

In this section, we present our proposed replay scheduling approach studied in Paper C and D for mitigating catastrophic forgetting in CL. More specifically, we focus on the contributions in Paper C, where we introduce a slightly new CL setting that considers real-world needs where all historical data can be available since data storage is cheap. Additionally, we demonstrate the advantages of replay scheduling in such setting where we select which tasks to replay, where we use Monte Carlo tree search (MCTS) [253] to search for efficient replay schedules by allowing multiple episodes in single CL environments.

New Problem Setting for Continual Learning

In this section, we describe the proposed CL setting presented in Paper C. We assume that historical data is accessible at any time for mitigating catastrophic forgetting in the model. However, re-training the model on all available historical data is prohibited due to limitations on compute and data transmission times. Therefore, the goal is to determine which historical tasks to replay and sample a small replay memory from the selected tasks to mitigate catastrophic forgetting as efficiently as possible.

The notation of this problem setting resembles the traditional CL setting for image classification. We let a neural network $f_{\boldsymbol{\phi}}$, parameterized by $\boldsymbol{\phi}$, learn T tasks from the datasets $\mathcal{D}_1, \dots, \mathcal{D}_T$ arriving sequentially one at a time. The t-th dataset $\mathcal{D}_t = \{(\boldsymbol{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^{N_t}$ consists of N_t samples where $\boldsymbol{x}_t^{(i)}$ and $y_t^{(i)}$ are the i-th data point and class label respectively. Additionally, the dataset is split into training, validation, and test sets such that $\mathcal{D}_{1:T} = \{(\mathcal{D}_t^{train}, \mathcal{D}_t^{val}, \mathcal{D}_t^{test})\}_{t=1}^{T}$. The training objective at task t is given by

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

where $\mathcal{L}(\cdot)$ is the loss function, which in our case is the cross-entropy loss. The challenge is for the network $f_{\boldsymbol{\phi}}$ to retain its classification performance on the previous $t-1$ tasks.

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 mitigating catastrophic forgetting of old tasks due to processing time constraints. The challenge then becomes how to select the M samples from the historical data to include in the replay memory that efficiently retain the previous knowledge. We focus on selecting the samples on task-level by deciding on the task proportions $(p_1, \dots, p_{t-1})$ on samples to fetch from each task, 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 $p_i \cdot M$. To simplify the selection of which tasks to replay, we construct a finite set of possible task proportions that can be used for constructing $\mathcal{M}$.

This problem setting shares several assumptions with traditional CL, such as:

- The model receives the datasets at different time steps from a continuous stream.
- Re-training on all historical data is prohibited.
- The model should perform well overall across all seen tasks and, hence, mitigate catastrophic forgetting.
- Replay memory size M is small.

The only difference with this setting and traditional CL is that all historical has been stored rather than deleted, and we can access this data to fill the replay memory $\mathcal{M}$ to mitigate catastrophic forgetting. We argue that this setting aligns well with real-world needs as data storage is cheap and easy to maintain but retraining large machine learning models is computationally expensive. Therefore, we use small replay memory sizes to limit the processing times of replay when the model is learning new tasks.

Monte Carlo Tree Search for Replay Scheduling

In this section, we describe how we used MCTS in Paper C to show the advantages of replay scheduling in the new CL setting. We focus on an ideal CL environment where executing multiple episodes is allowed to enable searching for replay schedules. For demonstration purposes, we use MCTS for finding replay schedules that efficiently mitigate catastrophic forgetting by selecting compositions of replay memories at different time steps.

We define a replay schedule as a sequence $S = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{T-1})$, where $\boldsymbol{p}_i = (p_1, \ldots, p_{T-1})$ 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 a discrete action space with a finite number of choices for how to construct the replay memory at different time steps. Algorithm 1 shows the procedure for creating this action space. At task i, we have $i-1$ historical tasks that we can choose from. We then generate all possible bin vectors $\boldsymbol{b}_i = [b_1, \ldots, b_i] \in \mathcal{B}_i$ of size i where each element are a task index $1, ..., i$. The elements in each bin vector are sorted by task index, and we only keep the unique bin vectors after sorting. 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 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. A replay schedule S is constructed by traversing through $\mathcal{T}$ and appending a single task proportion from every task level to S.

Algorithm 1 Discretization of action space with task proportions

Require: Number of tasks T

$\mathcal{T} = ()$
for $i = 1, \ldots, T-1$ **do**
  $\mathcal{P}_i = \{\}$
  $\mathcal{B} = \texttt{combinations}([1:i], i)$
  $\mathcal{B}^* = \texttt{unique}(\texttt{sort}(\mathcal{B}))$
  for $\boldsymbol{b}_i \in \mathcal{B}^*$ **do**
    $\boldsymbol{p}_i = \texttt{bincount}(\boldsymbol{b}_i)/i$
    $\mathcal{P}_i = \mathcal{P}_i \cup \{\boldsymbol{p}_i\}$
  end for
  $\mathcal{T}[i] = \mathcal{P}_i$
end for
return $\mathcal{T}$

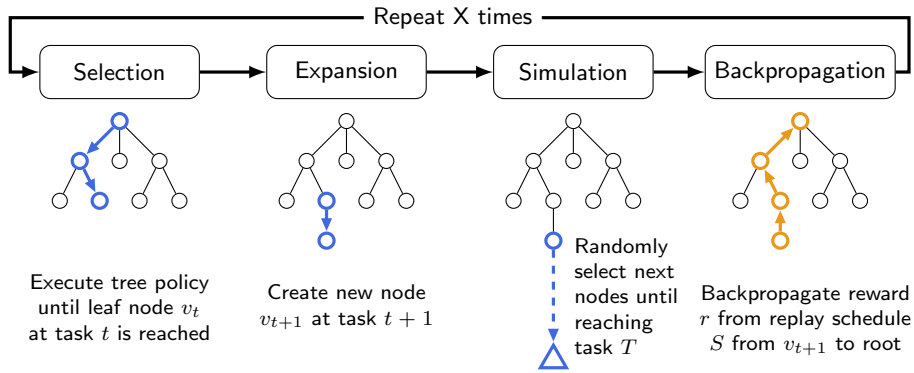


Figure 4.2: Outline of MCTS when searching for replay schedules.

The tree-shaped action space of task proportions grows fast with the number of tasks, which complicates studying replay scheduling in datasets with longer task-horizons. Therefore, we proposed using MCTS since it has been successful in applications with large action spaces [78, 254, 255]. In our case, MCTS concentrates the search for replay schedules in directions with promising CL performance in the environment. Each memory composition in the action space corresponds to a node that can be visited by MCTS. For instance, at task t, the node v_t is related to a task proportions $\boldsymbol{p}_t$ that can be used for retrieving a replay memory from the historical data. We store the related task proportion $\boldsymbol{p}_t$ from every visited node v_t in the replay schedule S. The final replay schedule is then used for constructing the replay memories at each task during the CL training. Figure 4.2 shows an illustration of the procedure for executing MCTS to search for replay schedules, where each step involves the following:

Selection. During a rollout, the tree policy either selects an unvisited child randomly from the current node v_t, or selects the next node by evaluating the Upper Confidence Tree (UCT) [256] if all children has been visited earlier. The child v_{t+1} with the highest UCT score is selected using the function from [255]:

$$UCT(v_t, v_{t+1}) = \max(q(v_{t+1})) + C\sqrt{\frac{2\log(n(v_t))}{n(v_{t+1})}}, \tag{4.2}$$

where $q(\cdot)$ is the reward function, C the exploration constant, and $n(\cdot)$ the number of node visits. The tree policy is executed until a leaf node v_t is reached.

Expansion. If the leaf node v_t has unvisited children, 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 succeeding nodes are selected randomly until reaching a terminal node v_T. The task proportions $\boldsymbol{p}_{1:T-1}$ from the visited nodes in the rollout constitutes the replay schedule S. After training the network using S for replay, we calculate the reward for the rollout given by $r = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}^{(val)}$, where $A_{T,i}^{(val)}$ is the validation accuracy of task i at task T.

Backpropagation. The reward r is backpropagated from the expanded node v_{t+1} to the root node, where the reward function $q(\cdot)$ and number of visits $n(\cdot)$ are

updated at each node in the path.

As mentioned earlier, we use MCTS to search for replay schedules to illustrate the importance of learning the time to learn. To this end, we study replay scheduling in an ideal CL environment where multiple episodes is allowed to find replay schedules. However, in the standard CL setting, multiple episodes are prohibited which currently prevents replay scheduling to be applied in real-world CL scenarios. In the next section, we present a policy learning framework based on RL to learn replay scheduling policies that generalize.

4.3 Policy Learning for Replay Scheduling

In this section, we introduce the RL-based framework for learning replay scheduling policies that was proposed in Paper D. In real-world CL settings, learning the policy with multiple episodes is infeasible, which is why we need a general policy for replay scheduling. Our intuition is that there may exist general patterns regarding the replay scheduling, e.g., tasks that are harder or have been forgotten should be replayed more often. Moreover, the policy may non-trivially take task properties into consideration. Therefore, we aim to learn policies that select which tasks to replay from states representing the current task performance in the CL environments. The policy can then be applied for mitigating catastrophic forgetting in new CL scenarios.

We model the CL environments as Markov Decision Processes [118] (MDPs) where each 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 $\mu_i(s_1)$, and discount factor γ. We assume that we have access to a fixed set of training environments $\mathcal{E}^{(train)} = \{E_1, \ldots, E_K\}$ sampled from a distribution of CL environments, i.e., $E_i \sim p(E)$ 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 by $f_{\boldsymbol{\phi}}$ at time step t. Note that we will use task and time step interchangeably. To generate diverse CL environments, we obtain environments with different initializations of the network $f_{\boldsymbol{\phi}}$ and shuffled task orders in the dataset when we sample environments from $p(E)$.

We define the state s_t of the environment as the validation accuracies $A_{t,1:t}^{(val)}$ on each seen task $1, ..., t$ from $f_{\boldsymbol{\phi}}$ at task t, i.e., $s_t = [A_{t,1}, ..., A_{t,t}, 0, ..., 0]$, where we use zero-padding on future tasks. The action space $\mathcal{A}$ is constructed as described in Algorithm 1 (Section 4.2), such that the $a_t \in \mathcal{A}$ corresponds to a task proportion $\boldsymbol{p}_t$ used for sampling the replay memory $\mathcal{M}_t$. We use a dense reward based on the average validation accuracies at task t, i.e., $r_t = \frac{1}{t}\sum_{i=1}^{t} A_{t,i}^{(val)}$. The state transition distribution $P_i(s'|s,a)$ represents the dynamics of the environment, which depend on the initialization of $f_{\boldsymbol{\phi}}$ and also the task order in the dataset. More specifically, the dynamics represent how the validation accuracies varies after $f_{\boldsymbol{\phi}}$ replays according to the action a_t when learning the current task.

The procedure for training the policy goes as follows: The state s_t is obtained by evaluating the network $f_{\boldsymbol{\phi}}$ on the validation sets $\mathcal{D}_{1:t}^{(val)}$ after learning the t-th task from $\mathcal{D}_t^{(train)}$. Action a_t is selected under the policy $\pi_{\boldsymbol{\theta}}(a|s_t)$ parameterized by $\boldsymbol{\theta}$. The action is converted into task proportion $\boldsymbol{p}_t$ that is used for sampling the replay memory $\mathcal{M}_t$ from the historical datasets. We then train classifier $f_{\boldsymbol{\phi}}$ with $\mathcal{D}_{t+1}^{(train)}$

and $\mathcal{M}_t$, and obtain the reward r_{t+1} and the next state s_{t+1} by evaluating f_ϕ on the validation sets $\mathcal{D}_{1:t+1}^{(val)}$. The collected transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ are used for updating the policy. A new episode starts after f_ϕ has learned the final task T.

We evaluate the learned policy by applying it to mitigate catastrophic forgetting in new CL environments at test time. To foster generalization across environments, we train the policy on multiple environments with different dynamics, such as task orders and datasets, to learn from a diverse set of training data. The goal for the agent is to maximize the sum of rewards in each training environment. At test time, the policy is applied on new CL classifiers and datasets in the test environments without added computational cost nor experience collection. In Section 4.4, we test the policies generalization capability to new CL environments where the task orders and datasets are unseen during training.

4.4 Experiments

In this section, we give an overview of the experimental results from Paper C and D. In Paper C, we evaluate the benefits of replay scheduling in single CL environments using MCTS for finding replay schedules. We perform extensive evaluation to show that the replay schedules from MCTS can outperform the baselines across several CL benchmarks and different backbones. In Paper D, we evaluate our RL-based framework using DQN [119] and A2C [124] for learning policies that generalize to new CL scenarios. We show that the learned policies can efficiently mitigate catastrophic forgetting in CL environments with new task orders and datasets that are unseen during training.

Results on Replay Scheduling with Monte Carlo Tree Search

Here, we provide a selection of the experimental results from Paper C. Our approach has been named Replay Scheduling MCTS (RS-MCTS). We evaluate the CL performance of RS-MCTS for varying memory sizes to illustrate the importance of replay scheduling. Moreover, we show that replay scheduling can benefit any replay-based method by combining scheduling with three recent CL methods, namely, HAL [257], MER [258], and DER [235]. Finally, we demonstrate the efficiency benefits of replay scheduling in settings when the memory size is smaller than the number of classes. We give an overview of the experimental settings, see Paper C for the full experimental results and hyperparameter settings.

Datasets and Architectures. We perform experiments on common CL benchmark datasets such as Split MNIST [216], Split Fashion-MNIST [259], Split notMNIST [260], Permuted MNIST [261], and Split CIFAR-100 [154, 213, 244] and Split miniImagenet [262]. We use a 2-layer MLP with 256 hidden units and ReLU activations for Split MNIST, Split FashionMNIST, Split notMNIST, and Permuted MNIST. For Split CIFAR-10 and CIFAR-100, we use the CNN architecture used in [220, 262, 263], and a reduced ResNet-18 from [213] for Split mini-Imagenet. We use a multi-head output layer for each dataset, except Permuted MNIST where the network uses single-head output layer, such that we assume that task labels are available at test time [264].

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. Memory sizes are $M = 10$ and $M = 100$ for the 5-task and 10/20-task datasets respectively. We report the mean and std. averaged over 5 seeds. * denotes where some seed in Heuristic did not converge.

		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)

Baselines. We compare our method against the following scheduling baselines:

- **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. We use the following evaluation metric for measuring the CL performance

$$\text{ACC} = \frac{1}{T}\sum_{j=1}^{T} A_{T,j}^{test}, \tag{4.3}$$

where $A_{T,j}^{test}$ is the accuracy of task j after learning the final task T.

Varying Memory Size. We show that the replay schedules found by MCTS improves the CL performance across different memory sizes. In Figure 4.3, 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. These results show that replay scheduling can outperform the baselines, especially for small M, on both small and large datasets across different backbone choices.

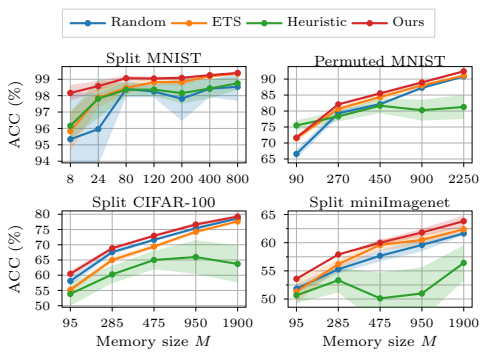


Figure 4.3: Performance comparisons with ACC over different replay memory sizes M for RS-MCTS (Ours) and the baselines. All results have been averaged over 5 seeds.

Applying Scheduling to Recent Replay Methods. We show that replay scheduling can be combined with any memory-based CL method to enhance the performance. Table 4.1 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

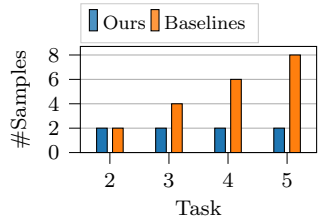


Figure 4.4: Number of replayed samples per task for the 5-task datasets for RS-MCTS (Ours) and the baselines in the tiny memory setting.

Method	5-task Datasets		
	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.

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. These results 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 a memory setting where we can only store 1 sample/class. 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 for our method to $M = 2$ for the 5-task datasets. We then compare against the most memory efficient CL baselines, namely A-GEM [265], ER-Ring [212] which have shown promising results with 1 sample/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 4.4. 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 this section, we present the experimental results from Paper D showing that the policies learned with our RL-based framework using DQN and A2C are capable of generalizing across CL environments with new task orders and datasets unseen during training. We run experiments on 5-task datasets Split MNIST [216], Split FashionMNIST [259], Split notMNIST [260], and Split CIFAR-10 [154] where there are 2 classes/task. We resort to CL benchmark datasets with $T = 5$ tasks to have feasible size of the discrete action space. The replay memory size is set to $M = 10$ in all experiments. See Paper D for full details on the experimental setup.

DQN and A2C Architectures. The input layer has size $T-1$ where each unit is inputting the task performances since the states are represented by the validation accuracies $s_t = [A_{t,1}^{(val)}, ..., A_{t,t}^{(val)}, 0, ..., 0]$. The current task can therefore be determined by the number of non-zero state inputs. The output layer has 35 units representing the possible actions at $T = 5$ in the discrete action space. We use action masking on the output units to prevent the network from selection invalid actions for constructing the replay memory at the current task. The DQN is a 2-layer MLP with 512 hidden units and ReLU activations. For A2C, we use separate networks for parameterizing the policy and the value function, where both networks are 2-layer MLPs with 64 hidden units of Tanh activations.

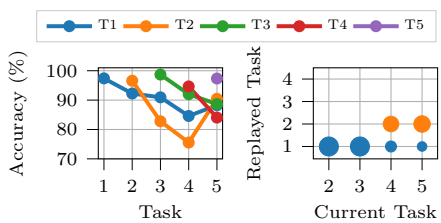
Baselines. We add two more heuristic scheduling baselines:

- **Heuristic Local Drop (Heur-LD).** Heuristic policy that replays tasks with validation accuracy below a threshold proportional to the previous achieved validation accuracy on the task.
- **Heuristic Accuracy Threshold (Heur-AT).** Heuristic policy that replays tasks with validation accuracy below a fixed threshold.

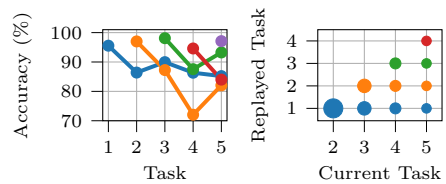
Here, we name the Heuristic from the experiments in Paper C as Heuristic Global Drop (Heur-GD). The Random and ETS baselines are the same as earlier.

Evaluation Protocol. We use ACC from Equation 4.3 to evaluate the performance in the CL environments. The reported results have been evaluated on the test set where the replay schedules are selected from evaluating the validation sets, where 15% of the training data is used for creating the validation sets. To assess generalization capability, we use a ranking method based on the ACC between the methods in every test environment for comparison (see Paper D for details).

Generalization to New Task Orders. We show that the learned replay scheduling policies can generalize to CL environments with previously unseen task orders. We generate training and test environments with unique task orders for three datasets, namely, Split MNIST, FashionMNIST, and CIFAR-10. Table 4.3 shows the average ranking for the DQN, A2C, and the baselines when being applied in 10 test environments. Our learned policies obtain the best average ranking across the datasets, where the DQN performs best for Split MNIST and FashionMNIST while A2C outperforms all methods on Split CIFAR-10. We provide further insights in the benefits of learning the replay scheduling policy by visualizing the replay schedule and task accuracy progress from A2C in one Split CIFAR-10 test environment in Figure 4.5.



(a) A2C - ACC: 89.75%



(b) ETS - ACC: 88.32%

Figure 4.5: Task accuracies and replay schedules for A2C and ETS for a Split CIFAR-10 environment.

Table 4.3: Average ranking (lower is better) for experiments on generalizing policies to environments with new task orders or a new dataset. We average the results over 10 test environments.

	New Task Order			New Dataset	
Method	S-MNIST	S-FashionMNIST	S-CIFAR-10	S-notMNIST	S-FashionMNIST
Random	4.23	3.60	5.03	3.87	3.95
ETS	3.80	4.57	5.37	4.27	3.63
Heur-GD	4.48	4.25	3.98	4.58	**2.77**
Heur-LD	4.65	3.65	3.75	4.97	5.10
Heur-AT	4.33	3.87	3.43	4.28	3.72
DQN (Ours)	**3.23**	**3.55**	3.68	3.20	4.37
A2C (Ours)	3.27	4.52	**2.75**	**2.83**	4.47

Additionally, we show the replay schedule and task accuracies from the ETS baseline in the same environment. The replay schedules are visualized with bubble plots showing the selected task proportion to use for composing the replay memories at each task. In Figure 4.5a, we observe that A2C decides to replay task 2 more than task 1 as the performance on task 2 decreases, which results in a slightly better ACC metric achieved by A2C than ETS. These results show that the learned policy can flexibly consider replaying forgotten tasks to enhance the CL performance.

Generalization to New Datasets. We show that the learned replay scheduling policy is capable of generalizing to CL environments with new datasets unseen in the training environments. We perform two sets of experiments, 1) train with environments generated with Split MNIST and FashionMNIST and test on environments generated with Split notMNIST, and 2) train with environments generated with Split MNIST and notMNIST and test on environments generated with Split FashionMNIST. Table 4.3 shows the average ranking for DQN, A2C, and the baselines when generalization to test environments with new datasets. We observe that both A2C and DQN successfully generalize to Split notMNIST environments outperforming all baselines. However, the learned policies have difficulties to generalize to Split FashionMNIST environments, which could be due to high variations in the dynamics between training and test environments. The policy may exhibit state transition dynamics which has not been experienced during training, which makes generalization difficult for both DQN and A2C. Potentially, the performance could be improved by generating more training environments for the agent to exhibit more variations in the CL scenarios or by using other advanced RL methods which may generalize better [266].

4.5 Discussion

In this chapter, we summarized our contributions in CL presented in Paper C and D. In Paper C, we focused on a new CL setting where historical data is accessible at any time for mitigating catastrophic forgetting, but replaying all historical data is prohibited due to processing time constraints. We then proposed learning the time to learn for a CL system where we learn replay schedules over which tasks to replay at different time steps. To demonstrate the importance of learning the time to learn, we use MCTS to find the proper replay schedule and show that it can significantly outperform fixed scheduling policies in terms of CL performance. In Paper D, we address the efficiency of replay scheduling by proposing to use RL to learn the replay

scheduling policies that can generalize to new continual learning scenarios without added computational cost. In the experiments, we show that the learned policies can propose replay schedules that efficiently mitigate catastrophic forgetting in environments with previously unseen task orders and datasets. The proposed approaches opens up for new research directions in replay-based CL that stems well with real-world needs where training CL systems can be limited by processing time constraints rather than storing historical data for maintaining overall performance.

We argue that replay scheduling brings current CL research closer to real-world CL scenarios. Take for example image classification apps on mobile phones that should assist VI people with recognizing objects in their surroundings. The users may want to customize their apps to recognize new and personal items continuously. The app should adapt fast from few samples while maintaining its performance on the already known classes which is where CL comes in. However, as we saw in Paper C and D, fixed replay policies show difficulties performing well given different task orders, which potentially complicates generalizing to different users and new environments. To this end, we could transfer a learned replay scheduling policy that generalizes to new users and their unseen environments to mitigate catastrophic forgetting efficiently for any user's personal image recognition app. Moreover, a more general case for the importance of replay scheduling in CL is scenarios where the allowed replay memory size is smaller than the number of seen classes to remember. Scheduling over which task to replay at different times then becomes crucial to balance efficient learning of new classes while retaining previously learned abilities.

Generalization in RL is a challenging research topic by itself [267–269]. The main limitation with the RL-based framework proposed in Paper D is the requirements of large amounts of diverse data and training time to enable the learned policy to generalize well. This can be expensive due to the computational cost to generate the CL environments since each state transition involves training the classifier on a CL task. Moreover, we have currently only considered a discrete action space which is hard to construct especially when the number of tasks is large. Thus, in future work, we would explore more advanced RL methods which can handle continuous action spaces and generalize well [266].

Finally, we want to highlight the privacy issues that can arise when applying replay scheduling. Any replay-based CL method inherently can violate privacy if raw samples are stored in the memory, but these risks can be mitigated by instead storing compressed feature representations. Furthermore, as mentioned above, we needed substantial amounts of data and training time for learning policies that generalize which increases the need for secure data storage solutions. As the states were defined using validation accuracies rather than images, we have circumvented the potential privacy issues as only the state transitions are stored from the generated CL environments to use for training the policy.

Chapter 5

Conclusions and Future Directions

In this chapter, we provide our conclusions from the works described in this thesis (Section 5.1). Thereafter, we discuss potential future directions for work towards better computer vision methods for assisting visually impaired (VI) people (Section 5.2).

5.1 Conclusions

We have divided our conclusions into two parts, where we first provide our insights on the work on fine-grained image recognition (FGIR) of grocery items presented in Chapter 3, and thereafter our insights on our proposed replay scheduling method for continual learning (CL) presented in Chapter 4.

Fine-Grained Image Recognition of Groceries

Here, we provide our insights around the work presented in Chapter 3.

Dataset Collection. In Paper A, we presented a realistic dataset of images of groceries with a mobile phone to simulate a grocery shopping scenario using an assistive vision app. Collecting these images in the grocery stores was indeed time consuming and required effort. However, as there is a gap between benchmark datasets and reality, we argue that evaluating computer vision methods on real-world data and scenarios is critical for assistive vision. One major learned lesson is that we should have collected the images by recording videos rather than taking snapshot images. First of all, recording videos to classify the groceries would be more user-friendly and probably more accurate than processing static images one at a time. The dataset size would probably be enhanced as well, and the dataset could be used for evaluating both image and video models on supervised classification. In addition to data quality, the selection of network architecture can also have high influence on the final performance. Therefore, we should have performed a broader study over different network types, especially those specialized for mobile computing [270, 271], while measuring memory consumption and compute requirements as computation should be on-device to preserve user privacy [272]. It is also important to collect meta-data, such as hardware information, image resolutions, and geographical loca-

tion, about the mobile phone images to develop image recognition models that are device-agnostic and robust in new environments. Finally, it would have been valuable to include images from blind/low-vision collectors to measure the FGIR performance with data from the actual users of assistive vision devices [273].

Utilizing Web-Scraped Views for FGIR. 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 FGIR performance. Our ablation study showed that the learned latent representations improves the classification performance as the iconic images structures the items based on visual similarities (e.g. color and shape) in the latent space, while the text descriptions structures the items based on ingredients and flavors. To provide more insights on how the web-scraped views boost the performance, we could analyze the natural images with visual explanation methods, e.g., with class activation maps [274, 275], to investigate whether the classifier recognizes the fine-grained details in the items. However, performing such evaluation automatically probably requires human annotations on the location of the details. Adding more examples of iconic images and text descriptions may enable better modeling of the view-specific variations to more accurately capture the shared information about the views into the learned latent representations. In combination with more advanced text processing [209, 211], this could potentially enhance the separation between visually similar packaged items with different ingredients to further improve the FGIR performance. Finally, it would be interesting to study how the web-scraped views can be utilized to enhance the data-efficiency in few-shot learning and CL scenarios, which are important applications for assistive vision systems.

Replay Scheduling in Continual Learning

In this section, we provide our insights around the work presented in Chapter 4.

Benefits of Replay Scheduling. In Paper C, we proposed a slightly new CL setting where 1) the historical is stored rather than deleted and 2) the replay memory size is limited by processing time constraints instead of storage capacity. This setting stems well with real-world needs as many major companies stores all their recorded data [66, 276, 277]. We then showed in an ideal CL environment that scheduling over which tasks to replay can significantly improve the final CL performance compared to using fixed policies, e.g., replaying all tasks equally, especially with small replay memory sizes. Consider a CL system where the number of seen classes is much larger than the number of samples that the system has time to replay. In such scenario, it becomes crucial for the system to have a policy scheduling over which tasks to replay for reducing catastrophic forgetting efficiently. Furthermore, advances in CL are critical for assistive vision devices to maintain recognition of new objects in ever-changing environments without the need for re-training the systems from scratch. As these systems could require on-device learning to maintain privacy, replay scheduling could potentially retain the previous knowledge better than fixed scheduling policies and reduce idle times when learning new abilities.

Learning Replay Scheduling Policies. In Paper D, we proposed a framework based on reinforcement learning (RL) for learning policies scheduling which tasks to replay based on the current CL performance. Furthermore, we showed that the

replay scheduling policies could generalize to new CL scenarios without additional computational cost nor interactions with the test environments. The proposed approach enables replay scheduling to be applied in standard CL settings. However, as we only considered a discrete action space in the experiments, we need to extend the framework to handle continuous action spaces that would scale better to larger number of tasks. Furthermore, learning a policy that generalizes well to new scenarios required large amounts of diverse data and training time, so we need to explore more sample-efficient methods. We believe that our policy learning framework could be beneficial for user personalization of assistive vision systems using CL. Since fixed scheduling policies show difficulties generalizing to new CL scenarios in our experiments, we could instead transfer a learned replay scheduling policy that potentially can generalize to new users and their surroundings to reduce catastrophic forgetting in their personal devices.

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 [278] 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 [278–280] in the machine learning community. Federated learning involves leveraging the storage and computational resources of remote devices, e.g., mobile phones, for training predictive models locally on the device. The local model updates are transmitted to a central server while user privacy is preserved by keeping the training data on the device. The server collects model updates from several connected devices for training a global model by aggregating the local model updates [280]. The global model can then be distributed to the connected devices if necessary without sharing any data between the devices.

Example Scenario. Here, we give an example application of federated learning in the context of assistive vision. Alice and Bob are using an assistive vision app installed on their mobile phones for recognizing groceries in a supermarket. Today, Alice needs to buy milk and wants to recognize a milk package held in her hand. She records a video with the phone camera which captures several frames of the milk package that are then processed in the app. However, the app responds that this milk package is an unknown item.Alice asks Charles, the store clerk, what item she is holding, who tells Alice that this type of milk brand is new in the store. Charles helps Alice with updating her app with the new milk class, such that her app will recognize the milk in the future. The next day, Bob does his grocery shopping in the same supermarket as Alice. Bob notices a new milk package he has never tried

before, which is of the same class as the milk Alice bought the day before. Bob's app can successfully recognize the correct class of the milk package, although this is the first time that Bob tries to recognize this kind of milk package with his app. This is enabled through updating the assistive apps using federated learning, where the central server has transmitted the current global model to Bob's app after being updated with Alice's local model.

There are many interesting challenges that remains to tackle in federated learning. Li et al. [278] mentions challenges with statistical heterogeneity in the data from different users [281, 282], variability of the systems on each device [283], privacy concerns [284], and expensive communication rounds between the server and the devices [285]. From the perspective of this thesis, it would be valuable to study robustness to heterogeneity in the data generated from different users in terms of data quality, sample frequency, and context of where the data has been recorded. Here, an interesting challenge is how to maintain fairness and accuracy across different devices, since the learned models may become biased towards devices with large amounts of data as well as commonly occurring groups of devices. Another relevant path to study is system heterogeneity among the devices to determine how much the federated learning method can tolerate differences in, e.g., hardware and network connectivity. This is an important step towards enabling image recognition methods to be more independent of the mobile phone type, which in return may yield assistive vision apps that work equally well across different geographical locations and income levels of the users.

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 [23,47,286–288]. Essentially, mobile phones have been demonstrated to be a useful tool for object recognition and navigation [14, 22, 47, 51], as many VI people use these in their everyday life. Although computer vision on mobile phones is evidently a good fit, it is important that technical experts remembers to involve VI people throughout the research and development processes to ensure that the technology is in fact useful for blind/low-vision people. An example of close collaboration between VI people and technical researchers is the collection of the recent ORBIT dataset [15, 273]. Here, the authors took a *disability-first approach* to produce an object recognition dataset to benchmark computer vision methods in real-world scenarios for VI people. The data collection was only performed by blind/low-vision participants, who were instructed on how to collect the data as well as informed of various benefits and challenges with current computer vision methods. By engaging continuously with the participants, the authors even re-evaluated the data collection process to ease the effort for the participants while ensuring that the data is useful for training object recognizers. Maintaining the disability-first approach could potentially bring more insights to technical experts in, e.g., computer vision, human-computer interaction, and user experience, on the challenges that occur for

VI people, such that assistive vision technologies can be built that satisfy the needs of the target user.

To further maintain the usefulness of developed assistive vision technologies, the technical experts needs to consider challenges on fairness in computer vision systems and privacy of the users. Several works have shown that current computer vision methods suffer from biases that can disadvantage users based on their gender [34, 289, 290], geographical location [32], and co-occurrences between objects and surrounding contexts [291]. Such challenges on fairness [292–294] and various types of distribution shifts [295] will be crucial to tackle for establishing trust-worthy image recognition systems for the practitioners. Additionally, preserving user privacy to avoid unintended leakages of sensitive information will be key to enhance the utility of developed computer vision systems [178, 296]. The privacy of bystanders also needs to be accounted for as they could be photographed by mobile and wearable devices for assistive vision [24, 297]. However, for a bystander, it is non-trivial how one should approach a VI person and ask them to turn off their device for the sake of avoiding being caught on camera. This matter of social acceptance between assistive vision users and bystanders need to be further studied in realistic settings to understand what requirements that the society wishes to establish on future assistive vision technologies.

Bibliography

[1] 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.

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

[3] 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-surroundings/. Accessed 2022-04-05.

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

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

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

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

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

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] Kyungjun Lee and Hernisa Kacorri. Hands holding clues for object recognition in teachable machines. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[18] Kyungjun Lee, Jonggi Hong, Simone Pimento, Ebrima Jarjue, and Hernisa Kacorri. Revisiting blind photography in the context of teachable object recognizers. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, pages 83–95, 2019.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4845–4854, 2019.

[31] Dengxin Dai and Luc Van Gool. Dark model adaptation: Semantic image segmentation from daytime to nighttime. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3819–3824. IEEE, 2018.

[32] 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.

[33] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[34] 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.

[35] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

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

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

[44] 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.

[45] Synskadades Riksförbund (SRF). Vardagstips för personer med synnedsättning, 2017. URL https://www.srf.nu/globalassets/informationsmaterial/for-medlemmar-och-andra-synskadade/vardagstips_2017_tillganglig.pdf. The Swedish Association of the Visually Impaired (SRF).

[46] Henrik Götesson. Challenges and possibilities when using screen readers, 2019. URL https://www.srf.nu/contentassets/d26ce8ef340141768331b0fbc6e20f59/screenreader_survey_english_summary.pdf. The Swedish Association of the Visually Impaired (SRF).

[47] 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.

[48] 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.

[49] Yuhang Zhao, Shaomei Wu, Lindsay Reynolds, and Shiri Azenkot. A face recognition application for people with visual impairments: Understanding use beyond the lab. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2018.

[50] Xu Liu. A camera phone based currency reader for the visually impaired. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 305–306, 2008.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] Tess Winlock, Eric Christiansen, and Serge Belongie. Toward real-time grocery detection for the visually impaired. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 49–56. IEEE, 2010.

[59] 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.

[60] Roger Boldu, Denys JC Matthies, Haimo Zhang, and Suranga Nanayakkara. Aisee: an assistive wearable device to support visually impaired grocery shoppers. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(4):1–25, 2020.

[61] Peter A Zientara, Sooyeon Lee, Gus H Smith, Rorry Brenner, Laurent Itti, Mary B Rosson, John M Carroll, Kevin M Irick, and Vijaykrishnan Narayanan. Third eye: A shopping assistant for the visually impaired. *Computer*, 50(2):16–24, 2017.

[62] Marian George, Dejan Mircic, Gabor Soros, Christian Floerkemeier, and Friedemann Mattern. Fine-grained product class recognition for assisted shopping. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 154–162, 2015.

[63] 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.

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

[65] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 620–629. IEEE, 2018.

[66] 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.

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

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

[69] 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.

[70] T. Landauer and Robert Bjork. Optimum rehearsal patterns and name learning. *Practical Aspects of Memory*, 1, 11 1977.

[71] 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.

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

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

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

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

[76] 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.

[77] 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.

[78] 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.

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

[80] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[81] 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.

[82] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in Neural Information Processing Systems*, 31, 2018.

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

[84] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[85] Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2):26–31, 2012.

[86] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[87] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[88] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[89] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[90] 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.

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

[92] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[93] 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.

[94] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[95] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[96] 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.

[97] Andrew Ng et al. Sparse autoencoder.

[98] 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.

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

[100] 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.

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

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

[103] 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.

[104] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

[105] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT Press, 2009.

[106] 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.

[107] 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.

[108] 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.

[109] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pages 1462–1471. PMLR, 2015.

[110] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. *arXiv preprint arXiv:1511.02793*, 2015.

[111] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. *Advances in Neural Information Processing Systems*, 29, 2016.

[112] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in Neural Information Processing Systems*, 32, 2019.

[113] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in Neural Information Processing Systems*, 28, 2015.

[114] Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[115] Yingzhen Li and Stephan Mandt. Disentangled sequential autoencoder. *arXiv preprint arXiv:1803.02991*, 2018.

[116] 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.

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

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

[119] 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.

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

[121] 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.

[122] 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.

[123] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 12, 1999.

[124] 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.

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

[126] Verena R Cimarolli, Kathrin Boerner, Mark Brennan-Ing, Joann P Reinhardt, and Amy Horowitz. Challenges faced by older adults with vision loss: a qualitative study with implications for rehabilitation. *Clinical rehabilitation*, 26(8):748–757, 2012.

[127] 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.

[128] 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.

[129] 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.

[130] 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.

[131] 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.

[132] 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.

[133] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014.

[134] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[135] Yu Zhang, Xiu-Shen Wei, Jianxin Wu, Jianfei Cai, Jiangbo Lu, Viet-Anh Nguyen, and Minh N Do. Weakly supervised fine-grained categorization with part-based image representation. *IEEE Transactions on Image Processing*, 25(4):1713–1725, 2016.

[136] 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.

[137] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5209–5217, 2017.

[138] 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.

[139] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2930, 2017.

[140] Abhimanyu Dubey, Otkrist Gupta, Pei Guo, Ramesh Raskar, Ryan Farrell, and Nikhil Naik. Pairwise confusion for fine-grained visual classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 70–86, 2018.

[141] Dongliang Chang, Yifeng Ding, Jiyang Xie, Ayan Kumar Bhunia, Xiaoxu Li, Zhanyu Ma, Ming Wu, Jun Guo, and Yi-Zhe Song. The devil is in the channels: Mutual-channel loss for fine-grained image classification. *IEEE Transactions on Image Processing*, 29:4683–4695, 2020.

[142] Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *European Conference on Computer Vision*, pages 301–320. Springer, 2016.

[143] Zhe Xu, Shaoli Huang, Ya Zhang, and Dacheng Tao. Webly-supervised fine-grained visual categorization via deep domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(5):1100–1113, 2016.

[144] Xiaoxiao Sun, Liyi Chen, and Jufeng Yang. Learning from web data using adversarial discriminative neural networks for fine-grained classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 273–280, 2019.

[145] Xiangteng He and Yuxin Peng. Fine-grained image classification via combining vision and language. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5994–6002, 2017.

[146] Hua Zhang, Xiaochun Cao, and Rui Wang. Audio visual attribute discovery for fine-grained object recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[147] Tianshui Chen, Liang Lin, Riquan Chen, Yang Wu, and Xiaonan Luo. Knowledge-embedded representation learning for fine-grained image recognition. *arXiv preprint arXiv:1807.00505*, 2018.

[148] 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.

[149] Li Niu, Ashok Veeraraghavan, and Ashutosh Sabharwal. Webly supervised learning meets zero-shot learning: A hybrid approach for fine-grained classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7171–7180, 2018.

[150] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2927–2936, 2015.

[151] 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.

[152] 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.

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

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

[155] 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.

[156] 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.

[157] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020.

[158] 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.

[159] 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.

[160] 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.

[161] 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.

[162] 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.

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

[164] 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.

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

[166] 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.

[167] 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.

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

[169] 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.

[170] 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.

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

[172] 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.

[173] 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.

[174] 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.

[175] 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.

[176] 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.

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

[178] 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.

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

[180] 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.

[181] 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.

[182] 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.

[183] 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.

[184] 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.

[185] 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.

[186] 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.

[187] 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.

[188] 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 Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[189] 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.

[190] 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.

[191] 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.

[192] 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.

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

[194] 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.

[195] 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.

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

[197] 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.

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

[199] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International Conference on Machine Learning*, pages 1247–1255, 2013.

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

[201] 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.

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

[203] 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.

[204] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[205] 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.

[206] 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.

[207] 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.

[208] 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.

[209] 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.

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

[211] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

[212] 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.

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

[214] 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.

[215] 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.

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

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

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

[219] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

[220] 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.

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

[222] 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.

[223] 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.

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

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

[226] Ju Xu and Zhanxing Zhu. Reinforced continual learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[227] 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.

[228] 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.

[229] 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.

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

[231] Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient based memory editing for task-free continual learning. *arXiv preprint arXiv:2006.15294*, 2020.

[232] Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9385–9394, 2021.

[233] 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.

[234] 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.

[235] 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.

[236] 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.

[237] 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.

[238] 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.

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

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

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

[242] 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.

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

[244] 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.

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

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

[247] 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.

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

[249] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8218–8227, 2021.

[250] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9516–9525, 2021.

[251] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3589–3599, 2021.

[252] 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.

[253] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International Conference on Computers and Games*, pages 72–83. Springer, 2006.

[254] 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.

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

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

[257] 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.

[258] 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.

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

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

[261] 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.

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

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

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

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

[266] 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.

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

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

[269] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[270] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[271] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[272] Tyler L Hayes and Christopher Kanan. Online continual learning for embedded devices. *arXiv preprint arXiv:2203.10681*, 2022.

[273] 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.

[274] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[275] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[276] 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.

[277] 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.

[278] 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.

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

[280] 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.

[281] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[282] 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.

[283] 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.

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

[285] 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.

[286] 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.

[287] 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.

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

[289] 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.

[290] 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.

[291] 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.

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

[293] 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.

[294] 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.

[295] 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.

[296] 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.

[297] 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.

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 technologies 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

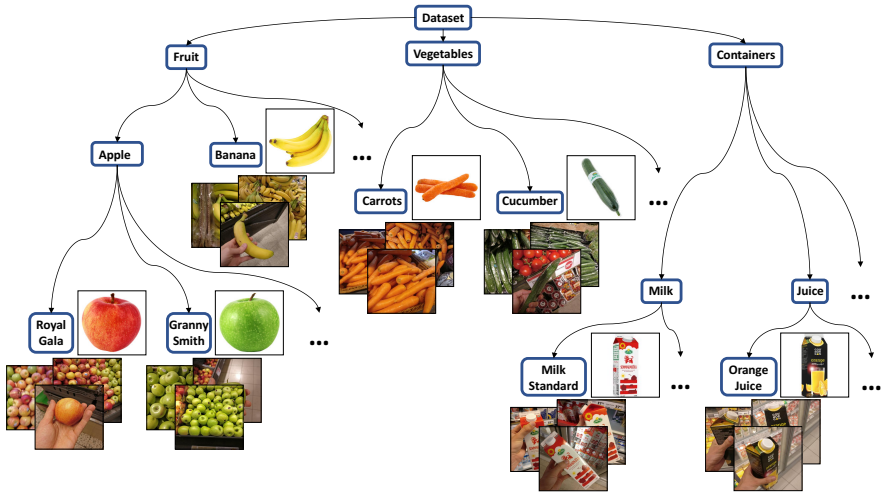


Figure 1: The primary contribution of this paper is a dataset of grocery items, for the purpose of training a visual recognition system to aid visually impaired people. The dataset is organized according to a hierarchical class structure, as illustrated above. A novel aspect of the dataset is that each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

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 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

(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.

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.

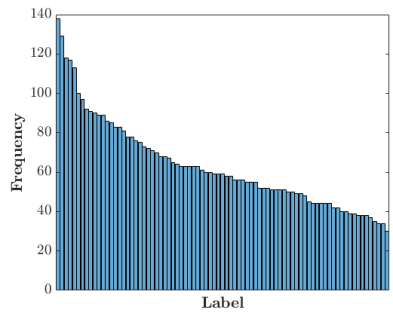


Figure 4: Histogram over the number of images in each class in the dataset.

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.

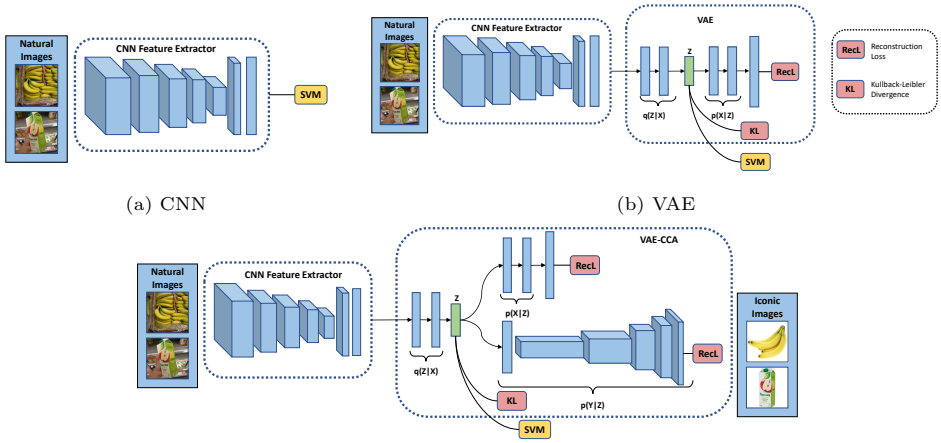
Variational Autoencoders with only natural images. Deep generative models, e.g. the variational autoencoder (VAE) [16, 35, 36], have become widely used in the machine learning community thanks to their generative nature. We thus use VAEs for representation learning as the second benchmarking method. 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 $\boldsymbol{\phi}$ 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.



(a) CNN (b) VAE

(c) VAE-CCA

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.

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

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

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-

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

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.

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.

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

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

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 3 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.

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 **??**). 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

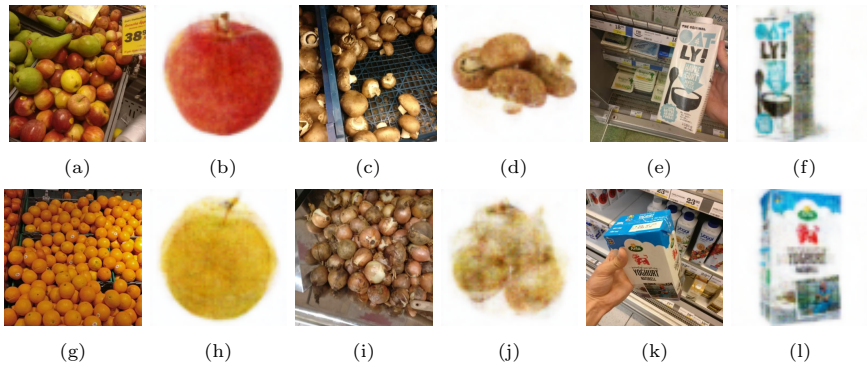


Figure 6: Examples of six natural images in the test set that have been decoded into product iconic images by the iconic image decoder $p_{\boldsymbol{\theta}^{(2)}}(\mathbf{y} \mid \mathbf{z})$ using VAE-CCA model as in Figure 5c. This result is obtained with the fine-tuned DenseNet-169 features, which 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.

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 represen-

tation 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

In recent years, computer vision-based assistive technologies have been developed for supporting people with visual impairments. Such technologies 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], Transsense [4], and the Sound of Vision system [5]. These products can 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. In this paper, we focus on an application that is essential for assistive vision, namely visual support when

Figure 1: Examples of natural images in our dataset, where each image has been taken inside a grocery store. Image examples of fruits, vegetables, and refrigerated products are presented in each row respectively.

Figure 2: Examples of iconic images downloaded from a grocery shopping website, which corresponds to the target items in the images in Figure 1.

shopping for grocery items considering a large range of eatable objects, including fruits, vegetables, and refrigerated products, e.g., milk and juice packages.

Grocery shopping with low vision capabilities can be difficult for various reasons. For example, in grocery store sections for raw groceries, the items are often stacked in large bins as shown in Figure 1(a-f). Additionally, similar items are usually stacked next to each other, and therefore, items are can be misplaced into neighboring bins. Humans can distinguish between groceries without vision to some degree, e.g., by touch and smell, but it requires prior knowledge about texture and fragrance of food items. Furthermore, packaged items, e.g., milk, juice, and yoghurt cartons, can only be differentiated with the help of visual information (see Figure 1(g-i)). Such items usually have barcodes, that are readable using the existing assistive vision devices described above. Even if using a barcode detector is a clever solution, it can be inconvenient and exhausting always having to detect barcodes of packaged items. Therefore, an assistive vision device that relies on natural images would be of significant value for a visually impaired person in a grocery store.

For an image classification model to be capable of recognizing grocery items in the setting of grocery shopping, we need image data from grocery store environments. In our previous work [6], we addressed this by collecting a novel dataset containing natural images of various raw grocery items and refrigerated products, where all images have been taken with a smartphone camera inside grocery stores to simulate a realistic shopping scenario. In addition to the natural images, we also looked upon alternative types of grocery item data that could facilitate the classification task. Grocery store chains commonly maintain websites where they store information about

the grocery items they sell and are currently available in the store for online grocery shopping. For every item, there is usually an iconic image of the item with white background, a text description about the item, and also information about nutrition values, origin country, etc. We downloaded such information from an online grocery shopping website to complement the natural images in the dataset.

To utilize the available information from the dataset for grocery item classification, we use a multi-view generative model Variational Canonical Correlation Analysis (VCCA) [7] that learns a shared representation of the natural images and the downloaded information. A view can be defined as any signal or data measured by some appropriate sensor and combining information from multiple views has previously been shown to be helpful for various image classification tasks [8–15]. However, naively adding more additional information may not lead to improved results and even harm the performance of the model [16,17]. We, therefore, perform an ablation study over the available views in the dataset with VCCA to gain insights into how each view can affect the classification performance. Moreover, VCCA allows separating the latent space into shared and private components, where the private latent spaces should hold information about a single view. This might prove useful for reducing view-specific noise in the shared latent space, which can ease the learning process of the representations we want to use for classification. We investigate the effects each view has on the learned representations of VCCA by measuring classification performances as well as visualizing the latent space for the different model settings. The contributions of this paper are the following:

- We present a novel dataset with natural images of grocery items as well as iconic images and text descriptions for every item class (see Section 2.1) [6]. The natural images are taken in grocery stores in different lighting conditions and backgrounds that can be challenging settings for a computer vision-based assistive device. The additional iconic images and text descriptions make the dataset suitable for multi-view learning by combining the natural images with the extra information to obtain better classification performance.

- We investigate how to use information from different views for the task of grocery item classification with the deep generative model VCCA (see Section 4.2). This model combines information from multiple views into a low-dimensional latent representation that can be used for classification. We also select a variant of VCCA denoted VCCA-private that separates shared and private information about each view through factorization of the latent representation (see subsection Extracting Private Information of Views with VCCA-private in Experimental Procedures). Furthermore, we use a standard multi-view autoencoder model called Split Autoencoder (SplitAE) [17,18] for benchmarking against VCCA and VCCA-private on classification.

- We conduct experiments with SplitAE, VCCA, and VCCA-private on the task of grocery item classification with our dataset (Results). We perform a thorough ablation study over all views in the dataset to demonstrate how each view contributes to enhancing the classification performance and conclude that utilizing the web-scraped views yields better classification results than only using the natural images (see subsection Classification Results in Results). To gain further insights

into the results, we visualize the learned latent representations of the VCCA models and discuss how the iconic images and textual descriptions impose different structures on the latent space that are beneficial for classification (see subsection Investigation of the Learned Representations in Results).

This work is an extended version of Klasson et al. [6], where we first presented this dataset. In this paper, we have added a validation set of natural images from two stores that were not present in the training and test set splits from [6] to avoid overfitting effects. We also demonstrate how the text descriptions can be utilized alone and along with the iconic images in a multi-view setting, while [6] only experimented with the combination of natural and iconic images to build better representations of grocery items. Finally, we decode iconic images from unseen natural images as an alternative to evaluate the usefulness of the latent representations (see subsection Decoding Iconic Images from Unseen Natural Images in Results). As we only evaluated the decoded iconic images qualitatively in Klasson et al. [6], we have extended the assessment by comparing the quality of the decoded images from different VCCA models with multiple image similarity metrics.

Next, we discuss image datasets, food datasets, and multi-view models related to our work:

Image Datasets Many image datasets used in computer vision have been collected by downloading images from the web [19–30]. Some datasets [19, 22, 24, 26, 28] use search words with the object category in isolation, which typically returns high-quality images where the searched object is large and centered. To collect images from more real-world scenarios, searching for combinations of object categories usually returns images of two searched categories but also numerous other categories [25, 30]. The simplest annotation of these images is to provide a class label for the present objects. Occasionally, the dataset can use a hierarchical labeling structure and provide a fine- and coarse-grained label to objects where it is applicable. The annotators can also be asked to increase the possible level of supervision for the objects by, for instance, providing bounding boxes, segmentation masks, keypoints, text captions that describe the scene, and reference images of the objects [21, 23, 25, 26, 28, 30]. Our dataset includes reference (iconic) images of the objects that were web-scraped from a grocery store website. We also downloaded text descriptions that describe general attributes of the grocery items, such as flavor and texture, rather than the whole visual scene. The grocery items have also been labeled hierarchically in a fine- and coarse-grained manner if there exist multiple variants of specific items. For example, fine-grained classes of apples such as *Golden Delicious* or *Royal Gala* belongs to the coarse-grained class *apple*.

Food Datasets Recognizing grocery items in their natural environments, such as grocery stores, shelves, and kitchens, have been addressed in plenty of previous works [6, 31–41]. The addressed tasks range from hierarchical classification, object detection, segmentation, and 3D model generation. Most of these works collect a dataset that resembles shopping or cooking scenarios, where the datasets vary in the degree of labeling, different camera views, and the data domain difference between the training and test set. The training sets in GroZi-120 [36], Grocery Products [32], and CAPG-GP [31] datasets were obtained by web-scraping product images of single instances on grocery web stores, while the test sets were collected in grocery stores

where there can be single and multiple instances of the same item and other different items. The RPC [39] and TGFS [41] datasets are used for object detection and classification of grocery products, where RPC is targeted for automatic checkout systems and TGFS is for the task of recognizing items purchased from self-service vending machines. The BigBIRD [37] dataset and datasets from [**?**, 33] contain images of grocery items from multiple camera views, segmentation masks, and depth maps for 3D reconstruction of various items. The Freiburg Groceries [34] dataset contains images taken with smartphone cameras of items inside grocery stores, while its test set consists of smartphone photos in home environments with single or multiple instances from different kinds of items. The dataset presented in [38] also contains images taken with smartphone cameras inside grocery stores to develop a mobile application for recognizing raw food items and provide details about the item, such as nutrition values and recommendations of similar items. Other works that collected datasets of raw food items, such as fruits and vegetables, focused on the standard image classification task [42, 43] and on detecting fruits in orchards for robotic harvesting [44, 45]. Our dataset – the Grocery Store dataset – shares many similarities with the mentioned works above, for instance, that all images of groceries are taken in their natural environment, the hierarchical labeling of the classes, and the iconic product images for each item in the dataset. Additionally, we have provided a text description for each item that was web-scraped along with the iconic image. As most grocery item datasets only include packaged products, we have also collected images of different fruit and vegetable classes along with packages in our dataset.

Other examples of food datasets are those with images of food dishes, where [46] provides a summary of existing benchmark food datasets. The contents of these datasets range from images of food dishes [47–50], cooking videos [51], recipes [52–54], and also restaurant-oriented information [55, 56]. One major difference between recognizing groceries and food dishes is the appearance of the object categories in the images. For instance, a fruit or vegetable is usually intact and present in the image, while ingredients that are significant for recognizing a food dish may not be visible at all depending on the recipe. However, recognizing raw food items and dishes share similarities since they can appear with many different visual features in the images compared to packaged groceries, e.g., carton boxes, cans, and bottles, where the object classes have identical shape and texture. Another key difference is the natural environments and scenes where the images of grocery items and food dishes have been taken. Food dish images usually show the food on a plate placed on a table and, occasionally, with cutlery and a glass next to the plate. Images taken in grocery stores can cover many instances of the same item stacked close to each other in shelves, baskets, and refrigerators, while there can be multiple different kinds of items next to each other in a kitchen environment. To summarize, recognizing grocery items and food dishes are both challenging tasks because examples from the same category can look very different and also appear in various realistic settings in images.

Multi-View Learning Models There exist lots of multi-view learning approaches for data fusion of multiple features [7, 9, 10, 17, 57–70]. A common approach is to obtain a shared latent space for all views with the assumption that each view has been generated from this shared space [68]. A popular example of this is approach

is Canonical Correlation Analysis (CCA) [71], which aims to project two sets of variables (views) into a lower-dimensional space so that the correlation between the projections is maximized. Similar methods propose maximizing other alignment objectives for embedding the views, such as ranking losses [9, 10, 59, 67]. There exist nonlinear extensions of CCA, e.g., Kernel CCA [72] and Deep CCA [73], that optimize their nonlinear feature mappings based on the CCA objective. Deep Canonically Correlated Autoencoders (DCCAE) [18] is a Deep CCA model with an autoencoding part, which aims to maximize the canonical correlation between the extracted features as well as reconstructing the input data. Removing the CCA objective reduces DCCAE to a standard multi-view autoencoder, e.g., Bimodal Autoencoders and Split Autoencoders (SplitAEs) [17, 18], that only aim to learn a representation that best reconstructs the input data. SplitAEs aim to reconstruct two views from a representation encoded from one of the views. This approach was empirically shown to work better than Bimodal Autoencoders in [17] in situations where only a single view is present at both training and test time.

Variational CCA (VCCA) [7] can be seen as an extension of CCA to deep generative models, but can also be described as a probabilistic version of SplitAEs. VCCA uses the amortized inference procedure from Variational Autoencoders (VAEs) [74,75] to learn the shared latent space by maximizing a lower bound on the data log-likelihood of the views. Succeeding works have proposed new learning objectives and inference methods for enabling conditional generation of views, e.g., generating an image conditioned on some text and vice versa [58, 62, 63, 65, 66]. These approaches rely on fusing the views into the shared latent space as the inference procedure, which often requires tailored training and testing paradigms when views are missing. However, adding information from multiple views may not lead to improved results and can even make the model perform worse on the targeted task [16, 17]. This is especially the case when views have noisy observations, which complicates learning a shared latent space that combines the commonalities between the views. To avoid disturbing the shared latent space with noise from single views, some works design models which extend the shared latent space with private latent spaces for each view that should contain the view-specific variations to make learning the shared variations easier [7, 61, 64, 69, 76]. VCCA can be extended to extract shared and private information between different views through factorization of the latent space into shared and private parts. In this paper, we investigate if the classification performance of grocery items in natural images can be improved by extracting the view-specific variations in the additional views (iconic images and product descriptions) from the shared latent space with this variant of VCCA called VCCA-private. We will experiment with treating each data point as pairs of natural images and either iconic images or text descriptions as well as triplets of natural images, iconic images, and text descriptions. A difference between how we apply VCCA to our dataset compared to the works above is that the iconic image and text description are the same for every natural image of a specific class.

2 Results

In this section, we begin by providing the details about the collected dataset, which we have named the Grocery Store dataset. Furthermore, we illustrate the utility of

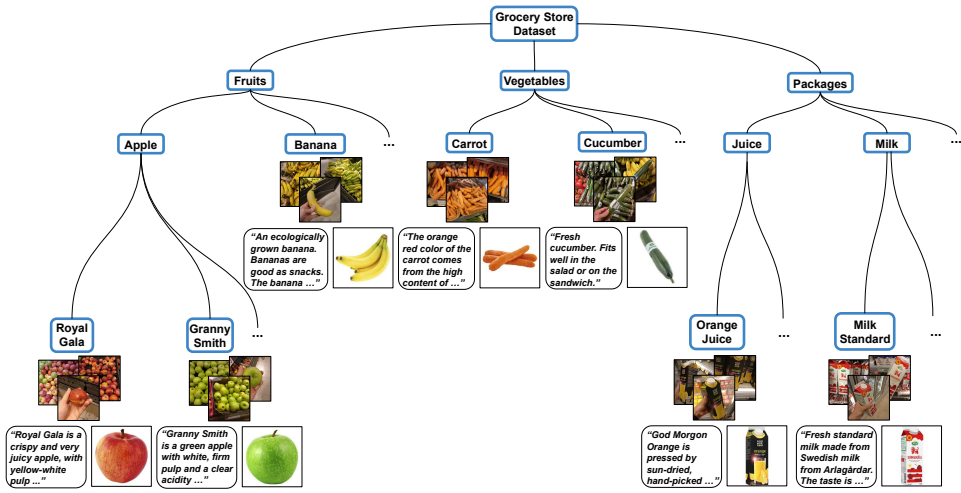


Figure 3: Illustration of the hierarchical class structure of the dataset. First, the classes are divided by their grocery item type, i.e., *Fruits*, *Vegetables*, and *Packages*, followed by a separating the items into coarse-grained classes, e.g., *Apple*, *Carrot*, and *Milk*, and then into fine-grained classes. There are 81 fine-grained classes in total and 46 coarse-grained classes. The figure also shows the iconic image and text description of the items next to the class label.

the additional information in the Grocery Store dataset to classify grocery items in the experiments. We compare SplitAE, VCCA, and VCCA-private with different combinations of views against two standard image classifiers. Additionally, we experiment with a vanilla autoencoder (denoted as AE) and a VAE that post-processes the natural image features to train a linear classifier cheaper and compare the performance against the multi-view models. We measure the classification performance on the test set for every model and also compare the classification accuracies when the number of words in the text description varies for the models concerned (see subsection Classification Results in Results). To gain insights on how the additional views affect the learned representations, we visualize the latent spaces of VCCA and VCCA-private with PCA and discuss how different views changes the structure of the latent space (see subsection Investigation of the Learned Representations in Results). Finally, we show how iconic images can be used for enhancing the interpretability of the classification (see subsection Decoding Iconic Images from Unseen Natural Images in Results), which was also illustrated by Klasson et al. [6].

2.1 The Grocery Store Dataset

In Klasson et al. [6], we collected images from fruit, vegetable, and refrigerated sections with dairy and juice products in 20 different grocery stores. The dataset consists of 5421 images from 81 different classes. For each class, we have downloaded an iconic image of the item, a text description, and information including origin country, appreciated weight and nutrient values of the item from a grocery store website. Some examples of natural images and downloaded iconic images can be seen in Figure 1 and 2 respectively. Furthermore, Table S1 displays a selection of

text descriptions with their corresponding iconic image. The text description length varies between 6 and 91 words with an average of 36 words. We also structure that classes hierarchically with three levels as illustrated in Figure 3. The first level divides the items into three categories; *Fruits*, *Vegetables*, and *Packages*. The next level consists of 46 coarse-grained classes which divides the items into groups contain groups of items types, e.g., *Apple*, *Carrot*, *Milk*. The bottom level consists of the 81 fine-grained classes, where the items are completely separated. Note that a coarse-grained class without any fine-grained classes in the dataset, e.g., *Banana* and *Carrot*, is considered as a fine-grained class in the classification experiments (see subsection Classification Results in Results), where we report both fine-grained and coarse-grained classification performance of the models.

We aimed to collect the natural images under similar conditions as if they would be taken with an assistive mobile application. 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 incorrect shelves along with the targeted item. 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 split the natural images 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 training set, validation set, or test set. In the first version of the dataset [6], we balanced the class sizes to a large extent as possible in both the training and test set, which resulted in a training and test set containing 2640 and 2485 images respectively. In this paper, we have extended the dataset with a validation set containing 296 images taken with the same smartphone camera as before. The validation set was collected from two different grocery stores than the ones in the first version to avoid the biasing correlations described above. Initially, we experimented with grabbing a validation set from the current training set and noticed that the trained classifiers performed exceptionally well on the validation set. However, the classifiers generalized poorly to images from the test set that were taken in other stores and therefore decided to collect the validation set in two unseen stores to avoid the biases from the training set. We show histograms representing the class distributions for the training, validation, and test splits in Figure S1.

The scenario that we wanted to depict with the dataset was using a mobile device to classify natural images visually impaired people with grocery shopping. The additional information such as the iconic images, text descriptions, and hierarchical structure of the class labels can be used to enhance the performance of the computer vision system. Since every class label is associated with a text description, the 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.

2.2 Models

In this section, we briefly describe the models that we apply to grocery classification. The notation of the views that are available in the Grocery Store dataset are the following:

- x: Natural image encoded into image feature space with an off-the-shelf convolutional neural network.
- i: Iconic image of the object class in the natural image.
- w: Text description of the object class in the natural image.
- y: Class label of the natural image.

We mainly focus on analyzing VCCA [7] for utilizing different combinations of the views and investigate how each view contributes to the classification performance of grocery items. Our primary baseline model is the SplitAE which extracts shared representations by reconstructing all views, while VCCA aims to maximize a lower bound on the data log-likelihood for all views. We also study a variant of VCCA called VCCA-private [7], which is used for extracting private information about each view in addition to shared information across all views by factorizing the latent space.

We compare the multi-view models against single-view methods only using the natural images for classification. As our first single-view baseline, we customize the output layer of DenseNet169 [77] to the Grocery Store dataset and train it from scratch to classify the natural images, which we refer to as DenseNet-scratch in the experiments. The second baseline called Softmax is a Softmax classifier trained on the off-the-shelf features from DenseNet169 pre-trained on the ImageNet dataset [19], where we extract 1664-dimensional from the average pooling layer before the classification layer in the architecture. We also experiment with AEs and VAEs for post-processing the off-the-shelf features and use a linear classifier to evaluate the models on classification. See Section 4.2 for a thorough description of the single- and multi-view autoencoders used in this paper. We name the single- and multi-view autoencoders using subscripts to denote the views utilized for learning the shared latent representations. For example, VCCA$_{xi}$ utilizes natural image features x and iconic images i, while VCCA$_{xiwy}$ uses natural image features x and iconic images i, text descriptions w, and class labels y.

2.3 Classification Results

We evaluated the classification accuracy on the test set for each model. We also calculate the accuracy for the coarse-grained classes with the following method: Let the input $x^{(i)}$ have a fine-grained class $y_{fg}^{(i)}$ and a coarse-grained class $y_{cg}^{(i)}$. Each fine-grained class can be mapped to its corresponding coarse-grained class using $Pa(y_{fg}^{(i)}) = y_{cg}^{(i)}$, where $Pa(\cdot)$ stands for "parent". Then we compute the coarse-grained accuracy using

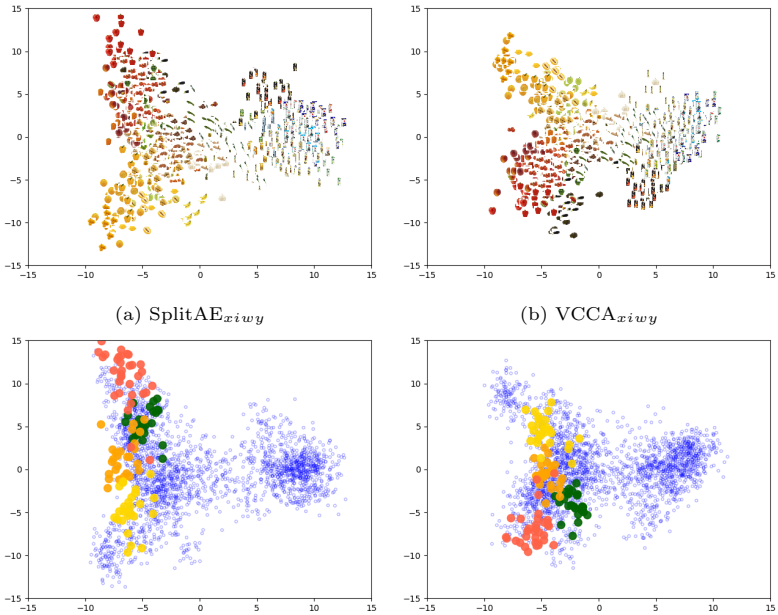
$$\begin{aligned} \text{coarse accuracy} &= \frac{1}{N}\sum_{i=1}^{N}\left[Pa(\hat{y}_{fg}^{(i)}) = y_{cg}^{(i)}\right], \\ \hat{y}_{fg}^{(i)} &= \arg\max_{y} p(y|x^{(i)}), \end{aligned} \tag{1}$$

Table 1: Classification accuracies on the test set for all models in percentage (%) along for each model. The subscript letters in the model names indicate the data views used in the model. The column Accuracy corresponds to the fine-grained classification accuracy. The column Coarse Accuracy corresponds to the classifying a class within the correct parent class. Results are averaged using 10 different random seeds and we report both means and standard deviations. Abbreviations: AE, Autoencoder; VAE, Variational Autoencoder; SplitAE, Split Autoencoder; VCCA, Variational Canonical Correlation Analysis.

Model	Accuracy (%)	Coarse Accuracy (%)
DenseNet-scratch	67.33 ± 1.35	75.67 ± 1.15
Softmax	71.67 ± 0.28	83.34 ± 0.32
AE_x+Softmax	70.69 ± 0.82	82.42 ± 0.58
VAE_x+Softmax	69.20 ± 0.46	81.24 ± 0.63
SplitAE_{xy}	70.34 ± 0.56	82.11 ± 0.38
VCCA_{xy}	70.72 ± 0.56	82.12 ± 0.61
SplitAE_{xi}+Softmax	77.68 ± 0.69	87.09 ± 0.53
VCCA_{xi}+Softmax	77.02 ± 0.51	86.46 ± 0.42
VCCA-private_{xi}+Softmax	73.04 ± 0.56	84.16 ± 0.51
SplitAE_{xiy}	77.43 ± 0.80	87.14 ± 0.57
VCCA_{xiy}	77.22 ± 0.55	86.54 ± 0.51
$\text{VCCA-private}_{xiy}$	74.04 ± 0.83	84.59 ± 0.83
SplitAE_{xw}+Softmax	76.27 ± 0.66	86.45 ± 0.56
VCCA_{xw}+Softmax	75.37 ± 0.46	86.00 ± 0.32
VCCA-private_{xw}+Softmax	75.11 ± 0.81	85.91 ± 0.55
SplitAE_{xwy}	75.78 ± 0.84	86.13 ± 0.63
VCCA_{xwy}	74.72 ± 0.85	85.59 ± 0.78
$\text{VCCA-private}_{xwy}$	74.92 ± 0.74	85.59 ± 0.67
SplitAE_{xiw}+Softmax	77.79 ± 0.48	87.12 ± 0.62
VCCA_{xiw}+Softmax	77.51 ± 0.51	86.69 ± 0.41
SplitAE_{xiwy}	78.18 ± 0.53	87.26 ± 0.46
VCCA_{xiwy}	77.78 ± 0.45	86.88 ± 0.47

where $\left[Pa(\hat{y}_{fg}^{(i)}) = y_{cg}^{(i)}\right] = 1$ when the condition is true and $\hat{y}_{fg}^{(i)}$ is the predicted fine-grained class from the selected classifier. The classification results for all models are shown in Table 1. We group the results in the table according to the utilized views and classifier. We see that Softmax trained on off-the-shelf features outperforms DenseNet-scratch by 4%. This result is common when applying deep learning to small image datasets, where pre-trained deep networks are transferred to a new dataset usually performs well compared to training neural networks on the dataset from scratch. Therefore, we present results using the off-the-shelf features for all other models.

The SplitAE and VCCA models surpass the Softmax baseline in classification performance when incorporating either the iconic image or text description view. We believe that the models using the iconic images achieve better classification accuracy over models using the text description because the iconic images contain visual features, e.g., color and shape of items, that are more useful for the image classification task. The text descriptions include more often information about the flavor, origin, and cooking details rather than describing visual features of the item, which can be



(a) SplitAE$_{xiwy}$ (b) VCCA$_{xiwy}$

(c) SplitAE$_{xiwy}$ (d) VCCA$_{xiwy}$

Figure 4: Visualizations of the latent representations of the test set from SplitAE$_{xiwy}$ and VCCA$_{xiwy}$. We plot the corresponding iconic image to each latent representation in Figure 4a and 4b. In Figure 4c and 4d, we plot the bell pepper representations according to the color of the class, while the blue points correspond to the other grocery items. Abbreviations: SplitAE, Split Autoencoder; VCCA, Variational Canonical Correlation Analysis.

less informative when classifying items from images. In most cases, the corresponding SplitAE and VCCA models perform on par for classification performance. However, VCCA-private with iconic images results in a significant drop in accuracy compared to its counterpart. We observed that the private latent variable simply models noise since there is only a single iconic image (and text description) for each class. We provide a further explanation of this phenomenon in Section 2.4.

We observe that VCCA models compete with their corresponding SplitAE models in the classification task. The main difference between these models is the Kullback-Leibler (KL) divergence [78] term in the ELBO that encourages a smooth latent space for VCCA (see Section 4). In contrast, SplitAE learns a latent space that best reconstructs the input data, which can result in parts of the space that does not represent the observed data. We showcase these differences by plotting the latent representations of SplitAE$_{xiwy}$ and VCCA$_{xiwy}$ using PCA in Figure 4. In Figure 4a and 4b, we have plotted the corresponding iconic image for the latent representations. We observe that VCCA$_{xiwy}$ tries to establish a smooth latent space by pushing visually similar items closer to each other, but at the same time prevent spreading out the representations too far from the origin. Figure 4c and 4d shows the positions of the bell peppers items in the latent spaces, where the color of the point corresponds to the specific bell pepper class. In Figure 4c, we observe that SplitAE$_{xiwy}$ has spread out the bell peppers across the space, while VCCA$_{xiwy}$ establishes shorter distances

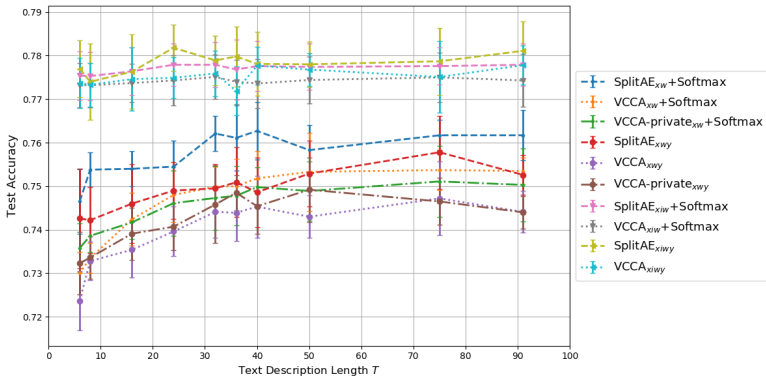


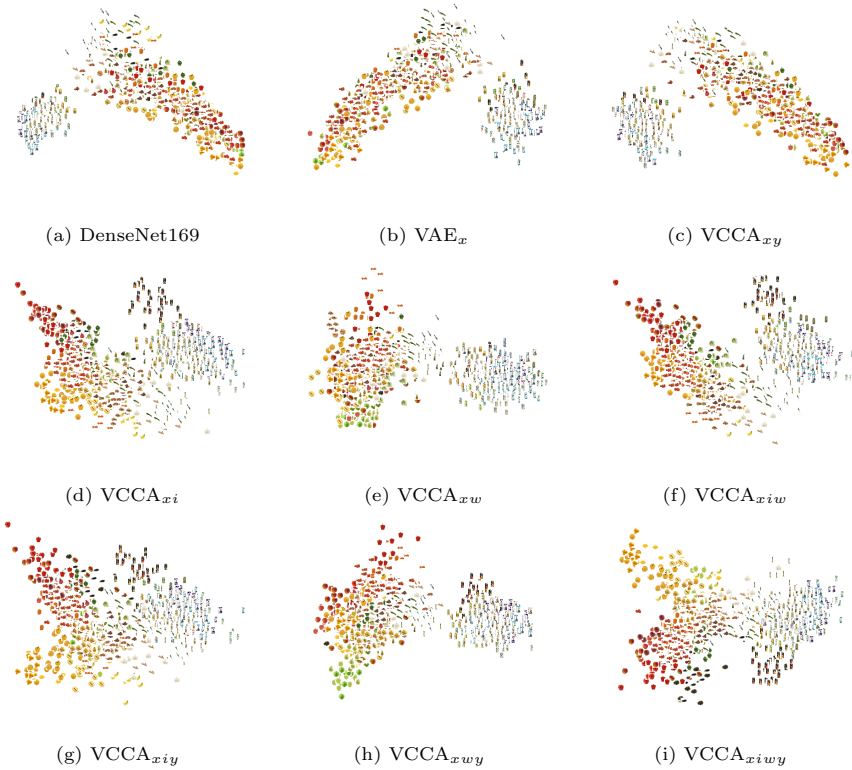
Figure 5: Test accuracy over the text description length T for all SplitAE, VCCA, and VCCA-private models using the text description. We show the accuracy for the models trained with $T = 6, 8, 16, 24, 32, 36, 40, 50, 75$, and 91 words. The results have been averaged for 10 different seeds and the error bars show the standard deviations for every setting of T. Abbreviations: SplitAE, Split Autoencoder; VCCA, Variational Canonical Correlation Analysis.

between them in Figure 4d due to the regularization.

We evaluated the classification performance achieved by each SplitAE, VCCA, and VCCA-private model using the text descriptions with different description lengths T. Figure 5 shows the fine-grained classification accuracies for the concerned models. For models using only the text descriptions, the classification accuracies increase as T increases in most cases. Setting $T \geq 32$ results in good classification performance, potentially since the models have learned to separate the grocery items based on that the text descriptions have become more dissimilar and unique. The classification accuracies are mostly stable as T varies for the models with the additional iconic images. Since including iconic images significantly increases the classification performance over models only using text descriptions, we conclude that the iconic images are more helpful when we want to classify the grocery items from natural images.

2.4 Investigation of the Learned Representations

To gain insights about the effects that each view has on the classification performance, we visualize the latent space by plotting the latent representations using PCA. Utilizing the additional views showed to have similar effects on the structure of the latent spaces from SplitAE and VCCA. Since our main interest lies in representation learning with variational methods, we focus on studying the latent representations of VCCA and VCCA-private. Firstly, we use PCA to visualize the latent representations in 2D and plot the corresponding iconic images of the representations (see Figure 6). Secondly, to illustrate the effects that the iconic images and text descriptions have on the learned latent space, we focus on two cases of grocery items where one of the views helps to separate two different types of classes and the other one does not (see Figure 7 and 8). Finally, we look into the shared and private latent spaces learned by VCCA-private$_{xw}$ and observe that variations in image backgrounds and structures of text sentences have been separated from the shared representations into the private ones.



(a) DenseNet169 (b) VAE$_x$ (c) VCCA$_{xy}$

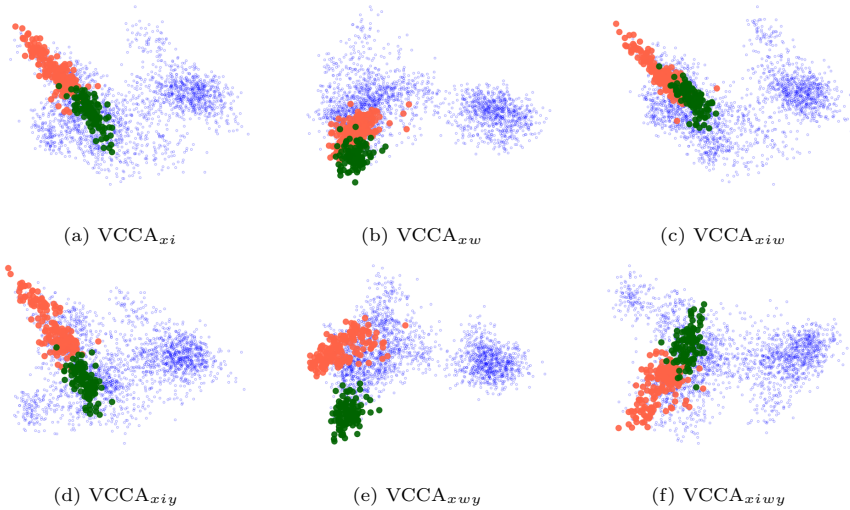
(d) VCCA$_{xi}$ (e) VCCA$_{xw}$ (f) VCCA$_{xiw}$

(g) VCCA$_{xiy}$ (h) VCCA$_{xwy}$ (i) VCCA$_{xiwy}$

Figure 6: Visualizations of the latent representations from the test set, where we plot the iconic image of the corresponding object classes. We also plot the PCA projection of the natural image features from the off-the-shelf DenseNet169 in Figure 6a. All models have been initialized with the same random seed before training.

In Figure 6, we show the latent representations for the VCCA models that were used in Table 1 (see subsection Classification Results in Results). We also plot the PCA projections of the natural image features from the off-the-shelf DenseNet169 in Figure 6a as a baseline. Figure 6b and 6c shows the latent space learned by n VAE$_x$ and VCCA$_{xy}$, which are similar to the DenseNet169 feature space since these models are performing compression of the natural image features into the learned latent space. We observe that these models have divided packages and raw food items into two separate clusters. However, the fruits and vegetables are scattered across their cluster and the packages have been grouped close to each other despite having different colors, e.g., black and white, on the cartons.

The structure of the latent spaces becomes distinctly different for the VCCA models that use either iconic images or text descriptions as an additional view and we can observe the different structures that the views bring to the learned latent space. In Figure 6d and 6g, we see that visually similar objects, in terms of color and shape, have moved closer together by utilizing iconic images in VCCA$_{xi}$ and VCCA$_{xiy}$. When using text descriptions in VCCA$_{xw}$ and VCCA$_{xwy}$, we also observe in the fruit and vegetable cluster that the items are more grouped based on their color in Figure 6e and 6h. Figure 6f and 6i shows the latent spaces in VCCA$_{xiw}$



(a) VCCA$_{xi}$ (b) VCCA$_{xw}$ (c) VCCA$_{xiw}$

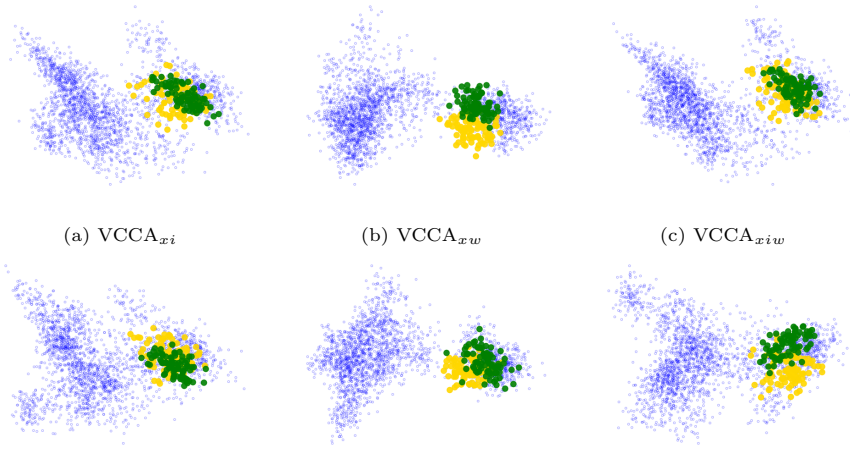
(d) VCCA$_{xiy}$ (e) VCCA$_{xwy}$ (f) VCCA$_{xiwy}$

Figure 7: Visualizations of the latent representations μ_z of the red and green apples in the Grocery Store dataset. The red points correspond to the red apple classes, while the green points correspond to the green apple. The blue points correspond to the other grocery items.

and VCCA$_{xiwy}$ respectively. These latent spaces are similar to the ones learned by VCCA$_{xi}$ and VCCA$_{xiy}$ in the sense that these latent spaces also group items based on their color and shape. We believe that this structure imposed by the iconic images could be softened by reducing the scaling weight λ_i, which potentially could reduce the classification accuracy as a consequence. The difference between the latent spaces is not evident comparing the models using the class label.

Red and Green Apples To showcase how the iconic images help to learn good representations, we consider all of the apple classes in the dataset, namely the red apples *Pink Lady*, *Red Delicious* and *Royal Gala*, and also the green apples *Golden Delicious* and *Granny Smith*. In Figure 7, we group the red apple classes and visualize their latent representations by red points. The green apples are grouped similarly and we visualize their latent representations with green points. Latent representations of all other grocery items are visualized as blue points. The models using iconic images as one view in Figure 7a, 7d, 7c, and 7f have managed to separate the red and green apples based on their color differences. The models using text description have instead moved the apples closer together in one part of the latent space, possibly because of their similarities mentioned in the description.

Juice and Yoghurt Packages To illustrate how the text descriptions can establish more useful latent representations, we consider a selection of juice and yoghurt classes. These packaged items have similar shapes and colors, which makes it difficult for a classifier to distinguish their content differences using only visual input. In Figure 8, we visualize the latent representations of the juice and yoghurt packages using yellow and green points respectively. We observe that only VCCA$_{xw}$ and VCCA$_{xwy}$ manages to separate the packages in Figure 8b and 8e due to their different text descriptions. Since the iconic images of the packages are visually similar, adding this information is insufficient for separating these packages in the latent space. This



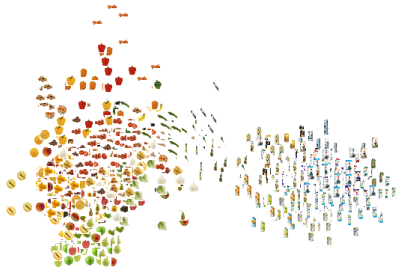
(a) VCCA$_{xi}$ (b) VCCA$_{xw}$ (c) VCCA$_{xiw}$

(d) VCCA$_{xiy}$ (e) VCCA$_{xwy}$ (f) VCCA$_{xiwy}$

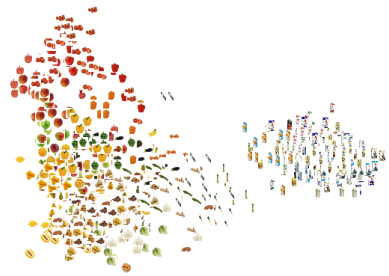
Figure 8: Visualizations of the latent representations μ_z of a selection of juice packages and yoghurt packages in the Grocery Store dataset. The yellow and green points correspond to the juice and yoghurt packages respectively. The blue points correspond to the other grocery items.

indicates that we gain different benefits from the iconic images and text descriptions when it comes to classifying grocery items.

Latent Spaces of VCCA-private We show the shared and private latent spaces of VCCA-private$_{xw}$ in Figure 9b-d, as well as the single latent space of VCCA$_{xw}$ for comparison in Figure 9a. Comparing with the latent space of VCCA$_{xw}$, the shared latent space of VCCA-private$_{xw}$ in Figure 9b, has structured the raw food items based on their class, color, and shape better than standard VCCA$_{xw}$. In Figure 9c, we plot the natural images corresponding to the latent representation for the private latent variable u_x. We zoom in on some natural images and found that the images are structured based on their similarities in background and camera view. On the left and bottom sides of the cluster, we found images of grocery items closely packed together in bins. Single items that are held in the hand of the photographer are placed on the right side, whereas images of items and the store floor are placed on the top and middle of the latent space. The model has therefore managed to separate the variations within the natural image view into the private latent variable u_x from the shared latent variable z, which probably is the main reason why similar raw food items are closer to each other in Figure 9b than in Figure 9a. We also plot the corresponding iconic image on the position of the text description representation for the private latent variable u_w in Figure 9d. Note that every text description is projected at the same location in the latent space since the text descriptions are the same for every class item. We highlighted some specific words in the descriptions and observed that descriptions with the same words are usually close to each other. Visually dissimilar items can be grouped close to each other in this latent space, which indicates that the private latent variable u_w contains information about the structure of the text sentences, i.e., word occurrences and how they are ordered in the text description. In Figure S4, we show the shared and private latent spaces of VCCA-private$_{xi}$ and provide a conclusion to the results in Section II.3.

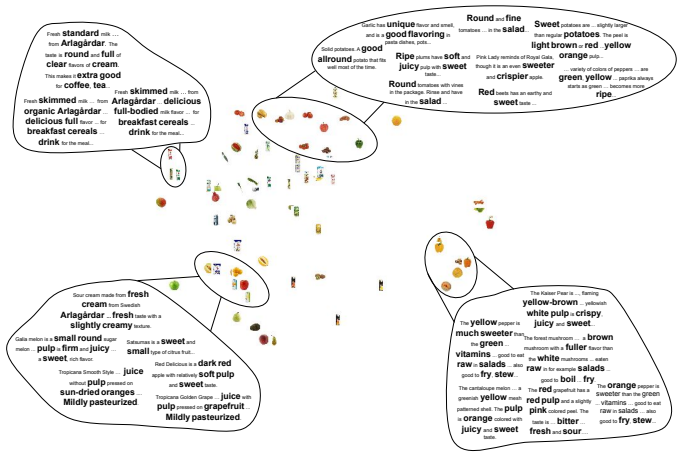


(a) μ_z from VCCA$_{xw}$



(b) μ_z from VCCA-private$_{xw}$

(c) μ_x from VCCA-private$_{xw}$



(d) μ_w from VCCA-private$_{xw}$

Figure 9: Visualizations of the latent representations μ_z of a selection of juice packages and yoghurt packages in the Grocery Store dataset. The yellow and green points correspond to the juice and yoghurt packages respectively. The blue points correspond to the other grocery items.

Table 2: Results on image quality of decoded iconic images for the Variational Canonical Correlation Analysis (VCCA) models using the iconic images. The subscript letters in the model names indicate the data views used in the model. ↑ denotes higher is better, ↓ lower is better. Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and Kullback-Leibler (KL) divergence are measured by comparing the true iconic image against the decoded one. Accuracy shows the classification performance for each model and has been taken from Table 1. We report means and standard deviations averaged over 10 random seeds for all metrics.

Model	PSNR ↑	SSIM ↑	KL ↓	Accuracy (%) ↑
VCCA$_{xi}$	20.13 ± 0.05	0.72 ± 0.00	4.43 ± 0.21	77.02 ± 0.51
VCCA$_{xiy}$	20.12 ± 0.09	0.73 ± 0.00	4.35 ± 0.22	77.22 ± 0.55
VCCA$_{xiw}$	20.11 ± 0.09	0.73 ± 0.00	4.29 ± 0.24	77.51 ± 0.51
VCCA$_{xiwy}$	20.16 ± 0.08	0.73 ± 0.00	4.32 ± 0.22	77.78 ± 0.45

2.5 Decoding Iconic Images from Unseen Natural Images

In this section, we show how the iconic image decoder can decode plausible iconic images of grocery items from unseen natural images in the test set. We apply the same approach as in [6], where we encode unseen natural images and then decode the retrieved latent representation back into an iconic image. We also report several image similarity metrics to investigate if the decoded image quality is correlated with classification performance. We report peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [79], and the KL divergence by comparing the decoded iconic images to the true ones. For computing the KL divergence, we model the decoded and true iconic image using two Gaussian Mixture Model (GMM) [80, 81]. The images are then represented as density functions, such that we can measure the similarity between the two densities with the KL divergence and use it as an image similarity metric. Since the KL divergence between two GMMs is not analytically tractable, we apply Monte Carlo simulation using n i.i.d. samples drawn from the decoded image density for approximating the KL divergence [82]. Due to the simple structure of the iconic images, we fit the GMMs with $K = 2$ Gaussian components using the RGB color values and draw $n = 100$ Monte Carlo samples to estimate the KL divergence in all experiments. Table 2 shows the image similarity metrics between the VCCA models using the iconic images. We also show the model classification accuracy for each model, which have been taken from Table 1. The models perform on par on the image similarity metrics, which indicates that the quality of the decoded images is intact if the model extends to utilizing text descriptions and class labels in addition to the iconic images.

In Figure 10, we display five different natural images from the test set, their true corresponding iconic image, the decoded iconic image from VCCA$_{xiwy}$. We also show the true and predicted labels from the class label decoder (see Pred. Label). Additionally, we report the image similarity metrics PSNR, SSIM, and KL divergence between the decoded and true iconic images. For the Mango and Royal Gala images, we observe that the decoded images are visually plausible, in terms of recognized item, color, and shape, in both cases, which coheres with the high PSNR and SSIM values and low KL values. The third row shows a shelf with orange and green bell peppers where the decoded image has indeed been decoded into a mix of a green and orange bell pepper. In the two succeeding rows, we display failure cases where the model confuses the true class label with other grocery items. We observe that each

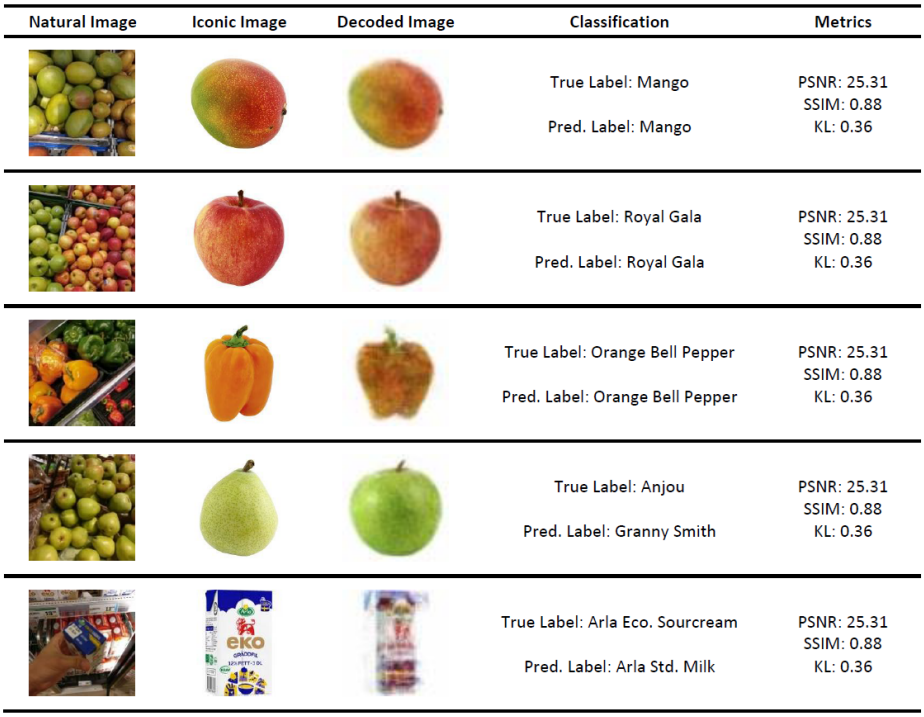


Figure 10: Examples of decoded iconic images from VCCA$_{xiwy}$ with their corresponding natural image and true iconic image as well as predicted labels and image similarity metrics. The column Classification shows the true label for the natural image (True Label) and the label predicted by the model (Pred. Label).

metric drops according to the mismatch between decoded and true iconic images. The fourth row shows a basket of Anjou pears, where the model confuses the pear with a Granny Smith apple which can be seen in the decoded image. In the fifth row, there are red milk packages stacked behind a handheld sourcream package, where the decoded image becomes a blurry mix of the milk and sourcream package. Although the predicted class is incorrect, we observe that the prediction is reasonable based on the decoded iconic image.

3 Discussion

In this section, we summarize the experimental results and discuss our findings.

Classification Results In the first experiments (see Section 2.3), we showed that utilizing all four views with SplitAE$_{xiwy}$ and VCCA$_{xiwy}$ resulted in the best classification performance on the test set. This indicates that these models take advantage of each view to learn representations that enhance their generalization ability compared to the single-view models. Moreover, using either the iconic images, the text descriptions or both views yields better representations for classification compared to using the natural image features alone. Note that it was necessary to properly set the scaling weights λ on the reconstruction losses of the additional views to achieve good

classification performance (see Table S2). Whenever the weight values are increased, the model tries to structure the representations according to variations between the items in the upweighted view rather than structuring the items based on visual features in the natural images, e.g., shape, color, and pose of items, image backgrounds, etc. Thus, the latent representations are structured based on semantic information that describes the item itself, which is important for constructing robust representations that generalize well in new environments. Furthermore, the class label decoders performed on par with the separate Softmax classifiers in most cases. The upside with training a separate classifier is that we only would have re-train the classifier if we receive a new class in the dataset, while we would have to train the whole model from scratch when the model uses a class label decoder. Note that the encoder for any of the multi-view models can be used for extracting latent representations for new tasks, whether the model utilizes the label or not, since the encoder only uses natural images as input.

Iconic Images vs. Text Descriptions In Table 1, the iconic images yielded higher classification accuracies compared to using the text descriptions. This was also evident in Figure 5 where the classification performance remains more or less the same regardless of the text description length T when the models utilize iconic images. We believe that the main reasons for the advantages with iconic images lie in the clear visual features of the items in these images, e.g., their color and shape, which carry lots of information that is important for image classification tasks. However, we also observed that iconic images and text descriptions can yield different benefits for constructing good representations. In Figure 6, we see that iconic images and text descriptions make the model construct different latent representations of the grocery items. Iconic images structures the representations with respect to color and shape of the items (see Figure 7), while the descriptions groups items based on their ingredients and flavor (see Figure 8). Therefore, the latent representations benefit differently from utilizing the additional views and a combination of all of them yields the best classification performance as shown in Table 1. We want to highlight the results in Figure 8, where the model manages to separate juice and yoghurt packages based on their text description. Refrigerated items, e.g., milk and juice packages, have in general very similar shapes and the same color if they come from the same brand. There are minor visual differences between items of the same brand that makes it possible to differentiate between them, e.g., the picture of the main ingredient on the package and ingredient description. Additionally, these items can be almost identical depending on which side of the package that is present on the natural image. When utilizing the text descriptions, we add useful information on how to distinguish between visually similar items that have different ingredients and contents. This is highly important for using computer vision models to distinguish between packaged items without having to use other kinds of information, e.g., barcodes.

Text Description Length We showed that the text descriptions are useful for the classification task, and that careful selection of the description length T is important for achieving the best possible performance (see Section 2.3), In Figure 5, we observed that most models achieve significantly better classification performance when the text description length T increases up until $T = 32$. The reason for this increase is due to the similarities between the descriptions of items from the same kind or brand,

such as milk and juice packages. For instance, in Table S1, the first sentence in the descriptions for the milk packages only differ by the ninth word, which is *organic* for the ecological milk package. This means that their descriptions will be identical when $T = 8$. Therefore, the descriptions will become more different from each other as we increase T, which helps the model to distinguish between items with similar descriptions. However, the classification accuracies have more or less saturated when setting $T > 32$, which is also due to the similarity between the descriptions. For example, the bell pepper descriptions in Table S1 only differ by the second word that describes the color of the bell pepper, i.e., the words *yellow* and *orange*. We also see that the third and fourth sentences in the descriptions of the milk packages are identical. The text descriptions typically have words that separate the items in the first or second sentence, whereas the following sentences provide general information on ingredients and how the item can be used in cooking. For items of the same kind but of different colors or brand, e.g., bell peppers or milk packages respectively, the useful textual information for constructing good representations of grocery items typically comes from a few words in the description that describes features of the item. Therefore, the models yield better classification performance when T is set to include at least the whole first sentence of the description. We could select T more cleverly, e.g., by using different T for different descriptions to make sure that we utilize the words that describe the item or filter out non-informative words for the classification task.

VCCA vs. VCCA-private The main motivation for using VCCA-private is to use private latent variables for modeling view-specific variations, e.g., image backgrounds and writing styles of text descriptions. This could allow the model to build shared representations that more efficiently combine salient information shared between the views for training better classifiers. This would then remove noise from the shared representation since the private latent variables are responsible for modeling the view-specific variations. For VCCA-private$_{xw}$, we observed that the private latent spaces managed to group different image backgrounds and grocery items with similar text descriptions respectively in Figure 9c and 9d respectively. This model also performed on par with VCCA$_{xw}$ regarding classification performance in Table 1. However, we also saw in the same table that the VCCA-private models using the iconic image perform poorly on the classification task compared to their VCCA counterpart. The reason for why this model fails is because of a form of *posterior collapse* [83] in the encoder for the iconic image, where the encoder starts outputting random noise. We noticed this as the KL divergence term for the private latent variable converged to zero when we trained the models for 500 epochs (see Figure S2 and S3), which means that the encoder outputs a distribution which equals a Gaussian prior. The same phenomenon occurs for VCCA-private with the text description as well. We have also experimented with other models with encoders, such as Deep CCA and DCCAE, which also suffered from the collapsing encoder problem for the additional views. Therefore, we believe that the collapsing effect is a consequence of only having access to a single iconic image and text description for every grocery item. Therefore, a potential solution would be to extend the dataset with multiple web-scraped iconic images and text descriptions for every grocery item, which would then establish some variability within the view for each item class. Another possible

solution would be to use data augmentation techniques to create some variability in the web-scraped views. For example, we could take a denoising approach and add noise to the iconic images which would force the decoder to reconstruct the real iconic images [84]. For the text descriptions, we could mask words at random in the encoder and let the decoder predict the whole description, which would work as a form of *word dropout* [83, 85]. We leave this for future work if such augmentation techniques can create the needed variability for learning more robust representations as well as discovering the structures of the private latent spaces that this approach would bring.

Decoded Iconic Images We observed with image similarity metrics that the quality of decoded iconic images coheres to some extent with the classification performance for VCCA models using the iconic images (see Section 2.5). We also showed that the decoded images are visually plausible decoded images with respect to colors, shapes, and identities of the grocery items in the dataset. The values for the metrics PSNR, SSIM, and KL divergence are similar across the different VCCA models. Since we used RGB values for estimating KL, we believe that including spatial information of pixels or using other color spaces, e.g., Lab, could provide more information about the dissimilarities between the decoded and true iconic images. To thoroughly assess the relationship between good classification performance and accurately decoding the iconic images, we also suggest evaluating the image quality on other image similarity metrics, e.g., perceptual similarity [86]. Finally, we see the decoding of iconic images as a promising method to evaluate the quality of the latent representations as well as enhancing the interpretability of the classification. For example, we could inspect decoded iconic images qualitatively or by using image similarity metrics to determine how certain the model was about the present items in the natural images, which could then be used as a tool for explaining misclassifications.

3.1 Conclusions

In this paper, we introduce a dataset with natural images of grocery items taken in real grocery store environments. Each item class is accompanied by web-scraped information in the form of an iconic image and a text description of the item. The main application for this dataset is for training image classifiers that can assist visually impaired people when shopping for groceries but is not limited to this use case only.

We selected the multi-view generative model VCCA that can utilize all of the available data views for image classification. To evaluate the contribution to the classification performance for each view, we conducted an ablation study comparing classification accuracies between VCCA models with different combinations of the available data types. We showed that utilizing the additional views with VCCA yields higher accuracies on classifying grocery items over models only using the natural images. The iconic images and text descriptions impose different structures of the shared latent space, where we observed that 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. These types of semantics that VCCA has learned can be useful for generalizing to new grocery items and other object recognition tasks. We also investigated VCCA-private, which introduces private latent variables

for view-specific variations, that separates the latent space into shared and private spaces for each view to provide high-quality representations. However, we observed that the private latent variables for the web-scraped views became uninformative by modeling noise due to the lack of variations in the additional web-scraped views. This encourages to explore new methods for extracting salient information from such data views that can be beneficial for downstream tasks.

An evident direction of future work would be to investigate other methods for utilizing the web-scraped views more efficiently. For instance, we could apply pre-trained word representations for the text description, e.g., BERT [85] or GloVe [87], to see if they enable the construction of representations that can more easily distinguish between visually similar items. Another interesting direction would be to experiment with various data augmentation techniques in the web-scraped views to create view-specific variations without the need for collecting and annotating more data. It is also important to investigate how the model can be extended to recognize multiple items. Finally, we see zero- and few-shot learning [67] of new grocery items and transfer learning [88] as potential applications where our dataset can be used for benchmarking of multi-view learning models on classification tasks.

4 Experimental Procedures

4.1 Resource Availability

Lead Contact: Marcus Klasson is the lead contact for this study and can be contacted by email at mklas@kth.se.

Materials Availability: There are no physical materials associated with this study.

Data and Code Availability:

1. The Grocery Store dataset along with documentation is available at the following Github repository: https://github.com/marcusklasson/GroceryStoreDataset.
2. The source code for the multi-view models along with documentation is available at the following Github repository: https://github.com/marcusklasson/vcca_grocerystore.

4.2 Methods

In this section, we outline the details of the models we use for grocery classification. We begin by introducing autoencoders and SplitAEs [18]. Then we describe VAEs [74] and how it is applied to single-view data, followed by the introduction of VCCA [7] and how we adapt it to our dataset. We also discuss a variant of VCCA called VCCA-private [7], which is used for extracting private information about each view in addition to shared information across all views by factorizing the latent space. The graphical model representations of the VAE, VCCA, and VCCA-private models that have been used in this paper are shown in Figure S5. The model names use subscripts to denote the views utilized for learning the shared latent representations. For example, VCCA$_{xi}$ utilizes natural image features x and iconic images i, while

VCCA$_{xiwy}$ uses natural image features x and iconic images i, text descriptions w, and class labels y.

4.2.1 Autoencoders and Split Autoencoders

The autoencoding framework can be used for feature extraction and learning latent representations of data in unsupervised manners [89]. It begins with defining a parameterized function called the encoder for extracting features. We denote the encoder as f_ϕ where ϕ includes its parameters, which commonly are the weights and bias vectors of a neural network. The encoder is used for computing a feature vector $h = f_\phi(x)$ from the input data x. Another parameterized function g_θ called the decoder is also defined, that maps the feature h back into input space, i.e., $\hat{x} = g_\theta(h)$. The encoder and decoder are learned simultaneously to minimize the reconstruction loss between the input and its reconstruction of all training samples. By setting the dimension of the feature vector smaller than the input dimension, i.e., $d_h < d_x$, the autoencoder can be used for dimensionality reduction which makes the feature vectors suitable for training linear classifiers in a cheap way.

As in the case for the Grocery Store dataset, we have multiple views available during training, while only the natural image view is present at test time. In this setting, we can use a Split Autoencoder (SplitAE) to extract shared representations by reconstructing all views during training from the one view that is available during the test phase [17, 18]. As an example, we have the two-view case with x present at both training and test while y is only available during training. We therefore define an encoder f_ϕ and two decoders g_{θ_x} and g_{θ_y} where both decoders inputs the same representation $h = f_\phi(x)$. The objective of the SplitAE is to minimize the sum of the reconstruction losses, which will encourage representations h that best reconstructs both views. The total loss is then

$$\mathcal{L}_{SplitAE}(\theta, \phi; x, y) = \lambda_x \mathcal{L}_x(x, g_{\theta_x}(h)) + \lambda_y \mathcal{L}_y(y, g_{\theta_y}(h)), \tag{2}$$

where $\theta_x, \theta_y \in \theta$ and λ_x, λ_y are scaling weights for the reconstruction losses. For images, the reconstruction loss can be the mean squared error, while the cross-entropy loss is commonly used for class labels and text. This architecture can be extended to more than two views by simply using view-specific decoders that input the shared representation extracted from natural images. Note that in the case when the class labels are available, we can use the class label decoder g_{θ_y} as a classifier during test time. Alternatively, we can train a separate classifier with the learned shared representations after the SplitAE has been trained.

4.2.2 Variational Autoencoders with only Natural Images

The Variational Autoencoder (VAE) is a generative model that can be used for generating data from single views. Here, we describe how the VAE learns latent representations of the data and how the model can be used for classification. VAEs define a joint probability distribution $p_\theta(x, z) = p(z)p_\theta(x|z)$, where $p(z)$ is a prior distribution over the latent variables z and $p_\theta(x|z)$ is the likelihood over the natural images x given z. The prior distribution is often assumed to be an isotropic Gaussian distribution, $p(z) = \mathcal{N}(z \,|\, \mathbf{0}, \mathbf{I})$, with the zero vector $\mathbf{0}$ as mean and the identity matrix $\mathbf{I}$ as the covariance. The likelihood $p_\theta(x|z)$ takes the latent variable z as input and

outputs a distribution parameterized by a neural network with parameters θ which is referred to as the decoder network. A common distribution for natural images is a multivariate Gaussian, $p_\theta(x|z) = \mathcal{N}(x \,|\, \mu_x(z), \boldsymbol{\sigma}_x^2(z) \odot \mathbf{I})$, where $\mu_x(z)$ and $\boldsymbol{\sigma}_x^2(z)$ are the means and standard deviations of the pixels respectively outputted from the decoder and $\odot$ denotes element-wise multiplication. We wish to find latent variables z that are likely to have generated x, which is done by approximating the intractable posterior distribution $p_\theta(z|x)$ with a simpler distribution $q_\phi(z|x)$ [75]. This approximate posterior $q_\phi(z|x)$ is referred to as the encoder network since it is parameterized by a neural network ϕ which inputs x and outputs a latent variable z. Commonly, we let the approximate posterior to be Gaussian $q_\phi(z|x) = \mathcal{N}(z \,|\, \mu_z(x), \sigma_z^2(x) \odot \mathbf{I})$, where the mean $\mu_z(x)$ and variance $\sigma_z^2(x)$ are the outputs of the encoder. The latent variable z is then sampled using the mean and variance from the encoder with the reparameterization trick [74, 90]. The goal is to maximize a tractable lower bound on the marginal log-likelihood of x using $q_\phi(z|x)$:

$$\log p_\theta(x) \geq \mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z) \right] - D_{\mathrm{KL}}(q_\phi(z|x) \,||\, p(z)). \tag{3}$$

The last term is the Kullback-Leibler (KL) divergence [78] of the approximate posterior from the prior distribution, which can be computed analytically when $q_\phi(z|x)$ and $p(z)$ are Gaussians. The expectation can be viewed as a reconstruction loss term, which can be approximated using Monte Carlo sampling from $q_\phi(z|x)$. The lower bound $\mathcal{L}$ is called the evidence lower bound (ELBO) and can be optimized with stochastic gradient descent via backpropagation. The mean outputs $\mu_z(x)$ from the encoder $q_\phi(z|x)$ are commonly used as the latent representation of the natural images x. We can use the representations $\mu_z(x)$ for training any classifier $p(y|\mu_z(x))$, e.g., a Softmax classifier, where y is the class label of x. We can therefore see training the VAE as a pre-processing step, where we extract a lower-dimensional representation of the data x which hopefully makes the classification problem easier to solve. We can also extend the VAE with a generative classifier by incorporating the class label y in the model [91, 92]. Hence, the VAE defines a joint distribution $p_\theta(x, y, z) = p(z)p_{\theta_x}(x|z)p_{\theta_y}(y|z)$ where the class label decoder $p_{\theta_y}(y|z)$ is used as the final classifier. We therefore aim to maximize the ELBO on the marginal log-likelihood over x and y:

$$\begin{aligned} \log p_\theta(x, y) &\geq \mathcal{L}(\theta, \phi; x, y) \\ &= \lambda_x \mathbb{E}_{q_\phi(z|x)} \left[\log p_{\theta_x}(x|z) \right] + \lambda_y \mathbb{E}_{q_\phi(z|x)} \left[\log p_{\theta_y}(y|z) \right] \\ &\quad - D_{\mathrm{KL}}(q_\phi(z|x) \,||\, p(z)). \end{aligned} \tag{4}$$

The parameters λ_x and λ_y are used for scaling the magnitudes of the expected values. When predicting the class label for an unseen natural image x^*, we can consider multiple output predictions of the class label by sampling K different latent variables for x^* from the encoder to determine the final predicted class. For example, we could either average the predicted class scores over K or use the maximum class score from K samples as the final predictions. In this paper, we compute the average of the predicted class scores using

$$\hat{y} = \arg\max_y \frac{1}{K} \sum_{k=1}^{K} p_{\boldsymbol{\theta_y}}(y|z^{(k)}), \;\; z^{(k)} \sim q_\phi(z|x^*), \tag{5}$$

where $\hat{y}$ is the predicted class for the natural image x^*. We denote this model as VCCA_{xy} due to its closer resemblance to VCCA than VAE in this paper.

4.2.3 Variational Canonical Correlation Analysis for Utilizing Multi-View Data

In this section, we describe the details of Variational Canonical Correlation Analysis (VCCA) [7] for our application. In the Grocery Store dataset, the views can be the natural images, iconic images, text descriptions, or class labels and we can use any combination of those three in VCCA. To illustrate how we can employ this model to the Grocery Store dataset, we let the natural images x and the iconic images i be the two views. We assume that both views x and i have been generated from a single latent variable z. Similarly as with VAEs, VCCA defines a joint probability distribution $p_\theta(x, i, z) = p(z)p_{\theta_x}(x \,|\, z)p_{\theta_i}(i \,|\, z)$. There are now two likelihoods for each view modeled by the decoders $p_{\theta_x}(x \,|\, z)$ and $p_{\theta_i}(i \,|\, z)$ are represented as neural networks with parameters θ_x and θ_i. Since we want to classify natural images, the other available views in the dataset will be missing when we have received a new natural image. Therefore, the encoder $\boldsymbol{q}_\phi(z|x)$ only uses x as input to infer the latent variable z shared across all views, such that we do not have to use inference techniques that handle missing views. With this choice of approximate posterior, we receive the following ELBO on the marginal log-likelihood over x and i that we aim to maximize:

$$\begin{aligned} \log p_\theta(x, i) &\geq \mathcal{L}(\theta, \phi; x, i) \\ &= \lambda_x \mathbb{E}_{q_\phi(z|x)} \left[\log p_{\theta_x}(x|z) \right] + \lambda_i \mathbb{E}_{q_\phi(z|x)} \left[\log p_{\theta_i}(i|z) \right] \\ &\quad - D_{\text{KL}}(q_\phi(z|x) \,||\, p(z)). \end{aligned} \tag{6}$$

The parameters λ_x and λ_i are used for scaling the magnitude of the expected values for each view. We provide a derivation of the ELBO for three or more views in the Supplemental Experimental Procedures. The representations $\mu_z(x)$ from the encoder $q_\phi(z|x)$ can be used for training a separate classifier. We can also add a class label decoder $p_{\theta_y}(y|z)$ to the model and use Equation 5 to predict the class of unseen natural images.

4.2.4 Extracting Private Information of Views with VCCA-private

In the following section, we show how the VCCA model can be altered to extract shared information between the views as well as view-specific private information to enable more efficient posterior inference. Assuming that a shared latent variable z is sufficient for generating all different views may have its disadvantages. Since the information in the views is rarely fully independent nor fully correlated, information only relevant to one of the views will be mixed with the shared information. This may complicate the inference of the latent variables, which potentially can harm the classification performance. To tackle this problem, previous works have proposed learning separate latent spaces for modeling shared and private information of the different views [7, 61, 69]. The shared information should represent the correlations between the views, while the private information represents the independent variations within each view. As an example, the shared information between natural and iconic images are the visual features of the grocery item, while their private informa-

tion is considered as the various backgrounds that can appear in the natural images and the different locations of non-white pixels in the iconic images. For the text descriptions, the shared information would be words that describe visual features in the natural images, whereas the private information would be different writing styles for describing grocery items with text. We adapt the approach from [7] called VCCA-private and introduce private latent variables for each view along with the shared latent variable. To illustrate how we employ this model to the Grocery Store dataset, we let the natural images x and the text descriptions w be the two views. The joint distribution of this model is written as

$$p_\theta(x, w, z, u_x, u_w) = p_{\theta_x}(x|z, u_x)p_{\theta_w}(w|z, u_w)p(z)p(u_x)p(u_w), \tag{7}$$

where u_x and u_w are the private latent variables for the x and w respectively. To enable tractable inference of this model, we employ a factorized approximate posterior distribution of the form

$$q_\phi(z, u_x, u_w|x, w) = q_{\phi_z}(z|x)q_{\phi_x}(u_x|x)q_{\phi_w}(u_w|w), \tag{8}$$

where each factor is represented as an encoder network inferring its associated latent variable. With this approximate posterior, the ELBO for VCCA-private is given by

$$\begin{aligned}
\log p_\theta(x, w) &\geq \mathcal{L}_{\text{private}}(\theta, \phi; x, w) \\
&= \lambda_x \mathbb{E}_{q_{\phi_z}(z|x), q_{\phi_x}(u_x|x)} \left[\log p_{\theta_x}(x|z, u_x)\right] \\
&\quad + \lambda_w \mathbb{E}_{q_{\phi_z}(z|x), q_{\phi_w}(u_w|w)} \left[\log p_{\theta_w}(w|z, u_w)\right] \\
&\quad - D_{\text{KL}}(q_{\phi_z}(z|x) \,||\, p(z)) \\
&\quad - D_{\text{KL}}(q_{\phi_x}(u_x|x) \,||\, p(u_x)) - D_{\text{KL}}(q_{\phi_w}(u_w|w) \,||\, p(u_w)).
\end{aligned} \tag{9}$$

The expectations in $\mathcal{L}_{\text{private}}(\theta, \phi; x, w)$ in Equation 9 can be approximated using Monte Carlo sampling from $q_\phi(z|x)$. The sampled latent variables are concatenated and then used as input to their corresponding decoder. We let the approximated posteriors over both shared and private latent variables to be multivariate Gaussian distributions and their prior distributions to be standard isotropic Gaussians $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The KL divergences in Equation 8 can then be computed analytically. Since only natural images are present during test time and because the shared latent variable z since it should contain information about similarities between the views, e.g., the object class, we use the encoder $q_\phi(z|x)$ to extract latent representations $\mu_z(x)$ for training a separate classifier. As for the VAE and standard VCCA, we can also add a class label decoder $p_{\theta_y}(y|z)$ only conditioned on z to the model and use Equation 5 to predict the class of unseen natural images. We evaluated the classification performance of VCCA-private and compare it to the standard VCCA model only using a single shared latent variable (see Section 2.3).

4.3 Experimental Setup

This section briefly describes the network architecture designs and the selection of hyperparameters for the models. See Section II.1 for full details on the network architectures and hyperparameters that we use.

Processing of Natural Images We use a DenseNet169 [77] as the backbone for processing the natural images since this architecture showed good classification performance in [6]. As our first baseline, we customize the output layer of DenseNet169

to our the Grocery Store dataset and train it from scratch to classify the natural images. For the second baseline, we train a Softmax classifier on off-the-shelf features from DenseNet169 pre-trained on the ImageNet dataset [19], where we extract 1664-dimensional from the average pooling layer before the classification layer in the architecture. Using pre-trained networks as feature extractors for smaller datasets has previously been proven to be a successful approach for classification tasks [93], which makes it a suitable baseline for the Grocery Store dataset. We denoted the DenseNet169 trained from scratch and the Softmax classifier trained on off-the-shelf features as DenseNet-scratch and Softmax respectively in Section 2.

Network Architectures We use the same architectures for SplitAE and VCCA for a fair comparison. We train the models using off-the-shelf features extracted from a pre-trained DenseNet169 for the natural images. No fine-tuning of the DenseNet backbone was used in the experiments, which we leave for future research. The image feature encoder and decoder consist of a single hidden layer, where the encoder outputs the latent representation and the decoder reconstructs the image feature. We use a DCGAN [94] generator architecture for the iconic image decoder reconstructing the iconic images. The text description is predicted sequentially using an LSTM network [95]. The label decoder uses a single hidden layer with 512 hidden units and Leaky ReLU activation. The latent space dimension to $d_z = 200$ for all SplitAE and VCCA models in the experiments. In the VCCA-private models, the encoder and decoders have the same architectures as in the VCCA models. We use a reversed DCGAN generator as the iconic image encoder. The text description encoder is an LSTM. We obtain an embedding for the description by averaging all of the hidden states h_t generated from the LSTM, i.e., $\frac{1}{T}\sum_{t=1}^{T} h_t$, and input it to a linear layer that outputs the latent representation. The dimensions of the private latent spaces are the same as the shared latent space dimension $d_z = 200$.

Training Details In all experiments, we train all models for 200 epochs using the Adam optimizer [96] with initial learning rate 10^{-4} and hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We apply the sum-of-squares loss for the natural and iconic images and the categorical cross-entropy loss for text descriptions and class labels. The latent representations for the AE and SplitAE are the output from the encoder, while for the VAE, VCCA, and VCCA-private we instead use the mean outputs $\mu_z(x)$ from the encoder. In VCCA-private, we use the the mean outputs $\mu_{u_x}(x)$ and $\mu_{u_w}(w)$ for visualizing the private latent representations (see Figure 9). The Softmax classifiers are trained for 100 epochs using with initial learning rate 10^{-4} and hyperparameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We run a hyperparameter search for the scaling weights λ for the reconstruction losses of each view (see Section II.1 for more details).

Choice of Text Description Length T We wanted to investigate how the classification performance is affected by the text description length T for the SplitAE and VCCA models using the text description w. Since the LSTM predicts the text description sequentially, the computational time increases as the text description length increases. We began by setting $T = 16$ because of how the sentence length was set for the image captioning model in [12]. Then we created an interval by taking steps with 8 words up to $T = 40$ and included the minimum, mean, and maximum text description lengths which are $T = 6, 36$, and 91 respectively. We added two

additional points at $T = 50$ and 75 since for most models there was a slight increase in classification performance when T was increased from 40 to 91.

Computational Time The computational time for the SplitAE and VCCA models differs depending on which views that are being used. We measured the number of seconds per epoch (s/epoch) on an Nvidia GeForce RTX 2080 Ti, where an epoch consists of the 2640 training samples. Running experiments with the text description view took around 0.4–2.7 s/epoch with text description length $T \in [6, 91]$. Adding the iconic image view and training the models took around 4.5–6.2 s/epoch depending on the text description length.

Visualization Method We use PCA to visualize the latent space by plotting the latent representations from the trained encoders. PCA is used for projecting the representations from dimension d_z to a 2D space. We obtain the principal components for the representations with the natural images from the training set. In all figures, we plot the representations of test set images given the principal components obtained from the training images (see Section 2.4). To distinguish more easily between the class labels of the images, we occasionally use the iconic images from the dataset and plot the corresponding iconic image of the representation on its location in the 2D space.

I Supplemental Figures

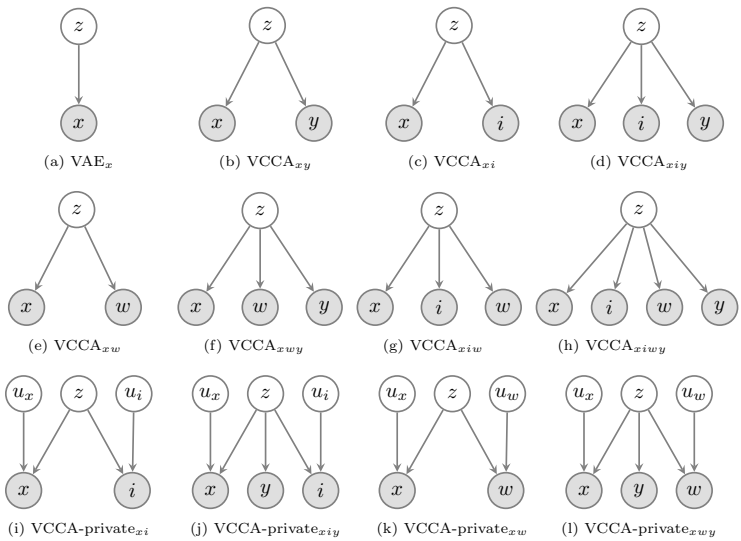
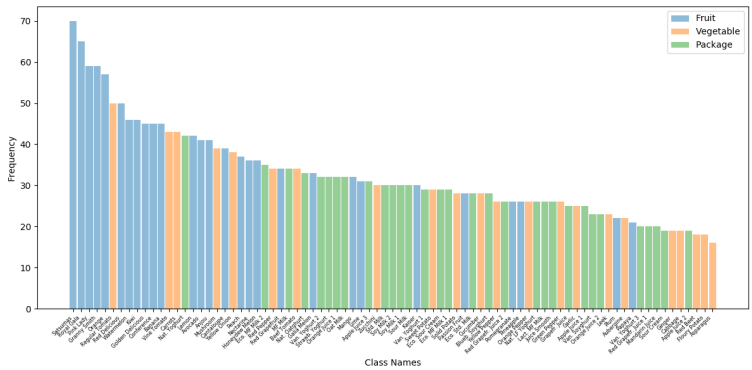
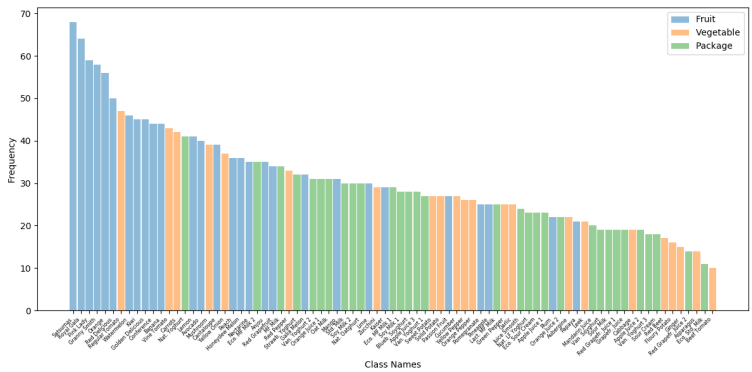


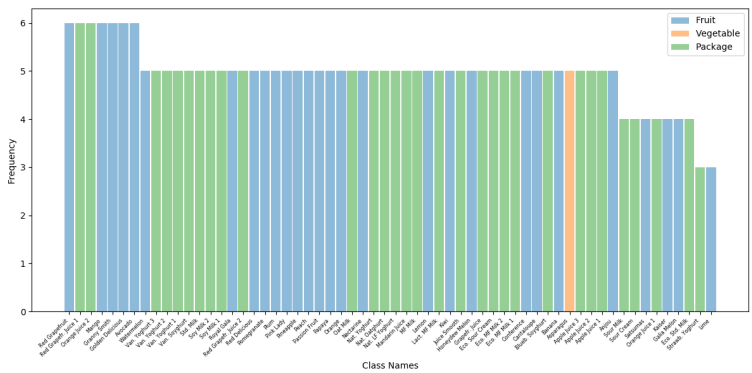
Figure 11: The graphical models of the VAE, VCCA, and VCCA-private, where nodes represent random variables and edges indicate possible dependence. Grey nodes are observed random variables, while white nodes are latent random variables. The joint distribution is given by the product of the conditional distributions of nodes given their parents. Abbreviations: VAE, Variational Autoencoder; VCCA, Variational Canonical Correlation Analysis.



(a) Histogram for the training split.



(b) Histogram for the test split.



(c) Histogram for the validation split.

Figure 12: Histograms of natural images for every class in the training (a), test (b), and validation (c) splits of the Grocery Store dataset. We also show with different colors on the bins if the class is either a Fruit, Vegetable, or Package item.

Table 3: Examples of text descriptions and their corresponding class label and iconic image from the Grocery Store dataset.

Class Label	Iconic Image	Text Description
Golden Delicious Apple		Golden Delicious has a white juicy pulp and a greenish yellow shell. The taste is mellow and sweet, making Golden Delicious suitable for desserts.
Red Delicious Apple		Red Delicious is a dark red apple with relatively soft pulp and sweet taste.
Orange		There are many different types of oranges that ripen and is sold during different parts of the year. The orange is a very important vitamin C source and the vitamins are best kept if the fruit is eaten naturally.
Yellow Bell Pepper		The yellow pepper is much sweeter than the green. It also contains more vitamins and antioxidants than the green. Peppers are good to eat raw in salads and as garnish, but are also good to fry, stew or gratinate, for example with filling. Paprika also fits well in pots, gratins and pies.
Orange Bell Pepper		The orange pepper is sweeter than the green. It also contains more vitamins and antioxidants than the green. Peppers are good to eat raw in salads and as garnish, but are also good to fry, stew or gratinate, for example with filling. Paprika also fits well in pots, gratins and pies.
Arla Medium Fat Milk		Fresh skimmed milk made from Swedish milk from Arlagårdar. Skimmed milk has a delicious full-bodied milk flavor and is a popular choice for breakfast cereals, porridge or as a drink for the meal. Milk is a natural source of, for example, protein, calcium and vitamin B12. Protein contributes to muscle building and calcium is needed to maintain a normal bone structure. The brand Arla Ko guarantees that the product is made of 100% Swedish milk.
Arla Ecological Medium Fat Milk		Fresh skimmed milk made from Swedish milk from organic Arlagårdar. Skimmed milk has a delicious full flavor and is a popular choice for breakfast cereals, porridge or as a drink for the meal. Milk is a natural source of, for example, protein, calcium and vitamin B12. Protein contributes to muscle building and calcium is needed to maintain a normal bone structure. The brand Arla Ko guarantees that the product is made of 100% Swedish milk. The new brown carton has 24 percent lower climate impact compared to the previous white carton.
God Morgon Orange Juice		God Morgon Orange is pressed by sun-dried, hand-picked oranges. The package contains juice from 2 kilograms of oranges!
Tropicana Golden Grapefruit Juice		Tropicana Golden Grape is a ready to drink juice with pulp pressed on grapefruit. Not from concentrate. Mildly pasteurized.

Table 4: Classification accuracies on the test set for all models in percentage (%) along with the best hyperparameter setting, i.e., the scaling weights λ and text description length T, for each model. The column Accuracy corresponds to the fine-grained classification accuracy. The column Coarse Accuracy corresponds to the classifying a class within the correct parent class. Results are averaged using 10 different random seeds and we report both means and standard deviations. Abbreviations: AE, Autoencoder; VAE, Variational Autoencoder; SplitAE, Split Autoencoder; VCCA, Variational Canonical Correlation Analysis.

Model	λ_x	λ_i	λ_w	λ_y	T	Accuracy (%)	Coarse Accuracy (%)
DenseNet-scratch	-	-	-	-	-	67.33 ± 1.35	75.67 ± 1.15
Softmax	-	-	-	-	-	71.67 ± 0.28	83.34 ± 0.32
AE_x+Softmax	1	-	-	-	-	70.69 ± 0.82	82.42 ± 0.58
VAE_x+Softmax	1	-	-	-	-	69.20 ± 0.46	81.24 ± 0.63
$SplitAE_{xy}$	1	-	-	1000	-	70.34 ± 0.56	82.11 ± 0.38
$VCCA_{xy}$	1	-	-	1000	-	70.72 ± 0.56	82.12 ± 0.61
$SplitAE_{xi}$+Softmax	1	1000	-	-	-	77.68 ± 0.69	87.09 ± 0.53
$VCCA_{xi}$+Softmax	1	1000	-	-	-	77.02 ± 0.51	86.46 ± 0.42
VCCA-private$_{xi}$+Softmax	1	10	-	-	-	73.04 ± 0.56	84.16 ± 0.51
$SplitAE_{xiy}$	1	1000	-	1000	-	77.43 ± 0.80	87.14 ± 0.57
$VCCA_{xiy}$	1	1000	-	1000	-	77.22 ± 0.55	86.54 ± 0.51
VCCA-private$_{xiy}$	1	10	-	1000	-	74.04 ± 0.83	84.59 ± 0.83
$SplitAE_{xw}$+Softmax	1	-	1000	-	40	76.27 ± 0.66	86.45 ± 0.56
$VCCA_{xw}$+Softmax	1	-	1000	-	75	75.37 ± 0.46	86.00 ± 0.32
VCCA-private$_{xw}$+Softmax	1	-	1000	-	75	75.11 ± 0.81	85.91 ± 0.55
$SplitAE_{xwy}$	1	-	1000	10	75	75.78 ± 0.84	86.13 ± 0.63
$VCCA_{xwy}$	1	-	1000	10	75	74.72 ± 0.85	85.59 ± 0.78
VCCA-private$_{xwy}$	1	-	1000	1000	50	74.92 ± 0.74	85.59 ± 0.67
$SplitAE_{xiw}$+Softmax	1	1000	1000	-	24	77.79 ± 0.48	87.12 ± 0.62
$VCCA_{xiw}$+Softmax	1	1000	1000	-	32	77.51 ± 0.51	86.69 ± 0.41
$SplitAE_{xiwy}$	1	1000	1000	1000	24	78.18 ± 0.53	87.26 ± 0.46
$VCCA_{xiwy}$	1	1000	1000	1000	91	77.78 ± 0.45	86.88 ± 0.47

II Supplemental Experimental Procedures

II.1 Details on Experimental Setup

In this section, we provide the full details on the experimental setups.

Training DenseNet169 from Scratch We train a DenseNet169 on the dataset from scratch as a baseline. We train the network using stochastic gradient descent for 300 epochs and follow the learning rate schedule of Huang et al. [77], i.e., using an initial learning rate of 0.1 and dividing it by 10 after 150 and 225 epochs. We use a weight decay of 10^{-4} and Nesterov momentum of 0.9 without dampening. We denote this network as DenseNet-scratch in the experiments.

Training the Softmax Classifier We use a Softmax classifier trained on off-the-shelf features as another baseline. We use a DenseNet169 [77] pre-trained on ImageNet 1K as the feature extractor, where we extract 1664-dimensional from the average pooling layer before the classification layer in the architecture. The Softmax classifier is trained for 100 epochs with batch size 64 to minimize the cross-entropy loss. We use the Adam optimizer [96] with initial learning rate 10^{-4} and hyperparameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Note that we used no regularization when training

the Softmax classifier. We denote this classifier as Softmax in the experiments. This training setup is also used when training the Softmax classifiers for SplitAE, VCCA, and VCCA-private.

Architectures for Single-View Autoencoders We use a vanilla autoencoder and a VAE as baselines that learn latent representations of the natural image features only. These models are denoted as AE_x and VAE_x respectively in the experiments. Their latent representations have dimension $d_z = 200$ throughout all experiments. The encoder and decoder networks consist of one hidden layer of 512 hidden units with Leaky ReLU activation. The decoder aims to reconstruct the natural image features and we use the sum-of-squares as the reconstruction loss during training. For the VAE, we use the mean outputs $\mu_z(x)$ from the encoder as the latent representations of the image features to train a Softmax classifier, which we denote as VAE_x+Softmax.

Architectures for SplitAE and VCCA We use the same encoder and decoder architecture for the natural image features as in the single-view autoencoders for all SplitAE, VCCA, and VCCA-private models, i.e., one hidden layer of 512 hidden units with Leaky ReLU activation. We set the latent dimension to $d_z = 200$ in all experiments. The class label decoders use the same architecture as the image feature decoder and predict the class label by optimizing the cross-entropy loss. The iconic image decoders use the generator architecture of the DCGAN [94]. This decoder reconstructs the iconic images $i \in \mathbb{R}^{64\times64\times3}$ by minimizing the sum of squares loss. The text descriptions are assumed to be a sequence of words $w = (w_1, \ldots, w_T)$, where T is the length of the description. We create a vocabulary $V \in \mathbb{R}^{658}$ of the total number of unique words from all text descriptions in the dataset. The text description decoder is an LSTM [95] followed by a linear layer that predicts the next word in the description. More specifically, at time step t, the decoder yields a multinomial probability distribution $p_{\theta_w}(w_t|w_{t-1}, z)$ defined over $w_t \in V$. The LSTM is trained using teacher forcing, meaning that we use the previous ground truth word as input at every time step during the training phase. We project each word into an embedding space of dimensionality $d_{emb} = 200$ by using a lookup table before the word is input to the LSTM. The hidden state h and memory state c of the LSTM is initialized with a linear projection of the latent representation z to provide the LSTM with some context about the natural image. We use the same dimension for h and c as for z, i.e., $d_z = d_h = d_c = 200$. We minimize the cross-entropy loss at every time step by comparing the predicted word with the true word in the description. We apply dropout [97] with a keep rate of 0.5 on the output hidden state h_t before we input it through the linear layer that predicts the word at time step t.

Architectures for VCCA-private In the VCCA-private models, the decoders has the same architectures as the VCCA models. The encoder for the shared latent variable z is also the same as the encoder for the previous described models. The encoder for the private latent variable u_x uses an identical architecture as encoder for z. We use a convolutional encoder for the iconic image private latent variable, which is a reversed DCGAN generator outputting the mean $\mu_{u_i}(i)$ and variance $\sigma^2_{u_i}(i)$. The text description encoder is an LSTM with the same architectural details as the LSTM decoder. We obtain an embedding for the description by averaging all of the hidden states h_t generated from the LSTM, i.e., $\frac{1}{T}\sum_{t=1}^{T} h_t$, and input it to a linear

layer outputting the mean $\mu_{u_w}(w)$ and the variance $\sigma^2_{u_w}(w)$ for the private latent variable for the text description. We use the same latent dimensions for the shared and private latent variables, i.e., $d_z = d_{u_x} = d_{u_i} = d_{u_w} = 200$.

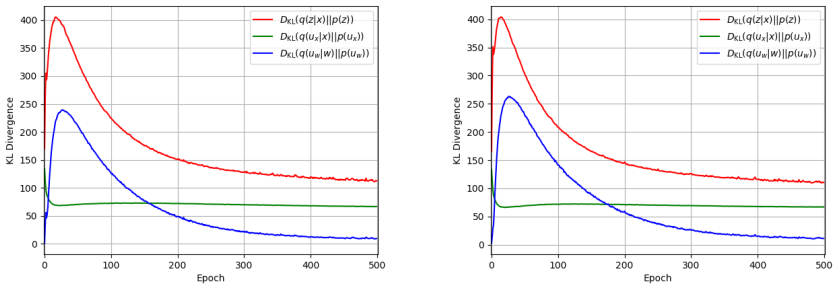
Training the Single- and Multi-View Autoencoders The single- and multi-view autoencoding models are trained for 200 epochs with batch size 64 and aims to minimize either their reconstruction losses or their corresponding ELBOs. We use the Adam optimizer with initial learning rate 10^{-4} and hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ in all experiments. The mean outputs $\mu_z(x)$ from the encoder are used as the latent representations of the image features to train a Softmax classifier. For the class label decoder, we use $K = 1$ posterior samples for predicting the class label during training, while we set $K = 5$ in the validation and test stages.

Grid Search We run a hyperparameter search for the scaling weights λ for the reconstruction losses of the views using a grid search with grid points $\{0.1, 1, 10, 100, 1000\}$. The grid search is performed for all VCCA and VCCA-private models. The weight for the natural image feature loss λ_x is used as a reference by setting it to $\lambda_x = 1$ throughout all experiments. This means that we only vary the scaling weights λ_i, λ_w, and λ_y. We run the grid searches using three different random seeds and average the resulting validation accuracies to select the best hyperparameter setting. Table 4 shows the best hyperparameter settings for the VCCA and VCCA-private models with their validation accuracies. Note that we use the same scaling weights for SplitAE as the ones we found for its corresponding VCCA model. To summarize the grid search results, it is always beneficial to use scaling weights $\lambda > 1$ for the models to enhance the classification accuracy. This will make the models add some semantically meaningful information to the latent space from the additional views, which makes the models more suitable for downstream tasks such as classification.

II.2 Posterior Collapse in VCCA-private

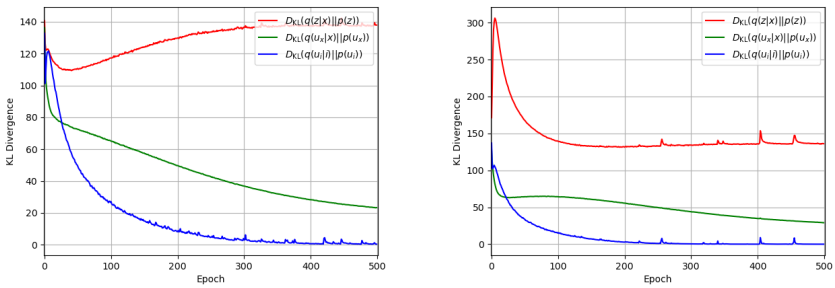
We noticed in Table 1 that VCCA-private achieves similar classification performance as standard VCCA when utilizing the text description w and even worse results when utilizing the iconic image i. The private latent variables cannot capture any view-specific variations when each class uses the same iconic image and text description for every natural image. A consequence of this is that the iconic image and text description can be identified by only using the shared latent variable z in the decoding phase. The private latent variables are thus not necessary for determining which iconic image or text description to generate, the generated view will be the same anyway for every natural image of a specific class. The model then finds out that the ELBO can be maximized by letting the approximate posteriors be equal to their prior distributions, which minimize the KL divergences of the private latent variables. This phenomenon is referred to as *posterior collapse* [83].

In Figure 13 and 14, we illustrate that the approximate posterior deduces to its prior distribution – a zero-mean Gaussian with unit variance – during training. Figure 13(a) and 13(b) shows the KL divergences over epochs for VCCA-private$_{xw}$ and VCCA-private$_{xwy}$ respectively. The number of words $T = 24$ is the same for both models and we train the models for 500 epochs to emphasize that $D_{\mathrm{KL}}(q_{\phi_w}(u_w|w) \,||\, p(u_w))$ goes towards zero. We believe that this is due to that there



(a) VCCA-private$_{xw}$ (b) VCCA-private$_{xwy}$

Figure 13: The measured KL divergences $D_{\mathrm{KL}}(q_{\phi_z}(z|x)\,||\,p(z))$ (red), $D_{\mathrm{KL}}(q_{\phi_x}(u_x|x)\,||\,p(u_x))$ (green), and $D_{\mathrm{KL}}(q_{\phi_w}(u_w|w)\,||\,p(u_w))$ (blue) over epochs. We increased the number of epochs from 200 to 500 to demonstrate that the KL divergence for the private latent variable u_w goes to zero. The number of words $T = 24$ is the same for both models. The likelihood weights are set to $\lambda_w = 1000$ and $\lambda_y = 100$. Abbreviations: VCCA, Variational Canonical Correlation Analysis; KL, Kullback-Leibler.



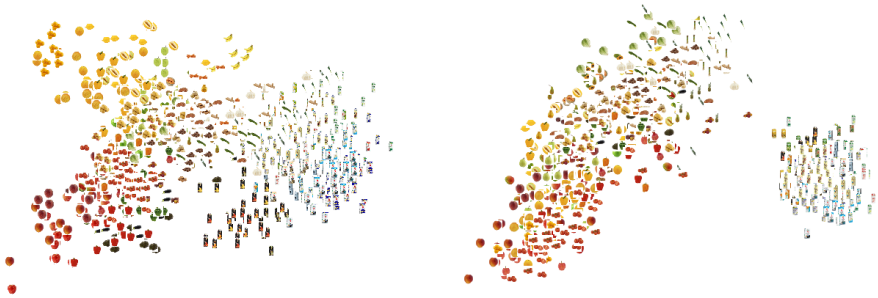
(a) VCCA-private$_{xi}$ (b) VCCA-private$_{xiy}$

Figure 14: The measured KL divergences $D_{\mathrm{KL}}(q_{\phi_z}(z|x)\,||\,p(z))$ (red), $D_{\mathrm{KL}}(q_{\phi_x}(u_x|x)\,||\,p(u_x))$ (green), and $D_{\mathrm{KL}}(q_{\phi_i}(u_i|i)\,||\,p(u_i))$ (blue) over epochs. We increased the number of epochs from 200 to 500 to demonstrate that the KL divergence for the private latent variable u_i goes to zero. The likelihood weights are set to $\lambda_i = 10$ and $\lambda_y = 1000$. Abbreviations: VCCA, Variational Canonical Correlation Analysis; KL, Kullback-Leibler.

are no variations in the text descriptions within one class (see subsection Investigations of the Learned Representations in Results). We perform a similar experiment with VCCA-private$_{xi}$ and VCCA-private$_{xiy}$ and plot their KL divergences in Figure 14. The KL divergence $D_{\mathrm{KL}}(q_{\phi_i}(u_i|w)\,||\,p(u_i))$ decreases to zero faster for these models than when we use the text description. We also observe that the KL divergence $D_{\mathrm{KL}}(q_{\phi_x}(u_x|x)\,||\,p(u_x))$ decreases to zero as well for both models. Why the KL divergence for the private latent variables decreases faster in the models with the iconic image i is mainly because their likelihood weight λ_i is smaller than λ_w. We noticed that the KL divergences of the private latent variables decreases slower towards zero when λ_i is increased, probably because the model foremost focuses on minimizing the reconstruction loss.

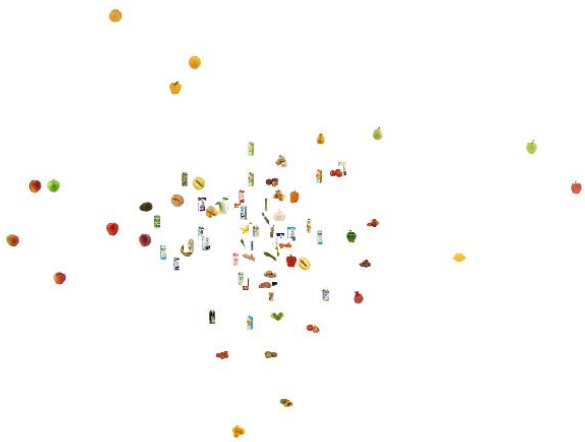
II.3 Investigating Latent Representations in VCCA-private$_{xi}$

Figure 15 shows the shared and private latent spaces of VCCA-private$_{xi}$, as well as the latent space of the standard VCCA$_{xi}$ for comparison. Note that the shared latent spaces in Figure 15(a) and 15(b) have different structures mainly due to the different settings of their likelihood weight λ_i, which is $\lambda_i = 1000$ for VCCA$_{xi}$ and $\lambda_i = 10$ for VCCA-private$_{xi}$. We observe in Figure 15(c) that the natural images are structured based on their similarities in background and camera setup. Across the upper right and the lower left parts of the cluster, we find images of grocery items closely packed together in bins. The middle and upper left part includes images with the hand of the photographer and grey backgrounds, e.g., the floor and shelves in the grocery store. Note that this private latent space is rather densely packed mainly because the KL divergence $D_{\mathrm{KL}}(q_{\phi_x}(u_x|x)\,||\,p(u_x))$ is steadily decreasing towards zero (see Figure 14(a)). This means that the approximate posterior $q_{\phi_x}(u_x|x)$ is collapsing to its prior distribution, which is why the mean values μ_{u_x} are close to the origin of the space. The mean values μ_{u_i} for the private latent variable u_i are shown in Figure 15(d) using the iconic image. Note that every iconic image i is projected at the same location in the latent space. We observe that similar iconic images are projected close to each other. For instance, an orange, grapefruit and yellow bell pepper have been projected in the upper part, packaged items are in the center parts and the left region we find round objects with a dark red color. To the far right, we see a green and a red apple that has been pushed far away from the similar iconic images on the left. If we look closer into these iconic images, we observe that the apples on the right have a higher stalk on top of the apple than the apples on the left side has, which could be the reason why the model has separated them in the latent space. Another interesting observation is that iconic images with multiple items, e.g., tomatoes, kiwis and satsumas, have been projected into the lower region of the space. We conclude that similar iconic images, based on color and their appearance in the image, are grouped closely in the private latent space.



(a) μ_z from VCCA$_{xi}$ (b) μ_z from VCCA-private$_{xi}$

(c) μ_{u_x} from VCCA-private$_{xi}$



(d) μ_{u_i} from VCCA-private$_{xi}$

Figure 15: Visualizations of the latent representations $\mu_z(x)$ from VCCA$_{xi}$ and VCCA-private$_{xi}$ on the first row followed by $\mu_{u_x}(x)$ and $\mu_{u_i}(i)$ for VCCA-private$_{xi}$. Abbreviations: VCCA, Variational Canonical Correlation Analysis.

II.4 Derivation of the ELBO for VCCA

Let $x_{1:M}$ denote the all observed data, i.e., $x_{1:M} = x_1, \ldots, x_M$, for M different views. We derive the ELBO for VCCA by introducing the approximate posterior $q_\phi(z|x_m)$, where x_m is the only view that we use to infer the latent variable z. We now derive the ELBO from the marginal log-likelihood $\log p_\theta(x_{1:M})$ as

$$\begin{aligned}
\log p_\theta(x_{1:M}) &= \log p_\theta(x_{1:M}) \int q_\phi(z|x_m)\, dz \\
&= \int q_\phi(z|x_m) \log p_\theta(x_{1:M})\, dz \\
&= \int q_\phi(z|x_m) \log \frac{p_\theta(x_{1:M}, z)}{p_\theta(z|x_{1:M})}\, dz \\
&= \int q_\phi(z|x_m) \log \frac{p_\theta(x_{1:M}, z)}{p_\theta(z|x_{1:M})} \frac{q_\phi(z|x_m)}{q_\phi(z|x_m)}\, dz \\
&= \int q_\phi(z|x_m) \left(\log \frac{q_\phi(z|x_m)}{p_\theta(z|x_{1:M})} + \log \frac{p_\theta(x_{1:M}, z)}{q_\phi(z|x_m)} \right) dz \\
&= D_{\mathrm{KL}}(q_\phi(z|x_m)\,||\,p_\theta(z|x_{1:M})) + \mathbb{E}_{q_\phi(z|x_m)} \left[\log \frac{p_\theta(x_{1:M}, z)}{q_\phi(z|x_m)} \right] \\
&\geq \mathbb{E}_{q_\phi(z|x_m)} \left[\log \frac{p_\theta(x_{1:M}, z)}{q_\phi(z|x_m)} \right] = \mathcal{L}(x_{1:M}; \theta, \phi)
\end{aligned}$$

We use the factorization property of the joint distribution $p_\theta(x_{1:M})$ to further derive the ELBO $\mathcal{L}(x_{1:M}; \theta, \phi)$:

$$\begin{aligned}
\mathcal{L}(\theta, \phi; x_{1:M}) =& \mathbb{E}_{q_\phi(z|x_m)} \left[\log \frac{p_{\theta_1}(x_1|z) \cdots p_{\theta_M}(x_M|z) p(z)}{q_\phi(z|x_m)} \right] \\
=& \mathbb{E}_{q_\phi(z|x_m)} \left[\log p_{\theta_1}(x_1|z) + ... + \log p_{\theta_M}(x_M|z) + \log \frac{p(z)}{q_\phi(z|x_m)} \right] \\
=& \mathbb{E}_{q_\phi(z|x_m)} \left[\log p_{\theta_1}(x_1|z) \right] + ... + \mathbb{E}_{q_\phi(z|x_m)} \left[\log p_{\theta_M}(x_M|z) \right] \\
& - D_{\mathrm{KL}}(q_\phi(z|x_m)\,||\,p(z))
\end{aligned}$$

It may be necessary to balance the terms in the ELBO with some constant for every term, especially when the dimensions and magnitudes differ between the modalities. Thus, we introduce the likelihood weights $\lambda_1, ..., \lambda_M$, such that the ELBO will be written as

$$\begin{aligned}
\mathcal{L}(\theta, \phi; x_{1:M}) =& \lambda_1 \mathbb{E}_{q_\phi(z|x_m)} \left[\log p_{\theta_1}(x_1|z) \right] + ... + \lambda_M \mathbb{E}_{q_\phi(z|x_m)} \left[\log p_{\theta_M}(x_M|z) \right] \\
& - D_{\mathrm{KL}}(q_\phi(z|x_m)\,||\,p(z))
\end{aligned}$$

The likelihood weights $\lambda_1, ..., \lambda_M$ that are optimal for the task at hand can be found with some hyperparameter search.

Acknowledgments

This research is funded by the Promobilia foundation and the Swedish e-Science Research Centre.

References

[1] Microsoft Seeing AI app. https://www.microsoft.com/en-us/seeing-ai/. Accessed on 2020-04-13.

[2] Aipoly Vision app. https://www.aipoly.com/. Accessed on 2020-04-13.

[3] Orcam. https://www.orcam.com/en/. Accessed on 2020-04-13.

[4] Transsense. https://www.transsense.ai/. Accessed on 2020-04-13.

[5] 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.

[6] Marcus Klasson, Cheng Zhang, and Hedvig Kjellström. A hierarchical grocery store image dataset with visual and semantic labels. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 491–500. IEEE, 2019.

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

[8] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[9] 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.

[10] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[11] Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2891–2903, 2013.

[12] 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.

[13] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297, 2016.

[14] Devi Parikh and Kristen Grauman. Relative attributes. In *2011 International Conference on Computer Vision*, pages 503–510. IEEE, 2011.

[15] Nitish Srivastava and Ruslan Salakhutdinov. Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research*, 15:2949–2980, 2014.

[16] Püren Güler, Yasemin Bekiroglu, Xavi Gratal, Karl Pauwels, and Danica Kragic. What's in the container? classifying object contents from vision and touch. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3961–3968. IEEE, 2014.

[17] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th International Conference on Machine Learning*, pages 689–696, 2011.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016.

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

[25] 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.

[26] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[27] 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.

[28] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[29] 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.

[30] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

[31] 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.

[32] 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.

[33] Edward Hsiao, Alvaro Collet, and Martial Hebert. Making specific features less discriminative to improve point-based 3d object recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2653–2660. IEEE, 2010.

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

[35] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE International Conference on Robotics and Automation*, pages 1817–1824. IEEE, 2011.

[36] 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.

[37] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 509–516. IEEE, 2014.

[38] 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.

[39] 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.

[40] Tess Winlock, Eric Christiansen, and Serge Belongie. Toward real-time grocery detection for the visually impaired. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 49–56. IEEE, 2010.

[41] Hao Yu, Fu Yanwei, and Jiang Yu-Gang. Take goods from shelves: A dataset for class-incremental object detection. In *ACM International Conference on Multimedia Retrieval (ICMR '19)*, 2019.

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

[43] Škrjanec Marko. Automatic fruit recognition using computer vision. (Mentor: Matej Kristan), Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2013.

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

[45] 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.

[46] Weiqing Min, Shuqiang Jiang, Linhu Liu, Yong Rui, and Ramesh Jain. A survey on food computing. *ACM Computing Surveys (CSUR)*, 52(5):1–36, 2019.

[47] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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 Transations on Pattern Analisys and Machince Intelligence*, 2019.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] Chris Cremer and Nate Kushman. On the importance of learning aggregate posteriors in multimodal variational autoencoders. *1st Symposium on Advances in Approximate Bayesian Inference*, 2018.

[59] Yanwei Fu and Leonid Sigal. Semi-supervised vocabulary-informed learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5337–5346, 2016.

[60] Alessandro Pieropan, Giampiero Salvi, Karl Pauwels, and Hedvig Kjellström. Audio-visual classification and detection of human manipulation actions. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3045–3052. IEEE, 2014.

[61] 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, 2010.

[62] Yuge Shi, N Siddharth, Brooks Paige, and Philip Torr. Variational mixture-of-experts autoencoders for multi-modal deep generative models. In *Advances in Neural Information Processing Systems*, pages 15692–15703, 2019.

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

[64] 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.

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

[66] Mike Wu and Noah Goodman. Multimodal generative models for scalable weakly-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5575–5585, 2018.

[67] 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.

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

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

[70] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

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

[72] Shotaro Akaho. A kernel method for canonical correlation analysis. *arXiv preprint cs/0609071*, 2006.

[73] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International Conference on Machine Learning*, pages 1247–1255, 2013.

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

[75] Cheng Zhang, Judith Butepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

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

[77] 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.

[78] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[79] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[80] Shiyong Cui and Mihai Datcu. Comparison of kullback-leibler divergence approximation methods between gaussian mixture models for satellite image retrieval. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 3719–3722. IEEE, 2015.

[81] Jacob Goldberger, Shiri Gordon, and Hayit Greenspan. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In *IEEE International Conference on Computer Vision*, 2003.

[82] John R Hershey and Peder A Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 4, pages IV–317. IEEE, 2007.

[83] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[84] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(12), 2010.

[85] 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.

[86] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[87] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

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

[89] 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.

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

[91] Yingzhen Li, John Bradshaw, and Yash Sharma. Are generative classifiers more robust to adversarial attacks? In *International Conference on Machine Learning*, pages 3804–3814, 2019.

[92] Chao Ma, Sebastian Tschiatschek, Konstantina Palla, José Miguel Hernández-Lobato, Sebastian Nowozin, and Cheng Zhang. Eddi: Efficient dynamic discovery of high-value information with partial vae. *arXiv preprint arXiv:1809.11142*, 2018.

[93] 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.

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

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

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

[97] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

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
‡Silo AI, Stockholm, Sweden
†Microsoft Research, Cambridge, United Kingdom

Abstract

Replay-based continual learning have shown to be successful in mitigating catastrophic forgetting despite having limited access to historical data. However, storing historical data is cheap in many real-world applications, yet replaying all seen data would be prohibited due to processing time constraints. In such settings, we propose learning the time to learn for a continual learning system, in which we learn replay schedules over which tasks to replay at different time steps. To demonstrate the importance of learning the time to learn, we use Monte Carlo tree search in an ideal continual learning scenario to find the proper replay schedule. We perform extensive evaluations to show the benefits of replay scheduling in various memory settings and in combination with different replay methods. Moreover, the results indicate that the found schedules are consistent with human learning insights. Our findings opens up for new research directions that can bring current continual learning research closer to real-world needs.

1 Introduction

Many organizations deploying machine learning systems receive large volumes of data daily [1,2]. 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 often need to continuously adapt to new tasks while retaining the previously learned abilities. Continual learning (CL) methods [3,38] address this challenge where, in particular, replay methods [4,5] have shown to be very effective in achieving great prediction performance. Replay 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, replay memories

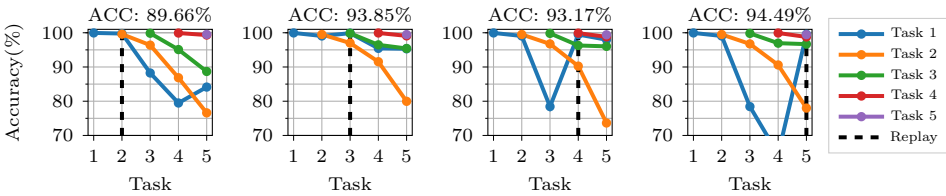


Figure 1: Task accuracies on Split MNIST [14] when replaying only 10 samples of classes 0/1 at a single time step. The black vertical line indicates when replay is used. ACC denotes the average accuracy over all tasks after learning Task 5. Results are averaged over 5 seeds. These results show that the time to replay the previous task is critical for the final performance.

are 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.

Recent research on replay-based CL has focused on the quality of memory samples [4, 6–11] or data compression to increase the memory capacity [5, 12, 13]. Most previous methods allocate equal memory storage space for samples from old tasks, and replay the whole memory 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, uniform selection policy of samples to revisit is commonly used which ignores the time of which tasks to learn again. This stands in contrast to human learning where education methods focus on scheduling of learning and rehearsal of previous learned knowledge. For example, spaced repetition [67, 68, 70], where the time interval between rehearsal increases, has been shown to enhance memory retention.

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 [14] 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 [15]. To show the importance of replay scheduling, we take an episodic-learning approach where a policy is learned from multiple trials selecting which tasks to replay in a CL scenario. In particular, we illustrate in single CL environments by using Monte Carlo tree search (MCTS) [16] as an example method that searches for good replay schedules. The replay schedules from MCTS are evaluated by measuring the final performance of a network trained on a sequence of CL tasks where the scheduled replay samples have been used for mitigating catastrophic forgetting. We use this way to show the importance of replay scheduling given an ideal environment to highlight the need for learning replay schedules in real-world large scale CL tasks. 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 where we learn the time of which tasks to replay (Section 3.2).
- We argue that learning the time to learn is essential for CL performance. We use MCTS as an example method to illustrate the benefits of replay scheduling in CL, where MCTS searches over finite sets of replay memory compositions 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 4.1 and 4.4). Moreover, we show that replay scheduling can be combined with any memory selection technique and replay-based method (Section 4.3 and 4.5), as well as being efficient in situations where the memory size is smaller than the number of classes (Section 4.6).

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 [9,14,17–22]. Architecture-based methods isolate task-specific parameters by either increasing network capacity [23–25] or freezing parts of the network [26,27] 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 [4,5,28–32] or generated using a generative model [33,34]. Regularization-based approaches and dynamic architectures have been combined with replay-based approaches to methods to overcome their limitations [9,13,18,35–44]. 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 [4,6–11,29,31,45]. Cahaudrhy et al. [4] reviews several selection strategies in scenarios with tiny memory capacity, e.g., reservoir sampling [46], first-in first-out buffer [31], k-Means, and Mean-of-Features [10]. However, more elaborate selection strategies have been shown to give little benefit over random selection for image classification problems [5,18]. More recently, there has been work on compressing raw images to feature representations to increase the number of memory examples for replay [5,12,13]. Selecting the time to replay old tasks has mostly been ignored in the literature, with an exception in [28] which replays memory samples that would most interfere with a foreseen parameter update.

Our replay scheduling approach differs from the above mentioned works since we focus on learning to select which tasks to replay. Nevertheless, our scheduling can be combined with any selection strategy and replay 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 [47]. Different learning techniques have been suggested for humans to memorize better [48, 49]. An example is spaced repetition where time intervals between rehearsal are gradually increased to improve long-term memory retention [15], where the earliest documented works are from Ebbinghaus [50]. Further studies have shown that memory training schedules with adjusted spaced repetition are better at preserving memory than using uniformly spaced rehearsal times [51,52]. Several works in CL with neural networks are inspired by human learning techniques, including spaced repetition [53–55], sleep mechanisms [22,26,56], and memory reactivation [5,57]. 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 of CL where historical data are available while the processing time is limited when learning new tasks. In Section 3.1 and 3.2, we present the considered problem setting, as well as our idea of learning schedules over which tasks to replay at different time steps to mitigate catastrophic forgetting. We use MCTS [16] in an ideal CL environment that searches for good replay schedules to show the importance of replay scheduling in CL (Section 3.3).

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 often prohibited from replaying all historical data due to processing time constraints. Therefore, the goal is to determine which historical tasks to revisit and sample a small replay memory from the selected tasks to mitigate catastrophic forgetting as efficiently as possible.

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 $\mathcal{D}_1, \ldots, \mathcal{D}_T$ 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. Next, we describe our method for constructing this replay memory.

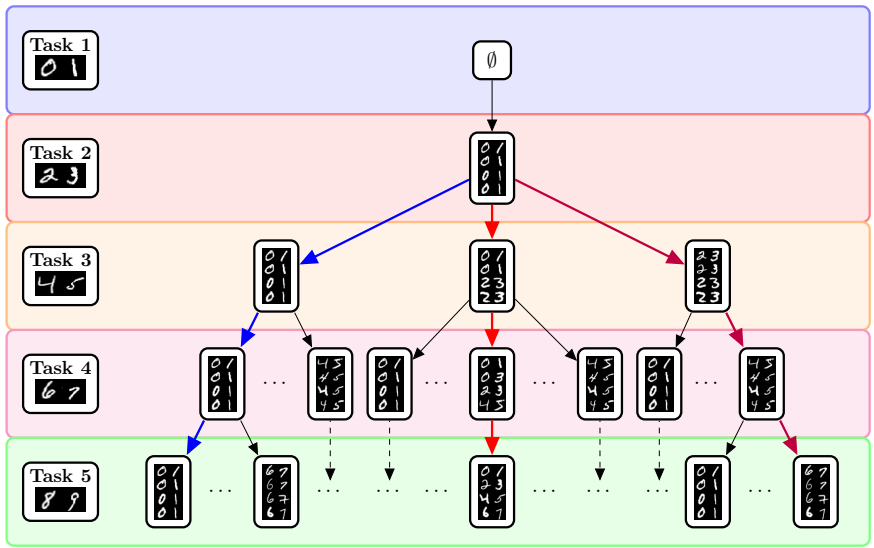


Figure 2: Tree-shaped action space of possible replay memory compositions with size $M = 8$ at every task from the discretization method described in Section **??** for Split MNIST. A replay schedule is represented as a path traversal of different replay memory compositions from task 1 to task 5.

We assume that historical data from old tasks are accessible at any time step t. However, due to processing time constraints, we can only fill a small replay memory $\mathcal{M}$ with M historical samples for replay. The challenge then becomes how to select the M replay samples to efficiently retain knowledge of old tasks. We focus on selecting the samples on task-level by deciding on the task proportion $(p_1, \ldots, p_{t-1})$ of samples to fetch from each task, where $p_i \geq 0$ is the proportion of M samples from task i to place in $\mathcal{M}$ and $\sum_{i=1}^{t-1} p_i = 1$. To simplify the selection of which tasks to replay, we construct a discrete set of possible task proportions that can be used for constructing $\mathcal{M}$.

3.2 Replay Scheduling in Continual Learning

In this section, we describe our setup for enabling the scheduling for selecting replay memories at different time steps. We define a replay schedule as a sequence $S = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{T-1})$, where the task proportions $\boldsymbol{a}_i = (a_1, \ldots, a_{T-1})$ for $1 \leq i \leq T-1$ are used for determining how many samples from seen tasks with which to fill the replay memory at task i. We construct an action space with a discrete number of choices of task proportions that can be selected at each task: At task t, we have $t-1$ historical tasks that we can choose samples from. We create $t-1$ bins $\boldsymbol{b}_t = [b_1, \ldots, b_{t-1}]$ and sample a task index for each bin $b_i \in \{1, \ldots, t-1\}$. The bins are treated as interchangeable and we only 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. We count the number of occurrences of each task index in $\boldsymbol{b}_t$ and divide by $t-1$ to obtain the task proportion, i.e., $\boldsymbol{a}_t = \texttt{bincount}(\boldsymbol{b}_t)/(t-1)$. We round the number of replay samples from task

i, i.e., $a_i \cdot M$, up or down accordingly to keep the memory size M fixed when filling the memory. From this specification, we can build a tree of different replay schedules to evaluate with the network. We outline the steps for creating the action space in Algorithm 1 (see Appendix V).

Figure 2 shows an example of a replay schedule tree with Split MNIST [14] where the memory size is $M = 8$. Each level corresponds to a task to learn and we show some examples of possible replay memories in the tree that can be evaluated at each task. A replay schedule is represented as a path traversal of different replay memory compositions from task 1 to task 5. At task 1, the memory $\mathcal{M}_1 = \emptyset$ is empty, while $\mathcal{M}_2$ is filled with samples from task 1 at task 2. The memory $\mathcal{M}_3$ can be composed with samples from either task 1 or 2, or equally fill $\mathcal{M}_3$ with samples from both tasks. All possible paths in the tree are valid replay schedules. We show three examples of possible schedules in Figure 2 for illustration: the blue path represents a replay schedule where only task 1 samples are replayed. The red path represents using memories with equally distributed tasks, and the purple path represents a schedule where the memory is only filled with samples from the most previous task.

3.3 Monte Carlo Tree Search for Replay Schedules

In this section, we describe how we enable using MCTS for studying the benefits of replay scheduling in CL scenarios. The tree-shaped action space of task proportions described in Section 3.2 grows fast with the number of tasks, which complicates studying replay scheduling in datasets with longer task-horizons. Since the search space is too big for using exhaustive searches, we need a scalable method that enables tree searches in large action spaces. To this end, we propose to use MCTS since it has been successful in applications with large action spaces [58–61]. In our case, MCTS concentrates the search for replay schedules in directions with promising CL performance in the environment. We use MCTS in an ideal CL setting, wherein multiple episodes are allowed, for demonstration purposes to show that replay scheduling can be critical for the CL performance.

Each memory composition in the action space corresponds to a node that can be visited by MCTS. For example, in Figure 2, the nodes correspond to the possible memory examples which can be visited during the MCTS rollouts. At task t, the node v_t is related to a task proportions $\boldsymbol{a}_t$ used for retrieving a replay memory from the historical data. We store the related task proportion $\boldsymbol{a}_t$ from every visited node v_t in the replay schedule S. The final replay schedule is then used for constructing the replay memories at each task during the CL training. Next, we briefly outline the MCTS steps for performing the replay schedule search (more details in Appendix V):

Selection. During a rollout, the current node v_t either moves randomly to unvisited children, or selects the next node by evaluating the Upper Confidence Tree (UCT) [62] if all children has been visited earlier. The child v_{t+1} with the highest UCT score is selected using the function from [255]:

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

where $q(\cdot)$ is the reward function, C the exploration constant, and $n(\cdot)$ the number of node visits.

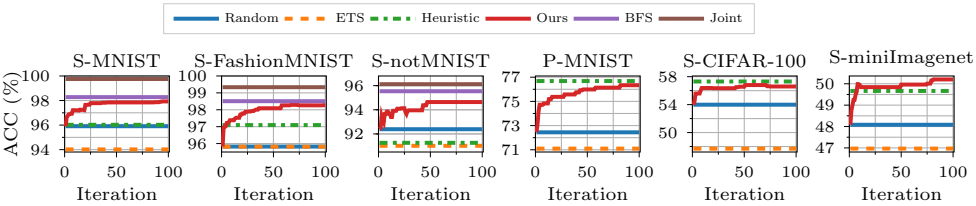


Figure 3: Average test accuracies over tasks after learning the final task (ACC) over the MCTS simulations for all datasets, where 'S' and 'P' are used as short for 'Split' and 'Permuted'. We compare performance for RS-MCTS (Ours) against random replay schedules (Random), Equal Task Schedule (ETS), and Heuristic Scheduling (Heuristic) baselines. For the first three datasets, we show the ACC for for training on all seen task datasets jointly (Joint), as well as the best ACC found from a breadth-first search (BFS) as two upper bounds. 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.

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 expansion, the succeeding nodes are selected randomly until reaching a terminal node v_T. The task proportions from the visited nodes in the rollout constitutes the replay schedule S. After training the network using S for replay, we calculate the reward for the rollout is given by $r = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}^{(val)}$, 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.

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 I and additional results are in Appendix III.

Datasets. We conduct experiments on six datasets commonly used as benchmarks in the CL literature: Split MNIST [14,63], Fashion-MNIST [64], Split notMNIST [65], Permuted MNIST [66], Split CIFAR-100 [67], and Split miniImagenet [68]. 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,

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. We provide the metrics for training on all seen task datasets jointly (Joint) as an upper bound, as well as include the results from a breadth-first search (BFS) with Uniform memory selection for the 5-task 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-miniImagenet
–	Joint	99.75 (± 0.06)	99.34 (± 0.08)	96.12 (± 0.57)	95.34 (± 0.13)	84.73 (± 0.81)	74.03 (± 0.83)
Uniform	BFS	98.28 (± 0.49)	98.51 (± 0.23)	95.54 (± 0.67)	–	–	–
Uniform	Random	95.91 (± 1.56)	95.82 (± 1.45)	92.39 (± 1.29)	72.44 (± 1.15)	53.99 (± 0.51)	48.08 (± 1.36)
	ETS	94.02 (± 4.25)	95.81 (± 3.53)	91.01 (± 1.39)	71.09 (± 2.31)	47.70 (± 2.16)	46.97 (± 1.24)
	Heuristic	96.02 (± 2.32)	97.09 (± 0.62)	91.26 (± 3.99)	**76.68** (± 2.13)	**57.31** (± 1.21)	49.66 (± 1.10)
	Ours	**97.93** (± 0.56)	**98.27** (± 0.17)	**94.64** (± 0.39)	76.34 (± 0.98)	56.60 (± 1.13)	**50.20** (± 0.72)
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 (± 2.70)
	ETS	92.89 (± 3.53)	96.47 (± 0.85)	**93.80** (± 0.82)	69.40 (± 1.32)	47.51 (± 1.14)	45.82 (± 0.92)
	Heuristic	96.28 (± 1.68)	95.78 (± 1.50)	91.75 (± 0.94)	75.57 (± 1.18)	54.31 (± 3.94)	49.25 (± 1.00)
	Ours	**98.20** (± 0.16)	**98.48** (± 0.26)	93.61 (± 0.71)	**77.74** (± 0.80)	**56.95** (± 0.92)	**50.47** (± 0.85)
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 (± 0.96)
	ETS	94.84 (± 1.40)	97.28 (± 0.50)	91.08 (± 2.48)	69.11 (± 1.69)	44.13 (± 1.06)	41.35 (± 1.23)
	Heuristic	94.55 (± 2.79)	94.08 (± 3.72)	92.06 (± 1.20)	74.33 (± 2.00)	50.32 (± 1.97)	44.13 (± 0.95)
	Ours	**98.24** (± 0.36)	**98.06** (± 0.35)	**94.26** (± 0.37)	**76.55** (± 1.16)	**51.37** (± 1.63)	**46.76** (± 0.96)
MoF	Random	95.18 (± 3.18)	95.76 (± 1.41)	91.33 (± 1.75)	77.96 (± 1.84)	61.93 (± 1.05)	54.50 (± 1.33)
	ETS	97.04 (± 1.23)	96.48 (± 1.33)	92.64 (± 0.87)	77.62 (± 1.12)	60.43 (± 1.17)	56.12 (± 1.12)
	Heuristic	96.46 (± 2.41)	95.84 (± 0.89)	93.24 (± 0.77)	77.27 (± 1.45)	55.60 (± 2.70)	52.30 (± 0.59)
	Ours	**98.37** (± 0.24)	**97.84** (± 0.32)	**94.62** (± 0.42)	**81.58** (± 0.75)	**64.22** (± 0.65)	**57.70** (± 0.51)

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 II 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 [22,68]. For Split miniImagenet, we apply the reduced ResNet-18 from [31].

Evaluation Metric. We use the average test accuracy over all tasks after learning the final task, i.e., ACC $= \frac{1}{T}\sum_{i=1}^{T} A_{T,i}^{(test)}$ where $A_{T,i}^{(test)}$ 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. Furthermore, we include the ACC achieved by training on all seen datasets jointly at every task (Joint) for the 5-task datasets. 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. Table 1 at the rows for Uniform selection shows the ACC for each method in 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 spaced 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 visualization for Split MNIST in Figure 7 in Appendix III.1 to bring more insights to the benefits of replay scheduling.

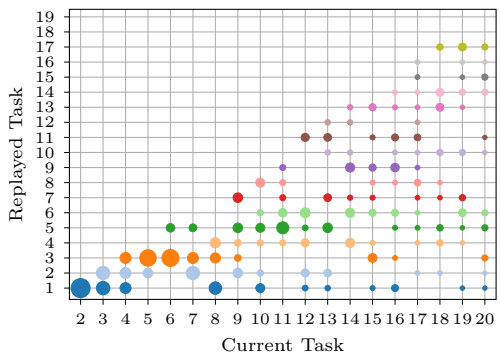


Figure 4: Replay schedule learned from Split CIFAR-100 visualized as a bubble plot. The task proportions vary dynamically over time which would be hard to replace by a heuristic method.

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 [9], and Mean-of-Features (MoF) [10]. 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

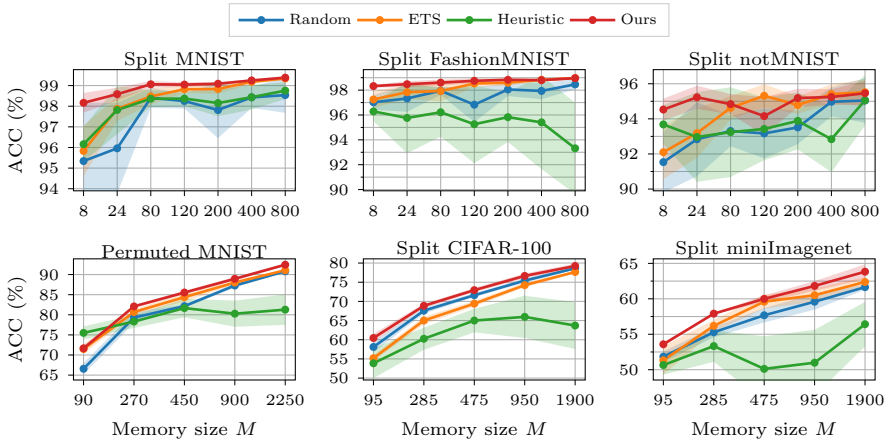


Figure 5: Average test accuracies over tasks after learning the final task (ACC) over different replay memory sizes M for the RS-MCTS (Ours) and the Random, ETS, and Heuristic baselines on all datasets. All results have been averaged over 5 seeds. The results show that replay scheduling can outperform 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.

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 Varying 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. These results show that replay scheduling can outperform the baselines, especially for small M, on both small and large datasets across different backbone choices.

4.5 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) [37], Meta-Experience Replay (MER) [69], Dark Experience Replay (DER) [35], and DER++. We provide the hyperparameter settings in Appendix III.3. Table 2 shows the performance comparison between our proposed replay scheduling against using Random, ETS, and Heuristic schedules for each method. The results confirm that replay scheduling is important for the final

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-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)
DER++	Random	97.90 (± 0.52)	97.10 (± 1.03)	93.29 (± 1.43)	87.89 (± 1.10)	58.49 (± 1.44)	48.40 (± 0.69)
	ETS	97.98 (± 0.52)	98.12 (± 0.40)	94.53 (± 1.02)	85.25 (± 0.88)	52.54 (± 1.06)	41.36 (± 2.90)
	Heuristic	92.35 (± 2.42)	*67.31 (± 21.20)	93.88 (± 1.33)	79.17 (± 2.44)	56.70 (± 1.27)	45.73 (± 0.84)
	Ours	**98.84** (± 0.21)	**98.38** (± 0.43)	**94.73** (± 0.20)	**89.84** (± 0.22)	**59.23** (± 0.83)	**49.45** (± 0.68)

performance given the same memory constraints and it can benefit any existing CL framework.

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 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 [36], ER-Ring [4] 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 IV shows the memory usage for the other datasets.

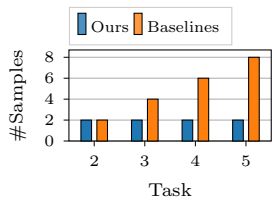


Figure 6: Number of replayed samples per task for the 5-task datasets in the tiny memory setting. Ours use $M = 2$ samples for replay, while the baselines increment their memory per task.

Table 3 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 indicate that replay scheduling is an important research direction in CL as storing 1 sample/class in the memory could be inefficient in settings with large number of tasks.

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			**10- and 20-task Datasets**		
Method	S-MNIST	S-FashionMNIST	S-notMNIST	P-MNIST	S-CIFAR-100	S-miniImagenet
Random	92.56 ± 2.90	92.70 ± 3.78	89.53 ± 3.96	70.02 ± 1.76	48.62 ± 1.02	48.85 ± 1.38
A-GEM	94.97 ± 1.50	94.81 ± 0.86	92.27 ± 1.16	64.71 ± 1.78	42.22 ± 2.13	32.06 ± 1.83
ER-Ring	94.94 ± 1.56	95.83 ± 2.15	91.10 ± 1.89	69.73 ± 1.13	53.93 ± 1.13	49.82 ± 1.69
Uniform	95.77 ± 1.12	97.12 ± 1.57	92.14 ± 1.45	69.85 ± 1.01	52.63 ± 1.62	50.56 ± 1.07
RS-MCTS (Ours)	96.07 ± 1.60	97.17 ± 0.78	93.41 ± 1.11	72.52 ± 0.54	51.50 ± 1.19	50.70 ± 0.54

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 CL inspired by human learning techniques. We demonstrated the benefits with replay scheduling by showing the performance improvements with schedules found using MCTS on several CL benchmarks when comparing against methods with fixed scheduling policies under the same memory budgets. Furthermore, the dynamic behavior of the replay schedules showed similarities to human learning techniques, such as spaced repetition, by replaying previous tasks with varying time intervals. We also 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. The proposed problem setting brings CL research closer to real-world needs, especially in scenarios where CL is applied under limited processing times but with rich amounts of historical data available for replay.

Limitations and Future Work. In this work, we assumed that multiple episodes are allowed in the CL environments to use MCTS. However, this is prohibited in the CL setting since tasks can never be fully revisited. Hence, it would be useful if we could learn replay scheduling policies that generalize to new CL scenarios for mitigating catastrophic forgetting. Using MCTS for searching after such policies can be inefficient as the algorithm needs to run 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. This requires defining the state using various task performance metrics such as accuracies and forgetting metrics, the action either in the same space as in this work or as in continuous space with fractions of different tasks in the memory, and the reward as final or accumulated CL performance. Finally, we would like to explore choosing memory samples on an instance level as the current work selects samples on task level. This would also require a policy learning method that scales to large action spaces which is a research challenge by itself.

Appendix

This supplementary material is structured as follows:

- Appendix I: Full details of the experimental settings.

- Appendix V: Pseudocode of RS-MCTS in Algorithm 2 to provide more details about our method.
- Appendix II describes the Heuristic scheduling baseline.
- Appendix III shows additional experimental results.
- Appendix IV includes additional figures.

I 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 [14] is a variant of the MNIST [63] 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 [64] 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 [65] 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 [39] for Split notMNIST. Permuted MNIST [66] 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 [67], the 100 classes are divided into 20 tasks with 5 classes for each task [10,31]. Similarly, Split miniImagenet [68] 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 [17, 22, 68], which consists of four 3x3 convolutional blocks, i.e. convolutional layer followed by batch normalization [70], with 64 filters, ReLU activations, and 2x2 Max-pooling. For Split mniImagenet, we use the reduced ResNet-18 from [31] with multi-head output layer.

Hyperparameters. We train all networks with the Adam optimizer [71] 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 [59].

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. [7] for the memory selection strategies Uniform sampling, k-means clustering, k-center clustering [9], and Mean-of-Features [10]. For HAL [37], MER [69], DER [35], and DER++, we follow the implementations released by [35] 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 [36] and ER-Ring [4]. 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.

II 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 I. 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: τ = [0.9, 0.93, 0.95, **0.96**, 0.97, 0.98, 0.99]
- Split FashionMNIST: τ = [0.9, 0.93, 0.95, 0.96, **0.97**, 0.98, 0.99]
- Split notMNIST: τ = [0.9, 0.93, 0.95, 0.96, 0.97, **0.98**, 0.99]
- Permuted MNIST: τ = [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: τ = [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: τ = [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.

III Additional Experimental Results

In this section, we bring more insights to the benefits of replay scheduling in Section III.1 as well as provide metrics for catastrophic forgetting in Section III.2.

III.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

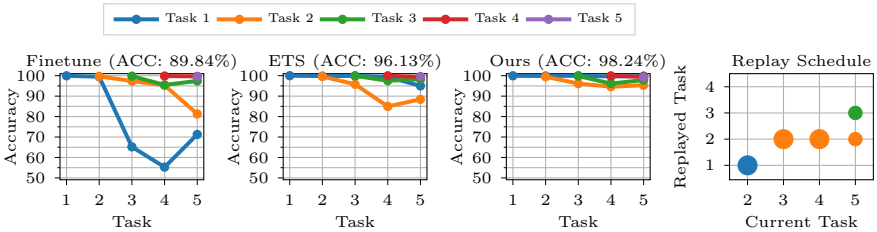


Figure 7: Comparison of test classification accuracies for Task 1-5 on Split MNIST from a network trained without replay (Finetune), ETS, and RS-MCTS (Ours). The ACC metric for each method is shown on top of each figure. We also visualize the replay schedule found by RS-MCTS as a bubble plot to the right.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

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.

III.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 [31], 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 4.3. 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 7 shows the ACC and BWT metrics for the experiments in Section 4.6, 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.

III.3 Applying Scheduling to Recent Replay Methods

In Section 4.5, 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) [37], Meta Experience Replay (MER) [69], and Dark Experience Replay (DER) [35]. 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.

We present the hyperparameters used for each method in Table 5. The hyperparameters for each method are denoted as

- **HAL.** η: learning rate, λ: regularization, γ: mean embedding strength, β: decay rate, k: gradient steps on anchors
- **MER.** γ: across batch meta-learning rate, β: within batch meta-learning rate
- **DER.** α: loss coefficient for memory logits
- **DER++.** α: loss coefficient for memory logits, β: loss coefficient for memory labels

We used the same architectures and hyperparameters as described in Appendix I for all datasets, except for the optimizer in HAL where we used the SGD optimizer since using Adam made the model diverge in our experiments. We used the Adam optimizer with learning rate $\eta = 0.001$ for MER, DER, and DER++.

IV 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 4.6. 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.

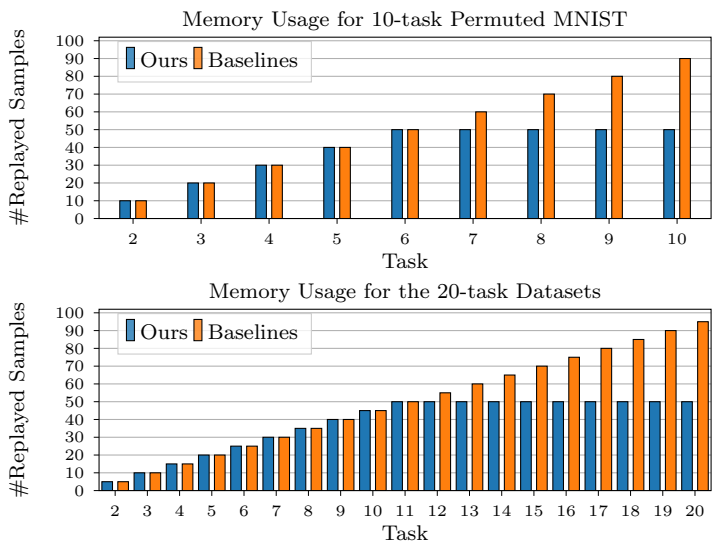


Figure 8: Number of replayed samples per task for 10-task Permuted MNIST (top) and the 20-task datasets in the experiment in Section 4.6. The fixed memory size $M = 50$ for our method is reached after learning task 6 and task 11 on the Permuted MNIST and the 20-task datasets respectively, while the baselines continue incrementing their number of replay samples per task.

Table 4: Performance comparison with ACC and BWT metrics for all datasets between MCTS (Ours) and the baselines with various memory selection methods. We provide the metrics for training on all seen task datasets jointly (Joint) as an upper bound. Furthermore, we include the results from a breadth-first search (BFS) with Uniform memory selection for the 5-task datasets. 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. 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 notMNIST**	
Selection	**Method**	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑
–	Joint	99.75 (± 0.06)	0.01 (± 0.06)	99.34 (± 0.08)	-0.01 (± 0.14)	96.12 (± 0.57)	-0.21 (± 0.71)
Uniform	BFS	98.28 (± 0.49)	-1.84 (± 0.63)	98.51 (± 0.23)	-1.03 (± 0.28)	95.54 (± 0.67)	-1.04 (± 0.87)
Uniform	Random	95.91 (± 1.56)	-4.79 (± 1.95)	95.82 (± 1.45)	-4.35 (± 1.79)	92.39 (± 1.29)	-4.56 (± 1.29)
	ETS	94.02 (± 4.25)	-7.22 (± 5.33)	95.81 (± 3.53)	-4.45 (± 4.34)	91.01 (± 1.39)	-6.16 (± 1.82)
	Heuristic	96.02 (± 2.32)	-4.64 (± 2.90)	97.09 (± 0.62)	-2.82 (± 0.84)	91.26 (± 3.99)	-6.06 (± 4.70)
	Ours	97.93 (± 0.56)	-2.27 (± 0.71)	98.27 (± 0.17)	-1.29 (± 0.20)	94.64 (± 0.39)	-1.47 (± 0.79)
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 (± 1.77)
	ETS	92.89 (± 3.53)	-8.66 (± 4.42)	96.47 (± 0.85)	-3.55 (± 1.07)	93.80 (± 0.82)	-2.84 (± 0.81)
	Heuristic	96.28 (± 1.68)	-4.32 (± 2.11)	95.78 (± 1.50)	-4.46 (± 1.87)	91.75 (± 0.94)	-5.60 (± 2.07)
	Ours	98.20 (± 0.16)	-1.94 (± 0.22)	98.48 (± 0.26)	-1.04 (± 0.31)	93.61 (± 0.71)	-3.11 (± 0.55)
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 (± 2.37)
	ETS	94.84 (± 1.40)	-6.20 (± 1.77)	97.28 (± 0.50)	-2.58 (± 0.66)	91.08 (± 2.48)	-6.39 (± 3.46)
	Heuristic	94.55 (± 2.79)	-6.47 (± 3.50)	94.08 (± 3.72)	-6.59 (± 4.57)	92.06 (± 1.20)	-4.70 (± 2.09)
	Ours	98.24 (± 0.36)	-1.93 (± 0.44)	98.06 (± 0.35)	-1.59 (± 0.45)	94.26 (± 0.37)	-1.97 (± 1.02)
MoF	Random	95.18 (± 3.18)	-5.73 (± 3.95)	95.76 (± 1.41)	-4.41 (± 1.75)	91.33 (± 1.75)	-5.94 (± 1.92)
	ETS	97.04 (± 1.23)	-3.46 (± 1.50)	96.48 (± 1.33)	-3.55 (± 1.73)	92.64 (± 0.87)	-4.57 (± 1.59)
	Heuristic	96.46 (± 2.41)	-4.09 (± 3.01)	95.84 (± 0.89)	-4.39 (± 1.15)	93.24 (± 0.77)	-3.48 (± 1.37)
	Ours	98.37 (± 0.24)	-1.70 (± 0.28)	97.84 (± 0.32)	-1.81 (± 0.39)	94.62 (± 0.42)	-1.80 (± 0.56)

		Permuted MNIST		**Split CIFAR-100**		**Split miniImagenet**	
Selection	**Method**	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑
–	Joint	95.34 (± 0.13)	0.17 (± 0.18)	84.73 (± 0.81)	-1.06 (± 0.81)	74.03 (± 0.83)	9.70 (± 0.68)
Uniform	Random	72.44 (± 1.15)	-25.56 (± 1.39)	53.99 (± 0.51)	-34.19 (± 0.66)	48.08 (± 1.36)	-15.98 (± 1.08)
	ETS	71.09 (± 2.31)	-27.39 (± 2.59)	47.70 (± 2.16)	-41.68 (± 2.37)	46.97 (± 1.24)	-18.32 (± 1.34)
	Heuristic	76.68 (± 2.13)	-20.82 (± 2.41)	57.31 (± 1.21)	-30.76 (± 1.45)	49.66 (± 1.10)	-12.04 (± 0.59)
	Ours	76.34 (± 0.98)	-21.21 (± 1.16)	56.60 (± 1.13)	-31.39 (± 1.11)	50.20 (± 0.72)	-13.46 (± 1.22)
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 (± 2.71)
	ETS	69.40 (± 1.32)	-29.23 (± 1.47)	47.51 (± 1.14)	-41.77 (± 1.30)	45.82 (± 0.92)	-19.53 (± 1.10)
	Heuristic	75.57 (± 1.18)	-22.11 (± 1.22)	54.31 (± 3.94)	-33.80 (± 4.24)	49.25 (± 1.00)	-12.92 (± 1.22)
	Ours	77.74 (± 0.80)	-19.66 (± 0.95)	56.95 (± 0.92)	-30.92 (± 0.83)	50.47 (± 0.85)	-13.31 (± 1.24)
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 (± 1.17)
	ETS	69.11 (± 1.69)	-29.58 (± 1.81)	44.13 (± 1.06)	-45.28 (± 1.04)	41.35 (± 0.96)	-23.71 (± 1.45)
	Heuristic	74.33 (± 2.00)	-23.45 (± 2.27)	50.32 (± 1.97)	-37.99 (± 2.14)	44.13 (± 0.95)	-18.26 (± 1.05)
	Ours	76.55 (± 1.16)	-21.06 (± 1.32)	51.37 (± 1.63)	-37.01 (± 1.62)	46.76 (± 0.96)	-16.56 (± 0.90)
MoF	Random	77.96 (± 1.84)	-19.44 (± 2.13)	61.93 (± 1.05)	-25.89 (± 1.07)	54.50 (± 1.33)	-8.64 (± 1.26)
	ETS	77.62 (± 1.12)	-20.10 (± 1.26)	60.43 (± 1.17)	-28.22 (± 1.26)	56.12 (± 1.12)	-8.93 (± 0.83)
	Heuristic	77.27 (± 1.45)	-20.15 (± 1.63)	55.60 (± 2.70)	-32.57 (± 2.77)	52.30 (± 0.59)	-9.61 (± 0.67)
	Ours	81.58 (± 0.75)	-15.41 (± 0.86)	64.22 (± 0.65)	-23.48 (± 1.02)	57.70 (± 0.51)	-5.31 (± 0.55)

Table 5: Hyperparameters for replay-based methods HAL, MER, DER and DER++ used in experiments on applying RS-MCTS to recent replay-based methods in Section 4.5.

		5-task Datasets			**10- and 20-task Datasets**		
Method	**Hyperparam.**	S-MNIST	S-FashionMNIST	S-notMNIST	P-MNIST	S-CIFAR-100	S-miniImagenet
HAL	η	0.1	0.1	0.1	0.1	0.03	0.03
	λ	0.1	0.1	0.1	0.1	1.0	0.03
	γ	0.5	0.1	0.1	0.1	0.1	0.1
	β	0.7	0.5	0.5	0.5	0.5	0.5
	k	100	100	100	100	100	100
MER	γ	1.0	1.0	1.0	1.0	1.0	1.0
	β	1.0	0.01	1.0	1.0	0.1	0.1
DER	α	0.2	0.2	0.1	1.0	1.0	0.1
DER++	α	0.2	0.2	0.1	1.0	1.0	0.1
	β	1.0	1.0	1.0	1.0	1.0	1.0

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 notMNIST**	
Method	**Schedule**	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑
HAL	Random	97.24 (± 0.70)	-2.77 (± 0.90)	86.74 (± 6.05)	-15.54 (± 7.58)	93.61 (± 1.31)	-2.73 (± 1.33)
	ETS	97.21 (± 1.25)	-2.80 (± 1.59)	96.75 (± 0.50)	-2.84 (± 0.75)	92.16 (± 1.82)	-5.04 (± 2.24)
	Heuristic	97.69 (± 0.19)	-2.22 (± 0.24)	*74.16 (± 11.19)	-31.26 (± 14.00)	93.64 (± 0.93)	-2.80 (± 1.20)
	Ours	97.96 (± 0.15)	-1.85 (± 0.18)	97.56 (± 0.51)	-2.02 (± 0.63)	94.47 (± 0.82)	-1.67 (± 0.64)
MER	Random	93.07 (± 0.81)	-8.36 (± 0.99)	85.53 (± 3.30)	-0.56 (± 3.36)	91.13 (± 0.86)	-5.32 (± 0.95)
	ETS	92.97 (± 1.73)	-8.52 (± 2.15)	84.88 (± 3.85)	-3.34 (± 5.59)	90.56 (± 0.83)	-6.11 (± 1.06)
	Heuristic	94.30 (± 2.79)	-6.46 (± 3.50)	96.91 (± 0.62)	-1.34 (± 0.76)	90.90 (± 1.30)	-6.24 (± 1.96)
	Ours	96.44 (± 0.72)	-4.14 (± 0.94)	86.67 (± 4.09)	0.85 (± 3.85)	92.44 (± 0.77)	-3.63 (± 1.06)
DER	Random	98.23 (± 0.53)	-1.89 (± 0.65)	96.56 (± 1.79)	-3.48 (± 2.22)	92.89 (± 0.86)	-3.75 (± 1.16)
	ETS	98.17 (± 0.35)	-2.00 (± 0.42)	97.69 (± 0.58)	-2.05 (± 0.71)	94.74 (± 1.05)	-1.94 (± 1.17)
	Heuristic	94.57 (± 1.71)	-6.08 (± 2.09)	*72.49 (± 19.32)	-20.88 (± 11.46)	*77.88 (± 12.58)	-12.66 (± 4.17)
	Ours	99.02 (± 0.10)	-0.91 (± 0.13)	98.33 (± 0.51)	-1.26 (± 0.63)	95.02 (± 0.33)	-0.97 (± 0.81)
DER++	Random	97.90 (± 0.52)	-2.32 (± 0.67)	97.10 (± 1.03)	-2.77 (± 1.29)	93.29 (± 1.43)	-3.11 (± 1.60)
	ETS	97.98 (± 0.52)	-2.24 (± 0.66)	98.12 (± 0.40)	-1.59 (± 0.52)	94.53 (± 1.02)	-1.82 (± 1.02)
	Heuristic	92.35 (± 2.42)	-8.83 (± 2.99)	*67.31 (± 21.20)	-24.86 (± 16.34)	93.88 (± 1.33)	-2.86 (± 1.49)
	Ours	98.84 (± 0.21)	-1.14 (± 0.26)	98.38 (± 0.43)	-1.17 (± 0.51)	94.73 (± 0.20)	-1.21 (± 1.12)

		Permuted MNIST		**Split CIFAR-100**		**Split miniImagenet**	
Method	**Schedule**	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑
HAL	Random	88.49 (± 0.99)	-7.03 (± 1.05)	36.09 (± 1.77)	-17.49 (± 1.78)	38.51 (± 2.22)	-6.65 (± 1.43)
	ETS	88.46 (± 0.86)	-7.26 (± 0.90)	34.90 (± 2.02)	-18.92 (± 0.91)	38.13 (± 1.18)	-8.19 (± 1.73)
	Heuristic	*66.63 (± 28.50)	-29.68 (± 27.90)	35.07 (± 1.29)	-24.76 (± 2.41)	39.51 (± 1.49)	-5.65 (± 0.77)
	Ours	89.14 (± 0.74)	-6.29 (± 0.74)	40.22 (± 1.57)	-12.77 (± 1.30)	41.39 (± 1.15)	-3.69 (± 1.86)
MER	Random	75.90 (± 1.34)	-21.69 (± 1.47)	42.96 (± 1.70)	-34.01 (± 2.07)	31.48 (± 1.65)	-6.99 (± 1.27)
	ETS	73.01 (± 0.96)	-25.19 (± 1.10)	43.38 (± 1.81)	-34.84 (± 1.98)	33.58 (± 1.53)	-6.80 (± 1.46)
	Heuristic	83.86 (± 3.19)	-12.48 (± 3.60)	40.90 (± 1.70)	-44.10 (± 2.03)	34.22 (± 1.93)	-7.57 (± 1.63)
	Ours	79.72 (± 0.71)	-17.42 (± 0.78)	44.29 (± 0.69)	-32.73 (± 0.88)	32.74 (± 1.29)	-5.77 (± 1.04)
DER	Random	87.51 (± 1.10)	-8.81 (± 1.28)	56.83 (± 0.76)	-27.34 (± 0.63)	42.19 (± 0.67)	-10.60 (± 1.28)
	ETS	85.71 (± 0.75)	-11.15 (± 0.87)	52.58 (± 1.49)	-32.93 (± 2.04)	35.50 (± 2.84)	-10.94 (± 2.21)
	Heuristic	81.56 (± 2.28)	-15.06 (± 2.51)	55.75 (± 1.08)	-31.27 (± 1.02)	43.62 (± 0.88)	-8.18 (± 1.16)
	Ours	90.11 (± 0.18)	-5.89 (± 0.23)	58.99 (± 0.98)	-24.95 (± 0.64)	43.46 (± 0.95)	-9.32 (± 1.37)
DER++	Random	87.89 (± 1.10	-8.35 (± 1.33)	58.49 (± 1.44)	-25.48 (± 1.55)	48.40 (± 0.69)	-4.20 (± 0.86)
	ETS	85.25 (± 0.88)	-11.60 (± 1.03)	52.54 (± 1.06)	-33.22 (± 1.51)	41.36 (± 2.90)	-4.07 (± 2.28)
	Heuristic	79.17 (± 2.44)	-17.68 (± 2.68)	56.70 (± 1.27)	-30.33 (± 1.41)	45.73 (± 0.84)	-6.09 (± 1.24)
	Ours	89.84 (± 0.22)	-6.13 (± 0.29)	59.23 (± 0.83)	-24.61 (± 0.91)	49.45 (± 0.68)	-3.12 (± 0.89)

Table 7: 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 (± 5.02)
A-GEM	94.97 (± 1.50)	-6.03 (± 1.87)	94.81 (± 0.86)	-5.65 (± 1.06)	92.27 (± 1.16)	-4.17 (± 1.39)
ER-Ring	94.94 (± 1.56)	-6.07 (± 1.92)	95.83 (± 2.15)	-4.38 (± 2.59)	91.10 (± 1.89)	-6.27 (± 2.35)
Uniform	95.77 (± 1.12)	-5.02 (± 1.39)	97.12 (± 1.57)	-2.79 (± 1.98)	92.14 (± 1.45)	-4.90 (± 1.41)
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 (± 1.56)

	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 (± 1.86)
A-GEM	64.71 (± 1.78)	-34.41 (± 2.05)	42.22 (± 2.13)	-46.90 (± 2.21)	32.06 (± 1.83)	-30.81 (± 1.79)
ER-Ring	69.73 (± 1.13)	-28.87 (± 1.29)	53.93 (± 1.13)	-34.91 (± 1.18)	49.82 (± 1.69)	-14.38 (± 1.57)
Uniform	69.85 (± 1.01)	-28.74 (± 1.17)	52.63 (± 1.62)	-36.43 (± 1.81)	50.56 (± 1.07)	-13.52 (± 1.34)
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 (± 1.13)

V Replay Scheduling Monte Carlo Tree Search Algorithm

In this section, we provide more details on the methodology for replay scheduling with MCTS. Algorithm 1 outlines the steps for how we discretized the action space of task proportions to enable searching for replay schedules (Section 3.2). Furthermore, we provide pseudo-code in Algorithm 2 outlining the steps for our method Replay Scheduling Monte Carlo tree search (RS-MCTS) described Section 3.3.

In Algorithm 2, 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 the function 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 1 Discretization of action space with task proportions

Require: Number of tasks T

$\mathcal{T} = ()$ ▷ Initialize sequence for storing actions
for $i = 1, \ldots, T-1$ **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 $1, ..., i$
  $\hat{\mathcal{B}} = \texttt{unique}(\texttt{sort}(\mathcal{B}))$ ▷ Only keep unique bin vectors
  for $\boldsymbol{b}_i \in \hat{\mathcal{B}}$ **do**
    $\boldsymbol{a}_i = \texttt{bincount}(\boldsymbol{b}_i)/i$ ▷ Calculate task proportion
    $\mathcal{P}_i = \mathcal{P}_i \cup \{\boldsymbol{a}_i\}$ ▷ 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

Algorithm 2 Replay Scheduling Monte Carlo tree search

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

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

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

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

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

function $\text{EvaluateReplaySchedule}(\mathcal{D}_{1:T}, S, M)$
  Initialize neural network $f_{\boldsymbol{\theta}}$
  for $t = 1, \ldots, T$ **do**
    $\boldsymbol{a} \leftarrow S[t-1]$
    $\mathcal{M} \leftarrow \text{GetReplayMemory}(M, \boldsymbol{a})$
    for $\mathcal{B} \sim \mathcal{D}_t^{(train)}$ **do**
      $\boldsymbol{\theta} \leftarrow SGD(\mathcal{B} \cup \mathcal{M}, \boldsymbol{\theta}, \eta)$
    end for
  end for
  $A_{1:T}^{(val)} \leftarrow \text{EvaluateAccuracy}(f_{\boldsymbol{\theta}}, \mathcal{D}_{1:T}^{(val)})$
  $\text{ACC} \leftarrow \frac{1}{T}\sum_{i=1}^{T} A_{T,i}^{(val)}$
  return ACC
end function

function $\text{Backpropagate}(v_t, R)$
  while v_t is not root **do**
    $N(v_t) \leftarrow N(v_t) + 1$
    $Q(v_t) \leftarrow R$
    $v_t \leftarrow$ parent of v_t
  end while
end function

References

[1] 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.

[2] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 620–629. IEEE, 2018.

[3] 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.

[4] 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.

[5] 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.

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

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

[8] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning*, 2020.

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

[10] 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.

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

[12] 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.

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

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

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

[16] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International Conference on Computers and Games*, pages 72–83. Springer, 2006.

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

[18] 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.

[19] 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.

[20] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[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] Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient based memory editing for task-free continual learning. *arXiv preprint arXiv:2006.15294*, 2020.

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

[32] Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9385–9394, 2021.

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

[34] 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.

[35] 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.

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

[37] 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.

[38] 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.

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

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

[41] 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.

[42] 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.

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

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

[45] 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.

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

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

[48] 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.

[49] 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.

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

[51] 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.

[52] T. Landauer and Robert Bjork. Optimum rehearsal patterns and name learning. *Practical Aspects of Memory*, 1, 11 1977.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

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

[60] 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.

[61] 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.

[62] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European Conference on Machine Learning*, pages 282–293. Springer, 2006.

[63] 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.

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

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

[66] 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.

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

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

[69] 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.

[70] 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.

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

Paper D

Policy Learning for Replay Scheduling in Continual Learning

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

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

Abstract

Scheduling over which tasks to select for replay at different times 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 missing, which makes replay scheduling infeasible in real-world scenarios. To this end, we propose using reinforcement learning to enable learning policies that can be applied in new continual learning scenarios without additional computational cost. In our experiments, we show that the learned policies can propose replay schedules that efficiently mitigate catastrophic forgetting in environments with previously unseen task orders and datasets. The proposed approach opens up for new research directions in replay-based continual learning that stems well with real-world needs.

1 Introduction

Scheduling over which historical tasks to replay has been shown to mitigate catastrophic forgetting efficiently in continual learning (CL) [1, 2]. Such replay strategies are important in real-world applications with machine learning systems that needs to learn continuously from streaming data, wherein the incoming data is stored rather than deleted. In such scenarios, re-training on all stored data is often prohibited due to limitations on computational resources and data transmission times. Therefore, we need a policy that schedules from immense amounts of recorded data which previously learned abilities to replay when learning new tasks.

Although memory scheduling is essential for real-world CL, an efficient policy that schedules the time to replay which tasks is currently missing from the literature. For example, the policy in [2] is searched after using multiple CL episodes which is prohibited in the CL setting. Moreover, the method in [1] performs forward passes on

all stored data for selecting the tasks to replay, which becomes challenging in scenarios where the amounts of historical data is huge. If we could learn the scheduling policy, we would want the possibility to transfer the policy to new CL scenarios without the need for additional training in the target domain. Ideally, the policy should be domain-agnostic such that it is capable of generalizing to any CL dataset unseen during training. Such policy could potentially enable replay methods to efficiently mitigate catastrophic forgetting in real-world CL applications where the number of seen classes exceeds the replay memory budget.

In this paper, we propose a framework based on reinforcement learning (RL) [3] for learning a general replay scheduling policy that selects which tasks to replay at different times. We focus on the CL setting introduced in [2] which simulates a scenario where a replay memory is sampled from a large historical data storage for mitigating catastrophic forgetting. The policy is learned from multiple episodes in a CL environment by selecting which tasks to replay to efficiently mitigate catastrophic forgetting in the CL network. Our goal is to learn a replay scheduling policy that generalizes to new CL scenarios without added computational cost. We enable the possibility for generalizing to new domains by using the intuition that the policy should replay tasks that are to be forgotten by the classifier. Hence, we provide the policy with the evaluated task performances of the classifier in the CL environment, such that the policy can select the tasks to efficiently retain overall CL performance. In summary, our contributions are:

- We take the first steps to enable replay scheduling in real-world CL scenarios by proposing an RL-based framework for learning policies that generalize across different CL environments (Section 4).
- We show that the learned policies can efficiently mitigate catastrophic forgetting in CL scenarios with new task orders and datasets unseen during training without added computational cost (Section 5.1).

2 Preliminaries

In this section, we recall the CL setting in [2] that we consider as well as some background on RL and the algorithms Deep Q-Networks (DQNs) [4,5] and Advantage Actor-Critic (A2C) [6].

2.1 Continual Learning Setting

We consider the CL setting presented in Klasson et al. [2]. In contrast to the traditional CL setting [7, 8], the historical data is assumed to be accessible at any time for replay to mitigate catastrophic forgetting. However, retraining on all seen data is prohibited due to processing time constraints, such as limitations on data transmission times or the allowed time for learning new tasks. The learning setup is the same as for CL in image classification, where a network parameterized by $\boldsymbol{\phi}$ should learn T tasks arriving sequentially in the form of T datasets $\mathcal{D}_{1:T} = \{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$. The dataset for the t-th task $\mathcal{D}_t = \{(\boldsymbol{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^{N_t}$ where $\boldsymbol{x}_t^{(i)}$ and $y_t^{(i)}$ are the i-th data point and corresponding class label among a total of N_t examples in the dataset. Furthermore, each dataset is split into a training, validation, and test set,

i.e., $\mathcal{D}_t = \{\mathcal{D}_t^{train}, \mathcal{D}_t^{val}, \mathcal{D}_t^{test}\}$. The objective at task t is to minimize the loss $\ell(f_{\boldsymbol{\phi}}(\boldsymbol{x}_t), y_t)$ where $\ell(\cdot)$ is the cross-entropy loss in our case.

The challenge is for the network $f_{\boldsymbol{\phi}}$ to retain its performance on the previous tasks. However, as the historical data can be huge, we are only allowed to fill a tiny replay memory $\mathcal{M}$ of size M with historical data due to limitations on processing time. Hence, we need a method for selecting which tasks to replay and the amount of samples to add to the replay memory from each selected task. We use the same approach as in [2] to enable the task selection, where a discrete action space of task proportions is created. At each task t, there is a finite set possible task proportions $\boldsymbol{p}_t = (p_1, \ldots, p_{T-1})$, where $\sum_j p_j = 1$ with $p_j \geq 0$ if $j < t$ otherwise $p_j = 0$, that can be used for constructing the replay memory for mitigating catastrophic forgetting. See Appendix II.1 for details on how to construct the action space. Replay scheduling involves finding a policy for selecting task proportions that are efficient in mitigating catastrophic forgetting in the CL network. In Section 4, we describe our RL-based framework for learning such policies.

2.2 Background on Reinforcement Learning

The RL setup considers an agent interacting with an environment E over a number of discrete time steps [3]. The environment is modeled with a Markov Decision Process (MDP) [9] represented as a tuple $E = (\mathcal{S}, \mathcal{A}, P, R, \mu, \gamma)$ consisting of the state space $\mathcal{S}$, action space $\mathcal{A}$, state transition probability $P(s'|s, a)$, reward function $R(s, a)$, initial state distribution $\mu(s_1)$, and discount factor γ. 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 for the agent is to learns policy that maximizes the expected return.

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. Deep Q-Networks (DQNs) [4,5] is a value-based RL algorithm which aims to approximate the optimal action value function as $Q^*(s, a) \approx Q_{\boldsymbol{\theta}}(s, a)$ with a neural network $Q_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$. For learning the function $Q_{\boldsymbol{\theta}}$, we collect data from the environment with by using an epsilon-greedy policy [3] to estimate $Q_{\boldsymbol{\theta}}$ by minimizing the loss $\mathcal{L}_{\text{DQN}}(\boldsymbol{\theta}) = (y_t - Q_{\boldsymbol{\theta}}(s_t, a_t))^2$ with $y_t = r_t + \gamma \max_{a'} Q_{\boldsymbol{\theta}^-}(s_{t+1}, a')$, where $\boldsymbol{\theta}^-$ is a previous copy of the network $\boldsymbol{\theta}$ referred to as the target network. In addition to the introduction of target networks, several methods have been proposed for stabilizing the learning process, such as using experience replay [10] to sample training data stored in a replay buffer, applying L1-smoothing to the loss, and correcting the action value estimates [11].

An alternative to value-based RL is policy gradient methods where the optimal policy is estimated directly with a parameterized form $\pi_{\boldsymbol{\theta}}(a|s)$ where $\boldsymbol{\theta}$ represents the parameters of a neural network. Similar to the action value function, the value function $V^\pi(s) = \mathbb{E}[G_t|s_t = s]$ defines the expected return following policy π from

state s. In actor critic methods [6, 12–14], we estimate the value function with a neural network $V_{\boldsymbol{\theta}_v}$ to reduce variance of the gradients. The policy network takes the state s_t as input and outputs a distribution over the possible actions a_t. The agent collects experiences from the environment with the current policy to use for updating the parameters $\boldsymbol{\theta}$ by minimizing the loss $\mathcal{L}_{\text{PG}}(\boldsymbol{\theta}) = \mathbb{E}[\log \pi_{\boldsymbol{\theta}}(a|s)\hat{A}(s_t, a_t)]$,, 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 policy and value function can be updated after t_{max} actions or when a terminal state is reached [6].

3 Related Work

In this section, we place this paper into context by providing a brief overview of CL as well as generalization in RL.

Continual Learning. The various CL approaches for mitigating catastrophic forgetting can broadly be divided into three categories, namely, regularization-based [15–21], architecture-based [22–28], and replay-based approaches [1,2,29–42]. Regularization-based methods generally focuses on applying regularization techniques on parameters important for recognizing old tasks and fit the remaining parameters to new tasks [15, 17, 21, 43]. Furthermore, knowledge distillation [44] has been used for regularizing network output units of previous tasks [16, 18], as well as constraining the parameter updates to subspaces with gradient projections to avoid interference with previous tasks [19,20]. Architecture-based approaches focuses on adding task-specific network modules for every seen task [22–24,27,28], or isolating parameters for predicting specific task in fixed-size networks [25, 26, 45]. Replay-based methods re-trains the network on samples of old tasks that are either stored in an external memory [1, 2, 29, 30, 34–38, 40, 41, 46], or synthesized with a generative model [32, 33, 42]. Scheduling over which samples or tasks to replay has been studied in [1,2]. This work builds on the replay scheduling idea in [2] where we propose an RL-based framework for learning policies that selects which tasks to replay at different times.

Generalization in Reinforcement Learning. Generalization is an active research topic in RL [47] as RL agents tend to overfit to their training environments [48–51]. The goal is often to transfer learned policies to environments with new tasks [52–55] and action spaces [56–58]. Some approaches aim to improve generalization capabilities by generating more diverse training data [49, 59, 60], using network regularization or inductive biases [61–63], or learning dynamics models [64, 65]. In this paper, we use RL for learning policies for selecting which tasks a CL network should replay. The goal is to learn policies that can be applied in new CL environments for replay scheduling on unseen task orders and datasets without additional computational cost.

4 Methodology: Policy Learning Framework for Replay Scheduling

In this section, we present an RL-based framework for learning replay scheduling policies that generalize across different CL scenarios. Our intuition is that there may

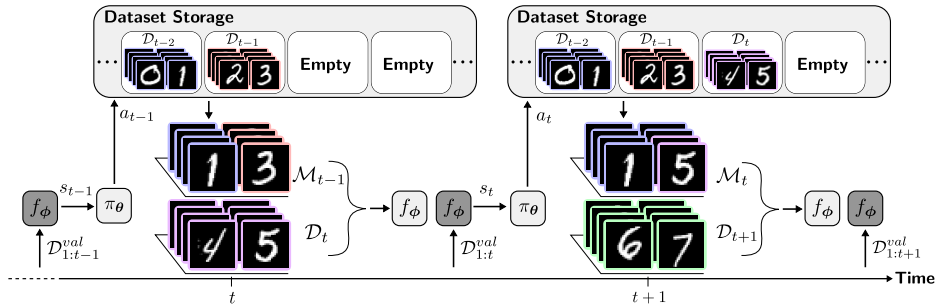


Figure 1: Illustration of the procedure of the proposed RL-based framework in a CL environment with Split MNIST. The goal is to mitigate catastrophic forgetting in the classifier $f_{\boldsymbol{\phi}}$ by composing replay memories $\mathcal{M}$ with the replay scheduling policy $\pi_{\boldsymbol{\theta}}$. The policy selects action a determining which tasks to replay given the state s represented as the validation task performances of $f_{\boldsymbol{\phi}}$. The colored border on the MNIST images indicates which task the data comes from. The classifier $f_{\boldsymbol{\phi}}$ is in evaluation mode when darker-shaded color over the box is used.

exist general patterns regarding the replay scheduling, e.g., tasks that are harder or have been forgotten should be replayed more often. Moreover, the policy may non-trivially take task properties into consideration. Therefore, we aim to learn policies that select which tasks to replay from states representing the current task performance in the CL environments. The policy can then be applied for mitigating catastrophic forgetting in new CL scenarios.

The CL environments are modeled using MDPs represented as tuples $E_i = (\mathcal{S}_i, \mathcal{A}, P_i, R_i, \mu_i, \gamma)$ sampled from a distribution of environments, such that $E_i \sim p(E)$ for $i = 1, ..., K$. We let the policy interact with a fixed set of environments $\mathcal{E}^{(train)} = \{E_1, \ldots, E_K\}$ sampled from the distribution $p(E)$ for training. Each environment E_i contains of network $f_{\boldsymbol{\phi}}$ and T datasets $\mathcal{D}_{1:T}$ where the t-th dataset is learned at time step t. To generate diverse CL environments, we get environments with different network initializations of $f_{\boldsymbol{\phi}}$ and shuffled task orders in the dataset when we sample environments from $p(E)$. We define the state s_t of the environment as the validation accuracies $A^{val}_{t,1:t}$ on each seen task $1, ..., t$ from $f_{\boldsymbol{\phi}}$ at task t, i.e., $s_t = [A_{t,1}, ..., A_{t,t}, 0, ..., 0]$, where we use zero-padding on future tasks. The action space $\mathcal{A}$ is constructed as described in Appendix II.1, such that the $a_t \in \mathcal{A}$ corresponds to a task proportion $\boldsymbol{p}_t$ used for sampling the replay memory $\mathcal{M}_t$. We use a dense reward based on the average validation accuracies at task t, i.e., $r_t = \frac{1}{t}\sum_{i=1}^{t} A^{val}_{t,i}$. The state transition distribution $P_i(s'|s, a)$ represents the dynamics of the environment, which depend on the initialization of $f_{\boldsymbol{\phi}}$ and also the task order in the dataset.

The procedure for training the policy goes as follows: The state s_t is obtained by evaluating the network $f_{\boldsymbol{\phi}}$ on the validation sets $\mathcal{D}^{val}_{1:t}$ after learning the t-th task from $\mathcal{D}^{(train)}_t$. Action a_t is selected under the policy $\pi_{\boldsymbol{\theta}}(a|s_t)$ parameterized by $\boldsymbol{\theta}$. The action is converted into task proportion $\boldsymbol{p}_t$ for sampling the replay memory $\mathcal{M}_t$ from the historical datasets $\mathcal{D}^{train}_{1:t}$. We then train classifier $f_{\boldsymbol{\phi}}$ with $\mathcal{D}^{(train)}_{t+1}$ and $\mathcal{M}_t$, and obtain the reward r_{t+1} and the next state s_{t+1} by evaluating $f_{\boldsymbol{\phi}}$ on the validation sets $\mathcal{D}^{val}_{1:t}$. The collected transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ are used for updating the policy. This procedure is followed until the final task T after which we start a new episode.

Figure 1 shows an illustration of this procedure for using the learned replay scheduling policy in a CL environment with the Split MNIST dataset [43]. The seen datasets are placed in the dataset storage to use for sampling replay memories given the actions from the policy $\pi_{\boldsymbol{\theta}}$ for mitigating catastrophic forgetting at each time step.

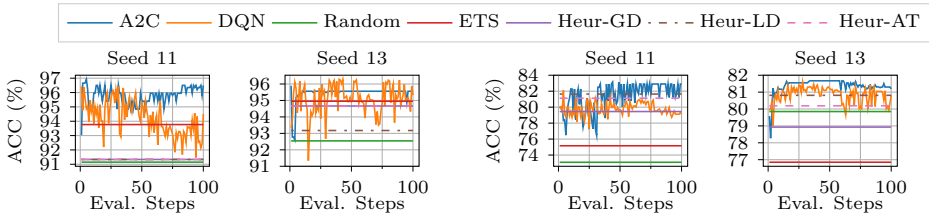
We evaluate the learned policy by applying it to mitigate catastrophic forgetting in new CL environments at test time. To foster generalization across environments, we train the policy on multiple environments with different dynamics, e.g., task orders and datasets, to learn from diverse sets of training data. The goal for the agent is to maximize the sum of rewards in each training environment. At test time, the policy is applied on new CL classifiers and datasets in the test environments without added computational cost nor experience collection. In Section 5, we test the policies generalization capability to new CL environments where the task orders and datasets are unseen during training.

5 Experiments

We evaluate our RL-based framework using DQN [4, 5] and A2C [6] for learning policies that generalize to new CL scenarios. We show that the learned policies can efficiently mitigate catastrophic forgetting in CL environments with new task orders and datasets that are unseen during training. Full details on experimental settings and additional results are in Appendix I and IV. Code will be made publicly available upon acceptance.

Datasets and Network Architectures. We conduct experiments on four CL benchmark datasets: Split MNIST [43,66], FashionMNIST [67], Split notMNIST [68], Permuted MNIST [69], and Split CIFAR-10 [70]. We randomly sample 15% of training data for each task to use for validation, and keep the original test sets intact. We use multi-head output layers for all datasets and assume task labels are available at test time [71]. A 2-layer MLP with 256 hidden units for Split MNIST, Split FashionMNIST, and Split notMNIST. For Split CIFAR-10, we the same ConvNet architecture used in [18, 21, 72].

Generating CL Environments. We generate multiple CL environments with pre-set random seeds for initializing the network parameters $\boldsymbol{\phi}$ and shuffling the task order. See Appendix I for details on the pre-set seeds. We shuffle the task order by permuting the class order and then split the classes into 5 pairs (tasks) with 2 classes/pair. Taking a step at task t in the CL environments involves training the CL network on the t-th dataset with a replay memory $\mathcal{M}_t$ from the discrete action space described in Appendix II.1. To speed up the experiments with the RL algorithms, we run a breadth-first search (BFS) through the discrete action space and save the classification results for re-use during policy learning. Note that the action space has 1050 possible paths of replay schedules for the 5-task datasets, which makes the environment generation time-consuming. More specifically, generating a CL environment for one seed with Split MNIST took on around 9.5 hours on average on a NVIDIA GeForce RTW 2080Ti. Hence, we only generate environments where the replay memory size $M = 10$ have been used, and leave analysis of different memory sizes as future work.



(a) Split MNIST (b) Split CIFAR-10

Figure 2: Performance progress measured in ACC (%) for all methods in the 2 test environments with (a) **Split MNIST** and (b) **Split CIFAR-10** in the New Task Orders experiment. We plot the performance progress for A2C and DQN for 100 evaluation steps equidistantly distributed over the training episodes. The performance for the Random, ETS, and Heuristic scheduling baselines are plotted as straight lines.

DQN and A2C Architectures. The input layer has size $T-1$ where each unit is inputting the task performances since the states are represented by the validation accuracies $s_t = [A_{t,1}^{(val)}, ..., A_{t,t}^{(val)}, 0, ..., 0]$. The current task can therefore be determined by the number of non-zero state inputs. The output layer has 35 units representing the possible actions at $T = 5$ in the discrete action space (see Appendix II.1). We use action masking on the output units to prevent the network from selection invalid actions for constructing the replay memory at the current task. The DQN is a 2-layer MLP with 512 hidden units and ReLU activations. For A2C, we use separate networks for parameterizing the policy and the value function, where both networks are 2-layer MLPs with 64 hidden units of Tanh activations.

Baselines. We compare our proposed method to the following scheduling baselines:

- **Random.** Random policy that randomly selects task proportions from the action space on how to structure the replay memory at every task.
- **Equal Task Schedule (ETS).** Policy that selects equal task proportion such that the replay memory aims to fill the memory with an equal number of samples from every seen task.
- **Heuristic Global Drop (Heur-GD).** Heuristic policy that replays tasks with validation accuracy below a certain threshold proportional to the best achieved validation accuracy on the task.
- **Heuristic Local Drop (Heur-LD).** Heuristic policy that replays tasks with validation accuracy below a threshold proportional to the previous achieved validation accuracy on the task.
- **Heuristic Accuracy Threshold (Heur-AT).** Heuristic policy that replays tasks with validation accuracy below a fixed threshold.

The heuristics are based on the intuition that forgotten tasks should be replayed. They fill the replay memory with M/k samples per task where k is the number of selected tasks. If $k = 0$, then replay is skipped at the current task.

Evaluation Protocol. We use the average test accuracy over all tasks after learning the final task, i.e., ACC $= \frac{1}{T}\sum_{i=1}^{T} A_{T,i}^{test}$ where $A_{T,i}^{test}$ is the test accuracy of task i after learning task T. To assess generalization capability, we use a ranking method

Table 1: Average ranking (lower is better) for experiments on generalizing policies to environments with new task orders or a new dataset. We average the results over 10 test environments.

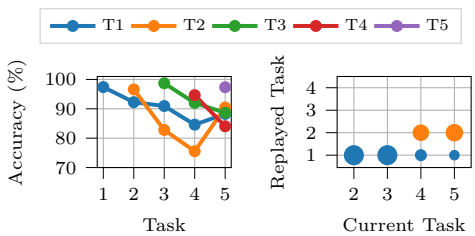
	New Task Order			New Dataset	
Method	S-MNIST	S-FashionMNIST	S-CIFAR-10	S-notMNIST	S-FashionMNIST
Random	4.23	3.60	5.03	3.87	3.95
ETS	3.80	4.57	5.37	4.27	3.63
Heur-GD	4.48	4.25	3.98	4.58	**2.77**
Heur-LD	4.65	3.65	3.75	4.97	5.10
Heur-AT	4.33	3.87	3.43	4.28	3.72
DQN (Ours)	**3.23**	**3.55**	3.68	3.20	4.37
A2C (Ours)	3.27	4.52	**2.75**	**2.83**	4.47

based on the ACC between the methods in every test environment for comparison (see Appendix I for more details).

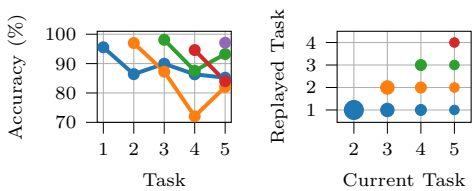
5.1 Policy Generalization to New Continual Learning Scenarios

In this section, we show that the policies learned with our RL-based framework using DQN and A2C are capable of generalizing across CL environments with new task orders and datasets unseen during training.

Generalization to New Task Orders. We show that the learned replay scheduling policies can generalize to CL environments with previously unseen task orders. We generate training and test environments with unique task orders for three datasets, namely, Split MNIST, FashionMNIST, and CIFAR-10. Table 1 shows the average ranking for the DQN, A2C, and the baselines when being applied in 10 test environments. Our learned policies obtain the best average ranking across the datasets, where the DQN performs best for Split MNIST and FashionMNIST while A2C outperforms all methods on Split CIFAR-10. Figure 2(a) and (b) shows the performance progress for two test environments with Split MNIST and Split CIFAR-10 respectively (see Figure 5 and 6 in Appendix IV for all test environments). We observe that DQN exhibits noisier progress of the achieved ACC in these test environments than A2C. We provide further insights in the benefits of learning the replay scheduling policy by visualizing the replay schedule and task accuracy progress from A2C in one Split CIFAR-10 test environment in Figure 3. Additionally, we show the replay schedule

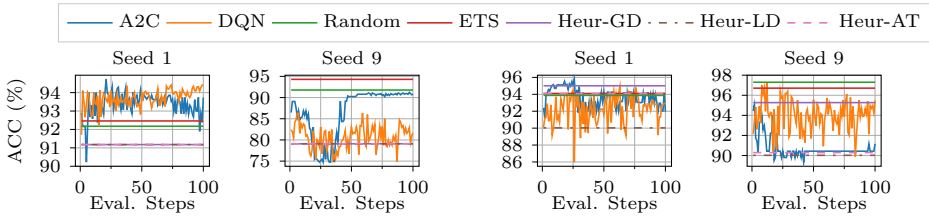


(a) A2C - ACC: 89.75%



(b) ETS - ACC: 88.32%

Figure 3: Task accuracies and replay schedules for A2C and ETS for a Split CIFAR-10 environment.



(a) Split notMNIST (b) Split FashionMNIST

Figure 4: Performance progress measured in ACC (%) for all methods in the 2 test environments with (a) **Split notMNIST** and (b) **Split FashionMNIST** in the New Dataset experiment. We plot the performance progress for A2C and DQN for 100 evaluation steps equidistantly distributed over the training episodes. The performance for the Random, ETS, and Heuristic scheduling baselines are plotted as straight lines.

and task accuracies from the ETS baseline in the same environment. The replay schedules are visualized with bubble plots showing the selected task proportion to use for composing the replay memories at each task. Each circle color corresponds to a historical task and its size represents the proportion of replay samples at the current task. The sum of points in all circles at each column is fixed at all current tasks. In Figure 3a, we observe that A2C decides to replay task 2 more than task 1 as the performance on task 2 decreases, which results in a slightly better ACC metric achieved by A2C than ETS. These results show that the learned policy can flexibly consider replaying forgotten tasks to enhance the CL performance.

Generalization to New Datasets. We show that the learned replay scheduling policy is capable of generalizing to CL environments with new datasets unseen in the training environments. We perform two sets of experiments:

1. Train with environments generated with Split MNIST and FashionMNIST, and test on environments generated with Split notMNIST.
2. Train with environments generated with Split MNIST and notMNIST, and test on environments generated with Split FashionMNIST.

Table 1 shows the average ranking for DQN, A2C, and the baselines when generalization to test environments with new datasets. We observe that both A2C and DQN successfully generalize to Split notMNIST environments outperforming all baselines. However, the learned policies have difficulties to generalize to Split FashionMNIST environments, which could be due to high variations in the dynamics between training and test environments. To gain insights, we show the performance progress for two test environments with Split notMNIST and Split FashionMNIST datasets in Figure 4(a) and (b) respectively (see Figure 7 and 8 in Appendix IV for all test environments). In Figure 4(b), we observe that both A2C and DQN has difficulties in converging to efficient replay scheduling policies, where A2C even collapses to a suboptimal replay schedule in second test environment (Seed 9). A possible reason could be that the policy encounters state transition dynamics in the test environments which has not been experienced during training, which makes generalization difficult for both DQN and A2C. Potentially, the performance could be improved by generating more training environments for the agent to exhibit more variations in the CL scenarios or by using other advanced RL methods which may generalize better [62].

6 Conclusions

In this paper, we introduce an an RL-based framework for learning the time to learn in a CL context. Building on the replay scheduling idea from [2], we are the first to propose a method for learning policies that schedules which tasks to replay at different times to improve memory retention of historical tasks. We showed that the framework learns replay scheduling policies that can generalize across different CL environments with unseen task orders and datasets without additional computational cost or training in the test environment. Moreover, the learned policies are capable of considering replaying forgotten tasks which can mitigate catastrophic forgetting more efficiently than fixed scheduling policies. The proposed policy learning framework opens up for new research directions in replay-based CL and memory scheduling, especially in real-world scenarios where the amount of available data can exceed the replay memory capacity and processing time constraints.

Limitations and Future Work. Generalization in RL is a challenging research topic by itself. With the current method, large amounts of diverse data and training time is required to enable the learned policy to generalize well. This can be costly due to generating the CL environments is expensive since each state transition involves training the classifier on a CL task. Moreover, we are currently considering a discrete action space which is hard to construct especially when the number of tasks is large. Thus, in future work, we would explore more advanced RL methods which can handle continuous action spaces and generalize well.

Supplementary Material

This supplementary material is structured as follows:

- Appendix I: Details of the experimental settings on the CL and RL parts with hyperparameter settings for CL networks, DQN, and A2C. Moreover, we provide details on the heuristic scheduling baselines used in the experiments, as well as the ranking method used for assessing generalization capability.
- Appendix II: Details on the methodology, such as construction of the action space and the training procedure of the policy with RL.
- Appendix III: Tables over the task splits in all test environments used in the experiments.
- Appendix IV: Additional figures and tables as part of the experimental results.

I Experimental Settings

In this section, we describe the full details of the experimental settings used in this paper. We first provide details on hyperparameter settings for the CL and RL parts, including hyperparameters for DQN [4,5] and A2C [6], followed by description of the heuristic baselines, and the ranking method for assesing the generalization capability of the learned policy.

Datasets. We generate CL environments with four datasets commonly used in the CL literature. Split MNIST [43] is a variant of the MNIST [66] 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 FashionMNIST [67] 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 [68] 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 [22] for Split notMNIST. In Split CIFAR-10 [70], the 10 classes are divided into 5 tasks with 2 classes for each task.

CL Network Architectures. We use a 2-layer MLP with 256 hidden units and ReLU activation for Split MNIST, Split FashionMNIST, and Split notMNIST. For Split CIFAR-10, we use the ConvNet architecture used in [18, 21, 72], which consists of four 3x3 convolutional blocks, i.e. convolutional layer followed by batch normalization [73], with 64 filters, ReLU activations, and 2x2 Max-pooling. We use a multi-head output layer for each dataset and assume task labels are available at test time for selecting the correct output head related to the task.

CL Hyperparameters. We train all networks with the Adam optimizer [74] 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: 10 epochs/task, batch size 128.
- Split notMNIST: 20 epochs/task, batch size 128.
- Split CIFAR-100: 20 epochs/task, batch size 256.

Generating CL Environments. We generate multiple CL environments with pre-set random seeds for initializing the network parameters $\boldsymbol{\phi}$ and shuffling the task order. The pre-set random seeds are in the range $0 - 49$, such that we have 50 environments for each dataset. We shuffle the task order by permuting the class order and then split the classes into 5 pairs (tasks) with 2 classes/pair. For environments with seed 0, we keep the original task order in the dataset. Taking a step at task t in the CL environments involves training the CL network on the t-th dataset with a replay memory $\mathcal{M}_t$ from the discrete action space described in Section II.1. Therefore, to speed up the experiments with the RL algorithms, we run a breadth-first search (BFS) through the discrete action space and save the classification results for re-use during policy learning. Note that the action space has 1050 possible paths of replay schedules for the datasets with $T = 5$ tasks, which makes the environment generation time-consuming. Hence, we only generate environments where the replay memory size $M = 10$ have been used, and leave analysis of different memory sizes as future work. The CL environments that we used will be provided in the public code.

Computational Cost. All experiments were performed on one NVIDIA GeForce RTW 2080Ti on an internal GPU cluster. Generating a CL environment for one seed with Split MNIST took on around 9.5 hours averaged over 10 runs of BFS. Similarly for Split CIFAR-10, generating one CL environment took on average 16.1

hours. Note that BFS runs 1050 iterations in total for all 5-task datasets. The wall clock time for ETS on Split MNIST was around 1.5 minutes.

DQN and A2C Architectures. The input layer has size $T-1$ where each unit is inputting the task performances since the states are represented by the validation accuracies $s_t = [A_{t,1}^{(val)}, ..., A_{t,t}^{(val)}, 0, ..., 0]$. The current task can therefore be determined by the number of non-zero state inputs. The output layer has 35 units representing the possible actions at $T = 5$ with the discrete action space we have constructed in Appendix II.1. We use action masking on the output units to prevent the network from selection invalid actions for constructing the replay memory at the current task. The DQN is a 2-layer MLP with 512 hidden units and ReLU activations. For A2C, we use separate networks for parameterizing the policy and the value function, where both networks are 2-layer MLPs with 64 hidden units of Tanh activations.

DQN and A2C Hyperparameters. We provide the hyperparameters for the both DQN and A2C in Table 2-5. Table 2 and 3 includes the hyperparameters on the New Task Order experiment for DQN and A2C respectively, while Table 4 and 5 includes the hyperparameters on the New Dataset experiment for DQN and A2C respectively. Regarding the training environments in Table 4 and 5, we use two different datasets in the training environments to increase the diversity. When Spit notMNIST is for testing, half the amount of training environments are using Split MNIST and the other half uses Split FashionMNIST. For example, in Table 5, A2C uses 10 training environments which means that there are 5 Split MNIST environments and 5 Split FashionMNIST environments. Similarly, half the amount of training environments are using Split MNIST and the other half uses Split notMNIST when the testing environments uses Split FashionMNIST.

Implementations. The code for DQN was adapted from OpenAI baselines [75] and the PyTorch [76] tutorial on DQN https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html. For A2C, we followed the implementations released by Kostrikov [77] and Igl et al. [78]. We make the code publicly available upon acceptance.

Table 2: DQN hyperparameters for the experiments on **New Task Orders** in Section 5.1.

Hyperparameters	**Split MNIST**	**Split FashionMNIST**	**Split CIFAR-10**
Training Environments	30	20	10
Learning Rate	0.0001	0.0003	0.0003
Optimizer	Adam	Adam	Adam
Buffer Size	10k	10k	10k
Target Update per step	500	500	500
Batch Size	32	32	32
Discount Factor γ	1.0	1.0	1.0
Exploration Start ϵ_{start}	1.0	1.0	1.0
Exploration Final ϵ_{final}	0.02	0.02	0.02
Exploration Annealing (episodes)	2.5k	2.5k	2.5k
Training Episodes	10k	10k	10k

Table 3: A2C hyperparameters for the experiments on **New Task Orders** in Section 5.1.

Hyperparameters	**Split MNIST**	**Split FashionMNIST**	**Split CIFAR-10**
Training Environments	10	10	10
Learning Rate	0.0001	0.0003	0.00003
Optimizer	RMSProp	RMSProp	RMSProp
Gradient Clipping	0.5	0.5	0.5
GAE parameter λ	0.95	0.95	0.95
VF coefficient	0.5	0.5	0.5
Entropy coefficient	0.01	0.01	0.01
Number of steps n_{steps}	5	5	5
Discount Factor γ	1.0	1.0	1.0
Training Episodes	100k	100k	100k

Table 4: DQN hyperparameters for the experiments on **New Dataset** in Section 5.1. Split notMNIST and Split FashionMNIST indicate the dataset used in the test environments.

Hyperparameters	**Split notMNIST**	**Split FashionMNIST**
Training Environments	30	30
Learning Rate	0.0001	0.0001
Optimizer	Adam	Adam
Buffer Size	10k	10k
Target Update per step	500	500
Batch Size	32	32
Discount Factor γ	1.0	1.0
Exploration Start ϵ_{start}	1.0	1.0
Exploration Final ϵ_{final}	0.02	0.02
Exploration Annealing (episodes)	2.5k	2.5k
Training Episodes	10k	10k

Heuristic Scheduling Baselines

We implemented three heuristic scheduling baselines to compare against our proposed methods. These heuristics are based on the intuition of re-learning tasks when they have been forgotten. We keep a validation set for each task to determine whether any task should be replayed by comparing the validation accuracy against a hand-tuned threshold. If the validation accuracy is below the threshold, then the corresponding task is replayed. Let $A_{t,i}^{(val)}$ be the validation accuracy for task t evaluated at time step i. The threshold is set differently in each of the baselines:

- **Heuristic Global Drop (Heur-GD).** Heuristic policy that replays tasks with validation accuracy below a certain threshold proportional to the best achieved validation accuracy on the task. The best achieved validation accuracy for task i is given by $A_{t,i}^{(best)} = \max\{(A_{1,i}^{(val)}, \ldots, A_{t,i}^{(val)})\}$. Task i is replayed if $A_{t,i}^{(val)} < \tau A_{t,i}^{(best)}$ where $\tau \in [0, 1]$ is a ratio representing the degree of how much the validation accuracy of a task is allowed to drop.
- **Heuristic Local Drop (Heur-LD).** Heuristic policy that replays tasks with validation accuracy below a threshold proportional to the previous achieved

Table 5: A2C hyperparameters for the experiments on **New Dataset** in Section 5.1. Split notMNIST and Split FashionMNIST indicate the dataset used in the test environments.

Hyperparameters	**Split notMNIST**	**Split FashionMNIST**
Training Environments	10	10
Learning Rate	0.0001	0.0003
Optimizer	RMSProp	RMSProp
Gradient Clipping	0.5	0.5
GAE parameter λ	0.95	0.95
VF coefficient	0.5	0.5
Entropy coefficient	0.01	0.01
Number of steps n_{steps}	5	5
Discount Factor γ	1.0	1.0
Training Episodes	100k	100k

Table 6: The threshold parameter τ used in the heuristic scheudling baselines Heuristic Global Drop (Heur-GD), Heuristic Local Drop (Heur-LD), and Heuristic Accuracy Threshold (Heur-AT). The search range is $\tau \in \{0.90, 0.95, 0.999\}$ for all methods and we display the number of environments used for selecting the parameter used at test time.

	New Task Order						**New Dataset**			
	S-MNIST		**S-FashionMNIST**		**S-CIFAR-10**		**S-notMNIST**		**S-FashionMNIST**	
Method	τ	#Envs	τ	#Envs	τ	#Envs	τ	#Envs	τ	#Envs
Heur-GD	0.9	10	0.95	20	0.9	10	0.9	10	0.9	10
Heur-LD	0.9	10	0.999	20	0.999	10	0.95	10	0.999	10
Heur-AT	0.9	10	0.999	20	0.9	10	0.9	10	0.95	10

validation accuracy on the task. Task i is replayed if $A_{t,i}^{(val)} < \tau A_{t-1,i}^{(val)}$ where *tau* again represents the degree of how much the validation accuracy of a task is allowed to drop.

- **Heuristic Accuracy Threshold (Heur-AT).** Heuristic policy that replays tasks with validation accuracy below a fixed threshold. Task i is replayed if if $A_{t,i}^{(val)} < \tau$ where $\tau \in [0, 1]$ represents the least tolerated accuracy before we need to replay the task.

The replay memory is filled with M/k samples from each selected task, where k is the number of tasks that need to be replayed according to their decrease in validation accuracy. We skip replaying any tasks if no tasks are selected for replay, i.e., $k = 0$.

Grid search for τ. We performed a grid search for the parameter τ for the three heuristic scheduling baselines for each experiment to compare against the learned replay scheduling policies. The validation set consists of 15% of the training data and is identical to the validation set used for learning the RL policies. We select the parameter based on ACC scores achieved in the same number of training environments used by either DQN or A2C. The search range we use is $\tau \in \{0.90, 0.95, 0.999\}$. In Table 6, we show the selected parameter value of τ and the number of environments used for selecting the value for each method and experiment in Section **??**. The same parameters are used to generate the results on the heuristics in Table 1.

Assessing Generalization with Ranking Method

We use a ranking method based on the CL performance in every test environment for performance comparison between the methods in Section 5.1. We use rankings because the performances can vary greatly between environments with different task orders and datasets. To measure the CL performance in the environments, we use the average test accuracy over all tasks after learning the final task, i.e.,

$$\text{ACC} = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}^{(test)},$$

where $A_{t,i}^{(test)}$ is the test accuracy of task i after learning task t. Each method are ranked in descending order based on the ACC achieved in an environment. For example, assume that we want to compare the CL performance from using learned replay scheduling policies with DQN and A2C against a Random scheduling policy in one environment. The CL performances achieved for each method are given by

$$[\text{ACC}_{\text{Random}}, \text{ACC}_{\text{DQN}}, \text{ACC}_{\text{A2C}}] = [90\%, 99\%, 95\%].$$

We get the following ranking order between the methods based on their corresponding ACC:

$$\texttt{ranking}([\text{ACC}_{\text{Random}}, \text{ACC}_{\text{DQN}}, \text{ACC}_{\text{A2C}}]) = [3, 1, 2],$$

where DQN is ranked in 1st place, A2C in 2nd, and Random in 3rd. When there are multiple environments for evaluation, we compute the average ranking across the ranking positions in every environment for each method to compare.

The average ranking for DQN and A2C are computed over the seed for initializing the network parameters as well as the seed of the environment. Similarly, the Random baseline is affected by the seed setting the random selection of actions and the environment seed. However, the performance of the ETS and Heuristic baselines are affected by the seed of the environment as these policies are fixed. We use copied values of the performance in environments for the ETS and Heuristic baselines when we need to compare across different random seeds for Random, DQN, and A2C. We show an example of such ranking calculation for ETS, a Heuristic baseline, DQN, and A2C. Consider the following performances for one environment:

$$\begin{bmatrix} \text{ACC}_{\text{ETS}}^1 & \text{ACC}_{\text{Heur}}^1 & \text{ACC}_{\text{DQN}}^1 & \text{ACC}_{\text{A2C}}^1 \\ \text{ACC}_{\text{ETS}}^2 & \text{ACC}_{\text{Heur}}^2 & \text{ACC}_{\text{DQN}}^2 & \text{ACC}_{\text{A2C}}^2 \end{bmatrix} = \begin{bmatrix} 90\% & 95\% & 95\% & 99\% \\ * & * & 97\% & 98\% \end{bmatrix},$$

where $*$ denotes a copy of the ACC value in the first row. The subscript on ACC denotes the method and the superscript the seed used for initializing the policy network $\boldsymbol{\theta}$. Therefore, we copy the values for ETS and Heur such that the $\text{ACC}_{\text{DQN}}^2$ for seed 2 can be compared against ETS and Heur. Note that there is a tie between $\text{ACC}_{\text{Heur}}^1$ and $\text{ACC}_{\text{DQN}}^1$ as they have ACC 95%. We handle ties by assigning tied methods the average of their ranks, such that the ranks for both seeds will be

$$\texttt{ranking}\left(\begin{bmatrix}\mathrm{ACC}^1_{\mathrm{ETS}} & \mathrm{ACC}^1_{\mathrm{Heur}} & \mathrm{ACC}^1_{\mathrm{DQN}} & \mathrm{ACC}^1_{\mathrm{A2C}}\\ \mathrm{ACC}^2_{\mathrm{ETS}} & \mathrm{ACC}^2_{\mathrm{Heur}} & \mathrm{ACC}^2_{\mathrm{DQN}} & \mathrm{ACC}^2_{\mathrm{A2C}}\end{bmatrix}, \texttt{axis=-1, keepdim=True}\right)$$

$$=\texttt{ranking}\left(\begin{bmatrix}90\% & 95\% & 95\% & 99\%\\ 90\% & 95\% & 97\% & 98\%\end{bmatrix}, \texttt{axis=-1, keepdim=True}\right)$$

$$=\begin{bmatrix}4 & 2.5 & 2.5 & 1\\ 4 & 3 & 2 & 1\end{bmatrix},$$

where we inserted the copied values, such that $\mathrm{ACC}^1_{\mathrm{ETS}} = \mathrm{ACC}^2_{\mathrm{ETS}} = 90\%$ and $\mathrm{ACC}^1_{\mathrm{Heur}} = \mathrm{ACC}^2_{\mathrm{Heur}} = 95\%$. The mean ranking across the seeds thus becomes

$$\texttt{mean}\left(\begin{bmatrix}4 & 2.5 & 2.5 & 1\\ 4 & 3 & 2 & 1\end{bmatrix}, \texttt{axis=0}\right) = \begin{bmatrix}4 & 2.75 & 2.25 & 1\end{bmatrix}$$

where A2C comes in 1st place, DQN in 2nd, Heur. in 3rd, and ETS on 4th place. We average across seeds and environments to obtain the final ranking score for each method for comparison.

II Additional Methodology

We provide additional information on the construction of the action space with task proportions in Appendix II.1, and outline the training procedure of the replay scheduling policy in Appendix II.2.

II.1 Discrete Action Space with Task Proportions

We show the procedure for creating the discrete action space of task proportions in Algorithm 1 which was used in [2]. At task i, we have $i-1$ historical tasks that we can choose from. We then generate all possible bin vectors $\boldsymbol{b}_i = [b_1, \ldots, b_i] \in \mathcal{B}_i$ 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 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.

II.2 RL Framework Algorithm

We provide pseudo-code for the RL-based framework for learning the replay scheduling policy with either DQN [4] or A2C [6] in Algorithm 2. The procedure collects experience from all training environments in $\mathcal{E}^{(train)}$ at every time step t. The datasets

Algorithm 1 Discretization of action space with task proportions

Require: Number of tasks T

$\mathcal{T} = ()$ ▷ Initialize sequence for storing actions
for $i = 1, \ldots, T-1$ **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

and classifiers are specific for each environment $E_i \in \mathcal{E}^{(train)}$. At $t = 1$, we obtain the initial state $s_1^{(i)}$ by evaluating the classifier on the validation set $\mathcal{D}_1^{(val)}$ after training the classifier on the task 1. Next, we get the replay memory for mitigating catastrophic forgetting when learning the next task $t+1$ by 1) taking action $a_t^{(i)}$ under policy $\pi_{\boldsymbol{\theta}}$, 2) converting action $a_t^{(i)}$ into the task proportion $\boldsymbol{p}_t$, and 3) sampling the replay memory $\mathcal{M}_t$ from the historical datasets given the selected proportion. We then obtain the reward r_t and the next state s_{t+1} by evaluating the classifier on the validation sets $\mathcal{D}_{1:t+1}^{(val)}$ after learning task $t+1$. The collected experience from each time step is stored in the experience buffer $\mathcal{B}$ for both DQN and A2C. In UpdatePolicy$(\cdot)$, we outline the steps for updating the policy parameters $\boldsymbol{\theta}$ with either DQN or A2C.

Algorithm 2 RL Framework for Learning Replay Scheduling Policy

Require: $\mathcal{E}^{(train)}$: Training environments, $\boldsymbol{\theta}$: Policy parameters, γ: Discount factor
Require: η: Learning rate, $n_{episodes}$: Number of episodes, M: Replay memory size
Require: n_{steps}: Number of steps for A2C
$\mathcal{B} = \{\}$ ▷ Initialize experience buffer
for $i = 1, \ldots, n_{\text{episodes}}$ **do**
  for $t = 1, \ldots, T-1$ **do**
    for $E_i \in \mathcal{E}^{(train)}$ **do**
      $\mathcal{D}_{1:t+1} = \textsc{GetDatasets}(E_i, t)$ ▷ Get datasets from environment E_i
      $f_{\boldsymbol{\phi}}^{(i)} = \textsc{GetClassifier}(E_i)$ ▷ Get classifier from environment E_i
      if $t == 1$ **then**
        $\textsc{Train}(f_{\boldsymbol{\phi}}^{(i)}, \mathcal{D}_t^{(train)})$ ▷ Train classifier $f_{\boldsymbol{\phi}}^{(i)}$ on task 1
        $A_{1:t}^{(val)} = \textsc{Eval}(f_{\boldsymbol{\phi}}^{(i)}, \mathcal{D}_{1:t}^{(val)})$ ▷ Evaluate classifier $f_{\boldsymbol{\phi}}^{(i)}$ on task 1
        $s_t^{(i)} = A_{1:t}^{(val)} = [A_{1,1}^{(val)}, 0, ..., 0]$ ▷ Get initial state
      end if
      $a_t^{(i)} \sim \pi_{\boldsymbol{\theta}}(a, s_t^{(i)})$ ▷ Take action under policy $\pi_{\boldsymbol{\theta}}$
      $\boldsymbol{p}_t = \textsc{GetTaskProportion}(a_t^{(i)})$
      $\mathcal{M}_t \sim \textsc{GetReplayMemory}(\mathcal{D}_{1:t}^{(train)}, \boldsymbol{p}_t, M)$
      $\textsc{Train}(f_{\boldsymbol{\phi}}^{(i)}, \mathcal{D}_{t+1}^{(train)} \cup \mathcal{M}_t)$ ▷ Train classifier $f_{\boldsymbol{\phi}}^{(i)}$
      $A_{1:t+1}^{(val)} = \textsc{Eval}(f_{\boldsymbol{\phi}}^{(i)}, \mathcal{D}_{1:t+1}^{(val)})$ ▷ Evaluate classifier $f_{\boldsymbol{\phi}}^{(i)}$
      $s_{t+1}^{(i)} = A_{1:t+1}^{(val)} = [A_{t+1,1}^{(val)}, ..., A_{t+1,t+1}^{(val)}, 0, ..., 0]$ ▷ Get next state
      $r_t^{(i)} = \frac{1}{t+1} \sum_{j=1}^{t+1} A_{1:t+1}^{(val)}$ ▷ Compute reward
      $\mathcal{B} = \mathcal{B} \cup \{(s_t^{(i)}, a_t^{(i)}, r_t^{(i)}, s_{t+1}^{(i)})\}$ ▷ Store transition in buffer
      if time to update policy **then**
        $\boldsymbol{\theta}, \mathcal{B} = \textsc{UpdatePolicy}(\boldsymbol{\theta}, \mathcal{B}, \gamma, \eta, n_{steps})$ ▷ Update policy with experience
      end if
    end for
  end for
end for
return $\boldsymbol{\theta}$ ▷ Return policy

function $\textsc{UpdatePolicy}(\boldsymbol{\theta}, \mathcal{B}, \gamma, \eta, n_{steps})$
  if DQN **then**
    $(s_j, a_j, r_j, s'_j) \sim \mathcal{B}$ ▷ Sample mini-batch from buffer
    $y_j = \begin{cases} r_j & \text{if } s'_j \text{ is terminal} \\ r_j + \gamma \max_a Q_{\boldsymbol{\theta}^-}(s'_j, a) & \text{else} \end{cases}$ ▷ Compute y_j with target net $\boldsymbol{\theta}^-$
    $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} (y_j - Q_{\boldsymbol{\theta}}(s_j, a_j))^2$ ▷ Update Q-function
  else if A2C **then**
    $s_t = \mathcal{B}[n_{steps}]$ ▷ Get last state in buffer
    $R = \begin{cases} 0 & \text{if } s_t \text{ is terminal} \\ V_{\boldsymbol{\theta}_v}(s_t) & \text{else} \end{cases}$ ▷ Bootstrap from last state
    for $j = n_{steps} - 1, ..., 0$ **do**
      $s_j, a_j, r_j = \mathcal{B}[j]$ ▷ Get state, action, and reward at step j
      $R = r_j + \gamma R$
      $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_j, s_j)(R - V_{\boldsymbol{\theta}_v}(s_j))$ ▷ Update policy
      $\boldsymbol{\theta}_v = \boldsymbol{\theta}_v - \eta \nabla_{\boldsymbol{\theta}_v} (R - V_{\boldsymbol{\theta}_v}(s_j))^2$ ▷ Update value function
    end for
    $\mathcal{B} = \{\}$ ▷ Reset experience buffer
  end if
  return $\boldsymbol{\theta}, \mathcal{B}$
end function

III Task Splits in Test Environments in Policy Generalization Experiments

Here, we provide the task splits of the test environments used in the policy generalization experiments in Section 5.1. We evaluated all methods using 10 test environments in all experiments. The test environments in the New Task Order experiments were generated with seeds 10-19. We show the task splits for the Split MNIST, Split FashionMNIST, and Split CIFAR-10 environments in Table 7, 8, and 9 respectively. The test environments in the New Dataset experiments were generated with seeds 0-9. We show the task splits for the Split notMNIST and Split FashionMNIST environments in Table 10 and 11 respectively.

Table 7: Task splits with their corresponding seed for test environments of Split MNIST datasets in the **New Task Orders** experiments in Section 5.1.

Seed	Task 1	Task 2	Task 3	Task 4	Task 5
10	8, 2	5, 6	3, 1	0, 7	4, 9
11	7, 8	2, 6	4, 5	1, 3	0, 9
12	5, 8	7, 0	4, 9	3, 2	1, 6
13	3, 5	6, 1	4, 7	8, 9	0, 2
14	3, 9	0, 5	4, 2	1, 7	6, 8
15	2, 6	1, 3	7, 0	9, 4	5, 8
16	6, 2	0, 7	8, 4	3, 1	5, 9
17	7, 2	5, 3	4, 0	9, 8	6, 1
18	7, 9	0, 4	2, 1	6, 5	8, 3
19	1, 7	9, 6	8, 4	3, 0	2, 5

Table 8: Task splits with their corresponding seed for test environments of Split FashionMNIST datasets in the **New Task Orders** experiments in Section 5.1.

Seed	Task 1	Task 2	Task 3	Task 4	Task 5
10	Bag, Pullover	Sandal, Shirt	Dress, Trouser	T-shirt/top, Sneaker	Coat, Ankle boot
11	Sneaker, Bag	Pullover, Shirt	Coat, Sandal	Trouser, Dress	T-shirt/top, Ankle boot
12	Sandal, Bag	Sneaker, T-shirt/top	Coat, Ankle boot	Dress, Pullover	Trouser, Shirt
13	Dress, Sandal	Shirt, Trouser	Coat, Sneaker	Bag, Ankle boot	T-shirt/top, Pullover
14	Dress, Ankle boot	T-shirt/top, Sandal	Coat, Pullover	Trouser, Sneaker	Shirt, Bag
15	Pullover, Shirt	Trouser, Dress	Sneaker, T-shirt/top	Ankle boot, Coat	Sandal, Bag
16	Shirt, Pullover	T-shirt/top, Sneaker	Bag, Coat	Dress, Trouser	Sandal, Ankle boot
17	Sneaker, Pullover	Sandal, Dress	Coat, T-shirt/top	Ankle boot, Bag	Shirt, Trouser
18	Sneaker, Ankle boot	T-shirt/top, Coat	Pullover, Trouser	Shirt, Sandal	Bag, Dress
19	Trouser, Sneaker	Ankle boot, Shirt	Bag, Coat	Dress, T-shirt/top	Pullover, Sandal

Table 9: Task splits with their corresponding seed for test environments of Split CIFAR-10 datasets in the **New Task Orders** experiments in Section 5.1.

Seed	Task 1	Task 2	Task 3	Task 4	Task 5
10	Ship, Bird	Dog, Frog	Cat, Automobile	Airplane, Horse	Deer, Truck
11	Horse, Ship	Bird, Frog	Deer, Dog	Automobile, Cat	Airplane, Truck
12	Dog, Ship	Horse, Airplane	Deer, Truck	Cat, Bird	Automobile, Frog
13	Cat, Dog	Frog, Automobile	Deer, Horse	Ship, Truck	Airplane, Bird
14	Cat, Truck	Airplane, Dog	Deer, Bird	Automobile, Horse	Frog, Ship
15	Bird, Frog	Automobile, Cat	Horse, Airplane	Truck, Deer	Dog, Ship
16	Frog, Bird	Airplane, Horse	Ship, Deer	Cat, Automobile	Dog, Truck
17	Horse, Bird	Dog, Cat	Deer, Airplane	Truck, Ship	Frog, Automobile
18	Horse, Truck	Airplane, Deer	Bird, Automobile	Frog, Dog	Ship, Cat
19	Automobile, Horse	Truck, Frog	Ship, Deer	Cat, Airplane	Bird, Dog

Table 10: Task splits with their corresponding seed for test environments of Split notMNIST datasets in the **New Dataset** experiments in Section 5.1.

Seed	Task 1	Task 2	Task 3	Task 4	Task 5
0	A, B	C, D	E, F	G, H	I, J
1	C, J	G, E	A, D	B, H	I, F
2	E, B	F, A	H, C	D, G	J, I
3	F, E	B, C	J, G	H, A	D, I
4	D, I	E, J	C, G	A, B	F, H
5	J, F	C, E	H, B	A, I	G, D
6	I, B	H, A	G, F	C, E	D, J
7	I, F	A, C	B, J	H, D	G, E
8	I, G	J, A	C, F	H, B	E, D
9	I, E	H, C	B, J	D, A	G, F

Table 11: Task splits with their corresponding seed for test environments of Split FashionMNIST datasets in the **New Dataset** experiments in Section 5.1.

Seed	Task 1	Task 2	Task 3	Task 4	Task 5
0	T-shirt/top, Trouser	Pullover, Dress	Coat, Sandal	Shirt, Sneaker	Bag, Ankle boot
1	Pullover, Ankle boot	Shirt, Coat	T-shirt/top, Dress	Trouser, Sneaker	Bag, Sandal
2	Coat, Trouser	Sandal, T-shirt/top	Sneaker, Pullover	Dress, Shirt	Ankle boot, Bag
3	Sandal, Coat	Trouser, Pullover	Ankle boot, Shirt	Sneaker, T-shirt/top	Dress, Bag
4	Dress, Bag	Coat, Ankle boot	Pullover, Shirt	T-shirt/top, Trouser	Sandal, Sneaker
5	Ankle boot, Sandal	Pullover, Coat	Sneaker, Trouser	T-shirt/top, Bag	Shirt, Dress
6	Bag, Trouser	Sneaker, T-shirt/top	Shirt, Sandal	Pullover, Coat	Dress, Ankle boot
7	Bag, Sandal	T-shirt/top, Pullover	Trouser, Ankle boot	Sneaker, Dress	Shirt, Coat
8	Bag, Shirt	Ankle boot, T-shirt/top	Pullover, Sandal	Sneaker, Trouser	Coat, Dress
9	Bag, Coat	Sneaker, Pullover	Trouser, Ankle boot	Dress, T-shirt/top	Shirt, Sandal

IV Additional Experimental Results

Here, we provide complementary results for the policy generalization experiments in Section 5.1. Figure 5-8 shows the performance progress measured in ACC in the test environments during training for the DQN and A2C. We also show the ACC achieved by the scheduling baselines as straight lines as these policies are fixed. Figure 5 and 6 shows the performance progress for test environments with Split MNIST and Split CIFAR-10 respectively in the New Task Orders experiment. Figure 7 and 8 shows the performance progress for test environments with Split notMNIST and Split FashionMNIST respectively in the New Dataset experiment. In general, we observe that DQN exhibits noisier progress of the achieved ACC in the test environments than A2C.

In Table 12-16, we display the ACC and rank in every test environment that was used for generating the average rankings in Table 1 (see Section 5.1). Table 12, 13, and 14 shows the ACCs and ranks for the New Task Orders experiments with datasets Split MNIST, FashionMNIST, and CIFAR-10 respectively. Table 15 and 16 shows the ACCs and ranks for the New Dataset experiments with datasets Split notMNIST and FashionMNIST respectively. The numbers for the average rankings in Table 1 are computed by averaging over the ranks in each test environment for every method separately.

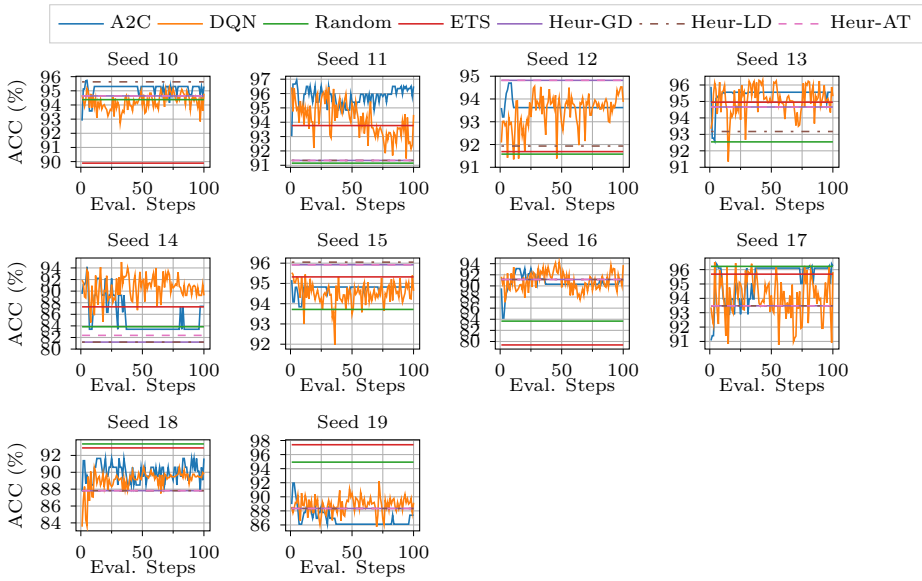


Figure 5: Performance progress measured in ACC (%) for all methods in the 10 test environments for **Split MNIST** in the New Task Orders experiment. We plot the performance progress for A2C and DQN for 100 evaluation steps equidistantly distributed over the training episodes. The performance for the Random, ETS, and Heuristic scheduling baselines are plotted as straight lines.

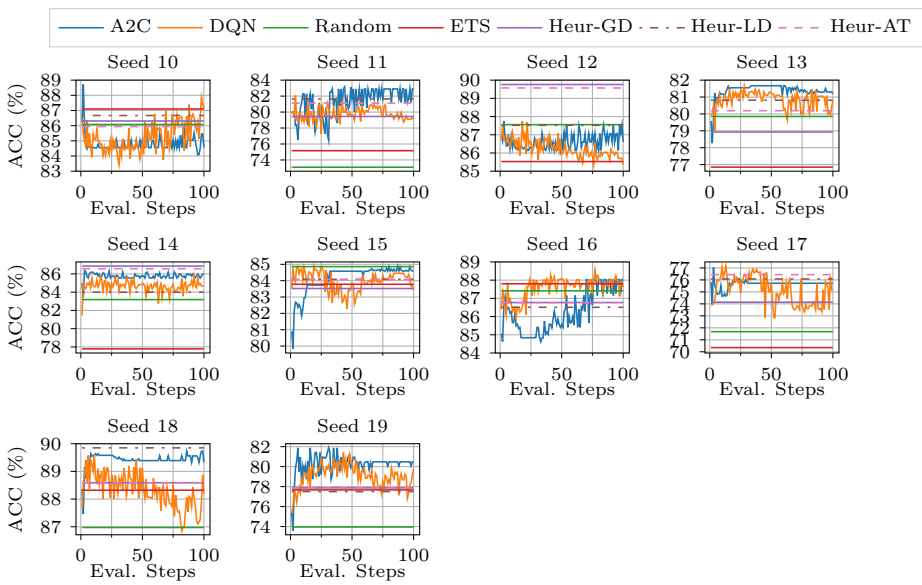


Figure 6: Performance progress measured in ACC (%) for all methods in the 10 test environments for **Split CIFAR-10** in the New Task Orders experiment. We plot the performance progress for A2C and DQN for 100 evaluation steps equidistantly distributed over the training episodes. The performance for the Random, ETS, and Heuristic scheduling baselines are plotted as straight lines.

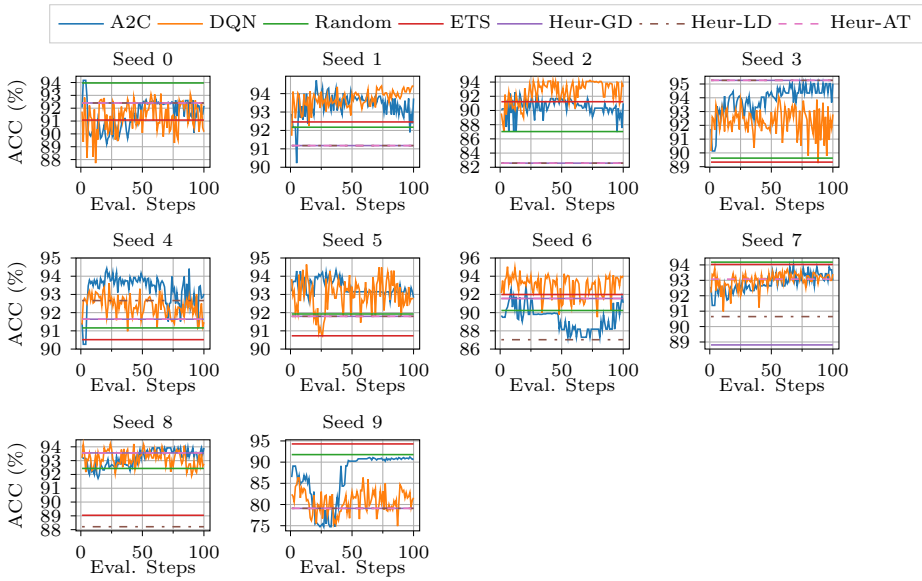


Figure 7: Performance progress measured in ACC (%) for all methods in the 10 test environments for **Split notMNIST** in the New Dataset experiment. We plot the performance progress for A2C and DQN for 100 evaluation steps equidistantly distributed over the training episodes. The performance for the Random, ETS, and Heuristic scheduling baselines are plotted as straight lines.

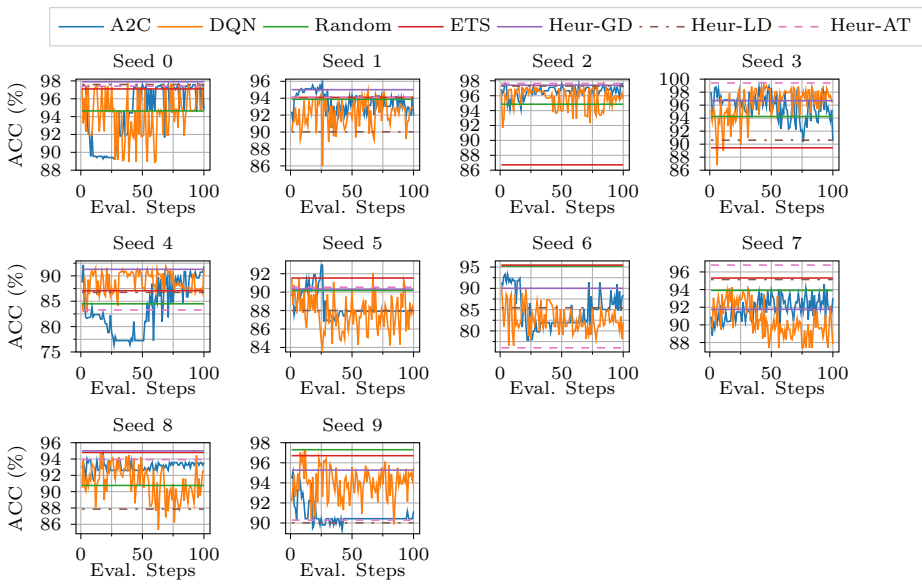


Figure 8: Performance progress measured in ACC (%) for all methods in the 10 test environments for **Split FashionMNIST** in the New Dataset experiment. We plot the performance progress for A2C and DQN for 100 evaluation steps equidistantly distributed over the training episodes. The performance for the Random, ETS, and Heuristic scheduling baselines are plotted as straight lines.

Table 12: The ACC and rank for each seed (10-19) in every test environment with **Split MNIST** for New Task Order experiment. Under each column named 'Seed X', we show the ACC (%) and, in parenthesis, the rank for the corresponding method for the specific seed. We also show the average rank of each method in the test environment under 'Rank'. Note that the ACC for ETS, Heur-GD, Heur-LD, and Heur-AT are copied across the seeds, since the seed only affects the policy initialization in DQN and A2C and the random selection in Random.

	Test Env. Seed 10				**Test Env. Seed 11**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	93.2 (6.0)	93.7 (6.0)	96.2 (1.0)	4.33	92.2 (3.0)	90.8 (7.0)	90.5 (7.0)	5.67
ETS	89.9 (7.0)	89.9 (7.0)	89.9 (7.0)	7	93.8 (2.0)	93.8 (3.0)	93.8 (3.0)	2.67
Heur-GD	94.6 (3.5)	94.6 (4.5)	94.6 (5.5)	4.5	91.3 (6.0)	91.3 (5.0)	91.3 (5.0)	5.33
Heur-LD	95.6 (1.0)	95.6 (1.0)	95.6 (3.0)	1.67	91.3 (6.0)	91.3 (5.0)	91.3 (5.0)	5.33
Heur-AT	94.6 (3.5)	94.6 (4.5)	94.6 (5.5)	4.5	91.3 (6.0)	91.3 (5.0)	91.3 (5.0)	5.33
DQN	93.2 (5.0)	95.5 (2.0)	95.7 (2.0)	3	91.8 (4.0)	96.0 (2.0)	95.7 (2.0)	2.67
A2C	95.3 (2.0)	95.3 (3.0)	95.3 (4.0)	3	96.5 (1.0)	96.5 (1.0)	96.5 (1.0)	1
	Test Env. Seed 12				**Test Env. Seed 13**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	90.7 (7.0)	88.8 (7.0)	95.2 (1.0)	5	93.4 (5.0)	94.4 (6.0)	89.9 (7.0)	6
ETS	91.7 (6.0)	91.7 (6.0)	91.7 (7.0)	6.33	95.0 (2.0)	95.0 (3.0)	95.0 (3.0)	2.67
Heur-GD	94.8 (1.5)	94.8 (1.5)	94.8 (2.5)	1.83	94.7 (3.5)	94.7 (4.5)	94.7 (4.5)	4.17
Heur-LD	91.9 (5.0)	91.9 (5.0)	91.9 (6.0)	5.33	93.2 (7.0)	93.2 (7.0)	93.2 (6.0)	6.67
Heur-AT	94.8 (1.5)	94.8 (1.5)	94.8 (2.5)	1.83	94.7 (3.5)	94.7 (4.5)	94.7 (4.5)	4.17
DQN	94.2 (3.0)	94.5 (3.0)	92.8 (5.0)	3.67	93.2 (6.0)	96.6 (1.0)	96.2 (1.0)	2.67
A2C	93.6 (4.0)	93.6 (4.0)	93.6 (4.0)	4	95.6 (1.0)	95.6 (2.0)	95.6 (2.0)	1.67
	Test Env. Seed 14				**Test Env. Seed 15**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	84.4 (3.0)	80.5 (7.0)	86.6 (3.0)	4.33	93.8 (7.0)	95.4 (4.0)	91.9 (7.0)	6
ETS	87.3 (2.0)	87.3 (2.0)	87.3 (2.0)	2	95.3 (4.0)	95.3 (5.0)	95.3 (4.0)	4.33
Heur-GD	81.2 (6.5)	81.2 (5.5)	81.2 (6.5)	6.17	95.9 (2.5)	95.9 (2.5)	95.9 (2.5)	2.5
Heur-LD	81.2 (6.5)	81.2 (5.5)	81.2 (6.5)	6.17	96.1 (1.0)	96.1 (1.0)	96.1 (1.0)	1
Heur-AT	82.4 (5.0)	82.4 (4.0)	82.4 (5.0)	4.67	95.9 (2.5)	95.9 (2.5)	95.9 (2.5)	2.5
DQN	92.8 (1.0)	86.8 (3.0)	88.1 (1.0)	1.67	94.8 (6.0)	94.8 (7.0)	94.3 (6.0)	6.33
A2C	83.4 (4.0)	95.3 (1.0)	83.4 (4.0)	3	94.8 (5.0)	94.8 (6.0)	94.8 (5.0)	5.33
	Test Env. Seed 16				**Test Env. Seed 17**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	80.5 (6.0)	87.5 (6.0)	83.0 (6.0)	6	96.4 (1.0)	96.1 (3.0)	96.1 (1.0)	1.67
ETS	79.4 (7.0)	79.4 (7.0)	79.4 (7.0)	7	95.7 (3.0)	95.7 (4.0)	95.7 (3.0)	3.33
Heur-GD	91.2 (3.0)	91.2 (4.0)	91.2 (3.0)	3.33	93.5 (6.0)	93.5 (6.0)	93.5 (6.0)	6
Heur-LD	91.2 (3.0)	91.2 (4.0)	91.2 (3.0)	3.33	93.5 (6.0)	93.5 (6.0)	93.5 (6.0)	6
Heur-AT	91.2 (3.0)	91.2 (4.0)	91.2 (3.0)	3.33	93.5 (6.0)	93.5 (6.0)	93.5 (6.0)	6
DQN	93.9 (1.0)	92.9 (1.0)	94.5 (1.0)	1	95.7 (4.0)	96.2 (2.0)	95.7 (4.0)	3.33
A2C	90.3 (5.0)	92.3 (2.0)	90.3 (5.0)	4	96.1 (2.0)	96.5 (1.0)	96.1 (2.0)	1.67
	Test Env. Seed 18				**Test Env. Seed 19**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	92.7 (2.0)	94.2 (1.0)	93.2 (1.0)	1.33	97.3 (2.0)	95.9 (2.0)	91.6 (2.0)	2
ETS	92.9 (1.0)	92.9 (2.0)	92.9 (2.0)	1.67	97.4 (1.0)	97.4 (1.0)	97.4 (1.0)	1
Heur-GD	87.8 (6.0)	87.8 (6.0)	87.8 (6.0)	6	88.3 (5.0)	88.3 (6.0)	88.3 (4.0)	5
Heur-LD	87.8 (6.0)	87.8 (6.0)	87.8 (6.0)	6	88.3 (5.0)	88.3 (6.0)	88.3 (4.0)	5
Heur-AT	87.8 (6.0)	87.8 (6.0)	87.8 (6.0)	6	88.3 (5.0)	88.3 (6.0)	88.3 (4.0)	5
DQN	89.5 (4.0)	89.4 (4.0)	90.5 (4.0)	4	89.4 (3.0)	90.5 (3.0)	87.5 (6.0)	4
A2C	91.6 (3.0)	91.6 (3.0)	91.6 (3.0)	3	86.1 (7.0)	89.9 (4.0)	86.1 (7.0)	6

Table 13: The ACC and rank for each seed (10-19) in every test environment with **Split Fashion-MNIST** for New Task Order experiment. Under each column named 'Seed X', we show the ACC (%) and, in parenthesis, the rank for the corresponding method for the specific seed. We also show the average rank of each method under 'Rank' in the corresponding test environment. Note that the ACC for ETS, Heur-GD, Heur-LD, and Heur-AT are copied across the seeds, since the seed only affects the policy initialization in DQN and A2C and the random selection in Random.

	Test Env. Seed 10				**Test Env. Seed 11**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	98.4 (1.0)	90.8 (7.0)	98.0 (1.0)	3	88.9 (5.0)	93.4 (3.0)	89.2 (5.0)	4.33
ETS	96.1 (6.0)	96.1 (5.0)	96.1 (5.0)	5.33	88.8 (6.0)	88.8 (6.0)	88.8 (6.0)	6
Heur-GD	98.0 (2.5)	98.0 (1.5)	98.0 (2.5)	2.17	93.2 (3.0)	93.2 (4.0)	93.2 (4.0)	3.67
Heur-LD	98.0 (2.5)	98.0 (1.5)	98.0 (2.5)	2.17	94.7 (1.0)	94.7 (1.0)	94.7 (1.0)	1
Heur-AT	92.0 (7.0)	92.0 (6.0)	92.0 (7.0)	6.67	93.5 (2.0)	93.5 (2.0)	93.5 (2.0)	2
DQN	96.9 (4.5)	97.0 (4.0)	95.4 (6.0)	4.83	91.3 (4.0)	93.2 (5.0)	93.3 (3.0)	4
A2C	96.9 (4.5)	97.1 (3.0)	96.9 (4.0)	3.83	85.3 (7.0)	86.9 (7.0)	85.3 (7.0)	7
	Test Env. Seed 12				**Test Env. Seed 13**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	97.6 (1.0)	96.8 (1.0)	93.4 (4.0)	2	94.0 (6.0)	88.9 (7.0)	93.4 (6.0)	6.33
ETS	91.2 (6.0)	91.2 (6.0)	91.2 (5.0)	5.67	91.2 (7.0)	91.2 (6.0)	91.2 (7.0)	6.67
Heur-GD	95.0 (5.0)	95.0 (4.0)	95.0 (3.0)	4	95.1 (5.0)	95.1 (5.0)	95.1 (5.0)	5
Heur-LD	96.1 (2.0)	96.1 (2.0)	96.1 (1.0)	1.67	96.2 (4.0)	96.2 (4.0)	96.2 (4.0)	4
Heur-AT	95.8 (3.0)	95.8 (3.0)	95.8 (2.0)	2.67	98.1 (1.0)	98.1 (2.0)	98.1 (1.0)	1.33
DQN	95.0 (4.0)	94.8 (5.0)	88.6 (6.0)	5	96.6 (3.0)	98.4 (1.0)	97.4 (2.0)	2
A2C	87.7 (7.0)	87.7 (7.0)	87.7 (7.0)	7	98.0 (2.0)	98.0 (3.0)	96.7 (3.0)	2.67
	Test Env. Seed 14				**Test Env. Seed 15**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	94.2 (5.0)	96.0 (2.0)	93.0 (5.0)	4	94.3 (1.0)	92.6 (2.0)	93.8 (1.0)	1.33
ETS	90.0 (6.0)	90.0 (6.0)	90.0 (6.0)	6	93.5 (2.0)	93.5 (1.0)	93.5 (2.0)	1.67
Heur-GD	95.4 (1.0)	95.4 (3.0)	95.4 (2.0)	2	79.3 (7.0)	79.3 (7.0)	79.3 (7.0)	7
Heur-LD	95.1 (3.0)	95.1 (4.0)	95.1 (3.0)	3.33	92.6 (3.0)	92.6 (3.0)	92.6 (3.0)	3
Heur-AT	82.0 (7.0)	82.0 (7.0)	82.0 (7.0)	7	88.5 (5.0)	88.5 (5.0)	88.5 (4.0)	4.67
DQN	95.3 (2.0)	96.1 (1.0)	95.5 (1.0)	1.33	89.6 (4.0)	90.1 (4.0)	88.4 (5.0)	4.33
A2C	94.7 (4.0)	94.7 (5.0)	94.7 (4.0)	4.33	81.0 (6.0)	81.0 (6.0)	80.5 (6.0)	6
	Test Env. Seed 16				**Test Env. Seed 17**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	90.0 (2.0)	88.1 (3.0)	92.3 (3.0)	2.67	99.1 (3.0)	98.6 (5.0)	99.2 (4.0)	4
ETS	94.4 (1.0)	94.4 (1.0)	94.4 (1.0)	1	98.1 (7.0)	98.1 (7.0)	98.1 (7.0)	7
Heur-GD	73.8 (7.0)	73.8 (7.0)	73.8 (7.0)	7	99.4 (1.0)	99.4 (1.0)	99.4 (1.0)	1
Heur-LD	80.4 (6.0)	80.4 (6.0)	80.4 (6.0)	6	99.3 (2.0)	99.3 (2.0)	99.3 (2.0)	2
Heur-AT	89.2 (4.0)	89.2 (2.0)	89.2 (4.0)	3.33	98.7 (4.0)	98.7 (4.0)	98.7 (6.0)	4.67
DQN	87.3 (5.0)	85.7 (4.0)	92.3 (2.0)	3.67	98.7 (5.0)	99.1 (3.0)	99.0 (5.0)	4.33
A2C	89.3 (3.0)	83.2 (5.0)	87.7 (5.0)	4.33	98.7 (6.0)	98.5 (6.0)	99.2 (3.0)	5
	Test Env. Seed 18				**Test Env. Seed 19**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	87.2 (7.0)	87.2 (7.0)	94.2 (4.0)	6	96.2 (5.0)	97.8 (1.0)	97.7 (1.0)	2.33
ETS	93.6 (4.0)	93.6 (4.0)	93.6 (5.0)	4.33	97.5 (1.0)	97.5 (3.0)	97.5 (2.0)	2
Heur-GD	92.9 (5.0)	92.9 (5.0)	92.9 (6.0)	5.33	95.8 (6.0)	95.8 (5.0)	95.8 (5.0)	5.33
Heur-LD	92.1 (6.0)	92.1 (6.0)	92.1 (7.0)	6.33	93.4 (7.0)	93.4 (7.0)	93.4 (7.0)	7
Heur-AT	94.2 (2.0)	94.2 (2.0)	94.2 (3.0)	2.33	96.6 (4.0)	96.6 (4.0)	96.6 (4.0)	4
DQN	95.3 (1.0)	94.7 (1.0)	96.0 (1.0)	1	96.9 (3.0)	95.7 (6.0)	95.7 (6.0)	5
A2C	93.8 (3.0)	93.8 (3.0)	95.0 (2.0)	2.67	97.0 (2.0)	97.7 (2.0)	96.9 (3.0)	2.33

Table 14: The ACC and rank for each seed (10-19) in every test environment with **Split CIFAR-10** for New Task Order experiment. Under each column named 'Seed X', we show the ACC (%) and, in parenthesis, the rank for the corresponding method for the specific seed. We also show the average rank of each method in the test environment under 'Rank'. Note that the ACC for ETS, Heur-GD, Heur-LD, and Heur-AT are copied across the seeds, since the seed only affects the policy initialization in DQN and A2C and the random selection in Random.

	Test Env. Seed 10				**Test Env. Seed 11**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	87.3 (2.0)	85.2 (6.0)	85.7 (6.0)	4.67	75.3 (6.0)	71.1 (7.0)	72.8 (7.0)	6.67
ETS	87.1 (3.0)	87.1 (1.0)	87.1 (1.0)	1.67	75.2 (7.0)	75.2 (6.0)	75.2 (6.0)	6.33
Heur-GD	86.3 (5.0)	86.3 (3.0)	86.3 (4.0)	4	79.5 (4.0)	79.5 (5.0)	79.5 (4.0)	4.33
Heur-LD	86.7 (4.0)	86.7 (2.0)	86.7 (3.0)	3	81.6 (2.0)	81.6 (2.0)	81.6 (2.0)	2
Heur-AT	85.9 (6.0)	85.9 (4.0)	85.9 (5.0)	5	81.1 (3.0)	81.1 (3.0)	81.1 (3.0)	3
DQN	88.9 (1.0)	85.5 (5.0)	86.9 (2.0)	2.67	78.2 (5.0)	79.9 (4.0)	79.3 (5.0)	4.67
A2C	84.6 (7.0)	84.6 (7.0)	84.6 (7.0)	7	82.9 (1.0)	82.9 (1.0)	82.9 (1.0)	1
	Test Env. Seed 12				**Test Env. Seed 13**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	86.2 (4.0)	88.9 (3.0)	87.6 (3.0)	3.33	77.3 (6.0)	81.3 (2.0)	80.9 (2.0)	3.33
ETS	85.5 (7.0)	85.5 (6.0)	85.5 (7.0)	6.67	76.9 (7.0)	76.9 (7.0)	76.9 (7.0)	7
Heur-GD	89.8 (1.0)	89.8 (1.0)	89.8 (1.0)	1	78.9 (5.0)	78.9 (6.0)	78.9 (6.0)	5.67
Heur-LD	87.5 (3.0)	87.5 (4.0)	87.5 (5.0)	4	80.8 (3.0)	80.8 (4.0)	80.8 (3.0)	3.33
Heur-AT	89.6 (2.0)	89.6 (2.0)	89.6 (2.0)	2	80.2 (4.0)	80.2 (5.0)	80.2 (5.0)	4.67
DQN	86.0 (6.0)	84.4 (7.0)	85.7 (6.0)	6.33	80.8 (2.0)	81.3 (1.0)	80.2 (4.0)	2.33
A2C	86.1 (5.0)	86.1 (5.0)	87.5 (4.0)	4.67	81.3 (1.0)	81.0 (3.0)	81.3 (1.0)	1.67
	Test Env. Seed 14				**Test Env. Seed 15**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	81.7 (6.0)	84.8 (5.0)	83.0 (6.0)	5.67	83.8 (4.0)	85.0 (2.0)	85.8 (1.0)	2.33
ETS	77.8 (7.0)	77.8 (7.0)	77.8 (7.0)	7	83.8 (5.0)	83.8 (6.0)	83.8 (5.0)	5.33
Heur-GD	86.9 (1.0)	86.9 (1.0)	86.9 (1.0)	1	83.5 (6.0)	83.5 (7.0)	83.5 (6.0)	6.33
Heur-LD	84.0 (5.0)	84.0 (6.0)	84.0 (4.0)	5	84.1 (2.5)	84.1 (4.5)	84.1 (3.5)	3.5
Heur-AT	86.6 (2.0)	86.6 (2.0)	86.6 (2.0)	2	84.1 (2.5)	84.1 (4.5)	84.1 (3.5)	3.5
DQN	85.0 (4.0)	85.2 (4.0)	83.8 (5.0)	4.33	83.2 (7.0)	85.4 (1.0)	81.7 (7.0)	5
A2C	86.2 (3.0)	86.2 (3.0)	85.5 (3.0)	3	84.6 (1.0)	84.6 (3.0)	84.6 (2.0)	2
	Test Env. Seed 16				**Test Env. Seed 17**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	85.2 (7.0)	89.4 (1.0)	87.6 (4.0)	4	68.9 (7.0)	72.1 (6.0)	74.0 (6.0)	6.33
ETS	87.8 (3.0)	87.8 (3.0)	87.8 (2.0)	2.67	70.3 (5.0)	70.3 (7.0)	70.3 (7.0)	6.33
Heur-GD	86.8 (4.5)	86.8 (5.5)	86.8 (5.5)	5.17	74.1 (4.0)	74.1 (5.0)	74.1 (5.0)	4.67
Heur-LD	86.5 (6.0)	86.5 (7.0)	86.5 (7.0)	6.67	76.1 (2.0)	76.1 (3.0)	76.1 (3.0)	2.67
Heur-AT	86.8 (4.5)	86.8 (5.5)	86.8 (5.5)	5.17	76.4 (1.0)	76.4 (1.0)	76.4 (2.0)	1.33
DQN	87.9 (2.0)	87.4 (4.0)	87.6 (3.0)	3	69.9 (6.0)	76.4 (2.0)	79.2 (1.0)	3
A2C	88.0 (1.0)	88.0 (2.0)	88.0 (1.0)	1.33	75.7 (3.0)	75.7 (4.0)	75.7 (4.0)	3.67
	Test Env. Seed 18				**Test Env. Seed 19**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	86.9 (7.0)	87.6 (7.0)	86.4 (7.0)	7	71.2 (7.0)	77.5 (7.0)	73.2 (7.0)	7
ETS	88.3 (6.0)	88.3 (6.0)	88.3 (5.0)	5.67	77.7 (5.0)	77.7 (5.0)	77.7 (5.0)	5
Heur-GD	88.6 (4.5)	88.6 (4.5)	88.6 (3.5)	4.17	77.9 (3.5)	77.9 (3.5)	77.9 (3.5)	3.5
Heur-LD	89.8 (1.0)	89.8 (2.0)	89.8 (1.0)	1.33	77.5 (6.0)	77.5 (6.0)	77.5 (6.0)	6
Heur-AT	88.6 (4.5)	88.6 (4.5)	88.6 (3.5)	4.17	77.9 (3.5)	77.9 (3.5)	77.9 (3.5)	3.5
DQN	89.1 (2.0)	88.6 (3.0)	86.8 (6.0)	3.67	80.5 (2.0)	80.5 (1.5)	78.5 (2.0)	1.83
A2C	88.7 (3.0)	89.9 (1.0)	89.4 (2.0)	2	80.5 (1.0)	80.5 (1.5)	80.5 (1.0)	1.17

Table 15: The ACC and rank for each seed (0-9) in every test environment with **Split notMNIST** for New Dataset experiment. Under each column named 'Seed X', we show the ACC (%) and, in parenthesis, the rank for the corresponding method for the specific seed. We also show the average rank of each method in the test environment under 'Rank'. Note that the ACC for ETS, Heur-GD, Heur-LD, and Heur-AT are copied across the seeds, since the seed only affects the policy initialization in DQN and A2C and the random selection in Random.

	Test Env. Seed 0				**Test Env. Seed 1**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	93.8 (1.0)	94.9 (1.0)	93.2 (2.0)	1.33	93.5 (2.0)	92.0 (4.0)	91.0 (7.0)	4.33
ETS	91.1 (5.0)	91.1 (7.0)	91.1 (6.0)	6	92.5 (4.0)	92.5 (3.0)	92.5 (3.0)	3.33
Heur-GD	92.4 (3.0)	92.4 (5.0)	92.4 (4.0)	4	91.2 (6.0)	91.2 (6.0)	91.2 (5.0)	5.67
Heur-LD	92.4 (3.0)	92.4 (5.0)	92.4 (4.0)	4	91.2 (6.0)	91.2 (6.0)	91.2 (5.0)	5.67
Heur-AT	92.4 (3.0)	92.4 (5.0)	92.4 (4.0)	4	91.2 (6.0)	91.2 (6.0)	91.2 (5.0)	5.67
DQN	90.7 (6.0)	94.5 (2.0)	85.3 (7.0)	5	94.2 (1.0)	94.6 (1.0)	94.5 (1.0)	1
A2C	90.6 (7.0)	92.7 (3.0)	93.3 (1.0)	3.67	93.5 (3.0)	93.9 (2.0)	93.9 (2.0)	2.33
	Test Env. Seed 2				**Test Env. Seed 3**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	82.1 (7.0)	90.5 (3.0)	88.5 (4.0)	4.67	94.2 (5.0)	89.5 (6.0)	85.2 (7.0)	6
ETS	91.2 (3.0)	91.2 (2.0)	91.2 (3.0)	2.67	89.3 (7.0)	89.3 (7.0)	89.3 (6.0)	6.67
Heur-GD	82.6 (5.0)	82.6 (6.0)	82.6 (6.0)	5.67	95.3 (2.0)	95.3 (2.0)	95.3 (2.0)	2
Heur-LD	82.6 (5.0)	82.6 (6.0)	82.6 (6.0)	5.67	95.3 (2.0)	95.3 (2.0)	95.3 (2.0)	2
Heur-AT	82.6 (5.0)	82.6 (6.0)	82.6 (6.0)	5.67	95.3 (2.0)	95.3 (2.0)	95.3 (2.0)	2
DQN	93.9 (1.0)	94.2 (1.0)	92.0 (1.0)	1	93.3 (6.0)	92.8 (5.0)	92.3 (5.0)	5.33
A2C	91.7 (2.0)	89.7 (4.0)	91.7 (2.0)	2.67	95.0 (4.0)	95.0 (4.0)	95.0 (4.0)	4
	Test Env. Seed 4				**Test Env. Seed 5**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	90.2 (7.0)	93.2 (1.0)	90.1 (7.0)	5	90.7 (7.0)	93.7 (1.0)	91.4 (6.0)	4.67
ETS	90.5 (6.0)	90.5 (7.0)	90.5 (6.0)	6.33	90.7 (6.0)	90.7 (7.0)	90.7 (7.0)	6.67
Heur-GD	91.6 (3.5)	91.6 (4.5)	91.6 (4.5)	4.17	91.8 (4.0)	91.8 (5.0)	91.8 (4.0)	4.33
Heur-LD	92.7 (2.0)	92.7 (3.0)	92.7 (2.0)	2.33	91.8 (4.0)	91.8 (5.0)	91.8 (4.0)	4.33
Heur-AT	91.6 (3.5)	91.6 (4.5)	91.6 (4.5)	4.17	91.8 (4.0)	91.8 (5.0)	91.8 (4.0)	4.33
DQN	91.3 (5.0)	91.2 (6.0)	92.0 (3.0)	4.67	93.2 (1.0)	92.1 (3.0)	93.0 (1.0)	1.67
A2C	93.1 (1.0)	93.1 (2.0)	93.0 (1.0)	1.33	92.6 (2.0)	93.1 (2.0)	92.8 (2.0)	2
	Test Env. Seed 6				**Test Env. Seed 7**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	91.6 (4.0)	87.2 (6.0)	91.9 (3.0)	4.33	95.0 (1.0)	94.3 (1.0)	93.3 (3.0)	1.67
ETS	92.0 (3.0)	92.0 (2.0)	92.0 (2.0)	2.33	94.0 (2.0)	94.0 (2.0)	94.0 (1.0)	1.67
Heur-GD	91.6 (5.5)	91.6 (3.5)	91.6 (4.5)	4.5	88.8 (7.0)	88.8 (7.0)	88.8 (7.0)	7
Heur-LD	87.0 (7.0)	87.0 (7.0)	87.0 (7.0)	7	90.6 (6.0)	90.6 (6.0)	90.6 (6.0)	6
Heur-AT	91.6 (5.5)	91.6 (3.5)	91.6 (4.5)	4.5	93.1 (4.0)	93.1 (4.0)	93.1 (4.0)	4
DQN	93.1 (2.0)	95.2 (1.0)	93.8 (1.0)	1.33	94.0 (3.0)	92.6 (5.0)	93.7 (2.0)	3.33
A2C	93.3 (1.0)	90.0 (5.0)	90.0 (6.0)	4	92.6 (5.0)	93.7 (3.0)	92.6 (5.0)	4.33
	Test Env. Seed 8				**Test Env. Seed 9**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	92.3 (5.0)	92.6 (4.0)	92.4 (4.0)	4.33	90.7 (3.0)	91.2 (2.0)	93.4 (2.0)	2.33
ETS	89.0 (6.0)	89.0 (6.0)	89.0 (6.0)	6	94.3 (1.0)	94.3 (1.0)	94.3 (1.0)	1
Heur-GD	93.6 (3.5)	93.6 (2.5)	93.6 (2.5)	2.83	79.1 (6.0)	79.1 (5.0)	79.1 (6.0)	5.67
Heur-LD	88.2 (7.0)	88.2 (7.0)	88.2 (7.0)	7	79.1 (6.0)	79.1 (5.0)	79.1 (6.0)	5.67
Heur-AT	93.6 (3.5)	93.6 (2.5)	93.6 (2.5)	2.83	79.1 (6.0)	79.1 (5.0)	79.1 (6.0)	5.67
DQN	94.6 (1.0)	91.6 (5.0)	92.1 (5.0)	3.67	82.7 (4.0)	79.1 (7.0)	82.9 (4.0)	5
A2C	93.9 (2.0)	93.9 (1.0)	93.9 (1.0)	1.33	90.8 (2.0)	90.8 (3.0)	90.8 (3.0)	2.67

Table 16: The ACC and rank for each seed (0-9) in every test environment with **Split Fashion-MNIST** for New Dataset experiment. Under each column named 'Seed X', we show the ACC (%) and, in parenthesis, the rank for the corresponding method for the specific seed. We also show the average rank of each method in the test environment under 'Rank'. Note that the ACC for ETS, Heur-GD, Heur-LD, and Heur-AT are copied across the seeds, since the seed only affects the policy initialization in DQN and A2C and the random selection in Random.

	Test Env. Seed 0				**Test Env. Seed 1**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	94.7 (7.0)	97.7 (2.0)	91.5 (7.0)	5.33	92.8 (5.0)	95.1 (1.0)	93.7 (6.0)	4
ETS	97.1 (6.0)	97.1 (6.0)	97.1 (6.0)	6	94.1 (2.0)	94.1 (3.0)	94.1 (3.0)	2.67
Heur-GD	97.9 (1.0)	97.9 (1.0)	97.9 (1.0)	1	95.0 (1.0)	95.0 (2.0)	95.0 (1.0)	1.33
Heur-LD	97.6 (3.0)	97.6 (3.0)	97.6 (3.0)	3	90.0 (7.0)	90.0 (7.0)	90.0 (7.0)	7
Heur-AT	97.4 (4.0)	97.4 (5.0)	97.4 (5.0)	4.67	94.1 (3.0)	94.1 (4.0)	94.1 (4.0)	3.67
DQN	97.7 (2.0)	97.4 (4.0)	97.6 (4.0)	3.33	94.0 (4.0)	90.1 (6.0)	94.6 (2.0)	4
A2C	97.3 (5.0)	89.1 (7.0)	97.7 (2.0)	4.67	90.9 (6.0)	90.9 (5.0)	94.0 (5.0)	5.33
	Test Env. Seed 2				**Test Env. Seed 3**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	94.6 (6.0)	95.1 (6.0)	94.8 (5.0)	5.67	89.5 (6.0)	99.4 (1.5)	93.9 (4.0)	3.83
ETS	86.7 (7.0)	86.7 (7.0)	86.7 (7.0)	7	89.4 (7.0)	89.4 (6.0)	89.4 (6.0)	6.33
Heur-GD	97.4 (2.0)	97.4 (3.0)	97.4 (3.0)	2.67	96.7 (3.0)	96.7 (3.0)	96.7 (2.0)	2.67
Heur-LD	97.3 (3.0)	97.3 (4.0)	97.3 (4.0)	3.67	90.6 (5.0)	90.6 (5.0)	90.6 (5.0)	5
Heur-AT	97.7 (1.0)	97.7 (1.0)	97.7 (1.0)	1	99.4 (1.0)	99.4 (1.5)	99.4 (1.0)	1.17
DQN	96.6 (4.0)	95.9 (5.0)	94.3 (6.0)	5	97.3 (2.0)	96.6 (4.0)	96.0 (3.0)	3
A2C	96.2 (5.0)	97.6 (2.0)	97.6 (2.0)	3	93.8 (4.0)	88.6 (7.0)	89.2 (7.0)	6
	Test Env. Seed 4				**Test Env. Seed 5**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	88.6 (4.0)	84.6 (6.0)	80.3 (7.0)	5.67	88.7 (5.0)	93.9 (1.0)	88.0 (5.0)	3.67
ETS	87.1 (5.0)	87.1 (4.0)	87.1 (4.0)	4.33	91.5 (1.0)	91.5 (2.0)	91.5 (1.0)	1.33
Heur-GD	91.3 (3.0)	91.3 (3.0)	91.3 (2.0)	2.67	90.3 (3.0)	90.3 (4.0)	90.3 (3.0)	3.33
Heur-LD	86.8 (6.0)	86.8 (5.0)	86.8 (5.0)	5.33	88.0 (6.0)	88.0 (5.0)	88.0 (4.0)	5
Heur-AT	83.3 (7.0)	83.3 (7.0)	83.3 (6.0)	6.67	90.5 (2.0)	90.5 (3.0)	90.5 (2.0)	2.33
DQN	91.3 (2.0)	92.2 (1.0)	90.5 (3.0)	2	89.2 (4.0)	87.1 (7.0)	86.1 (7.0)	6
A2C	91.8 (1.0)	91.8 (2.0)	91.8 (1.0)	1.33	87.9 (7.0)	87.9 (6.0)	87.9 (6.0)	6.33
	Test Env. Seed 6				**Test Env. Seed 7**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	95.3 (2.0)	94.0 (2.0)	96.3 (1.0)	1.67	93.3 (4.0)	94.0 (4.0)	94.4 (5.0)	4.33
ETS	95.5 (1.0)	95.5 (1.0)	95.5 (2.0)	1.33	95.3 (2.0)	95.3 (2.0)	95.3 (3.0)	2.33
Heur-GD	90.0 (3.0)	90.0 (4.0)	90.0 (4.0)	3.67	91.8 (5.0)	91.8 (5.0)	91.8 (6.0)	5.33
Heur-LD	85.4 (4.0)	85.4 (5.0)	85.4 (5.0)	4.67	95.1 (3.0)	95.1 (3.0)	95.1 (4.0)	3.33
Heur-AT	76.0 (7.0)	76.0 (7.0)	76.0 (7.0)	7	96.8 (1.0)	96.8 (1.0)	96.8 (1.0)	1
DQN	80.1 (6.0)	84.3 (6.0)	81.6 (6.0)	6	87.4 (7.0)	88.8 (7.0)	87.4 (7.0)	7
A2C	81.9 (5.0)	91.7 (3.0)	91.5 (3.0)	3.67	91.7 (6.0)	91.3 (6.0)	96.1 (2.0)	4.67
	Test Env. Seed 8				**Test Env. Seed 9**			
Method	Seed 1	Seed 2	Seed 3	Rank	Seed 1	Seed 2	Seed 3	Rank
Random	88.6 (6.0)	87.9 (6.0)	95.8 (1.0)	4.33	97.2 (1.0)	97.0 (1.0)	97.7 (1.0)	1
ETS	94.8 (2.0)	94.8 (3.0)	94.8 (3.0)	2.67	96.7 (3.0)	96.7 (2.0)	96.7 (2.0)	2.33
Heur-GD	95.0 (1.0)	95.0 (1.0)	95.0 (2.0)	1.33	95.3 (4.0)	95.3 (3.0)	95.3 (4.0)	3.67
Heur-LD	87.9 (7.0)	87.9 (7.0)	87.9 (7.0)	7	90.0 (7.0)	90.0 (7.0)	90.0 (7.0)	7
Heur-AT	93.9 (3.0)	93.9 (4.0)	93.9 (4.0)	3.67	90.3 (6.0)	90.3 (6.0)	90.3 (6.0)	6
DQN	91.8 (5.0)	95.0 (2.0)	91.0 (6.0)	4.33	96.9 (2.0)	93.7 (4.0)	95.9 (3.0)	3
A2C	93.6 (4.0)	93.6 (5.0)	93.6 (5.0)	4.67	90.4 (5.0)	92.6 (5.0)	90.4 (5.0)	5

References

[1] 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.

[2] Marcus Klasson, Hedvig Kjellström, and Cheng Zhang. Learn the time to learn: Replay scheduling for continual learning. *International Conference on Machine Learning (ICML) 2021 Workshop on Theory and Foundations of Continual Learning*, 2021.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

[14] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 12, 1999.

[15] 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.

[16] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

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

[18] 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.

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

[20] 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.

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

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

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

[28] Ju Xu and Zhanxing Zhu. Reinforced continual learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[29] 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.

[30] 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.

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

[34] Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9385–9394, 2021.

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

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

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

[38] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning*, 2020.

[39] 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.

[40] Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient based memory editing for task-free continual learning. *arXiv preprint arXiv:2006.15294*, 2020.

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

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

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

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

[45] 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.

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

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

[48] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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

[50] Chenyang Zhao, Olivier Sigaud, Freek Stulp, and Timothy M Hospedales. Investigating generalisation in continuous deep reinforcement learning. *arXiv preprint arXiv:1902.07015*, 2019.

[51] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

[52] 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.

[53] 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.

[54] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pages 1480–1490. PMLR, 2017.

[55] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

[56] 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.

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

[58] Yash Chandak, Georgios Theocharous, Chris Nota, and Philip Thomas. Lifelong learning with a changing action set. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3373–3380, 2020.

[59] 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.

[60] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

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

[62] 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.

[63] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

[64] 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.

[65] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[66] 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.

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

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

[69] 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.

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

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

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

[73] 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.

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

[75] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

[76] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[77] Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

[78] Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.