Review of Class-Incremental Methods

Giulia Ghisolfi, g.ghisolfi@studenti.unipi.it, Roll number: 664222
Master Degree in Computer in Science (Artificial Intelligence Curriculum), University of Pisa
Continual Learning course (791AA), Academic Year: 2023-2024

Abstract

Class-incremental learning is a scenario within incremental learning where algorithms must continuously differentiate between all classes encountered over time, with new classes introduced at each step while previous ones are not revisited. This report provides an analysis of methods for addressing the challenges of class-incremental learning, focusing on both memory-based and rebalancing classifiers: iCaRL, IL2M, and a Unified Classifier learned Incrementally via Rebalancing, as well as generative approaches like Memory Replay GANs. Each method introduces unique strategies to mitigate catastrophic forgetting and handle class imbalance in incremental learning scenarios. We explore the strengths and weaknesses of these approaches, highlighting their contributions to the field and identifying areas for potential improvement.

1 Introduction

Incremental learning is a paradigm in artificial intelligence, where models are designed to continuously learn new tasks without retraining from scratch. This capability is highly desirable, as it allows the reuse of resources and minimizes the computational cost of model updates.

In the context of incremental learning, the *class-incremental learning* (class-IL) scenario focuses on the ability of algorithms to distinguish between all classes encountered across different time steps, not just those from the current task. In this setting, new classes are introduced at each step, and older ones are not revisited, leading to a strong bias towards the most recent classes.

This report reviews methods addressing the class-incremental learning problem, dividing them into two main categories: memory-based and rebalancing classifiers, and generative approaches.

Memory-based and rebalancing classifiers rely on memory storage or rebalancing techniques to mitigate catastrophic forgetting and address class imbalance. Among these methods, we analyzed iCaRL (Incremental Classifier and Representation Learning) Rebuffi et al. [2017], which combines a nearest-mean classifier with a memory of exemplar images to balance new and old classes. We also examined IL2M (Class-Incremental Learning with Dual Memory) Belouadah and Popescu [2019], a method that introduces a dual memory mechanism to reweight predictions and maintain performance on older classes. Finally, we considered the model proposed in Hou et al. [2019], a Unified Classifier learned Incrementally via Rebalancing, which leverages feature rebalancing to address the bias toward newer classes and achieve a more balanced learning process.

Following this, we also analyzed generative approaches, which offer an alternative to memory-based strategies. Instead of storing real data, these methods utilize a generative model capable of synthesizing examples of previously learned classes during training. Specifically, we explored MeRGANs (Memory Replay Generative Adversarial Networks) Wu et al. [2018], which generate synthetic images of old classes, allowing the model to maintain performance without relying on real exemplars.

The aim of this report is to analyze these methods, identifying their strengths, weaknesses, and key differences, as well as their effectiveness in mitigating catastrophic forgetting in class-incremental scenarios.

2 Problem Setting

The class-incremental learning problem is a specific scenario in incremental learning where the goal is to develop models capable of learning new classes over time without forgetting previously learned ones.

Formally, we adopt the definition provided in Rebuffi et al. [2017]. An algorithm qualifies as class-incremental if it satisfies the following three properties:

1. **Incremental Training:** The algorithm must be trainable from a data stream where examples of different classes are encountered at different times.
2. **Multi-Class Classification:** At any point in time, it should provide a competitive multi-class classifier for all classes observed so far.
3. **Efficiency:** The computational requirements and memory footprint should remain bounded, or at least grow very slowly, with respect to the number of classes observed.

The first two criteria capture the core essence of class-incremental learning, while the third criterion ensures scalability and prevents trivial solutions.

2.1 Challenges

One of the main challenges in incremental learning is *catastrophic forgetting*, which occurs when a model abruptly forgets previously learned tasks while adapting to new ones. This phenomenon arises because neural networks update their weights to minimize the error on the current task, often overwriting representations of prior tasks. To mitigate catastrophic forgetting, it is crucial to approximate the loss on previous data or apply regularization techniques to preserve knowledge of past tasks.

Another core challenge in continual learning is the *stability-plasticity trade-off*. Stability refers to the model's ability to retain knowledge from old tasks, which can be maximized by freezing parameters. Plasticity, on the other hand, is the model's ability to learn new tasks, often achieved by fine-tuning or retraining the model. These two objectives are inherently conflicting, requiring careful tuning to strike a balance that minimizes forgetting while enabling effective learning of new tasks.

A further challenge is the *class imbalance problem*, which is specific to class-incremental learning tasks. Classes are best modeled when they are first learned with all data available. However, recently introduced classes often dominate the model's predictions due to their higher representation in the training data. This results in a disparity in class representation and leads to a phenomenon known as *classifier bias*, where a class-incremental learning algorithm tends to favor new classes over previously learned ones. Addressing this challenge requires strategies such as memory rebalancing, feature rectification, or weighted loss adjustments to mitigate the imbalance and ensure fair performance across all classes.

Addressing these challenges requires innovative methodologies, including memory-based techniques, generative approaches, and regularization strategies, as explored in this report.

2.2 Knowledge Distillation

Knowledge Distillation is technique that has been proven to be effective for transferring knowledge from a larger model, called the *teacher*, to a smaller model, called the *student*. The student model is typically easier to train and faster to deploy, making this approach valuable for achieving a reduction in model size while maintaining performance.

This method has been found to be useful in a class-IL setting, where, at each step, new classes are added to the single output head. However, this approach presents a significant challenge, as the classifier grows with time, the output probability distributions of the teacher and student models have different sizes, this hinders the direct comparison of these models using KL-divergence.

To address this challenge, Knowledge Distillation is computed over the *old units* (the previously observed classes) using the new data, while the Cross-Entropy loss is applied to units (both old and new classes) using the new data. A critical aspect of this approach is distinguishing between the old and new units during training.

3 Methodology: Class-Incremental Classifier

3.1 iCaRL: Incremental Classifier and Representation Learning

iCaRL (Incremental Classifier and Representation Learning), presented in Rebuffi et al. [2017] is a method designed to incrementally learn a large number of classes over an extended period. It simultaneously learns both classifiers and feature representations in the class-incremental setting, addressing the challenges of catastrophic forgetting and class imbalance.

The methodology of iCaRL is based on three main components, which in combination allow it to meet the criteria for class-incremental learning: Nearest-Mean-of-Exemplars Classification, Representation Learning with Knowledge Distillation and Prototype Rehearsal, and Prioritized Exemplar Selection.

By combining knowledge representation with knowledge distillation and employing these strategies, iCaRL effectively addresses the core challenges of class-incremental learning, such as catastrophic forgetting and maintaining fair performance across all classes.

Prototype-based Classification (Nearest-Mean-of-Exemplars)

To classify a new image x, iCaRL computes a prototype vector μ_y for each class y as the mean of the feature vectors of the stored exemplars.

iCaRL uses a classification strategy based on the nearest-mean rule. For each class, a mean feature vector μ_y (prototype) is computed using a subset of exemplars. At inference time, the class of an input is determined by finding the nearest prototype in the feature space.

$$\mu_y = \frac{1}{|P_y|} \sum_{p \in P_y} \phi(p)$$

Where P_y is the set of exemplars for class y, and $\phi(p)$ is the feature representation of exemplar p.

The image x is classified by assigning it to the class with the nearest prototype in terms of distance:

$$y^* = \arg \min_{y=1,\dots,t} \|\phi(x) - \mu_y\|$$

Representation Learning with Knowledge Distillation and Prototype Rehearsal

iCaRL updates its model parameters incrementally by processing batches of classes at a time. Whenever data for new classes becomes available, iCaRL executes an update routine that adjusts the model parameters and its internal knowledge. This update combines the newly available data with a set of exemplars representing previously learned classes. The training process ensures that the model can integrate new knowledge while preserving performance on old classes.

iCaRL combines exemplar replay with *knowledge distillation* to preserve previously acquired knowledge. During training, the dataset is composed of stored exemplars P_y and new data X_y:

$$\mathcal{D} = \bigcup_{y=s}^{t} \{(x, y) : x \in X_y\} \cup \bigcup_{y=1}^{s-1} \{(x, y) : x \in P_y\}$$

The loss function consists of two terms:

1. Classification loss: encourages the network to correctly classify new data by maximizing the likelihood of correct predictions for new classes.
2. Distillation loss: preserves knowledge of old classes by replicating the outputs of the previous model and maintaining consistency in predictions.

$$\begin{aligned}\mathcal{L}(\Theta) =& \mathcal{L}_{\text{classification}} + \mathcal{L}_{\text{distillation}} \\ =& - \sum_{(x_i,y_i)\in\mathcal{D}} \left[\sum_{y=s}^{t} \delta_{y=y_i} \log g_y(x_i) + \delta_{y\neq y_i} \log(1 - g_y(x_i)) \right. \\ &\left. + \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1 - q_i^y) \log(1 - g_y(x_i)) \right]\end{aligned}$$

Where $g_y(x_i)$ is the output of the current model for class y given input x_i, representing the predicted probability, q_i^y is the output of the previous model for class y given the same input, representing the prior prediction probability, and $\delta_{y=y_i}$ is an indicator function that equals 1 if $y = y_i$ (correct class), and 0 otherwise.

The incremental training process ensures a balance between stability (retaining knowledge of old classes) and plasticity (learning new classes effectively), allowing iCaRL to perform well in class-incremental learning scenarios. Additionally, prototype rehearsal is used by replaying stored exemplars during training, reinforcing the representation of old classes.

Prioritized Exemplar Selection (Herding)

After each update of the model representation, the exemplar set is updated to maintain a fixed memory size. iCaRL employs a herding algorithm to select the most representative exemplars for each class, ensuring that the memory is distributed evenly across all classes.

The number of exemplars allocated to each class m is computed as $\frac{K}{t}$ where K is the total memory size, and t is the number of classes observed so far.

For previously learned classes (*old classes*), whose exemplar sets were already computed, the size of their exemplar set is reduced by keeping only the first m elements.

For newly introduced classes (*new classes*), the exemplar set P is constructed iteratively to approximate the class mean vector μ. The herding algorithm selects the exemplar p_k at each step to minimize the deviation from the class mean:

$$p_k = \arg\min_{x \in X} \left\| \mu - \frac{1}{k} \left(\phi(x) + \sum_{j=1}^{k-1} \phi(p_j) \right) \right\|,$$

where X is the full set of training samples for the class, $\phi(x)$ is the feature representation of sample x, and $\mu = \frac{1}{n} \sum_{x \in X} \phi(x)$ is the mean feature vector of the class.

This process ensures that the selected exemplars are the most representative of the class in the learned feature space, allowing the model to maintain knowledge of old classes effectively.

3.2 IL2M: Class-Incremental Learning with Dual Memory

The model introduced in Belouadah and Popescu [2019] builds on the basic idea that classes are best modeled when first learned with all data available. Initial class statistics are reused in each subsequent incremental state to rectify the prediction scores of past classes.

IL2M uses a fixed deep neural network (DNN) architecture and a bounded memory for handling past classes. Specifically, it employs two types of memory. The first memory stores exemplar images of past classes, serving as a representative subset of their data.

The second memory stores the initial class statistics obtained when the classes were first learned, which are considered the best representations the model can provide. These class representations require minimal storage space, allowing for an optimal representation of each class while minimizing memory usage.

IL2M also leverages a distillation-based loss to reduce the discrepancy between the activations of past classes. This approach mitigates catastrophic forgetting by ensuring the alignment of predictions for old classes. Compared to other methods addressing the same problem, including iCaRL, IL2M distinguishes itself by its explicit use of initial class statistics to refine predictions in later stages.

Bounded Memory and Partial Access to Past Data

In the first memory, IL2M provides partial access to past data, maintaining a fixed number of samples $m = \frac{K}{t}$ for each class, where K is the total memory size, and t is the number of classes seen so far.

At every incremental step, new classes are introduced, causing the total number of classes to increase from P to N ($N > P$). However, due to the bounded memory, only a small fraction of the data from past classes can be retained. This leads to an increasing imbalance in the data as more classes are learned ($\frac{K}{N} < \frac{K}{P}$).

Prediction Rectification for Past Classes

To compensate for the prediction bias toward new classes caused by this imbalance, IL2M rectifies the predictions of past classes C_i ($i = 1, \ldots, P$) using the following correction formula:

$$p^r(C_i) = \begin{cases} p(C_i) \times \frac{\mu(M_N)}{\mu(M_P)} \times \frac{\mu_N(C_i)}{\mu_P(C_i)}, & \text{if pred} = \text{new}, \\ p(C_i), & \text{otherwise.} \end{cases}$$

Above P is the initial state in which C_i was learned, N is the current incremental state, $p(C_i)$ is the raw prediction score for C_i in state N, $\mu_P(C_i)$ is the mean classification score of C_i in state P, obtained using all training data when the class was initially learned, $\mu_N(C_i)$ is the mean classification score of C_i in state N, obtained using the current exemplar set, $\mu(M_P)$ is the model confidence in state P, computed as the averaged prediction score over all new training data available at state P, and $\mu(M_N)$ is the model confidence in state N, computed as the averaged prediction score over all new training data available at state N.

The rectification is applied only if an image is initially predicted as belonging to a new class. If a past class is directly predicted, no rectification is applied since there is no inherent bias toward new classes.

The additional computational complexity introduced by the rectification formula is very low compared to the overall complexity of a deep neural network architecture. This ensures that the method remains computationally efficient while effectively mitigating the bias toward new classes.

3.3 Learning a Unified Classifier Incrementally via Rebalancing

In Hou et al. [2019], the authors presented a novel framework for incrementally learning a unified classifier, with the objective of treating both old and new classes uniformly.

This framework addresses the limitations of iCaRL by introducing three components to balance the bias toward newly added classes in the model. First, the imbalance in the magnitudes of old and new class predictions is mitigated through *cosine normalization*, which enforces balanced magnitudes across all classes, including both old and new ones. Second, the forgetting of old class configurations is addressed with the *less-forget constraint*, which preserves the geometric configuration of old classes. Finally, the overlap between old and new classes in the feature space is addressed using *inter-class separation*, which encourages a large margin to separate old and new classes.

To effectively rebalance the training process, these three components are incorporated into the proposed method. While this method belongs to the distillation-based category, it differs from previous works in a critical aspect: rather than simply combining different objective terms to balance old and new classes, it systematically investigates the adverse effects of imbalance and provides a comprehensive solution to address the issue from multiple perspectives.

Cosine Normalization

This component normalizes the weight vectors of the classes in the neural network, ensuring balanced magnitudes for both old and new classes. The predicted probability for a class C_i is defined as:

$$p(x) = \frac{\exp(\eta \cdot \langle \bar{\theta}_i, \bar{f}(x) \rangle)}{\sum_j \exp(\eta \cdot \langle \bar{\theta}_j, \bar{f}(x) \rangle)},$$

where η is a scalar parameter controlling the peakiness of the softmax distribution (as $\langle \bar{v}_1, \bar{v}_2 \rangle$ is restricted to $[-1, 1]$), $f(.)$ is the feature extractor, θ_i represents the weight vector (class embedding) for class C_i, $\langle \bar{\theta}_i, \bar{f}(x) \rangle$ denotes the normalized dot product between the input feature vector and the weight vector of class C_i, with $\bar{v} = v/\|v\|_2$ denoting the l_2-normalized vector and $\langle \bar{v}_1, \bar{v}_2 \rangle = \bar{v}_1^T \bar{v}_2$ measuring the cosine similarity between two normalized vectors.

Less-forget Constraint

This constraint preserves the geometric configuration of the old classes in the feature space. It is implemented using a distillation loss:

$$L_{\text{LF}} = 1 - \langle \bar{f}^*(x), \bar{f}(x) \rangle,$$

where $\bar{f}^*(x)$ and $\bar{f}(x)$ are respectively the normalized features extracted by the original model and those by the current one.

The degree of need to preserve previous knowledge varies depending on the number of new classes introduced in each phase. To address this, the loss was weighted by a set of adaptive weights, denoted as λ, defined as follows:

$$\lambda = \lambda_{\text{base}} \cdot \frac{|C_n|}{|C_o|},$$

where $|C_o|$ and $|C_n|$ represent the number of old and new classes in each phase, respectively, and λ_{base} is a fixed constant for each dataset. In general, λ increases as the ratio of the number of new classes to the number of old classes increases.

Inter-class Separation

To enhance the separation between old and new classes, the method introduces a margin-based term that encourages larger distances between classes:

$$L_{\text{margin}} = \sum_{k=1}^{K} \max\left(0, m - \langle \bar{\theta}(x), \bar{f}(x) \rangle + \langle \bar{\theta}_k, \bar{f}(x) \rangle\right).$$

Above m is the margin threshold, $\bar{\theta}(x)$ represents the ground-truth class embedding of x, and $\bar{\theta}_k$ denotes one of the top-K new class embeddings selected as hard negatives for x.

Final Loss Function

The total loss combines the terms described above:

$$L = \frac{1}{|N|} \sum_{x \in N} \left(L_{\text{ce}}(x) + \lambda L_{\text{LF}}(x)\right) + \frac{1}{|N_o|} \sum_{x \in N_o} L_{\text{margin}}(x),$$

where $L_{\text{ce}}(x)$ is the cross-entropy loss for sample x, N is a training batch drawn from X, $N_o \subset N$ represents the reserved old samples contained in N, and $|N|$ and $|N_o|$ are the sizes of the full batch N and the reserved old sample set N_o, respectively.

4 Methodology: Generative Replay

Generative tasks are inherently more complex than classification tasks, and in the context of class-incremental learning, they are equally susceptible to the challenge of *catastrophic forgetting*. In this setting, the model not only needs to preserve its ability to generate data from previously learned classes but must also integrate new tasks without overwriting prior knowledge.

4.1 Memory Replay GANs

The task addressed by Memory Replay GANs (MeRGANs), presented in Wu et al. [2018], involves sequential learning in the context of image generation with GANs, where the primary challenge is to effectively mitigate catastrophic forgetting.

Framework Overview

MeRGANs utilize a conditional GAN framework that incorporates memory replay as the main mechanism to prevent forgetting. The framework consists of three key components: a generator, a discriminator, and a classifier.

The generator is responsible for generating synthetic images from both newly introduced and previously learned classes, conditioned on class labels c. The discriminator distinguishes between real and generated images. It shares most layers with the classifier but uses a separate output layer for its specific task. The classifier is a task-specific component that shares the feature extraction layers with the discriminator but has its own output layer for classification. Its purpose is to predict class labels, forcing the generator to produce meaningful images aligned with target labels.

The conditional GAN framework generates data conditioned on the class label c and latent variable z. The generator produces synthetic samples as $\tilde{x} = G(z, c)$, where z is a latent vector and c is the class label.

During training, the network addresses both the adversarial game and the classification task.

Sequential Learning in GANs

To adapt the conditional GAN framework to a sequential learning scenario, the model must not only learn from new data arriving with each task but also retain knowledge of previously learned tasks.

The joint training process for the GAN framework can be naturally extended to a sequential learning scenario. For a given task t, the generator and discriminator are optimized using the following objectives:

$$\min_{\theta_G^t} \mathcal{L}_{\text{GAN}}^{G}(\theta_t, S_t),$$
$$\min_{\theta_D^t} \mathcal{L}_{\text{GAN}}^{D}(\theta_t, S_t),$$

where $\theta_t = (\theta_G^t, \theta_D^t)$ represents the parameters of the generator (G) and discriminator (D) during experience t, S_t is the training set for experience t, and $\mathcal{L}_{\text{GAN}}^{G}$ and $\mathcal{L}_{\text{GAN}}^{D}$ are the loss functions for the generator and discriminator, respectively. In this framework, the parameters θ_t are initialized as $\theta_t = \theta_{t-1}$, allowing the model to build on knowledge from previous experiences, and notably, the authors in Wu et al. [2018] assume that each experience t contains data from only one category, simplifying the sequential learning process.

Elastic Weight Consolidation (EWC), a method commonly used to prevent forgetting by penalizing significant changes in important parameters, is also employed by the authors in this framework. The generator's objective is augmented as:

$$\min_{\theta_G^t} \mathcal{L}_{\text{GAN}}(\theta_G^t, S_t) + \frac{\lambda_{\text{EWC}}}{2} \sum_i F_{t-1,i}(\theta_{G,t,i} - \theta_{G,t-1,i})^2,$$

where $F_{t-1,i}$ is the Fisher information matrix indicating the importance of parameter $\theta_{G,t,i}$.

Memory Replay Mechanism

MeRGANs extend traditional GAN frameworks by incorporating a memory replay mechanism to mitigate forgetting by synthesizing data for previous tasks. This is achieved through two main strategies: *Joint Retraining with Replay* and *Replay Alignment*.

In *Joint Retraining with Replay*, the replay generator produces synthetic data that replicates data from previous task, which are combined with real data from the current task to create an extended training set, enabling the generator and discriminator to optimize jointly for both old and new tasks.

In *Replay Alignment*, the current generator G_t and the replay generator G_{t-1} are synchronized to generate the same image for a given z, c. A pixelwise loss between the two generated images is used to align the outputs.

This mechanism allows the model to retain task knowledge without storing real data.

Advantages of Generative Replay

By replaying synthetic samples generated by G, MeRGANs effectively address catastrophic forgetting without requiring access to the original data. This approach reduces memory requirements and computational overhead while maintaining the ability of the model to recognize and generate data from both old and new classes.

5 Analysis and Comparison between Methods

In this section, we analyze and compare the methods presented so far, focusing on their similarities, differences, and performance results.

5.1 Class-incremental learning classifiers comparison

For the class-incremental learning classifiers (iCaRL, IL2M, and Unified Classifier), we compare iCaRL with the other two methods, published later, to highlight the improvements made in terms of classification accuracy using different datasets. Although the publications do not provide a direct comparison between IL2M and Unified Classifier, both methods distinctly offer approaches to enhance iCaRL.

There are several similarities between these methods. All of them employ a memory buffer to retain exemplars from past classes and use distillation-based techniques to preserve knowledge of old classes. Additionally, they all explicitly or implicitly address the issue of class imbalance in class-incremental learning, though in different ways. However, the primary differences between the classifiers lie in how they approach these challenges.

iCaRL relies on a nearest-mean classifier and a fixed-size memory. It does not explicitly address the imbalance between old and new classes, which can result in lower accuracy for past classes.

IL2M, introduces two types of memory: an exemplar memory and initial class statistics. This allows the model to rectify predictions for past classes, improving performance on old classes while maintaining a low memory overhead.

The Unified Classifier combines several techniques, including cosine normalization, a less-forget constraint, and inter-class separation. These strategies explicitly balance training and reduce bias toward new classes. As a result, it achieves the highest final accuracy and the lowest forgetting among the three methods.

5.2 Experimental Results: iCaRL vs IL2M

In Belouadah and Popescu [2019], the authors compare their model (IL2M) to strong baseline methods addressing the same problem, including iCaRL, presented previously in Rebuffi et al. [2017], on three distinct datasets: ILSVRC, VGGFace2, and Landmarks.

States	$Z = 10$												$K = 5000$					
Dataset	ILSVRC				VGGFace2				Landmarks				ILSVRC		VGGFace2		Landmarks	
States	K=20k	K=10k	K=5k	K=0k	K=20k	K=10k	K=5k	K=0k	K=20k	K=10k	K=5k	K=0k	Z=5	Z=20	Z=5	Z=20	Z=5	Z=20
iCaRL	35.1	33.6	32.9	20.8	66.8	65.3	64.4	26.1	68.9	66.9	65.6	27.0	32.7	29.6	74.1	49.5	73.8	52.6
IL2M	**56.4**	**50.8**	**44.1**	**18.3**	**92.0**	**89.7**	**86.5**	**20.8**	**93.4**	**90.8**	**86.9**	**21.0**	**44.9**	**42.0**	**90.1**	**85.7**	**88.5**	**85.0**
Full	0.73				0.97				97.1				0.73		0.97		97.1	

Table 1: Average accuracy (%) for iCaRL and IL2M on the datasets ILSVRC, VGGFace2, and Landmarks using varying memory sizes (K) and step size (Z). *Full* represents the non-incremental upper-bound performance obtained with all data available for all classes. Best results are in bold.

The results in Table 1 report the average accuracy for iCaRL and IL2M on the datasets, using K (memory size) values of 20k, 10k, 5k, and 0 (indicating no memory buffer to store past information), with Z (step size) fixed at 10. The results for memory size equal to 5k (the smallest value) are presented, and tests were also conducted with step sizes of 5 and 20.

The results show a significant improvement of IL2M over iCaRL across all memory sizes and datasets. IL2M maintains superior performance across different memory sizes and datasets, demonstrating its robustness.

5.3 Experimental Results: iCaRL vs Unified Classifier

In Hou et al. [2019], the authors evaluated their model using iCaRL as a baseline, since it is considered a representative method for multi-class incremental learning. The experiments were conducted on two popular datasets for multi-class incremental learning: CIFAR-100 and ImageNet.

More specifically, the authors tested both their model and iCaRL using predictions from two classification strategies: CNN predictions and nearest-mean-of-exemplars classification. This allowed for a comprehensive comparison of the two models under different prediction methods.

The results on CIFAR-100 and ImageNet show that the Unified Classifier model outperforms iCaRL, improving accuracy on CIFAR-100 by 6% and reducing error on ImageNet by 13%.

Additionally, CNN-based predictions perform better than nearest-mean-of-exemplars classification, in contrast to iCaRL, which achieved higher accuracy using NME, highlighting a better handling of the imbalance between old and new classes.

The results demonstrate that the Unified Classifier provides a more robust solution to class-incremental learning compared to iCaRL. The model proposed in Hou et al. [2019] effectively balances the training process and reduces the bias towards new classes, addressing key challenges such as catastrophic forgetting and class imbalance more efficiently.

5.4 Experimental Results: MeRGANs

The experimental results show that Memory Replay GANs effectively address catastrophic forgetting in class-incremental learning scenarios. MeRGANs outperform traditional methods by leveraging a generative replay mechanism to synthesize data from previously learned tasks.

In Wu et al. [2018] the authors conducted experiments on benchmark datasets, including CIFAR-10 and CIFAR-100, show that MeRGANs maintain significantly high accuracy on old tasks while learning new ones. The experiments highlight that MeRGANs reduce forgetting by generating high-quality samples of old classes, allowing for balanced training alongside new classes.

Generative replay helps maintain class balance. Unlike methods that rely on memory buffers of real data, MeRGANs mitigate classifier bias toward new classes by generating representative synthetic samples, which are presented along with new class data during training. Experimental results show that the quality of generated images directly affects classifier performance. On complex datasets like CIFAR-100, MeRGANs show slight performance degradation, whereas on simpler datasets like CIFAR-10, the model achieves better results over time due to the higher quality of generated data.

The experiments highlight that MeRGANs provide a scalable and memory-efficient alternative for class-incremental learning, with competitive accuracy. However, further work is needed to improve the stability and quality of generated samples to handle more challenging datasets effectively.

Authors also demonstrate that the generated images provide a visual representation of task interference and potential forgetting. By directly comparing the generated images with the original data, it is possible to understand how accurately the model retains knowledge of previously learned classes, without the need for explicit numerical comparisons.

6 Strengths and Weaknesses

The first method we analyzed, iCaRL, is an efficient approach that has also been used as a baseline for subsequent works. It employs a simple yet effective nearest-mean classifier with a fixed memory strategy. Knowledge distillation helps preserve knowledge of previously learned classes. However, iCaRL has notable limitations: it lacks explicit mechanisms to address class imbalance, relies solely on knowledge distillation for mitigating forgetting, and its flexibility is constrained by the fixed-size memory, making it less adaptable to scenarios with a large number of classes or varying amounts of data per class.

After we explore IL2M, an improvement over iCaRL due to its use of initial class statistics to correct predictions. This model introduces a dual memory structure used to store both old data and statistics used to correct predictions. Prediction correction has a low computational cost and provides a strong theoretical justification for the use of statistics, making it an optimal choice. The main weakness lies in scalability, which is limited by the compact nature of the statistics. Although this model outperforms iCaRL in experimental results, it still suffers from similar issues related to fixed memory size as the number of classes grows.

Next we analysed the Unified Classifier, the last classifier we looked at. It combines different techniques to deal with forgetting and explicitly balances the training. Again, this model is presented as an improvement over the limitations of iCaRL, offering a more effective solution for class incremental learning. Experimental results show that this model achieves the highest accuracy and lowest forgetting compared to iCaRL and other previously presented models. However it has a higher computational complexity and requires careful hyperparameter tuning to achieve optimal results.

Finally, we also analyzed a generative approach that employs generative replay to avoid storing real data, thereby reducing memory requirements. This model is able to maintain classification performance on both new and old classes by generating data for new classes while retaining the knowledge learned from previous classes. In addition, it allows the visualization of forgetting through generated images. However, the generative approach suffers from challenges related to the instability of adversarial training and the computational costs involved in generating and maintaining the models for replay. These factors can limit the practical use of this approach, especially in large-scale settings.

7 Conclusions

This report analyzed four methods for addressing the challenges of class-incremental learning: iCaRL, IL2M, Unified Classifier, and Memory Replay GANs. Each method employs unique strategies to mitigate catastrophic forgetting and class imbalance, with varying levels of success.

While the methods discussed provide significant advancements in class-incremental learning, there are possibilities for further improvements. For example, enhancing the separation between old and new classes in the feature space without adding significant computational cost could lead to better performance. Additionally, finding ways to stabilize adversarial training in generative approaches could improve the quality of generated images and enhance the overall performance of generative replay methods. These improvements could lead to more efficient and robust solutions for class-incremental learning in real-world applications.

References

Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 583–592, 2019.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 831–839, 2019.

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.

Chenshen Wu, Luis Herranz, Xialei Liu, Joost Van De Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. *Advances in neural information processing systems*, 31, 2018.