Ministry of Education and Science of Ukraine
National University «Kyiv-Mohyla Academy»
Informatics Department of Faculty of Informatics

Master's thesis

educational level — master

on the topic: **«CONTINUAL LEARNING METHOD FOR IMAGE CLASSIFICATION IN COMPUTER VISION»**

By: 2-nd year student

of the educational program «Computer Science», 122

Kreshchenko Taras Oleksandrovych

Supervisor: Yushchenko Yu. O.
Candidate of Physico-Mathematical Sciences

Reviewer _____

The master's thesis was defended with a grade: _____

EC secretary _____

«____» _____ 2024

Київ – 2024

TABLE OF CONTENTS

ABSTRACT

The paper explores the hypothesis that Continual Learning (CL) methods can improve the performance of a deep learning model in a traditional machine learning scenario. By augmenting an existing state-of-the-art ML solution to a problem with CL techniques, this research aims to demonstrate that AI can still achieve more accurate and adaptive performance. This hypothesis is tested on a parking lot occupancy detection problem, a binary classification problem that is well-suited to CL due to the continuous stream of image data. Experiments are conducted to compare the proposed CL-based solution and a contemporary solution that is non-CL based.

Keywords: deep learning, continual learning, incremental learning, lifelong learning, domain adaptation, contrastive learning, CNN, computer vision, binary classification, parking, occupancy detection

INTRODUCTION

For decades, researchers everywhere around the world have been trying to replicate human cognitive behavior in the field of Artificial Intelligence (AI). Many machine learning (ML) methodologies have been proposed, as well as extensively tested and used, to achieve this goal. However, there is at least one feature of a human brain that has proven to be very difficult to model using a machine: the ability to learn continually throughout the lifetime.

This ability allows humans to improve their knowledge over time with every new experience they come across. It allows humans to still perform better at numerous tasks in comparison with standard ML models, which are trained once and never improved afterwards. Although, in the last decade a new paradigm of ML has been gaining scientific interest: Continual Learning (CL). This paradigm discards the notion of training a model once and using it in production afterwards and proposes to see the latter stage as a part of the training process, in other words, learning continually from the observed experiences.

In the recent years the field of continual learning has been in the peak of research interest. Numerous methods have been proposed that allow the machine to learn from a continuous stream of data. A lot of these methods focus on solving the problem of "catastrophic forgetting," which causes the model to perform worse and worse on old tasks.

Nowadays, most of the research in this field try to use Continual Learning to solve completely new types of problems – scenarios, that cannot be approached using traditional (non-CL) machine learning methods. Here, the Class-Incremental scenario has caught the most attention, in which the model, which is solving a classification task, is expected to learn completely new classes that have not been encountered previously. Similarly, the Task-Incremental scenario attempts to solve different tasks every time. This way the researchers are trying to expand the space of problems that can be solved using AI.

However, to the extent of the author's knowledge, no research has attempted to improve the performance of non-CL models in traditional scenarios using CL. One of such scenarios is the problem of parking lot occupancy detection, which is a binary classification problem in the field of Computer Vision (CV). This problem presents a continual stream of data in the form of video camera feed and requires the model to decide whether a parking space is occupied or free. The presence of such continual data stream allows to apply CL in an attempt to improve the performance of the model over time.

This paper explores the hypothesis that CL can lead to better performance in standard machine learning tasks. To do this, the problem of parking lot occupancy detection is considered, already existing solutions to this problem are explored, the current CL scene is studied and summarized, CL-based solution to this problem is proposed, and experiments are conducted in order to compare the performance of the CL-based and non-CL based solutions.

The paper consists of three sections.

Section one studies the current CL scene, state-of-the-art methods, and common usage scenarios. The problem of parking lot occupancy detection and existing solutions to this problem are explored. A CL-based solution to the problem is proposed, which has the potential to achieve better performance than the traditional solution.

The second section describes the implementation of the model and the specifics of the reproduction process of an existing solution.

Experiment results are reported and discussed in the third section.

1 MOTIVATION AND RELATED WORKS

1.1 AI as a tool to automate complex tasks

Humans are lazy creatures. We try to automate as many mundane tasks as we can, so that there is as little work left to do as possible. However, automation, the process of performing an action with minimal human assistance, has truly been a cornerstone of human advancement. The desire to automate has driven humans to innovate, transforming industries and everyday life. Historically, early examples of automation can be traced back to simple mechanical tools designed to reduce human labor in tasks like agriculture and manufacturing. These rudimentary systems laid the groundwork for more sophisticated automated processes that would emerge in later centuries.

With the development of various cultural areas of human life, such as science or arts, more emphasis has been put on mental work and decision-making. The evolution of computers has played a pivotal role in the advancement of automation. Early mechanical computers, such as Charles Babbage's Difference Engine in the 19th century, represented the first attempts to mechanize complex calculations. However, it was the transition to electronic computers in the mid-20th century that truly revolutionized computing. The development of the first general-purpose electronic computers in 1940s and 1950s, such as ENIAC, Zuse Z4, and MESM, marked the beginning of a new era. Subsequent innovations paved the way for modern computers, which are characterized by their incredible processing power and versatility.

With the appearance of electronic computers, the potential for automation expanded dramatically. Computers began to be integrated into industrial processes, leading to significant improvements in efficiency and precision. This integration was particularly evident in manufacturing, where machines and robots, controlled by a computer, automated repetitive or dangerous tasks. The impact of computing on automation then further extended to other sectors such as healthcare, finance, and logistics. Now, with simpler calculations taken care of by the machines, more complex and data-driven decisions could be made by the humans.

The next logical question that occurs is, how much mental work can really be automated? It seems that at a certain point the level of complexity becomes so high that human intelligence is required to perform the task. Artificial Intelligence (AI) represents the frontier of automation, aiming to create systems capable of performing tasks that typically require human intelligence. AI includes a wide range of technologies, starting from robotics and ending with natural language processing and machine learning. Historical milestones in AI development include the creation of the first AI programs in the 1950s, the development of expert systems in the 1970s and 1980s, and the recent advancements in deep learning. Machine learning (ML), a subset of AI, involves the development of algorithms that allow computers to learn from and make predictions based on data. This capability is essential for automating tasks that involve pattern recognition, data analysis, and decision-making.

The automation of complex tasks has been a major focus of AI research. Early AI systems relied on rule-based approaches, which required the programmer to explicitly define every rule as part of its domain-specific knowledge. The advent of ML algorithms transformed AI, allowing systems to learn and improve through data without having to explicitly define their behavior. AI systems now excel at tasks such as image recognition, understanding natural language, autonomous navigation, and text or image (or even video) generation, thanks to their ability to learn from large datasets. These advancements have made it possible to automate tasks that were once considered too complex for machines.

Still, there is one key limitation that makes ML models inferior in comparison to the way humans learn: the inability to learn continuously from a stream of data over time. Continual Learning (CL) is a set of approaches within ML that aim to address this limitation. Unlike static learning models that are trained once and then applied without further modification, CL models are designed to evolve as they process new information. This capability is natural in human learning, where knowledge is continuously updated based on new experiences. However, modeling this process in AI presents significant challenges, for instance, the now classical problem of catastrophic forgetting, where new information can overwrite previously learned knowledge.

In the next section, the current CL scene is explored. Common CL problems and scenarios as well as state-of-the-art methods and strategies are investigated.

1.2 Continual Learning methodology

While the ability to adapt and learn continually much like a human is the core idea behind CL, it can be unfolded into a set of primary goals. Firstly, models should efficiently incorporate new data without the need for retraining from scratch. At the same time, said models must retain previously learned knowledge while integrating new information. Secondly, there is a point to make about the practicality of using these models: the required amount of computational resources and memory should stay constant; the time-scale should stay realistic, and there should not be access to previously encountered data. Next, a lot of emphasis is being put on scalability. Models should be able to incrementally scale with the increasing complexity of knowledge and skills. Finally, significant value can be obtained from the ability to learn autonomously: either in an unsupervised, self-supervised, or at least minimally-supervised manner, as data labelling can be a very expensive task. Some of these and more principles of Continual Learning are outlined by Cossu et. al. in their work "Sustainable Artificial Intelligence through Continual Learning." [1]

As mentioned previously, one of the most core challenges in CL remains catastrophic forgetting. This tendency to abruptly forget previously learned information arises due to the static nature of traditional neural networks, which tend to overwrite old weights when updated with new data. The balance between remembering old information and learning new information is an inherent problem in lifelong learning, both in natural and artificial intelligence, therefore this is an issue that has to be addressed in almost every CL scenario.

The stability-plasticity dilemma, which is also a very important concept in Reinforcement Learning, encapsulates this issue: a continually learning model must balance stability (retaining old knowledge) with plasticity (adapting with new knowledge). High stability can hinder the learning of new information, while high plasticity can lead to forgetting old information. Effective CL strategies strive to find

an optimal balance between these two aspects. Some of the CL methods that address this issue are discussed further in this section.

Continual Learning can be applied in various scenarios, each presenting unique challenges and requirements. According to van de Ven et al., there are three main CL scenarios: Task-Incremental, Class-Incremental, and Domain-Incremental [2].

- Task-IL: the most general case, where every consequent task (containing a batch of streaming data) can be different. There are clear boundaries between each task, and the model itself is aware of task identities during training and testing. An example of this could be a robot, which gets various commands (e.g., to clean up the floor, or to do the dishes), and must perform that task as its output.
- Class-IL: classification scenario where each new task might contain new classes that have not been encountered previously. Task identities and boundaries are not specified, and the model's goal is to distinguish between all classes seen so far. For example, a robot is being shown a new object and is told what this object is, later must be able to recall this object when exposed to it again.
- Domain-IL: this scenario involves changes only in the distribution of data. The number of classes stays the same, but the same input data may belong to a different class. The main challenges here are domain shift and concept drift. The changes in data distribution or the relationship between input data and target classes makes a traditional ML model's performance deteriorate, forcing ML engineers to constantly monitor the model's performance and take action when this happens. In CL terminology, different domains can be represented by the task identities, which the model is unaware of. It may change arbitrarily, and the model is required to adapt to new domains, while retaining performance on previous ones.

The definition of task identities, i.i.d and non-i.i.d data, the size of incoming batches, the amount of supervision – all these factors can differ depending on the type of the CL problem that is being solved. This makes it rather difficult to compare CL methods. However, we can point out several types of approaches.

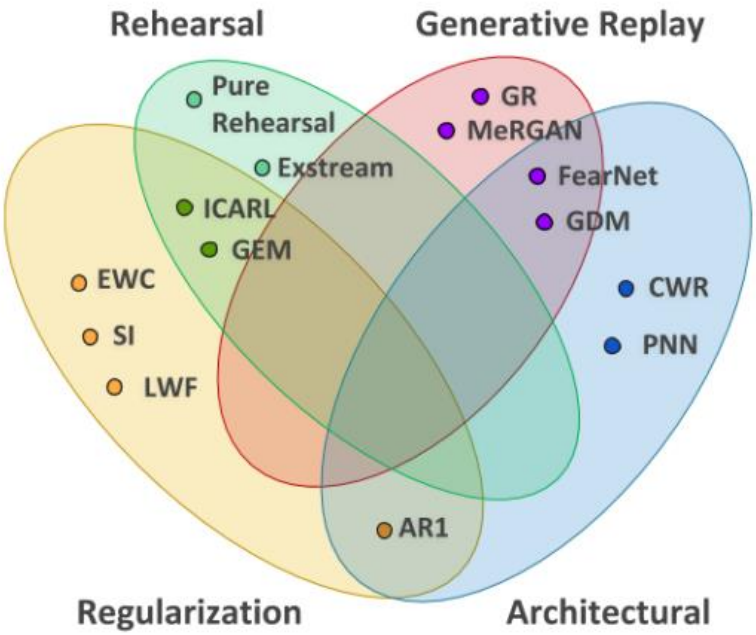

Figure 1: Venn diagram of the most popular CL strategies categorized into four approaches [3]

According to Lesort et. al., the most popular CL strategies can be divided into five general approaches: Rehearsal-based, Generative Replay, Regularization-based, Architectural, and Hybrid strategies [3].

- Rehearsal-Based Methods. Rehearsal is one of the most common ways to deal with forgetting for humans, so naturally it made sense to try the same technique in the context of AI. These methods attempt to maintain past knowledge by storing raw samples of data and periodically revisiting them during training. Such methods can differ in the way they choose the exemplars to store in memory. Examples include Maximally Interfered Retrieval (MIR) [4] and Latent Replay [5].

- Generative Replay. These methods utilize generative models to recreate previous data rather than storing raw data. This approach leverages the power of generative models like GANs (Generative Adversarial Networks) to generate pseudo-experiences of past tasks. Some of the known strategies to utilize this are Deep Generative Replay (DGR) [6] and Memory Replay GAN (MeRGAN) [7].

- Regularization-Based Methods. These approaches introduce additional terms to the loss function during training to protect the weights important for previously

learned tasks. These methods penalize changes to crucial parameters, thereby reducing catastrophic forgetting. The most popular strategies here are: Learning without Forgetting (LwF) [8], Elastic Weight Consolidation (EWC) [9], and Synaptic Intelligence (SI) [10].

- Architectural Methods. The aim is to modify the neural network architecture to accommodate new tasks without interfering with the knowledge gained from previous tasks. These methods often involve adding new network modules or expanding existing structures to ensure that learning new tasks does not overwrite the old knowledge. Here, the more known methods include Copy Weights with Re-Init (CWR) [11] and Progressive Neural Networks (PNN) [12] among others.
- Hybrid Strategies. Finally, these methods combine multiple approaches to leverage their strengths and mitigate their weaknesses. By integrating architectural, regularization-based, and rehearsal-based methods, hybrid strategies aim to achieve a more balanced and robust continual learning model. The more known hybrid methods include Incremental Classifier and Representation Learning (iCaRL) [13] and Gradient Episodic Memory (GEM) [14], which combine replay and regularization, and AR1 [15], which combines architectural, regularization, and replay components.

As can be seen, a variety of approaches have been proposed that try to deal with the issue of catastrophic forgetting and perform model adaptation in a Task-IL, Class-IL or Domain-IL setting. However, these methods still have a very niche use case. The problem is that all models and strategies discussed so far require supervision to perform adaptation of the model. The presence of labels as a supervised signal is not always present in a real-world scenario. This paper aims to challenge the notion of requiring additional information to perform model adaptation by attempting to apply some of the existing CL techniques in a very standard, traditional machine learning scenario, with the emphasis on not requiring any additional data.

Machine learning algorithms that learn from data without explicit labels are usually referred to as Unsupervised Learning. This is a particularly challenging area,

as the algorithms have to either do some sort of self-supervision by utilizing generative models or perform cluster analysis to make use of patterns that emerge from the data distribution. In the context of CL, the ability to adapt the model during its lifetime without requiring supervision can be very useful, since it drastically expands the scope of problems that CL models can be applied to.

One idea that is currently being explored is Self-Supervised Learning, where techniques like contrastive learning are used to distinguish between similar and dissimilar data points. Prototypical Contrastive Learning (PCL) [16] is a method that bridges contrastive learning with clustering, outperforming other state-of-the-art contrastive learning methods.

To test the hypothesis that CL can be applied in a traditional ML setting, this setting needs to be defined in Continual Learning terms. For simplicity, the problem of binary classification has been considered. Since there are always two classes to choose from, this is a Domain-IL problem. Here, the model must continually learn and adapt to domain changes to be able to effectively respond to domain shift and concept drift, while also keeping good performance when domain returns to its previous states. Traditionally, the binary classification problem would be solved by collecting a training dataset and training a model to distinguish between the classes. In CL, we can split this into two phases: the training phase, where the model is trained in a supervised manner; and the adaptation phase, where the model is used in the real world and, at the same time, an Unsupervised CL (UCL) technique constantly analyses the input data and adapts the model's weights or parameters. In traditional ML, the differences between the training dataset and the real world would inevitably hurt the model's performance, so some data would need to be collected and labelled again to re-train the model. Therefore, with the use of UCL it might become possible to develop a model that can continuously adapt to domain changes without the need for re-training.

In this section the current Continual Learning scene has been explored. In the understudied field of UCL an opportunity has been found to make an AI model that autonomously adapts to ever-changing data. In the next section, a problem suitable for testing this hypothesis is presented – the problem of parking lot occupancy detection.

1.3 Parking lot occupancy detection

In today's world having your own vehicle has become a social norm. With the ever-increasing number of vehicles on the streets the problem of efficient parking is becoming more and more evident. Drivers in big cities often encounter a problem finding an empty parking space during peak hours. There are many ways to address this problem on a city management scale, such as balancing road traffic, building more parking lots, or encouraging the use of public transport.

On a smaller scale it is possible to improve drivers' quality of life by making parking more convenient. One of the ways to do this is to implement an automated parking management system, which tracks the availability of each parking spot and provides it to the managers of the parking lot, as well as the drivers who may consider using it. This way drivers will have access to the state of each parking lot in advance, which makes planning a lot easier. At the same time, it also makes managing a parking lot easier, because a lot of manual work gets automated. Such a system could have various additional functionalities, which could help improve drivers' lives. Examples of such functionality are number plate tracking, automated payment system, fraud detection, or statistics collection.

In practice such automated systems are partially achieved by using sensors that are installed on each parking space. Unfortunately, this solution is often very costly, thus making it impractical in less developed countries. Amato et al. in their article "Car Parking Occupancy Detection Using Smart Camera Networks and Deep Learning" proposed a cheaper solution, which involves using "smart cameras" in conjunction with a Convolutional Neural Network (CNN) model to detect parking space occupancy in real time. [17]

The relevance of this problem has also been highlighted by Kreshchenko and Yushchenko in their article "Parking spot occupancy classification using deep learning." The need to improve the parking infrastructure arises, which may include increasing the number of parking lots, setting parking fees, and automating certain processes. The authors claim that an excellent way to make parking easier without requiring significant costs is to create a parking information system. In addition, such

system is easily scalable, because adding one parking lot to a system only requires adding one or two cameras. [18]

Very recently, de Almeida et al. released a comprehensive review of existing public datasets, created specifically for solving Computer Vision problems for parking lots, as well as scientific works that use such datasets [19]. The authors defined criteria to choose useful datasets, dropping ones that would not be useful in real world scenarios. Finally, the ones that were chosen were PKLot, CNRPark-EXT, and PLds. These datasets include parking lot images taken from video cameras during a certain period. The authors conclude that only PLds contains images at night and during snowy weather, though all of them contain sunny, overcast, and rainy weather images. CNRPark-EXT, however, has eleven camera angles available, which is more than the other ones.

De Almeida et al. also compare the results of different feature extraction-based and deep learning-based approaches when solving the problem of parking lot occupancy detection. They specify the performance each work has achieved in three different cases: train and test on the same camera angle, test on a different camera angle but same parking lot, and finally, test on a different parking lot. The third variant is the most interesting, as it resembles the real-life scenario the most, provided no additional data needs to be collected when the model is used. In this variant, as expected, the average performance on reproducible works was reported as 91.8%, which is 4.3% lower than the average performance on the first variant. The authors also specify whether each reviewed work is biased, reproducible, or uses private datasets, other than the three publicly available ones stated above. Among these experiments one was chosen for reproduction and improved using CL methodologies. It was a work by Amato et al., which was reproducible, achieved excellent results, and was well-documented [20]. The specifics of the reproduction process of the experiment are described in section 2.

This section explored the problem of parking lot occupancy detection, the relevance of the problem, and existing solutions. One such solution has been chosen as a baseline to compare with the CL method. The following section describes how CL

can be used to achieve better results when solving the problem of parking lot occupancy detection.

1.4 Improving parking lot occupancy detection with CL

The problem of parking lot occupancy detection does not involve any changes in the number of classes, as it is always two: either free or occupied. The data domain, however, changes: in this case the domain is represented by the weather, the time of day, and the season. On a lower level, this affects the lighting conditions and the color scheme of the image. Such changes in the domain can introduce additional challenge to any form of intelligence: if a human that has been watching a parking lot only during the day is suddenly put in a poorly lit nighttime environment, it can become more difficult to do something even as simple as telling whether the parking spot is busy or free. Admittedly, the human eye is good at adapting to gradual changes, which makes it a lot better suited for this kind of job.

Gradual changes in the environment, however, can potentially be utilized by a machine, too. In fact, Taufique et al. in their paper "Unsupervised Continual Learning for Gradually Varying Domains" proposed a CL method called UCL-GV that performs Unsupervised Domain Adaptation (UDA) in the CL paradigm [21]. To address the issue of catastrophic forgetting and to minimize local domain shift, the authors utilized episodic memory replay with buffer management. The authors consider a setting of UCL for domain adaptation in gradually varying domains, meaning that it is assumed that the domain change in each consecutive batch of data is minimal. This assumption makes it possible to utilize Prototypical Contrastive Loss [16] to improve domain alignment between the incoming batch samples and the existing samples stored in the buffer.

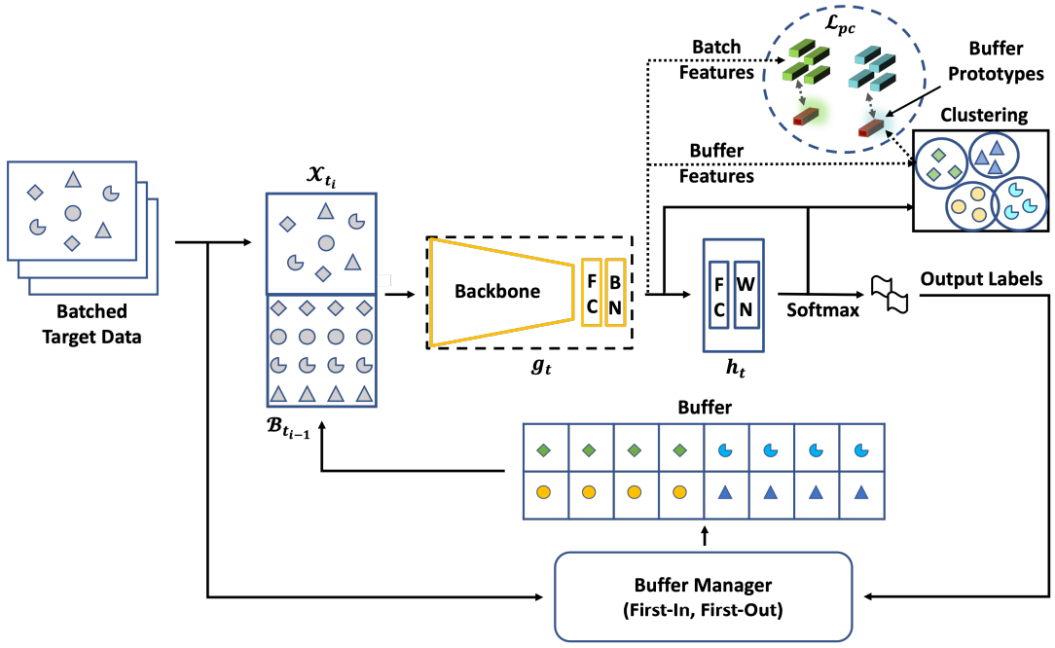

Figure 2: The architecture of UCL-GV [21]

UCL-GV uses a model architecture proposed by Liang et. al. [22], where the model consists of two parts: the feature extractor with a backbone, a fully connected layer, and a batch normalization layer; and the classifier (hypothesis layer) that has a fully convolutional layer and a weight normalization. Both batch and buffer samples are used with a custom loss function to execute model adaptation on the feature extractor part of the model, while the classifier remain frozen. The method was tested on the rotating MNIST dataset [23] and CORe50 [11]. The reported results of the proposed method significantly outperform many existing UDA methods.

In the case of parking lot occupancy detection, the nature of continual video feed of a camera is the perfect case of a gradually changing environment. This makes it possible to utilize UCL-GV, showing that even in the most optimized and studied problems such as binary classification it is possible to apply CL methodology to achieve more accurate and adaptive performance.

2 IMPLEMENTATION

2.1 Datasets and Benchmarks

The datasets used for both model reproduction and CL experiments are PKLot, CNRPark, and CNRParkEXT.

2.2 Reproduction of the baseline model

As stated previously, the chosen baseline model is mAlexNet, proposed by Amato et. al. [20] This model was chosen because of its excellent performance, but also because it was specifically made to have small resource requirements to be able to run on a Raspberry Pi microcomputer. The architecture of the model can be seen in Figure 3. This paper did provide a link to the source code, but it was implemented using the Caffe library – a C++ library for machine learning. All experiments were performed using the PyTorch library, so the model implementation had to be re-written.

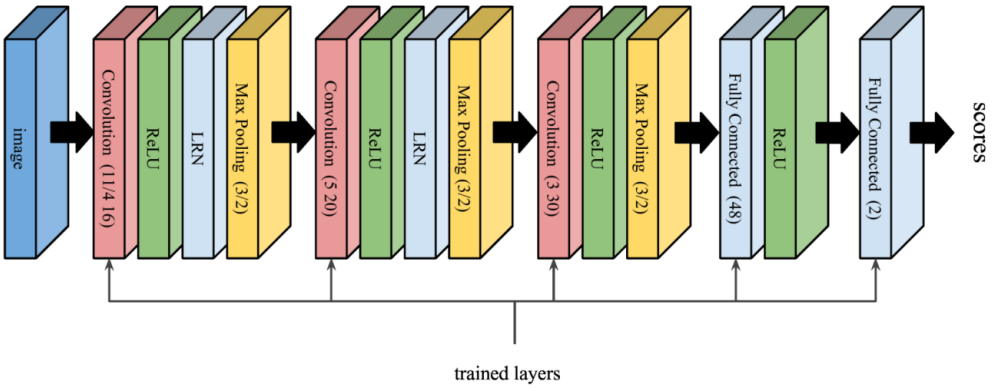

Figure 3: The architecture of mAlexNet [20]

This model is a simplified version of AlexNet, that the authors claim to hit the perfect balance between computation time and accuracy for small devices. It contains a sequence of three convolutional layers with an increasing number of output channels (16, 20, and 30), decreasing kernel size (11, 5, and 3), and the first one has a stride of 4. They are all followed by a Rectified Linear Unit, and a Max Pooling layer with a kernel size of 3 and a stride of 2. Additionally, the authors also used Local Response Normalization in the first two convolutional layers, but the parameters of these layers were not specified, so the parameters of the original AlexNet were taken. The

convolutional layers are followed by two fully connected layers. In the end, a softmax function is applied.

The authors did not report on weight and bias initialization; however, they did provide source code to their experiments. In the source code all weights were initialized using Xavier uniform distribution, and the bias was initialized with ones. During training, cross-entropy loss was used for backwards propagation, and SGD was used as the optimizer. It is also worth noting that biases had a learning rate multiplier of 2, and the weight decay parameter was 0.0005 for all layers except for convolution biases. Learning rate started at 0.0008 and was multiplied by 0.75 every two epochs, for a total of 6 epochs.

2.3 Applying Continual Learning

To implement UCL-GV the Avalanche library was used. This is a Continual Learning library based on PyTorch that was specifically created to make it easier to conduct CL experiments for researchers. It is still being actively developed and extended with newly emerging methods, therefore another reason it was chosen to implement UCL-GV was to potentially add this paper's implementation of it to the Avalanche library.

In Avalanche, all CL methods are implemented in the form of either a strategy, which is a wrapper that takes in a PyTorch model, an optimizer, and a loss criterion, and implements the training and evaluation loops; or a plugin that can be used to extend a strategy using callback functions. The latter was ultimately chosen for better simplicity.

On initialization the plugin allocates a memory buffer. The callback functions used were before_eval_exp, which executes before the beginning of each evaluation experience, after_eval_forward, which executes in the evalutation loop right after the forward pass, and after_eval_exp, which executes after the end of each experience. Before the beginning of the experience the data loader gets extended with samples from the memory buffer, and after the end of the experience the buffer gets updated with new samples.

Because the algorithm requires splitting the model into two parts – the feature extractor and the classifier – it is required that the model implements a corresponding function to only run the feature extractor. After the forward pass the algorithm adapts the feature extractor part of the model. To do this, K-means clustering is used to generate pseudo-labels and cluster centers from the output features. Next, UCL-GV uses a combination of prototypical contrastive loss, entropy, equal diversity loss, and cross-entropy loss to get overall loss, as a weighted sum with weights as hyperparameters:

$$L = L_{en} + \gamma_1 L_{eq} + \gamma_2 L_{ce} + \gamma_3 L_{pc}$$

Only cross-entropy loss was already implemented in the PyTorch library, so the other ones had to be implemented manually, according to their definitions in the paper. Finally, the overall loss was used to backpropagate model adaptation.

3 RESULTS

3.1 Reproduction of the baseline model

Table 1 shows reproduction results of some of the experiments reported by Amato et. al. [20] The method marked as "Reproduced" refers to the experiments run by the author. "Original" refers to the experiments reported in the original paper.

Method	Train DS / split	Test DS / split	Accuracy	AUC
Reproduced	CNRPark / Odd	CNRPark / Even	91.43%	0.97
Original	CNRPark / Odd	CNRPark / Even	90.13%	0.94
Reproduced	CNRPark / Even	CNRPark / Odd	91.5%	0.97
Original	CNRPark / Even	CNRPark / Odd	90.71%	0.92
Reproduced	CNRPark / All	CNRParkEXT / Test	93.64%	0.98
Original	CNRPark / All	CNRParkEXT / Test	93.52%	0.98
Reproduced	CNRPark / All	PKLot / TwoDays	85.08%	0.93

Original	CNRPark / All	PKLot / TwoDays	90.38%	0.99
Reproduced	PKLot / Train	CNRParkEXT / Test	74.77%	0.85
Original	PKLot / Train	CNRParkEXT / Test	83.83%	0.91
Reproduced	PKLot / Train	PKLot / Test	84.29%	0.97
Original	PKLot / Train	PKLot / Test	98.07%	0.99
Reproduced	CNRParkEXT / Cam1	CNRParkEXT / Cam2	95.34%	0.99
Original	CNRParkEXT / Cam1	CNRParkEXT / Cam2	95%	-

Table 1: experiment reproduction results of Amato et. al. [20]

The goal of running these experiments was to make sure that the model was reproduced correctly. The results show that the differences are quite marginal, even though there are some experiments that differ quite substantially. Still, it can be concluded that the model has been reproduced successfully enough.

3.2 Applying Continual Learning

To utilize UCL-GV a scenario had to be simulated where all images are ordered by the time of being taken.

Method	**Train DS / split**	**Test DS / split**	**Accuracy**
mAlexNet	CNRParkEXT / All	CNRPark / All	95.21%
mAlexNet+ UCL-GV	CNRParkEXT / All	CNRPark / All	95.73%
mAlexNet	CNRParkEXT / Cam1-8	CNRParkEXT / Cam9	96.01%
mAlexNet+ UCL-GV	CNRParkEXT / Cam1-8	CNRParkEXT / Cam9	96.85%
mAlexNet	PKLot / TwoDays	PKLot / NotTwoDays	91.74%

mAlexNet+ UCL-GV	PKLot / TwoDays	PKLot / NotTwoDays	92.11%

Table 2: experiment results of improving the mAlexNet with UCL-GV

The results indicate that augmenting mAlexNet with UCL-GV leads to a better model performance. The findings can have significant implications in the field of traditional classification, highlighting the potential of CL to enhance the capability and efficiency of standard ML models.

CONCLUSION

This paper attempts to view the paradigm of Continual Learning from a completely new perspective. An attempt has been made to utilize CL methodologies to achieve better performance on traditional problems, as opposed to solving new types of problems. This suggests that even solutions to problems that are often considered as "already solved" can be further improved by introducing contemporary CL methods. Future work may include attempts to improve more classical ML problems using this technique.

WORKS CITED

1. Cossu, Andrea, Marta Ziosi, and Vincenzo Lomonaco. “Sustainable Artificial Intelligence through Continual Learning.” CAIP 2021: Proceedings of the 1st International Conference on AI for People: Towards Sustainable AI, CAIP 2021, 20-24 November 2021, Bologna, Italy. European Alliance for Innovation, 2021.
2. van de Ven, G.M., Tuytelaars, T. and Tolias, A.S. “Three types of incremental learning.” Nat Mach Intell 4, 2022, pp. 1185–1197. https://doi.org/10.1038/s42256-022-00568-3.
3. Lesort, Timothée, et al. "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges." Information fusion 58 (2020): 52-68.
4. Aljundi, Rahaf, et al. "Online continual learning with maximal interfered retrieval." Advances in neural information processing systems 32 (2019).
5. Pellegrini, Lorenzo, et al. "Latent replay for real-time continual learning." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.
6. Shin, Hanul, et al. "Continual learning with deep generative replay." Advances in neural information processing systems 30 (2017).
7. Wu, Chenshen, et al. "Memory replay gans: Learning to generate new categories without forgetting." Advances in neural information processing systems 31 (2018).
8. Li, Zhizhong, and Derek Hoiem. "Learning without forgetting." IEEE transactions on pattern analysis and machine intelligence 40.12 (2017): 2935-2947.
9. Kirkpatrick, James, et al. "Overcoming catastrophic forgetting in neural networks." Proceedings of the national academy of sciences 114.13 (2017): 3521-3526.
10. Zenke, Friedemann, Ben Poole, and Surya Ganguli. "Continual learning through synaptic intelligence." International conference on machine learning. PMLR, 2017.

11. Lomonaco, Vincenzo, and Davide Maltoni. "Core50: a new dataset and benchmark for continuous object recognition." Conference on robot learning. PMLR, 2017.
12. Rusu, Andrei A., et al. "Progressive neural networks." arXiv preprint arXiv:1606.04671 (2016).
13. Rebuffi, Sylvestre-Alvise, et al. "icarl: Incremental classifier and representation learning." Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2017.
14. Lopez-Paz, David, and Marc'Aurelio Ranzato. "Gradient episodic memory for continual learning." Advances in neural information processing systems 30 (2017).
15. Maltoni, Davide, and Vincenzo Lomonaco. "Continuous learning in single-incremental-task scenarios." Neural Networks 116 (2019): 56-73.
16. Li, Junnan, et al. "Prototypical contrastive learning of unsupervised representations." arXiv preprint arXiv:2005.04966 (2020).
17. Amato, Giuseppe, et al. “Car parking occupancy detection using smart camera networks and Deep Learning.” 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, 2016, pp. 1212-1217, http://doi.org/10.1109/ISCC.2016.7543901.
18. Kreshchenko, Taras, and Yury Yushchenko. “Parking Spot Occupancy Classification Using Deep Learning.” Наукові записки НаУКМА. Комп'ютерні науки 5, 2022, pp. 72-78. https://doi.org/10.18523/2617-3808.2022.5.72-78.
19. de Almeida, P. R. L., Alves, J. H., Parpinelli, R. S., Barddal, J. P. “A Systematic Review on Computer Vision-Based Parking Lot Management Applied on Public Datasets.” Expert Systems with Applications, 198(116731), 2022. http://doi.org/10.1016/j.eswa.2022.116731.
20. Amato, Giuseppe, et al. “Deep learning for decentralized parking lot occupancy detection.” Expert Systems with Applications 72, 2017, pp. 327-334. https://doi.org/10.1016/j.eswa.2016.10.055.

21. Taufique, Abu Md Niamul, Chowdhury Sadman Jahan, and Andreas Savakis. “Unsupervised continual learning for gradually varying domains.” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.
22. Liang, Jian, Dapeng Hu, and Jiashi Feng. "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation." International conference on machine learning. PMLR, 2020.
23. Kumar, Ananya, Tengyu Ma, and Percy Liang. "Understanding self-training for gradual domain adaptation." International conference on machine learning. PMLR, 2020.

APPENDIX A

List of abbreviations

AI – Artificial Intelligence

CL – Continual Learning

CNN – Convolutional Neural Network

CV – Computer Vision

ML – Machine Learning

APPENDIX B

Link to the codes

https://github.com/74R45/parking-ucl