uOttawa

L'Université canadienne
Canada's university

Université d'Ottawa
Faculté de génie

University of Ottawa
Faculty of Engineering

ELG5901 Electrical Engineering Project

Final Report:

Student name: Ahmed Osama Ahmed Abdelaal **ID:** 300389360

Student name: Hussien Tarek Ismail Abdelrazik **ID:** 300389897

Student name: Moataz Soliman Mohamed Habib **ID:** 300389907

Student name: Shahd Mohamed Ali Mohammed **ID:** 300389911

Student name: Youssef Shaaban Sayed Mohamed **ID:** 300389908

Graduate Program: DEBI

Semester to Register: 2023, Fall

Project Title: Continuous Learning

Table of Contents

Contents

1 Introduction **4**
1.1 Problem Definition 4
1.2 Background 4
1.3 Project Context 5

2 Design Overview **6**
2.1 Requirements 7
2.2 Detailed Design 7
2.3 Implementation 10
2.4 Testing 13
2.4.1 Data Plan 13
2.4.2 Validation and Verification 13

3 Overall Results and Analysis **15**
3.1 Project Results: 15
3.2 Results Analysis: 16
3.3 Learning Outcomes: 17
3.4 Career Objectives: 17

4 Deployment Plan **18**

5 Conclusions and Future Works **19**

6 References **20**

Table of Figures

1. Figure 1: Architectural Diagram of SLDA - page 10
2. Figure 2: System architecture showing the workflow from data input through feature extraction and model training to making predictions and incorporating new data - page 11
3. Figure 3: The continuous learning process of the model with new object classes added over time displaying the model's ability to adapt without forgetting previous knowledge - page 11
4. Figure 4: A collection of images from the CORe50 dataset depicting the diversity of objects and conditions under which the data was captured - page 12

5. Figure 5: Incremental learning process visualized through the addition of new classes over time - page 13

6. Figure 6: The model correctly classifies a known object (glasses) demonstrating its proficiency in recognizing trained categories - page 13

7. Figure 7: An example of the model initially misclassifying an unrecognized object highlighting the need for continual learning and model updating - page 14

8. Figure 8: After updating the model with new data, it accurately classifies a previously unknown object showcasing the system's adaptive learning capability - page 14

9. Figure 9: The accuracy graph demonstrates the model's improvement in correctly classifying objects across successive learning batches - page 14

10. Figure 10: The forgetfulness curve illustrates the model's ability to retain knowledge of previously learned objects while learning new ones indicating effective mitigation of catastrophic forgetting - page 14

11. Figure 11: AR1-Based Latent Replay Model's Forgetting of First Batch across Experiences - page 15

Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
AR1	Common Continuous Learning Strategy
CNN	Convolutional Neural Network
CORe50	Continual Object Recognition dataset
CPU	Central Processing Unit
ImageNet	A large visual database for object recognition software research
JSON	JavaScript Object Notation
MgSvF	Multi-Grained Slow vs. Fast Framework for Few-Shot Class-Incremental Learning
PyTorch	An open-source machine learning library
ResNet-18	Residual Network with 18 Layers
RGB	Red Green Blue
RGB-D	RGB with Depth information
SDK	Software Development Kit
SLDA	Streaming Linear Discriminant Analysis

1 Introduction

1.1 Problem Definition

The foundation for intelligent systems to adapt to changing environments lies in the learning process, akin to the adaptability observed in biological organisms. Similar to the evolutionary mechanism that grants humans and other organisms a high degree of adaptability, artificial intelligence (AI) systems are expected to exhibit a comparable capability. This has led to the exploration of continual learning, also known as incremental or lifelong learning, where the goal is to enable AI systems to continuously acquire, update, accumulate, and leverage knowledge in response to external changes. In response to the ever-evolving landscape of technology, our company has identified a crucial need for the development of a cutting-edge system capable of continual learning for object classification. The primary objective is to enhance the adaptability and intelligence of our technologies, particularly in applications such as service robots, self-driving vehicles, and monitoring systems. The challenge at hand lies in addressing the inherent limitations of traditional deep neural networks, notably catastrophic forgetting, where incremental updates displace established representations, hindering the assimilation of new knowledge. Continual learning addresses the sequential acquisition of knowledge, encompassing new skills, additional examples, environmental variations, and diverse challenges. One prominent challenge is catastrophic forgetting, wherein adapting to new information leads to a reduction in the ability to capture knowledge from previous contexts. By investing in the development of a continuous learning system for object classification, our company aims to overcome this challenge, providing our technologies with the ability to seamlessly assimilate new information while retaining past knowledge. This strategic approach not only ensures our systems stay relevant in dynamic environments but also positions us at the forefront of technological advancements, offering unparalleled benefits in terms of adaptability, intelligence, and real-world applications.

1.2 Background

In the exploration, attention is given to noteworthy contributions, with a focus on practical applications and insights from industry leaders. Notable methodologies include an innovative "Latent Replay" technique (Pellegrini et al., 2020) [1], addressing memory limitations during continual learning by storing activations at an intermediate layer. However, potential limitations may arise, such as limited scalability due to the storage of activations and sensitivity to the choice of the layer. This method reduces storage and computation needs compared to traditional rehearsal-based approaches, achieving promising results in real-time scenarios. The "Deep SLDA" framework proposed by T. L. Hayes and C. Kanan. (2020) [2] introduces a deep streaming linear discriminant analysis for continual learning. While it utilizes a sliding window mechanism to process data streams and adapts the model with minimal catastrophic forgetting through forgetting-free updates, there might be potential cons. For instance, sensitivity to the window size may impact the ability to capture relevant information over time, and the resource-intensive nature of processing data streams could pose challenges.

Chen et al.'s (2023) [3] MgSvF framework handles few-shot class-incremental learning

scenarios by employing slow and fast learning processes within a single network. One potential limitation could be the complexity in balancing these learning processes and potential performance degradation in dynamic scenarios. However, this method achieves improved performance compared to existing approaches in memory-constrained settings. A. Ayub and A. R. Wagner. (2020) [4] focus on robots learning new object categories with few-shot examples in an incremental setting. They employ a dynamic memory network with task-specific mechanisms to achieve efficient learning from limited data while reducing catastrophic forgetting. Mirzadeh et al. (2022) [5] investigate the influence of network architecture on performance in continual learning tasks. They introduce a novel architecture using progressive distillation and adaptive batch normalization, demonstrating the potential for architectural choices to enhance continual learning capabilities. Potential cons may include the added complexity of progressive distillation and batch normalization, along with limited generalizability of architectural choices."A Complete Introduction to Continual Learning" [6] This blog post on iMerit provides a comprehensive introduction to Continual Learning (CL), a significant area of machine learning research. CL focuses on developing models that learn new tasks while retaining information from previous ones, addressing real-world scenarios where data and tasks are dynamic. The post discusses the challenges of CL, including catastrophic forgetting, and the techniques to overcome it, such as regularization and memory-augmented networks. It explores various aspects of CL, including biologically-inspired methods, multimodal multitask learning, the use of rehearsal buffers, and deep generative replay. The post also highlights practical applications of CL in anomaly detection, personalization, and forecasting. "Avalanche" [7] is a comprehensive library designed for Continual Learning (CL) based on PyTorch. It serves as an integrated framework for CL, supporting researchers and practitioners with tools for reproducible, comparable, and accessible CL research. Avalanche is built to facilitate the easy implementation, training, and evaluation of CL algorithms. Key features include benchmarks for CL scenarios, a flexible API for defining new benchmarks, comprehensive metrics for CL evaluation, and a training and evaluation engine. This library aims to standardize CL experiments and accelerate research in this field.

1.3 Project Context

Here we outline the external systems, APIs, tools, and collaborations required for the deployment of a real-time object recognition system with continuous learning capabilities, as part of our research project. Emphasis is placed on ensuring memory and resource efficiency, real-time capturing, and the ability to add new classes dynamically. Deployment using Streamlit: For an industry-ready deployment, we leverage Streamlit, a robust web application framework. This choice is driven by its capacity to provide an intuitive and user-friendly interface, crucial for real-time updates and interactive visualization. Our commitment includes ensuring compatibility with the latest version of Streamlit and verifying the availability of resources essential for hosting the web application in a corporate environment. Memory and Resource Efficient APIs: Our dedication to minimizing resource consumption led us to select efficient APIs for model deployment. We prioritize integration with lightweight frameworks or libraries suitable for real-time applications, ensuring reliability and adherence to stringent memory and resource efficiency requirements essential for seamless integration within corporate infrastructures. Real-Time Object Detection and Classification API: Integral to

our system is the integration of a robust API for real-time object detection and classification. This involves utilizing pre-trained models or custom-trained models optimized for rapid inference. Dependencies encompass confirming access to the chosen API, assessing compatibility with specific company requirements, and ensuring robust real-time capabilities to meet industry standards. Continuous Learning Infrastructure: Companies require an infrastructure that supports continuous learning, allowing dynamic model updates and the addition of new classes over time. Our commitment involves identifying or developing a system that seamlessly integrates with existing corporate frameworks, potentially requiring collaboration with experts in machine learning systems and frameworks. External Data Sources for New Classes: The addition of new classes through continuous learning mandates connections to external data sources providing pertinent information for model training. Dependencies include coordinating with data providers or organizations to ensure seamless access to updated datasets relevant to the specific industry needs. Our comprehensive integration strategy ensures the success of our project by facilitating real-time object recognition with continuous learning capabilities tailored to the specific requirements of companies. Regular communication and collaboration with external partners, API providers, and data sources are fundamental to overcoming potential challenges, ensuring a smooth implementation, and delivering a solution aligned with corporate objectives.

2 Design Overview

The system architecture outlined here introduces a pioneering algorithm, deep Streaming Linear Discriminant Analysis (deep SLDA), representing a significant advancement in large-scale image classification. Operating by classifying features derived from a deep Convolutional Neural Network (CNN), our system is the first to implement SLDA for this purpose on extensive image classification datasets. Key contributions of our system include: Algorithm Description: - The deep SLDA algorithm is intricately described, showcasing its role in classifying features from a deep CNN. - An innovative approach is demonstrated, highlighting the system's effectiveness for large-scale image classification datasets. Performance Superiority: - Empirical evidence substantiates the claim that deep SLDA surpasses state-of-the-art streaming learning algorithms. - This underscores the system's applicability and superiority in real-world scenarios. Efficiency and Speed: - Our system excels by outperforming recent methods in incremental batch learning, achieving remarkable speed and using significantly less memory. Our proposed Deep Streaming LDA, detailing its incremental training approach for a CNN in a streaming manner. The system decomposes the network into two functions, concentrating on training the last fully-connected layer. Variants of SLDA are introduced, incorporating a frozen covariance matrix and streaming updates, with a discussion on shrinkage regularization for precision matrix computation. Data flow is outlined, encompassing input from large-scale datasets, feature extraction using a pre-trained deep CNN, and the output of predicted categories for input images. The high-level description underscores the primary goal of our system: providing an efficient and memory-conservative solution for large-scale image classification in real-time. Key features include the novel deep SLDA algorithm, efficiency gains in both speed and memory usage, and adaptability for real-time applications. Researchers and developers anticipate innovative solutions contributing to

the advancement of large-scale image classification methodologies. They expect insights into overcoming challenges associated with streaming learning and incremental batch learning, emphasizing practical applications. For industry and technology companies, the expectation is for practical, efficient, and fast solutions addressing real-time image classification needs. This includes considerations for technologies with reduced memory usage and faster training speeds, aligning with the industry's demand for swift decision-making based on extensive image datasets.

2.1 Requirements

Nonfunctional requirements In terms of end-user requirements, efficiency is paramount. There is a need for a system that efficiently processes and classifies large-scale image datasets, enabling timely decision-making. The system should demonstrate efficiency gains, being faster in training and utilizing significantly less memory. Functional requirements such as Real-time adaptability is also crucial, addressing challenges associated with waiting for batch information accumulation before making inferences. End users prioritize a system designed for applications requiring prompt decision-making based on streaming data. Usability is another key requirement, emphasizing the need for a user-friendly system that is easy to implement in various scenarios without extensive computational resources. Practical usability ensures seamless integration into existing workflows. Accuracy and reliability are top priorities. Stakeholders and end users expect the system to consistently deliver reliable predictions, especially in dynamic and changing environments.

2.2 Detailed Design

Imagine a robot that can learn to recognize new objects without forgetting the ones it already knows. That's the kind of intelligence researchers are trying to achieve through continual learning. To test and compare algorithms developed for this purpose, scientists need datasets that mimic real-world scenarios. This is where CORe50 [8] comes in. CORe50 isn't filled with exotic animals or rare plants. Instead, it focuses on 50 familiar objects we encounter in everyday life, like mugs, remote controls, and light bulbs. These objects belong to 10 categories, allowing researchers to evaluate algorithms at both the individual object and broader category levels. Unlike most datasets, CORe50 doesn't just offer plain old photographs. Each image captures the object not only in color (RGB) but also in depth (D). This extra information makes the task more challenging and realistic, as humans rely on both color and depth perception to recognize objects. Instead of static images, it provides a series of 164,866 pictures taken over 11 different sessions. These sessions vary in lighting, background, and even location (some are indoors, some outdoors). Think of it like observing the object over time and from different angles. CORe50 puts the objects on the move! Within a designated space, these objects are rotated and shifted, ensuring the robot sees them from various perspectives and poses. This adds complexity and better reflects real-world situations where objects don't always stay perfectly still. The true test of a continual learning algorithm is its ability to acquire new knowledge without forgetting the old. CORe50 is designed specifically for this purpose. The 11 sessions are split into training and testing sets, simulating a scenario where the robot learns about some objects first and

then encounters new ones later. By evaluating how well the algorithm performs on both sets, researchers can assess its ability to avoid "catastrophic forgetting," a common issue in this field. Using the Avalanche library to load CORe50 data involves a streamlined process that simplifies the handling of continual learning experiments. Begin by installing the Avalanche library in our programming environment, using the appropriate command for your package manager. This initial step sets the foundation for leveraging the library's capabilities.

With the necessary modules imported from the Avalanche library, proceed to inform the library about your intention to use the CORe50 dataset. During this step, specify whether you are focusing on object-level or category-level classification, aligning with your experimental goals. Avalanche's strength lies in its ability to organize CORe50 data into a sequential stream of experiences, each representing a distinct learning task. This stream generation is a pivotal step that sets the stage for simulating continual learning scenarios efficiently. Once the data stream is generated, the subsequent task involves iterating through it. Each iteration corresponds to an experience, where you can seamlessly train or test your model on the specific data associated with that learning task. The simplicity of this process enables a straightforward approach to handling evolving scenarios. Within each experience, you have easy access to images and labels. This accessibility proves advantageous during the model development phase, allowing for efficient interaction with the data and fostering a clearer understanding of the continual learning process. The structured organization of CORe50 data into labeled tasks enhances clarity and facilitates evaluation. By labeling experiences (e.g., "session1", "session2"), Avalanche ensures a well-organized framework for tracking and referencing specific learning tasks throughout the experiment.

Before undergoing feature extraction, the input data is systematically divided into nine batches, each serving a distinct purpose. The initial batch, comprising the first set, is characterized by the presence of 10 classes. In contrast, the subsequent eight batches uniformly consist of 5 classes each. This strategic division of data into batches with varying class compositions potentially reflects an intentional experimental design or scenario where different classes are introduced at different stages. Such a pre-defined structure in the data organization adds an additional layer of complexity and diversity to the training and testing processes, offering a realistic simulation of scenarios where class distributions evolve over time or under specific conditions. This structured data organization aligns with the subsequent feature extraction process, where the tailored ResNet-18 model can effectively capture and differentiate features corresponding to the unique characteristics of each batch. A feature extraction model using the ResNet-18 architecture, catering specifically to data loaded from the Avalanche library. ResNet-18, a well-established convolutional neural network architecture known for its efficacy in image classification, serves as the foundation for this feature extraction model. It allows for the initialization of the model with pre-trained weights from the ImageNet dataset, enhancing its adaptability. Customization is facilitated through the adjustment of the fully connected layer (, where the desired feature size (and the number of output classes are specified as desired. Then a tailored ResNet-18 feature extraction model, ready for deployment in tasks such as continual learning within the Avalanche library. The Streaming Linear Discriminant Analysis (SLDA) algorithm is designed for continual learning applications, functioning in a streaming manner to adapt to incoming data while retaining knowledge from previous experiences. The model is initialized with essential parameters, such as the feature dimension, total number of classes, batch size for inference, a shrinkage

parameter governing matrix adjustments, and a flag indicating whether streaming updates to the covariance matrix should be performed. Initialization involves setting up weights and crucial variables. The model dynamically adjusts means, counts, and the covariance matrix (Sigma) in response to incoming data. Additionally, predictions or probabilities for test data are calculated through the predict method, with updates to the covariance matrix contingent on a specific parameter. The Streaming Linear Discriminant Analysis model is also responsible for fitting the model to base initialization data. This involves estimating initial means and initializing the covariance matrix. To facilitate model persistence, the provides methods for saving and loading model parameters, allowing for the reuse of trained models without retraining. The model operates on a specified device (typically the CPU), and data is transferred to this device for processing. We extend the functionality beyond the training of the Streaming Linear Discriminant Analysis (SLDA) model. After training the SLDA model on the existing data, we introduce a subsystem designed for real-time capturing of new input data. This subsystem allows the model to continuously adapt and learn from incoming data streams. The subsystem is responsible for the classification of an input image using the trained SLDA model. The image undergoes preprocessing through resizing and normalization before being fed into the model. The resulting features are extracted using a feature extraction wrapper, and the pooled features are obtained, potentially involving a pooling operation depending on the model architecture. The SLDA model is then utilized for prediction, and the predicted class is determined. This real-time learning subsystem is particularly valuable for scenarios where the model encounters new data on-the-fly. After making predictions, the code incorporates a mechanism to capture the real-time input and retrain the SLDA model with this new information. The model is subsequently updated, and the acquired knowledge is stored for future reference. This continuous learning process ensures the adaptability of the model to evolving data distributions, contributing to its robustness and efficacy in dynamic environments. We employ the Streamlit framework to deploy a continuous learning system as a web page. The interface allows real-time webcam classification using a pre-trained Streaming Linear Discriminant Analysis (SLDA) model, complemented by a retraining mechanism for dynamic adaptation to new classes. The system features start/stop buttons for webcam classification and enables users to initiate the addition of a new class. During retraining, users input the new class name, capture images for it, and trigger the model update process. The SLDA model is dynamically retrained using the captured images, incorporating data augmentation techniques, and the updated model is saved. Status messages provide real-time feedback on ongoing tasks, and success/error messages signify completion. Persistent storage is employed for category mappings and the trained model, enhancing knowledge retention between sessions. The Streamlit framework ensures a user-friendly and interactive continuous learning experience through the web page. our experiments offer insights into the trade-offs and evaluation of alternative approaches to continuous learning in image classification. The Naive Approach, employing a simple CNN, stands out for its efficiency, completing in 49 minutes, but may sacrifice model sophistication. The Cumulative Approach, still under experimentation, leaves room for assessing its efficacy compared to the Naive method. The Latent Replay method, based on AR1, requires an estimated 17 days for processing the entire dataset, demonstrating a trade-off between computational resources and comprehensive model training. Resource constraints lead to a decision to work on a portion of the data, emphasizing a pragmatic approach.

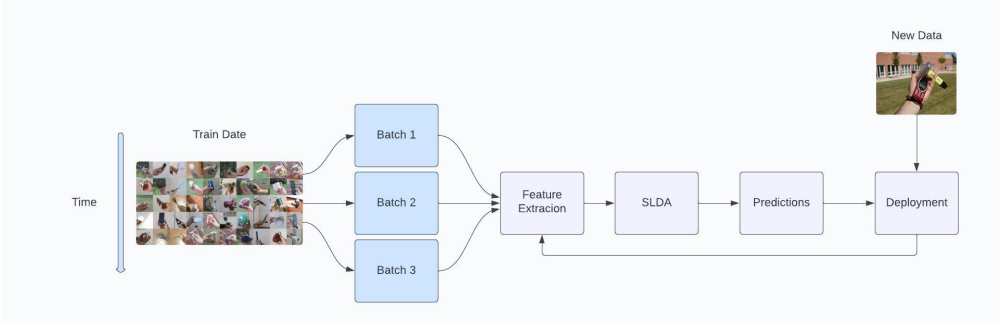

Figure 1: Architectural Diagram of SLDA

The SLDA approach, taking 19 minutes, strikes a balance between efficiency and potential model sophistication. These results underscore the nuanced considerations in choosing an approach, involving speed, resource utilization, and model effectiveness in the context of practical constraints.

2.3 Implementation

This section outlines the systematic approach to implementing our project, which aimed to develop a self-learning system capable of rapid adaptation. The discussion includes the final status of the project, the achievement of our goals, and how we utilized various tools and technologies.

Programming Languages and Technologies The project was developed using Python, with data handling and computational tasks managed through Kaggle and Google Colab. PyCharm served as our integrated development environment for coding and debugging. The Avalanche library, built on PyTorch, facilitated rapid prototyping and model evaluation, while Streamlit was employed to create interactive user interfaces, enhancing the adaptability and efficiency of our development process.

Development Environment Our development environment combined Kaggle, Google Colab, and PyCharm, creating a versatile platform that supported the project's versatile needs.

Implementation Process The implementation process included several essential steps, starting with the preparation of the CORe50 dataset. Transformations were applied using PyTorch's transforms module, and DataLoader instances were set up for batch processing. Following this, the ResNet-18 model was integrated with our Streaming Linear Discriminant Analysis (SLDA) model for feature extraction and incremental learning. This hybrid model was then trained incrementally, improving its accuracy with each new batch of data.

The workflow from data input to model updates, capturing the entire process from feature extraction to training and making predictions, is visualized in Figure 2. This illustration outlines the structured approach we employed to train and continually update the model. To assess the model's effectiveness, we calculated test accuracy and other key metrics. Figure 3

demonstrates the model's streaming learning capability, where it processes new data sample-by-sample in real-time. This method allows for ongoing and incremental adaptation, which is especially beneficial for embedded applications requiring immediate learning from each new data instance.

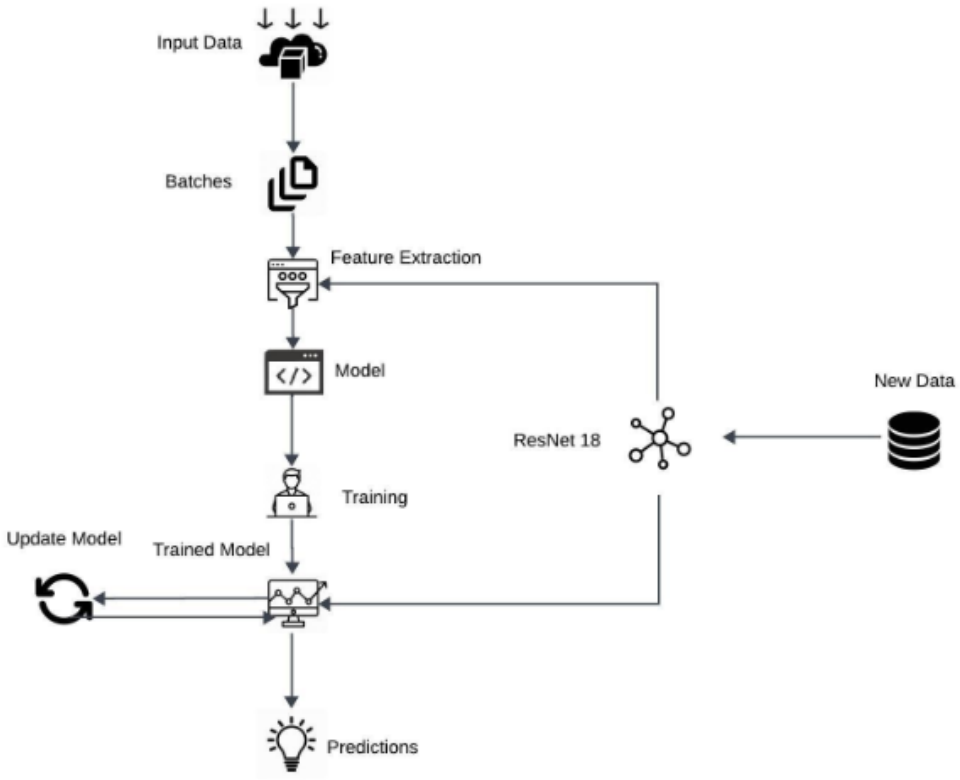

Figure 2: System architecture showing the workflow from data input through feature extraction and model training, to making predictions and incorporating new data.

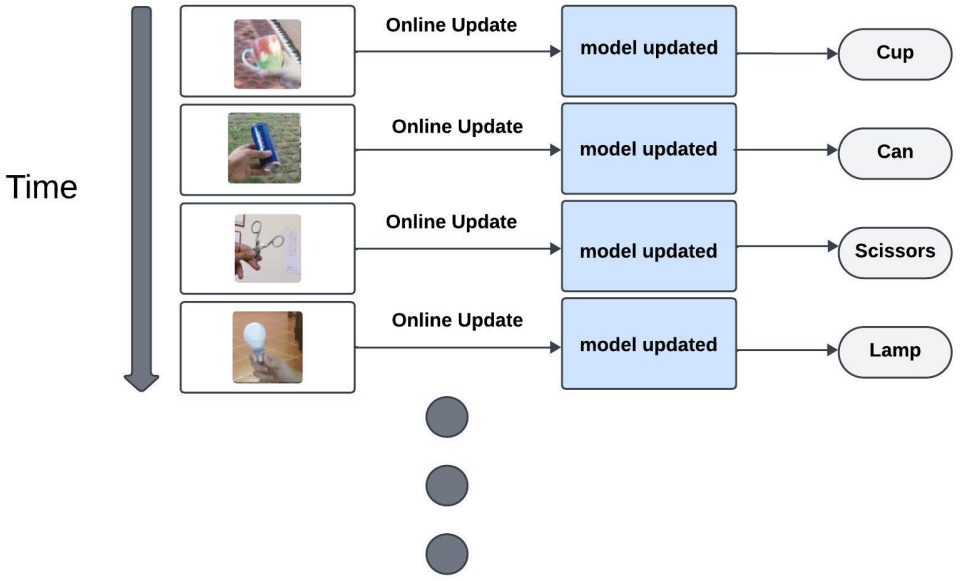

Figure 3: The continuous learning process of the model, with new object classes added over time, displaying the model's ability to adapt without forgetting previous knowledge.

Final Status and Goal Achievement The project culminated in the successful development of a system with an 80% accuracy rate in object classification, effectively managing to update the model with new classes. This milestone was a testament to the project's efficacy and was positively received by our sponsor.

Project Management Employing an Agile methodology, we maintained a flexible and iterative approach to development, with regular feedback sessions from our sponsor and mentor guiding the project's evolution. This collaborative approach was instrumental in navigating the challenges of continual learning and model adaptation.

2.4 Testing

2.4.1 Data Plan

The CORe50 dataset, specifically designed for continuous learning scenarios, was employed for both the development and testing of our project. This publicly available dataset comprises 164,866 RGB-D images of 50 objects categorized into 10 classes. The images were collected by the dataset authors in a variety of settings to allow for diverse testing conditions, encompassing different backgrounds and lighting conditions (Figure 4). The dataset, including details on its collection and use, is available at the official CORe50 website. The data was collected using a Kinect 2.0 sensor over 11 sessions, capturing 300 RGB-D frames per object. For our purposes, the data was normalized to ImageNet standards and resized to 224x224 pixels.

Figure 4: A collection of images from the CORe50 dataset depicting the diversity of objects and conditions under which the data was captured.

2.4.2 Validation and Verification

The validation and verification process of our project adhered to the design specifications that demanded accurate object classification and the ability to incrementally

learn and classify new objects. Extensive testing was conducted over 9 incremental batches, effectively simulating continuous model updates with new classes (Figure 5). The model's deployment capabilities were demonstrated using a Streamlit application, which facilitated real-time classification through a webcam feed, allowing for the addition of new classes, such as 'watch', and confirming the model's ability to integrate new information without forgetting previously learned classes (Figures 6, 7, and 8).

Accuracy and Forgetfulness The model's performance was quantitatively assessed by tracking accuracy and forgetfulness metrics. The accuracy improvements were documented over the course of encountering new batches, achieving a final accuracy of 80% (Figure 9). The model's ability to retain prior knowledge when new data was introduced was a critical measure of success, as indicated by the stability of the accuracy curve in (Figure 10).

Operational Testing Operational tests underscored the practicality of our system. Deployed via a Streamlit application, the system demonstrated high proficiency in real-time object classification. This interface allowed users to label and incorporate new object classes, such as 'watch', directly into the model's knowledge base, which is detailed in (Figures 6, 7, and 8).

Fault Prediction and Gap Analysis We identified that the presence of background noise could potentially impact the model's classification performance. Plans to incorporate noise reduction techniques and advanced data augmentation methods will be explored in future work to address these challenges.

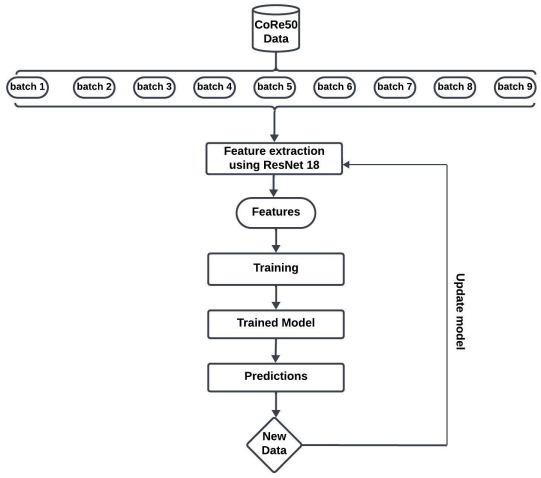

Figure 5: Incremental learning process visualized through the addition of new classes over time.

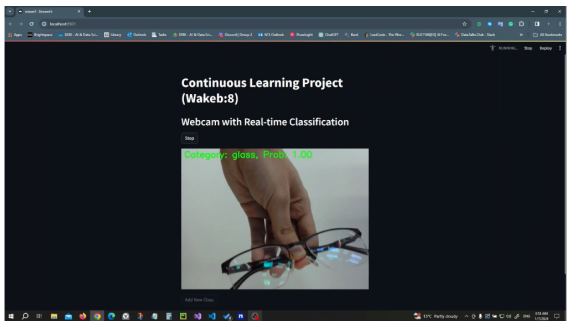

Figure 6: The model correctly classifies a known object (glasses), demonstrating its proficiency in recognizing trained categories.

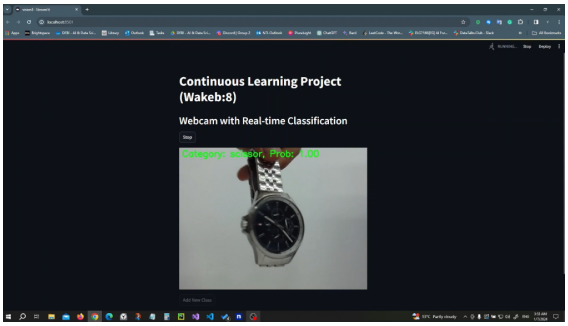

Figure 7: An example of the model initially misclassifying an unrecognized object, highlighting the need for continual learning and model updating.

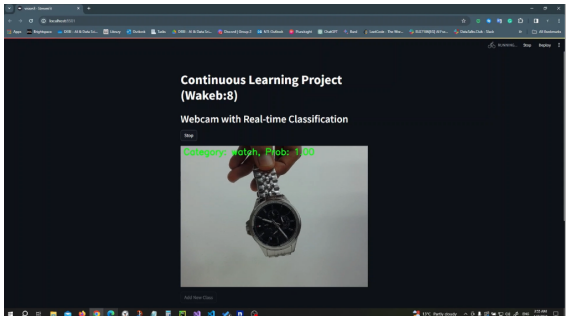

Figure 8: After updating the model with new data, it accurately classifies a previously unknown object, showcasing the system's adaptive learning capability.

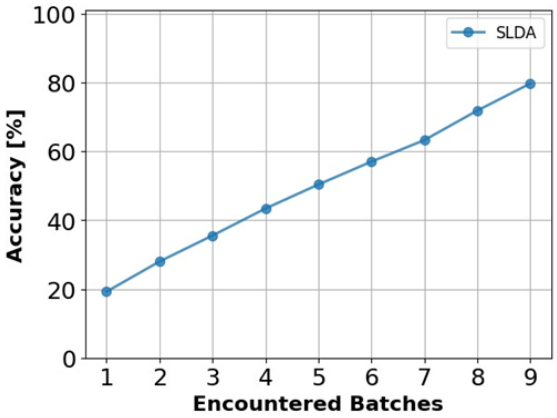

Figure 9: The accuracy graph demonstrates the model's improvement in correctly classifying objects across successive learning batches.

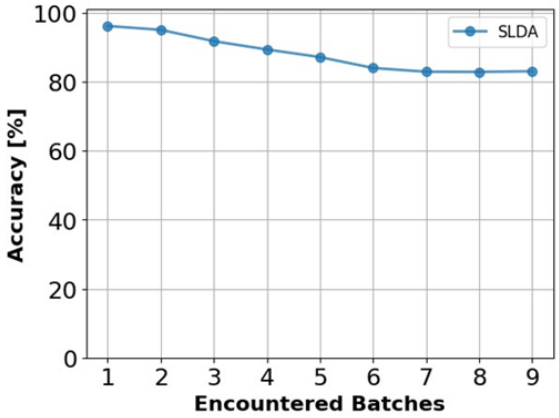

Figure 10: The forgetfulness curve illustrates the model's ability to retain knowledge of previously learned objects while learning new ones, indicating effective mitigation of catastrophic forgetting.

3 Overall Results and Analysis

3.1 Project Results:

To evaluate our model forgetting we will measure the model accuracy on the objects of the first batch since the first batch represents the catastrophic forgetting worst-case among batches. Because the earliest batch includes the objects that the model initially learned to detect and twice the number of objects within other batches. The model forgetting across batches didn't surpass 20% (Figure 10). On the other hand, the model's overall

accuracy on all objects of the whole dataset increased across batches reaching 80% (Figure 9), which indicates adding new objects to the model knowledge without forgetting much of the previously learned objects' knowledge.

3.2 Results Analysis:

The model shows very accurate performance concerning the time and computation resources it requires. It provides a very lightweight yet effective solution as it only required 19 min to train on the whole CORe50 dataset which is a very efficient time concerning other continual learning strategies that might give a slightly higher accuracy but require a lot of resources making it not applicable especially on low computational resources (e.g., edge devices, embedded systems, etc.). An example of those strategies that we encountered in our experiment was Latent Replay (specifically AR1-based); this solution was very challenging to apply to the whole dataset as it requires a lot of time and resources. Approximately 400 hr (about 17 days) of running resources with GPU parallelism is required to apply it to the whole dataset. This was estimated based on applying the AR1 latent replay on a portion of the dataset to be able to experiment with this method on the data. The dataset portion compromised only a random 6% of all the images for objects belonging to the same class of each of the 10 classes within the dataset and applied the strategy on the object level instead of the class level to lower the overhead on computational resources. Even though it required 24 hr running time to complete and didn't provide promising results, especially on the catastrophic forgetting mitigation across batches learning (Figure 11).

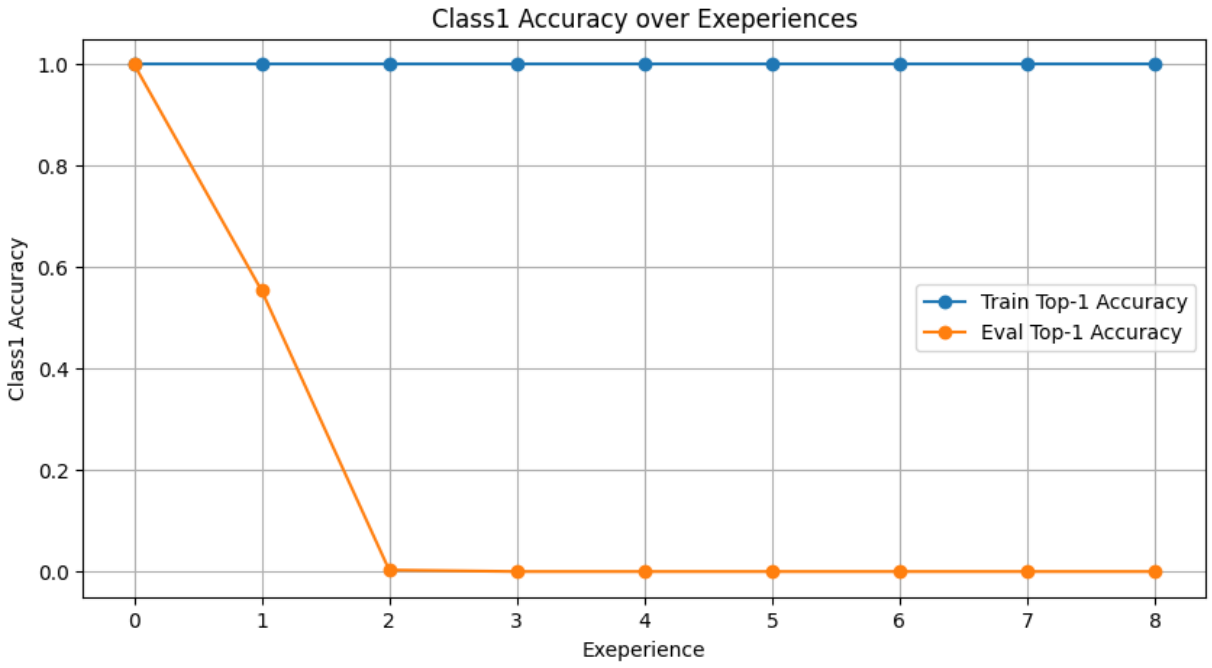

Figure 11: AR1-Based Latent Replay Model's Forgetting of First Batch across Experiences Approximately 0% on Train Set and 100% on Test Set

Although the SLDA model provides a lightweight and efficient solution, it has some drawbacks as the maximum number of new objects that the model can learn continually is constrained by the dimension of the SLDA model covariance matrix that was initially defined before the training phase by 1,000 objects in our solution experiment. Also, it doesn't handle features drift because the SLDA model (as any other continuous learning model) only mitigates the catastrophic forgetting and doesn't eliminate it; consequently, after training on a very large number of batches and new objects added to model knowledge, the model might forget the objects trained on in the earlier batches on a long term. A possible approach for handling this issue would be defining a forgetting threshold that if the model performs poorly exceeding it during continuous learning of new batches, the model will automatically retrain over the whole data in addition to all the new batches it previously learned single time until this threshold is exceeded again, but this will introduce the main issue of large scalable memory required for saving all this previously learned new batches in addition to original train data. It's possible to save only a compact features representation for the knowledge learned from the earlier batches instead of saving the whole batches' data such as the mechanism used in Latent Replay strategy but this will require more computational resources and time introducing the trade-off among memory, computation resources and processing time with model accuracy and forgetting mitigation.

3.3 Learning Outcomes:

Research and Innovation Skills: Engaging with innovative methodologies from recent research papers, such as Latent Replay, Deep SLDA, MgSvF, and dynamic memory network, is a key aspect of our project. This hands-on experience fosters our ability to stay at the forefront of cutting-edge advancements in the field.

Technical Proficiency: Implementing the project using advanced technologies and frameworks like PyTorch, Streamlit, and the Avalanche library enhances our technical skills in machine learning, deep learning, and web application development. This practical exposure is invaluable for our academic growth.

Problem-Solving and Critical Thinking: Our project involves identifying and addressing challenges in traditional deep neural networks, proposing novel algorithms like deep SLDA, and conducting a comparative analysis. These activities showcase our problem-solving and critical thinking abilities, crucial aspects of our academic development.

Collaboration and Communication: The collaborative nature of our project, involving external partners, API providers, and data sources, highlights our skills in managing dependencies and ensuring effective communication and teamwork. This aligns well with the collaborative aspects emphasized in our graduate program.

3.4 Career Objectives:

Industry-Relevant Skills: Developing a real-time object recognition system with continuous learning capabilities directly addresses the demands of the industry for adaptive and

intelligent systems. Our project equips us with practical skills that are directly applicable to the development of AI applications in various industries.

Deployment and Integration Experience: Integrating the system with tools like Streamlit and efficient APIs demonstrates our capability to deploy AI solutions for practical use. This hands-on experience is highly valuable for our future careers, showcasing our ability to deliver solutions that can be seamlessly integrated into real-world applications.

Problem-Specific Solutions: Our project concentrates on tackling particular issues related to continual learning, such as the challenge of catastrophic forgetting with low resources in an efficient manner. Through the proposal and execution of customized solutions, we enhance the depth of our skill set, ensuring our readiness to tackle practical challenges in our prospective professional journeys.

Project Management and Agile Methodology: The Agile methodology in our project management approach demonstrates our understanding of efficient development processes. This experience is a valuable asset for our future careers, where project management skills are often essential.

Project Success and Outcomes: In summation, our collaborative efforts on this project provide a unique opportunity for us to achieve academic success and simultaneously align with our career objectives. The practical experience gained and the skills developed throughout the project will undoubtedly contribute to our growth as professionals in the dynamic field of artificial intelligence. We look forward to our continued success as a cohesive project team.

4 Deployment Plan

In the initial phase of the project, collaboration with the sponsor led to the development of a comprehensive dashboard showcasing various model performance metrics such as accuracy, rate of forgetting, and the number of objects recognized. After careful consideration and team discussions, it was realized that a dashboard might not provide the anticipated level of practical utility. Therefore, the decision was made to pivot and deploy the project as web-based software, using Streamlit as recommended by the sponsor.

The current deployment is intentionally basic, serving as a foundational component for a larger system to be developed in the future. This expanded system will include advanced features like web scraping for image acquisition and object detection for multiple classifications.

The full deployment plan, as envisioned by the sponsor, involves utilizing the model as a safety monitoring tool. This application will monitor workers in operational environments, such as factories, to ensure compliance with safety protocols like wearing helmets.

Given this broader context, it's evident that the current deployment lacks several critical features required for full-scale implementation. Below is the current state of the deployment:

- *Overview of the Application*: The application combines real-time image capture, machine learning-based image classification, and an interactive interface. It's designed to classify images from a live webcam feed.

- *Real-Time Image Capture and Processing*: The core feature of the deployment is the integration of a webcam for real-time image capture. This capability enables the application to process and classify images as they are captured, providing immediate classification results. This is particularly useful in scenarios requiring instant decision-making based on visual data.

- *Image Classification Using Pre-Trained Model*: The heart of the application is a sophisticated image classification model. Pre-trained to identify various categories, it classifies images from the webcam in real-time. The classification process not only identifies the object but also quantifies the confidence of the prediction, providing a probability score alongside the classification result.

- *Dynamic Category Mapping*: A key feature is dynamic category mapping, translating numerical class predictions into human-readable category names. This mapping, stored in a JSON file, allows for easy updates and modifications, crucial for adapting the application to new requirements or extending its capabilities to new categories.

- *Interactive Model Updating and Extension*: The application includes interactive features enabling users to add new categories to the classification model. Users can input a new label, capture images corresponding to this label, and use these images to retrain the model. This functionality ensures that the application remains relevant and up to date with a constantly changing environment.

- *Deployment Considerations*: For deployment, the application is packaged to include all dependencies, ensuring seamless integration with webcam hardware and compatibility across various platforms. Attention is given to ensuring that the model runs efficiently, maintaining a balance between accuracy and performance. User-friendly error handling and feedback mechanisms are integrated to guide users through the process, especially during model retraining or in case of any operational issues.

Moving forward, let's discuss the potential processes to ensure the system or application is ready for distribution in the operational environment. We'll also address what is required in the operational environment and highlight maintainability issues to create a smooth end-user experience.

5 Conclusions and Future Works

In our project, we have accomplished the successful implementation of Streaming Linear Discriminant Analysis (SLDA) using the PyTorch framework. Our work has focused on utilizing the CORe50 dataset for effective streaming image classification, and we have achieved significant benchmarks that set our model apart from existing methodologies.

Our achievements are threefold:

1. **Effective Implementation:** We have demonstrated SLDA's potential in continual learning applications, showcasing its robustness against catastrophic forgetting—a prevalent challenge in neural networks.

2. **High Performance:** The deep SLDA model we developed not only meets but exceeds performance standards, achieving an optimal balance between efficiency and effectiveness.

3. **Speed and Memory Efficiency:** Our model demonstrates exceptional operational speed, significantly surpassing the performance of traditional models. Additionally, it showcases remarkable memory efficiency, requiring substantially less memory compared to conventional benchmarks, such as those set by the ImageNet dataset

As we look to the future, we envision a roadmap for furthering the capabilities of our model:

1. **Full Automation:** The next step involves moving towards full automation, enabling the model to identify new objects autonomously, search for relevant data, and self-train effectively.

2. **Object Localization and Detection:** We aim to enhance the model's ability to precisely localize and detect objects within an image, which will further refine its classification precision.

3. **Model Accuracy Enhancement:** Through the integration of advanced machine learning techniques and ongoing data analysis, we plan to continually improve the accuracy of our model.

We believe that the paths we have charted not only respond to the immediate needs of adaptive AI systems but also lay the groundwork for transformative future developments. Our model's adaptability and efficiency make it a potential paradigm in AI research, with broad implications for the way machines learn and interact with their environment. The motivation behind our research is to push the boundaries of machine learning, fostering a future where technology is more intuitive, responsive, and aligned with the dynamic nature of real-world data.

6 References

[1] L. Pellegrini, G. Graffieti, V. Lomonaco and D. Maltoni, "Latent Replay for Real-Time Continual Learning," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 10203-10209, doi: 10.1109/IROS45743.2020.9341460. vol. 00, no. 00, pp. 00-00, 2020.

[2] T. L. Hayes and C. Kanan, "Lifelong Machine Learning with Deep Streaming Linear Discriminant Analysis," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 2020, pp. 887-896, doi: 10.1109/CVPRW50498.2020.00118.

[3] H. Zhao, Y. Fu, M. Kang, Q. Tian, F. Wu and X. Li, "MgSvF: Multi-Grained Slow vs. Fast Framework for Few-Shot Class-Incremental Learning," in IEEE Transactions on Pattern Analysis and Machine Intelligence, doi: 10.1109/TPAMI.2021.3133897.

[4] A. Ayub and A. R. Wagner, "Tell me what this is: Few-Shot Incremental Object Learning by a Robot," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 8344-8350, doi: 10.1109/IROS45743.2020.9341140.

[5] S. I. Mirzadeh et al., "Architecture Matters in Continual Learning," ArXiv, Feb. 2022, Accessed: Jan. 15, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Architecture-Matters-in-Continual-Learning-Mirzadeh-Chaudhry/f31ad882c11f5e4ebd731323972d318307634d74

[6] iMerit, "A Complete Introduction to Continual Learning," Apr. 05, 2023. [Online]. Available: https://shorturl.at/agyE2. [Accessed: Jan. 15, 2024].

[7] Medium, "Avalanche: An End-to-End Library for Continual Learning Based on PyTorch." [Online]. Available: https://shorturl.at/vAHUV. [Accessed: Jan. 15, 2024].

[8] V. Lomonaco and D. Maltoni, "CORe50: a New Dataset and Benchmark for Continuous Object Recognition," in *Proceedings of the 1st Annual Conference on Robot Learning*, vol. 78, pp. 17-26, 2017. [Online]. Available: https://proceedings.mlr.press/v78/lomonaco17a.html