Università degli Studi di Padova

Department of Physics

Master Thesis in Physics of data

Web-based Continual Learning for Semantic Segmentation with Vision-Language Guidance

Supervisor
Prof. Jacopo Pazzini

Co-supervisor
Prof. Pietro Zanuttigh
Donald Shenaj

Master Candidate
Francesco Zane

Academic Year

2023 - 2024

A mia mamma e mio papà, Anna e Michele,
a mio fratello Raffaele
e alla mia fidanzata Chiara.
A loro voglio dedicare questa tesi, grazie per aver condiviso con me i momenti felici di questo percorso e, ancor di più, per avermi sostenuto e aiutato nei momenti più difficili.

Abstract

Deep learning has achieved great success in semantic segmentation but typically requires large, annotated datasets employed in a single-shot training. In real-world applications, this is impractical, especially when new tasks are introduced incrementally, and prior data is unavailable. Continual learning aims to address this by allowing models to learn new classes over time without forgetting previous ones, but it struggles with catastrophic forgetting. Traditional regularisation strategies often fail when multiple incremental steps are needed or when background class distributions shift. Recently, a novel approach has been explored: leveraging web-crawled data to retrieve images of old classes from online databases. The most advanced study in this direction introduced an image selection strategy that retrieves only the most informative replay images from the web. This method was applied in the particularly challenging scenario of weakly-supervised incremental learning for semantic segmentation (WILSS), where pixel-level annotations are replaced by image-level labels.

In this thesis, I extend this line of research by proposing three advanced web image selection strategies. The first method explores various techniques for extracting feature vectors from both replay images and dataset images, retaining only those replay images whose feature vectors are sufficiently similar to those previously computed and stored for the dataset images used as queries during the retrieval process. The second method introduces class prototypes, which are representations that encapsulate the average features of a class. Replay images are selected based on their similarity to at least one prototype of the classes they are intended to represent. The third method further refines the second by utilising more generalised information, reducing the dimensionality of both feature vectors and class prototypes.

Experimental results demonstrate that the proposed strategies significantly enhance the performance of previous methods, achieving state-of-the-art results. Notably, in the most challenging and realistic setting, the best method outperforms prior work by 3.02%, showcasing the effectiveness of these novel image selection techniques for continual learning in semantic segmentation.

Contents

Abstract v

List of figures ix

List of tables xi

1 Introduction 1

2 Semantic Segmentation: A Deep Learning Approach 5
- 2.0.1 Encoder-Decoder Architectures 6
- 2.0.2 ResNet: The Impact of Residual Blocks on Deep Segmentation Models 7
- 2.0.3 DeepLab model and its Innovations 8

3 Continual Learning 11
- 3.1 Problem Formulation 12
- 3.2 Continual learning setup in continual segmentation 13
- 3.3 Incremental Learning techniques 15
  - 3.3.1 Knowledge distillation 15
  - 3.3.2 Parameter freezing 16
  - 3.3.3 Geometrical feature-level regularisation 16
  - 3.3.4 Weights Initialisation 17
  - 3.3.5 Generative replay 17
  - 3.3.6 Webly-based Learning 18

4 Web-Driven Replay Solutions for Continual Semantic Segmentation 19
- 4.1 RECALL: Replay-Based Continual Learning in Semantic Segmentation 20
  - 4.1.1 General architecture 20
  - 4.1.2 Algorithm 23
- 4.2 RECALL+: Adversarial Web-based Replay for Continual Learning in Semantic Segmentation 24
  - 4.2.1 Image selection strategies 24
  - 4.2.2 Self-teaching strategies 26
- 4.3 Learning from the Web: Language Drives Weakly-Supervised Incremental Learning for Semantic Segmentation 27

4.3.1 Labelling for weakly-labelled images 28
4.3.2 Rehearsal strategies 29

5 Image Selection Techniques for Replay Images in Semantic Segmentation 33
5.1 CLIP-Based Strategy for Image-to-Image Similarity Assessment 35
5.1.1 Similarity by comparing text-based descriptors 35
5.1.2 Comparison of text descriptors of dataset images and web images descriptors 38
5.1.3 Comparison of dataset images descriptors and descriptors of the web images 40
5.1.4 Similarity by comparing image-based descriptors 42
5.2 Image Selection Strategy Based on CLIP Class Prototypes 45
5.2.1 Image-based image selection method with CLIP 46
5.2.2 Text-based image selection method with CLIP 49
5.3 Image Selection Strategy Based on Reduced CLIP Class Propotypes using PCA 52
5.3.1 Image-based image selection method with CLIP and PCA 52
5.3.2 Text-based image selection method with CLIP and PCA 55

6 Dataset 59

7 Results 61
7.1 Experimental setting 61
7.1.1 Web dataset 61
7.1.2 Implementation detail 62
7.2 Setting 62
7.3 Results on CLIP-Based Strategy for Image-to-Image Similarity Assessment . . 65
7.4 Results on Image Selection Strategy Based on CLIP Class Prototypes 68
7.5 Results on Image Selection Strategy Based on Reduced CLIP Class Prototypes using PCA 71
7.6 Final Performance Comparison of Image Selection Strategies 73

8 Conclusion 75

References 77

Listing of figures

2.1 Image classification, object detection and image segmentation. 6
2.2 Example of encoder-decoder CNN architecture for image segmentation, from [1]. 7
2.3 Residual block scheme, from [2] 8
2.4 (Top) Standard convolution followed by upsampling, (Bottom) Atrous convolution producing a denser feature map. Image took from [3] 9
2.5 Scheme of the Atrous Spatial Pyramid Pooling, from [3] 9

3.1 From [4], a graphical representation of the class-incremental continual learning framework. The model is progressively updated to identify new classes while preserving its knowledge of previously learned ones. 12
3.2 Overview of the various setups for class-incremental continual learning in semantic segmentation, from [4]. 14

4.1 Background self-inpainting, from [5] 22
4.2 RECALL architecture, from [5] 23
4.3 RECALL+ architecture, from [6] 24
4.4 RECALL+ Self-teaching strategies, from [6] 26
4.5 General architecture, from [7] 28
4.6 Illustration of caption-based image selection method, from [7] 31

5.1 Graphical representation of the batch training step in the CLIP model, from [8]. 34
5.2 Graphic representation of the image selection strategy applies on a replay image in the Text-to-Text method. 36
5.3 Graphic representation of the image selection strategy applies on a replay image in the Text-to-Image method. 40
5.4 Graphic representation of the image selection strategy applies on a replay image in the Image-to-Text method. 42
5.5 Graphic representation of the image selection strategy applies on a replay image in the Image-to-Image method. 44
5.6 Graphical representation of the firsts stage of step t 47
5.7 Graphical representation of the second stage of step t 47
5.8 Graphical representation of the firsts stage of step t 49
5.9 Graphical representation of the second stage of step t 50

5.10 Graphical representation of the firsts stage of step t 53
5.11 Graphical representation of the second stage of step t 53
5.12 Graphical representation of the firsts stage of step t 56
5.13 Graphical representation of the second stage of step t 56

6.1 Aeroplane 59
6.2 Bicycle 59
6.3 Bird 59
6.4 Boat 59
6.5 Bottle 59
6.6 Bus 60
6.7 Car 60
6.8 Cat 60
6.9 Chair 60
6.10 Cow 60
6.11 Table 60
6.12 Dog 60
6.13 Horse 60
6.14 Motorbike 60
6.15 Person 60
6.16 Plant 60
6.17 Sheep 60
6.18 Sofa 60
6.19 Train 60
6.20 Monitor 60

7.1 Graphical representation of *IoU* metric, from [9]. 64
7.2 Distribution of cosine similarity between the feature vectors of replay images and those of the dataset images across the four methods in the first image selection strategy. The red dashed lines represent the values of the threshold shown in Table 7.2 65
7.3 Behaviour of the *mIoU* for classes across the steps in the 10-10s-ov setting using the Image-Image_0.6 method. 67
7.4 Distribution of cosine similarity for images related to some class in the image-driven method (top) and in the text-driven method (bottom). 68
7.5 Behaviour of the *mIoU* for classes across the steps in the 10-10s-ov setting using the Image_0 method. 70
7.6 Behaviour of the *mIoU* for classes across the steps in the 10-10s-ov setting using the PCA_Text_-1 method. 72

Listing of tables

7.1 *mIoU* performances of the *baseline* model, the *upper bound*, and the *LfW* image selection strategy across the three experimental settings. 64
7.2 Results of the three settings using the CLIP-Based Strategy for Image-to-Image Similarity Assessment. This table compares the image selection strategy against the baseline, the upper bound, and the method from the previous paper. The Δ values represent the differences between the obtained results and those of LfW. 66
7.3 Results of the three settings using the Image Selection Strategy Based on CLIP Class Prototypes. This table compares the image selection strategy against the baseline, the upper bound, and the method from the previous paper. The Δ values represent the differences between the obtained results and those of LfW. 69
7.4 Results of the three settings using the Image Selection Strategy Based on Reduced CLIP Class Prototypes using PCA. This table compares the image selection strategy against the baseline, the upper bound, and the method from the previous paper. The Δ values represent the differences between the obtained results and those of LfW. 71
7.5 Results of the best method for each image selection strategy. This table compares the image selection strategy against the baseline, the upper bound, and the method from the previous paper. The Δ values represent the differences between the obtained results and those of LfW. 73

1 Introduction

In recent years, the field of computer vision has experienced remarkable advancements, with increasingly complex tasks being addressed through progressively sophisticated techniques. A major turning point occurred with the rise of neural networks, which enabled performance levels previously considered unattainable with traditional methods. This revolution began with breakthroughs in image classification, where a model assigns a single label to an entire image. It quickly evolved to tackle more intricate challenges, such as object detection, where models not only identify objects within an image but also pinpoint their locations using bounding boxes. Building on this progress, the next step involved tackling even more complex tasks like semantic segmentation, where the goal is to assign a label to each individual pixel, offering a finer level of granularity and a deeper understanding of the image's content.

To address these tasks, a variety of increasingly complex strategies have emerged over the years. In the early stages, convolutional layers [10] were the fundamental building blocks for solving image classification problems, representing a significant leap in the field. The widespread availability of pretrained convolutional neural network (CNN)-based architectures further accelerated progress, particularly in semantic segmentation. This led to the development of Fully Convolutional Networks (FCNs) [11], which utilised pre-trained classification models as backbones, marking a significant breakthrough in pixel-level prediction tasks. However, additional innovations were needed to push the boundaries of semantic segmentation. One such advancement came with the introduction of convolutional-based encoder-decoder architectures.

These architectures improved the ability to capture detailed information by first encoding the image into feature maps and then decoding them to generate precise pixel-wise predictions. While increasing the depth of these networks promised better performance, they encountered the vanishing gradient problem: as networks grew deeper, their ability to effectively propagate gradients across layers diminished, leading to performance degradation. The introduction of residual blocks [2] provided a crucial solution to this challenge. By allowing information to bypass certain layers through shortcut connections, Residual Networks (ResNets) helped mitigate the vanishing gradient problem, enabling deeper architectures to maintain high performance and achieve greater success in semantic segmentation tasks. DeepLab-v3 [12] is an excellent example of this advancement, utilising ResNet [13] as its backbone along with atrous convolutions to preserve dense feature maps. This architecture set a new standard in semantic segmentation and serves as the foundation for the model discussed in this thesis.

In the field of computer vision, it has often been assumed that all data samples and tasks are available during training, allowing models to be trained on the entire dataset at once. However, this assumption is often unrealistic in real-world scenarios. A significant challenge in practical applications is the necessity for models to continuously learn and adapt to new tasks while retaining previously acquired knowledge, all without access to prior data. This has led to the emergence of the field of continual learning, which aims to develop strategies that enable models to incrementally learn new tasks without forgetting what they have previously learned. A common challenge in continual learning arises when models must expand the range of classes they recognise, either by incorporating entirely new classes or by refining existing ones into more specific subclasses. Such task expansion can have a profound impact on the model's performance, particularly when training data becomes available incrementally. A model's performance on previously learned tasks can rapidly decline due to a phenomenon known as catastrophic forgetting. This occurs when the network's parameters are optimised for new tasks, often at the expense of older ones, unless specific mechanisms are implemented to counteract this issue. Only recently have continual learning strategies been applied to the domain of semantic segmentation, a task that requires a model to predict a label for every pixel in an image. The complexity of semantic segmentation, combined with the need for continuous learning without forgetting, presents unique challenges that demand innovative solutions.

To address catastrophic forgetting, researchers have explored a variety of techniques. Knowledge distillation is one such approach, where the new model is trained to replicate the behaviour of its predecessor, thus transferring knowledge from the previous model. Other solutions, such as parameter freezing or clever parameter initialisation, aim to preserve the most relevant parameters for previously learned tasks or ensure that new parameters are initialised in a way that respects prior learning. There are also approaches that operate in the feature space, seeking to maintain a well-organised and efficient representation of different classes. More recently, two promising strategies have emerged that aim to recreate previous data in order to mitigate forgetting: generative models and web-based replay. Generative models attempt to recreate images from previous tasks using learned representations, while web-based replay leverages the web by using information about previous tasks to retrieve relevant images online. These replayed images can then be used to retain knowledge of earlier tasks without needing access to the dataset. The most advanced study, conducted by Zanuttigh et al.[7], employing the web replay technique tackles an even more complex challenge: weakly-supervised incremental learning for semantic segmentation (WILSS), where image-level labels replace traditional pixel-level annotations. In this work, the authors introduced an image selection mechanism to ensure that only the most relevant images are retained during the replay process, enhancing the efficiency and effectiveness of the technique.

The goal of this thesis is to build upon the work presented in the paper [7] to develop a more efficient image selection technique. In this research, I introduce three novel image selection methods. The first method investigates various strategies for extracting feature vectors from both replay images and dataset images. It retains replay images only if their feature vectors are sufficiently similar to those previously computed and stored for the dataset images used as queries during the retrieval process. The second technique exploits the concept of class prototypes—representations that encapsulate the average characteristics of a class. Replay images are preserved if they exhibit sufficient similarity to at least one prototype of the classes they are intended to represent. Finally, the third method builds on the second by utilising more generalised information, thereby reducing the dimensionality of both feature vectors and prototypes.

The thesis is organised as follows: Chapter 2 describes the semantic segmentation task and how it is tackled within the deep learning framework. Chapter 3 introduces the framework of continual learning, detailing its formalism and principal strategies employed in the field. Chapter 4 examines three papers that have utilised web strategies in semantic segmentation for continual learning. Chapter 5 elaborates on the three image selection strategies developed in this thesis. Chapter 6 presents the dataset used to evaluate performance, while Chapter 7 discusses the implementation aspects and reports results across multiple settings. Finally, Chapter 8 concludes with key considerations and insights derived from the thesis.

2 Semantic Segmentation: A Deep Learning Approach

With the introduction of new algorithms and models, particularly the advent of neural networks and deep learning, the field of computer vision has made significant strides in tackling increasingly complex problems. Notably, we can identify three primary tasks in the field of computer vision, each representing a progressively more difficult challenge and building upon the previous one:

- **Image classification**
  The task is to assign the image to one class from a set of possible classes;
- **Object Detection**
  In addition to identification, the model must also approximately localise one or more objects within the image, typically using bounding boxes;
- **Image Segmentation**
  This is the most challenging task, requiring the model to identify and localise objects at the pixel level without differentiating between individual objects of the same class.

In the computer vision task of semantic segmentation, accurately identifying each object within an image means that the machine learning model assigns a label to every pixel. This task is not only one of the most challenging but also one of the most crucial, as it brings us closer to achieve the paradigm of complete understanding of an image in its entirety. Its applications span a wide range of real-world scenarios, from industrial uses to autonomous driving.

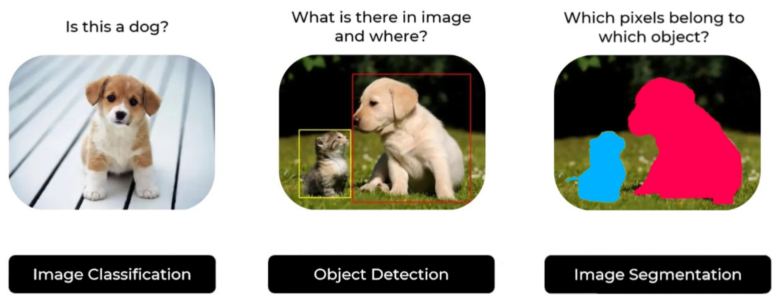

Is this a dog?
What is there in image and where?
Which pixels belong to which object?
Image Classification
Object Detection
Image Segmentation

Figure 2.1: Image classification, object detection and image segmentation.

One of the first significant breakthrough in semantic segmentation was achieved by [11], who introduced the Fully Convolutional Network (FCN) architecture. This innovative approach demonstrated that a network composed solely of convolutional layers (CNN) [10], trained end-to-end for semantic segmentation, could outperform existing state-of-the-art methods without the need for additional components. A key feature of their work was the introduction of skip connections, which integrate high-level semantic information with low-level spatial details.
Additionally, the authors presented a method for adapting well-known image classification architectures for use in semantic segmentation, further advancing the field.

2.0.1 Encoder-Decoder Architectures

As previously mentioned, the predominant deep learning architecture used for semantic segmentation is the encoder-decoder model. In this setup, the encoder's role is to extract meaningful features from the image. This is typically accomplished through a series of convolutional layers interspersed with max-pooling blocks. The convolutional layers identify patterns, while the max-pooling layers progressively enlarge the receptive field. Conversely, the decoder's task is to map the features from the compressed representation back to the original image dimensions, effectively reconstructing the spatial information that was reduced during the encoding process.

In semantic segmentation architectures, it is common to use a pre-trained encoder from a model originally designed for image classification. The decoder, on the other hand, is typically customised for the specific segmentation task, often incorporating novel approaches to better suit the needs of the application.

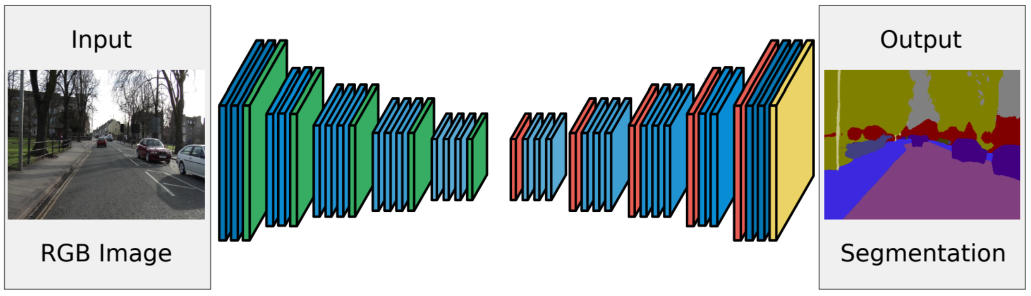


Figure 2.2: Example of encoder-decoder CNN architecture for image segmentation, from [1].

2.0.2 ResNet: The Impact of Residual Blocks on Deep Segmentation Models

Over the years, to achieve increasingly higher performance with more detailed images and a growing number of classes, researchers have developed progressively deeper encoder-decoder architectures. However, they observed that due to the *vanishing gradient problem*, increasing the number of layers beyond a certain point no longer improved performance. In fact, beyond a certain depth, adding more layers began to degrade the model's performance.
A breakthrough in addressing this issue was the invention of the *residual block* [2], which allows information to bypass layers through a shortcut connection. This parallel path simplifies the learning task, thus reducing the vanishing gradient problem. This block was crucial in allowing architectures to become increasingly deeper while simultaneously enhancing their performance.

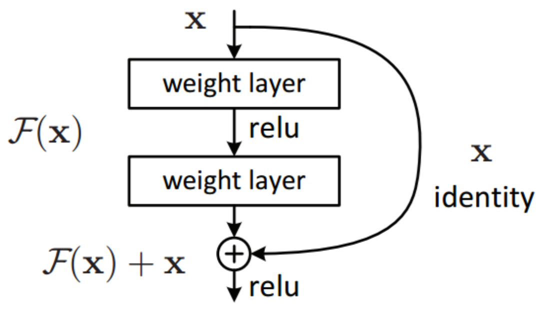

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.3: Residual block scheme, from [2]

2.0.3 DeepLab model and its Innovations

Although encoder-decoder architectures showed strong performance, they were based on architectures originally designed for image classification tasks. As a result, they inherited characteristics optimised for classification, such as invariance to local image transformations. While this enhances abstraction, it is less suitable for tasks like image segmentation. To address these limitations, DeepLab architectures were introduced, with a focus on improving the following key aspects of CNN-based encoder-decoder models:

- reduced feature resolution
- existence of object at multiple scales
- reduced localisation accuracy due to deep convolutional neural network (DCNN) invariance.

Classical encoder-decoder CNN architectures use repeated combinations of max-pooling and striding across consecutive layers. This increases the receptive field, enabling the model to capture larger-scale patterns. However, this approach also significantly reduces the spatial resolution of the resulting feature maps. To enlarge the receptive field of the layers without relying on max-pooling, *Atrous Convolution* layers were introduced [3]. These layers apply filters with inserted holes between the non-zero filter taps. This approach not only addresses the issue of reduced feature resolution but also achieves this improvement without increasing the number of parameters.

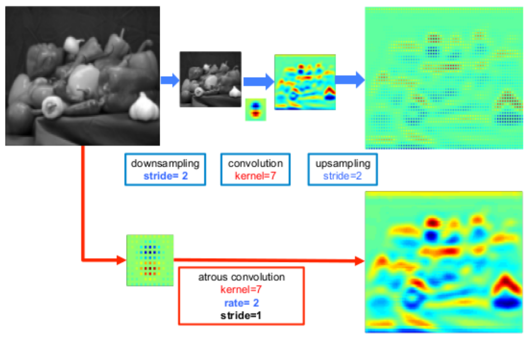


Figure 2.4: (Top) Standard convolution followed by upsampling, (Bottom) Atrous convolution producing a denser feature map. Image took from [3]

As previously mentioned, semantic segmentation models must also handle objects appearing at different scales. To address this, *Atrous Spatial Pyramid Pooling (ASPP)* was introduced [3]. The key idea is to apply multiple Atrous convolutions with different dilation rates and then combine the resulting outputs, allowing the model to capture features at various scales effectively.

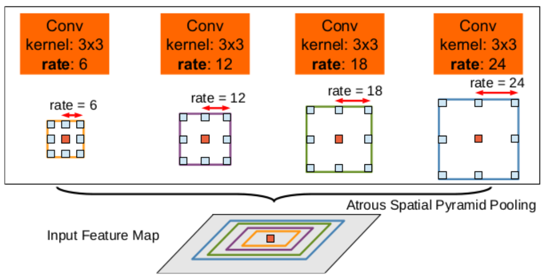


Figure 2.5: Scheme of the Atrous Spatial Pyramid Pooling, from [3]

3 Continual Learning

In the previous chapter, we explored how deep learning techniques in computer vision have rapidly advanced, enabling models to tackle a wide range of tasks and often achieve performance comparable to humans. Up to this point, we have discussed these techniques in the fully supervised setting where all data samples and tasks are assumed to be available during the training phase, allowing the model to train on the entire dataset at once.
However, as described in [4], this assumption is not always practical in real-world scenarios, where various challenges may arise. A key challenge is the need for models to continuously learn and adapt to new tasks without forgetting previously acquired knowledge. Continual learning strategies address this issue by enabling models to progressively acquire new tasks while retaining their past knowledge, all without the need to retrain from scratch.
A common challenge in continual learning arises from the need to increase the number of classes, either by adding new ones or by subdividing existing ones into more specific subclasses. Such changes in the training tasks can significantly influence the model's performance. When a model is trained on sequential tasks, where new data samples are introduced over time and previous data cannot be accessed, the model's performance on previously learned tasks can degrade rapidly. This occurs because the network's parameters are optimised for the new tasks without considering the older ones, unless specific measures are taken to prevent it. This phenomenon is known as *catastrophic forgetting*.
It is only recently that continual learning strategies have been studied in the context of semantic segmentation tasks.

3.1 Problem Formulation

Continual Learning (CL) can be seen as a specific form of transfer learning, where the data distribution changes at each step, and the model must perform well across all distributions. In practice, when discussing continual learning tasks, people often refer to the simplest setting, where tasks are introduced one at a time, and training is performed on the available data.
For example, in class-incremental learning, the model is updated to recognise new classes while preserving its knowledge of previously learned ones.

Formally, we consider the incremental step t, where $t = 1, 2, 3, ..., T_{MAX}$. At each step, we have access to the model from the previous step, M_{t-1}, as well as two sets of data $(\mathcal{X}^{(t)}, \mathcal{Y}^{(t)})$, randomly sampled from the distribution $\mathcal{D}^{(t)}$, which is a subset of the full domain $\mathcal{D}$. Here, $\mathcal{X}^{(t)}$ represents the data samples for step t, while $\mathcal{Y}^{(t)}$ contains the corresponding ground truth labels.
In the class-incremental setup we are exploring, each step introduces a new learning task by adding new classes to those from previous steps.

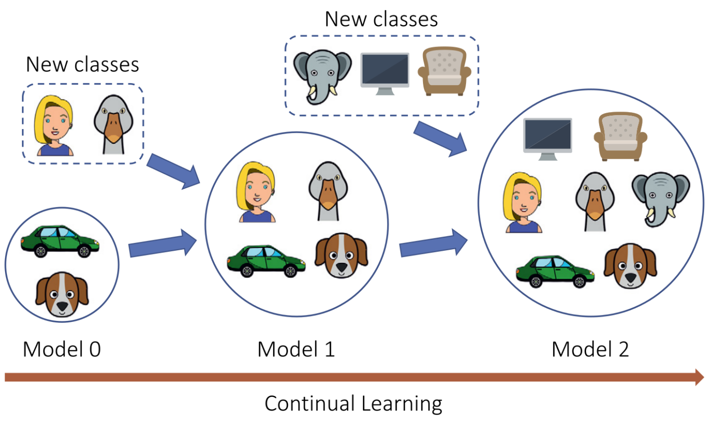


Figure 3.1: From [4], a graphical representation of the class-incremental continual learning framework. The model is progressively updated to identify new classes while preserving its knowledge of previously learned ones.

An additional challenge in many real-world scenarios is that, due to storage or privacy constraints, it is often not possible to retain any sample data $\{\mathcal{X}^{(s)}, \mathcal{Y}^{(s)}\}$ from any previously learned step s. This makes the continual learning setting even more difficult, as the objective becomes to optimise a function that encompasses all previously learned tasks without access to the original samples.

At step t, the empirical risk minimisation framework seeks to find the optimal parameters across all classes by optimising the following expression:

$$\arg\min_{\theta} \sum_{\tilde{t}=0}^{T} \mathbb{E}_{(\mathcal{X}^{(\tilde{t})},\mathcal{Y}^{(\tilde{t})})} \left[\mathcal{L}\left(M_t\left(\mathcal{X}^{(\tilde{t})}; \theta \right), \mathcal{Y}^{(\tilde{t})} \right) \right] \tag{3.1}$$

with model's parameters θ, loss function $\mathcal{L}$ and M_t the model at step t. Since samples related to previous classes may not be available, this objective function cannot be directly optimised using standard techniques.

We refer to *joint training* as the scenario where all samples are available from the start. This setup represents an upper bound on the performance of a continual learning system.

3.2 Continual learning setup in continual segmentation

Although the application of continual learning to semantic segmentation is a relatively recent area of study, several settings have already emerged. In particular, the differences between these approaches, which are related to various target applications, lie in how $\mathcal{D}^{(t)}$ is defined and how the data $(\mathcal{X}^{(t)}, \mathcal{Y}^{(t)})$ are sampled from it.

Let us denote with $\mathcal{S}^{(t-1)}$ the previous label set, which is expanded with a set of new classes $\mathcal{C}^{(t)}$ at step t, yielding a new label set $\mathcal{S}^{(t)} = \mathcal{S}^{(t-1)} \cup \mathcal{C}^{(t)}$.

In task-incremental settings, it is typically assumed that the sets of new labels discovered at each step are disjoint, with the exception of a special *background* or *void* class, whose behaviour and meaning depend on the specific scenario. There are various scenarios, and one key difference among them is how the background class is handled, which is a common aspect in many semantic segmentation benchmarks. Previous approaches have identified four main scenarios:

- **Sequential masked** This setup represents the most straightforward approach to continual semantic segmentation: each learning step involves a unique set of images where pixels are classified either as belonging to new classes or as part of the void class. The void class is not predicted by the model and is masked out from both the results and the training process.
- **Sequential** This setup has been proposed in Michieli and Zanuttigh ([14], [15]). Each learning step involves a unique set of images, where pixels are classified into classes seen either in the current step or in previous ones. At each step, labels for both new and old

classes are present; however, the presence of a specific old class is closely related to the set of new classes being introduced.

- **Disjoint** This setup has been proposed in Cermelli et al. ([16]) and in Michieli and Zanuttigh ([15]). At each learning step, the unique set of images is identical to the sequential setup. The key difference from the sequential setup lies in the set of labels. In this approach, each step includes only labels for pixels of new classes, while previously learned classes are labeled as background in the segmentation maps. Consequently, the background class distribution changes with each step.

- **Overlapped** This setup moves from the work of Shmelkov et al. ([17]) for object detection and has been addressed in Cermelli et al. ([16]); Michieli and Zanuttigh ([18]) for semantic segmentation. In this setup, each training step includes all images containing at least one pixel from a new set of classes, with only these new classes annotated while all other classes are labeled as background. Unlike other settings, images in this scenario may include pixels from classes that will be introduced in future steps but are labeled as background in the current step. As a result, similar to the previous setting, the distribution of the background class changes with each incremental step.

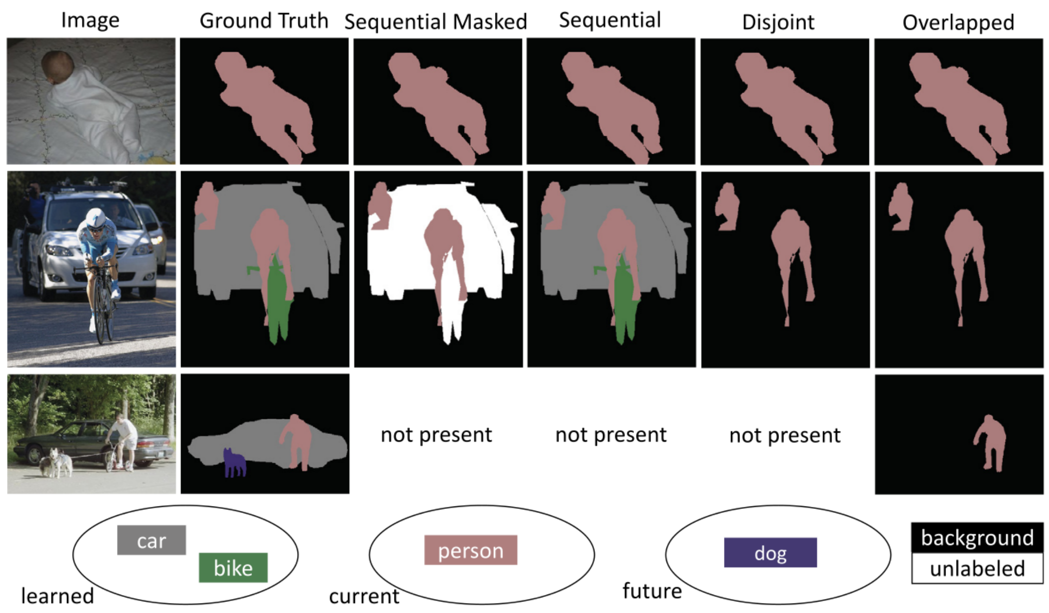


Figure 3.2: Overview of the various setups for class-incremental continual learning in semantic segmentation, from [4].

3.3 Incremental Learning techniques

In this section we are going to describe the main techniques used to handle the semantic segmentation task in the continual learning framework.

3.3.1 Knowledge distillation

Due to its simplicity and effectiveness, knowledge distillation is one of the most widely used techniques in continual learning, this technique was originally proposed by Bucilua et al. ([19]) and Hinton et al. ([20]). To preserve knowledge from previous tasks, this approach aims to maintain the model's responses to past data while updating it with new samples from the current task. This is typically achieved by adding a strong regularisation term during the learning process for the current classes, which helps the model mimic its responses to previous tasks. This approach often leads to improved performance on both the previous and current classes. Ozdemir and Goksel ([21]) successfully extended knowledge distillation techniques to dense tasks by formulating a knowledge distillation loss as the cross-entropy between the output probabilities of the previous and current models. One of the earliest attempts at knowledge distillation in dense tasks was conducted by Michieli and Zanuttigh (2019), who expanded the approach by applying knowledge distillation not only at the output level but also within the intermediate feature space, thereby preserving the geometric relationships of the extracted features.
A more recent study by Phan et al. [22] explores in greater detail how to transfer knowledge from old to new classes in incremental learning scenarios. The authors observed that previous works treat all old classes equally during knowledge distillation, which often leads to the forgetting of older classes, particularly those that are visually similar to the new ones being introduced. To address this issue, they proposed a novel method called Class Similarity Weighted Knowledge Distillation (CSW-KD). The main idea behind this approach is to revise the knowledge of old classes that are likely to be forgotten, specifically those that are similar to a new class. When a new class is introduced, the method computes its similarity to the existing old classes. Based on these similarity scores, the predictions of the previous model on old classes are reweighed, with more emphasis placed on those that are similar to the new class. This similarity-weighted knowledge is then distilled into the current model.

3.3.2 Parameter freezing

Another approach to address catastrophic forgetting is parameter freezing, where a portion of the network's weights are frozen. First introduced by Rebuffi et al. ([17]), this technique is a key strategy in continual learning and has been widely adopted in contemporary methods as a regularisation technique to prevent knowledge degradation.
Parameter freezing has also been explored as a potential solution to prevent forgetting in dense labelling tasks. The specific method of applying this technique is not fixed, and various approaches have been attempted over the years. In [14], Michieli and Zanuttigh proposed freezing all the layers of the encoder to preserve the feature extraction capabilities while training only the decoder parameters. Two years later, they Michieli and Zanuttigh ([15]) refined this approach by freezing only the first few layers of the encoder, focusing on preserving the most task-agnostic parts of the feature extractor.
Although the technique has proven effective across various tasks, the question of which layers to freeze remains unresolved. There is an inherent trade-off between the model's ability to efficiently learn new tasks and the preservation of previously acquired knowledge.
A first attempt of automatic selection of which layers to freeze has been recently introduced by Nguyen et al. ([23]) checking the most plastic layers of the network.

3.3.3 Geometrical feature-level regularisation

This alternative approach to addressing catastrophic forgetting focuses on the organisation of the latent space. The idea is to find a method for disentangling features within the latent space to better separate features related to different classes. In the continual learning setting, this approach can help the model reduce the overlap between features associated with new and old classes.
One of the first work that applied this idea in dense tasks was Michieli and Zanuttigh ([18]), where the latent space was constrained to reduce forgetting whilst improving the recognition of novel classes.

3.3.4 Weights Initialisation

Cermelli et al. ([16]) were the first to use this approach to address the atypical behaviour of the background class in both disjoint and overlapped scenarios. To prevent the model from being biased toward the background class when encountering unseen classes, they initialised the classifier's parameters for the new classes in a way that distributes the probability of the background uniformly among the novel classes.
Building on this work, Goswami et al. [24] introduced a more advanced method for addressing background shift. Their approach employs a novel classifier initialisation technique using gradient-based attribution to identify the most relevant weights from the previous classifier's background weights. These selected weights are then transferred to the new classifier, which helps accelerate the learning of new classes while reducing catastrophic forgetting.

3.3.5 Generative replay

One alternative solution is to use generative models to recreate past samples. In this approach, generative models are first trained on the current data distribution. Later, these models can generate data from past experiences when learning new information.
This approach was initially applied to image classification tasks. In their deep generative replay framework, introduced by Shin et al. [25], the model preserves previously learned knowledge by replaying generated pseudo-data alongside current tasks. Specifically, a deep generative model, based on the generative adversarial networks (GANs) framework, is trained to imitate past data. These generated data are then paired with the corresponding outputs from the past task solver, effectively representing older tasks and enabling the model to retain prior knowledge.
More recently, this approach has also been explored for solving dense prediction tasks. In particular, in [5], they used a pretrained GAN model to generate samples of old classes based on their class labels, with the goal of mitigating the problem of catastrophic forgetting.
However, it's important to note that only weak classification labels are available for the generated data, and some pseudo-labels need to be estimated for segmentation tasks.

3.3.6 Webly-based Learning

This approach leverages web search samples, offers a powerful method for retrieving accurate past examples.
The core idea of this approach is to leverage the web to retrieve images related to past classes that can serve as alternatives to samples that are no longer available. This method was introduced by [5], who initially tried to retrieve images using the names of old classes as queries.
However, given the vast number of images on the web, even though the class name is used as a query, not all the retrieved images are optimal for training a semantic segmentation model. A major challenge is controlling the quality of the retrieved images. Additionally, like other methods, this approach provides only weak classification labels, necessitating a pseudo-labelling scheme.
In more recent work, [6] and [7], the same team developed new strategies to both select web images to retain only the most useful ones and to create pseudo-labels for the retrieved images. We will see more in detail these methods in the next chapter.

4

Web-Driven Replay Solutions for Continual Semantic Segmentation

Convention used by the papers

In this chapter, we will explore in detail the research that utilises web-based rehearsal strategies to mitigate the phenomenon of catastrophic forgetting.

The primary objective of these studies is to develop a semantic segmentation model within the continual learning framework that can effectively differentiate among $\mathcal{C}$ classes introduced across multiple steps. Here, $\mathcal{C}_0$ denotes the set of classes present at step 0, while $\mathcal{C}_k$ represents the set of classes added at step k. A special class, denoted as b, represents the *background*, which is included at every step. In the standard supervised deep learning setting, the model is trained in a single phase on a dataset $\mathcal{T} \subset \mathcal{X} \times \mathcal{Y}$, where all classes and samples are accessible. In contrast, the class-incremental setting involves training over a sequence of steps $k = 0, 1, \cdots, K$. At each step k, only the training dataset $\mathcal{T}_k \subset \mathcal{X}_k \times \mathcal{Y}_k$ is available, where the pixel annotations are restricted to the classes in the subset $(\mathcal{C}_k \cup b) \subset \mathcal{C}$. Pixels not belonging to the categories in $\mathcal{C}_k$ are labelled as background. For any given step k, an image in the training set $\mathcal{T}_k$ is denoted $\mathbf{X} \in \mathcal{X}_k \subset \mathbb{R}^{H*W*3}$, while its corresponding dense label is represented as $\mathbf{Y} \in \mathcal{Y}_k \subset \mathbb{R}^{H*W*(|\mathcal{C}_k \cup b|)}$. At this stage, the model M is expected to make predictions across all classes that have been added up to that point, represented as $\mathcal{C}_{0 \to k}$.

4.1 RECALL: Replay-Based Continual Learning in Semantic Segmentation

In the context of continual learning for semantic segmentation, a novel method called RECALL [5] (REplay in ContinuAL Learning) has been introduced; it generates representations of previously learned classes using two primary strategies. The first utilises a pre-trained Generative Adversarial Network (GAN) that is conditioned to generate samples of specific classes. The second strategy, which we will focus on, involves sourcing images from the web using old class names as search queries. Both methods effectively provide a substantial amount of weakly labelled data.
Additionally, the work tackles the background shift problem by employing a self-inpainting strategy that reassigns background regions using predictions from previous models.

4.1.1 General architecture

The model M used to perform on semantic segmentation task is an encoder-decoder architecture. In the incremental environment they examined, it was assumed that during the incremental step k, only samples related to the new classes $\mathcal{C}_k$ are available.
At the first step only the samples related to $\mathcal{C}_0$ are available ($b \in \mathcal{C}_0$) and the model at that step is denoted as:

$$M_0 = D_0 \circ E_0 \tag{4.1}$$

$$M_0 : \mathcal{X} \to \mathbb{R}^{H \cdot W \cdot |\mathcal{C}_0|} \tag{4.2}$$

At the generic step k, the model must be able to segment the classes $\mathcal{C}_{0 \to k}$, which are given by the union of the new set of classes $\mathcal{C}_k$ and the classes learned up to that point $\mathcal{C}_{0 \to (k-1)}$. The model at the k-th step is:

$$M_k = D_k \circ E_0 \tag{4.3}$$

$$M_k : \mathcal{X} \to \mathbb{R}^{H \cdot W \cdot |\mathcal{C}_{0 \to k}|} \tag{4.4}$$

Replay Block

In this work, they introduced a Replay Block with two primary goals. First, it aims to retrieve images related to classes from previous steps, either by generating them from scratch or by sourcing them from alternative databases (e.g. web databases). Second, since these images are weakly labelled, the Replay Block is designed to obtain reliable semantic labels for them.
The Replay Block's image retrieval task is executed by what they called Source Block:

$$S = \mathcal{C}_k \to \mathcal{X}^{rp}_{\mathcal{C}_k} \tag{4.5}$$

This module can take a set of class names as input and provide images whose semantic content corresponds to those classes.
In particular, in the specific case where they retrieved images from the web, they used the available Flickr website. Assuming we are at the incremental step t and have access to the names of all previous classes, they downloaded images whose tags and descriptions contained the class names through Flickr's web crawler.

Since the retrieved images are weakly labelled, an additional module was needed to produce dense predictions corresponding to the classes identified in previous steps. This module is referred to as $\{L_{\mathcal{C}_k}\}_{\mathcal{C}_k \subset \mathcal{C}}$. For each specific set of semantic categories $\mathcal{C}_k$, a distinct Label Evaluation Block is used. These blocks consist of two components: a fixed part, denoted as E_0, which is common to all blocks, and a variable part, $D^H_{\mathcal{C}_k}$, which is adapted to the specific set of classes $\mathcal{C}_k$.

$$L_{\mathcal{C}_k} = D^H_{\mathcal{C}_k} \circ E_0 \tag{4.6}$$

$$L_{\mathcal{C}_k} : \mathcal{X}_{\mathcal{C}_k} \to \mathbb{R}^{H \cdot W \cdot (|\mathcal{C}_k \cup b|)} \tag{4.7}$$

Given the Source S and the Label Evaluation Block $L_{\mathcal{C}_k}$, a replay image dataset associated with a specific set of past classes $\mathcal{C}_k$ can be retrieved. Using $\mathcal{C}_k$ as a query, a replay image $\mathbf{X}^{rp}_{\mathcal{C}_k} = S(\mathcal{C}_k)$ is generated and then passed through the Label Evaluation Block $L_{\mathcal{C}_k}$ to produce a dense label. This is computed as:

$$\mathbf{Y}^{rp}_{\mathcal{C}_k} = argmax_{c \in \mathcal{C}_k \cup \{b\}} L_{\mathcal{C}_k}(\mathbf{X}^{rp}_{\mathcal{C}_k})\,[c] \tag{4.8}$$

By collecting multiple replay images, it is possible to construct a replay dataset corresponding to the specific set of classes $\mathcal{C}_k$:

$$\mathcal{R}_{\mathcal{C}_k} = \{(\mathbf{X}^{rp}_{\mathcal{C}_k}, \mathbf{Y}^{rp}_{\mathcal{C}_k})\}_{n=1}^{N_r} \tag{4.9}$$

where N_r is a hyperparameter that defines the number of replay images.

Background self-inpainting

In continual learning settings, a common challenge is the phenomenon known as background shift. This occurs because, at each learning step, the distribution of background pixels changes. Specifically, as shown in figure 3.2, in both the disjoint and overlapping settings, only the pixels corresponding to the current classes in $\mathcal{C}_k$ are visible, while all pixels corresponding to previously learned classes are relabelled as background.
To effectively address this problem, an inpainting mechanism was developed to transfer knowledge from the previous model to the current one. At each step k, with the training set $\mathcal{T}_k$, the background region in each ground truth map is filled with the predictions from the previous model M_{k-1}, resulting in $\mathcal{T}_k^{bi} = M_{(k-1)}(\mathcal{T}_k)$.
This allows the model to retain information from earlier tasks, as illustrated in figure 4.1.

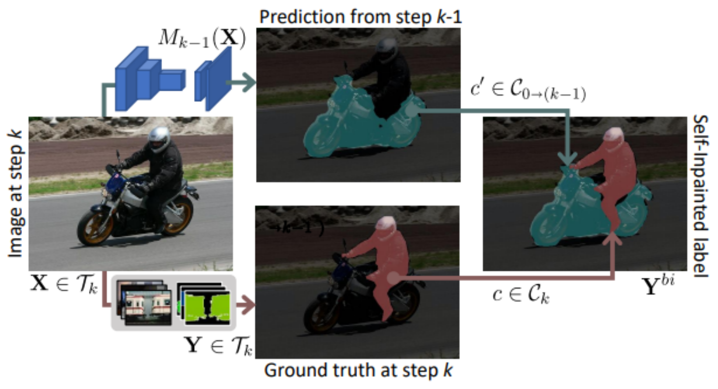


Figure 4.1: Background self-inpainting, from [5]

4.1.2 Algorithm

The complete training algorithm of RECALL is illustrated in Figure 4.2. Suppose we are at the incremental step k, where only the dataset $\mathcal{T}_k$, containing images related to the class set $\mathcal{C}_k$, is available.
The first stage of the algorithm involves the Source Block, which creates $k-1$ sets, containing images from previous learned class sets $\mathcal{C}_{(0\rightarrow k-1)}$. These dataset are then passed through the Label Evaluation Block to generate labelled replay datasets $\mathcal{R}_{\mathcal{C}_i}$ for each previous class $\mathcal{C}_i$, where $i = 0, \dots, k-1$. The final replay dataset is constructed by aggregating the individual replay datasets for each set of classes, denoted as $\mathcal{R}_{\mathcal{C}_0\rightarrow(k-1)} = \bigcup_{i=0}^{k-1} \mathcal{R}_{\mathcal{C}_i}$.
To obtain the complete training dataset for the model M_k, the replay dataset $\mathcal{R}_{\mathcal{C}_0\rightarrow(k-1)}$ is combined with $\mathcal{T}_k^{bi}$, which is generated by passing the dataset $\mathcal{T}_k$ through the model's background self-inpainting module. The resulting dataset is denoting as $\mathcal{T}_k^{rp} = \mathcal{T}_k^{bi} \cup \mathcal{R}_{\mathcal{C}_0\rightarrow(k-1)}$.
At this point, the segmentation model M_k can be effectively trained using the cross-entropy loss function $\mathcal{L}_{ce}(M_k; \mathcal{C}_{0\rightarrow k}, \mathcal{T}_k^{rp})$.
In the final stage of step k, the decoder component $D^H_{\mathcal{C}_k}$ is trained using the dataset $\mathcal{T}_k$, by minimising the cross-entropy loss $\mathcal{L}_{ce}(L_{\mathcal{C}_k}; \mathcal{C}_k \cup b, \mathcal{T}_k)$.

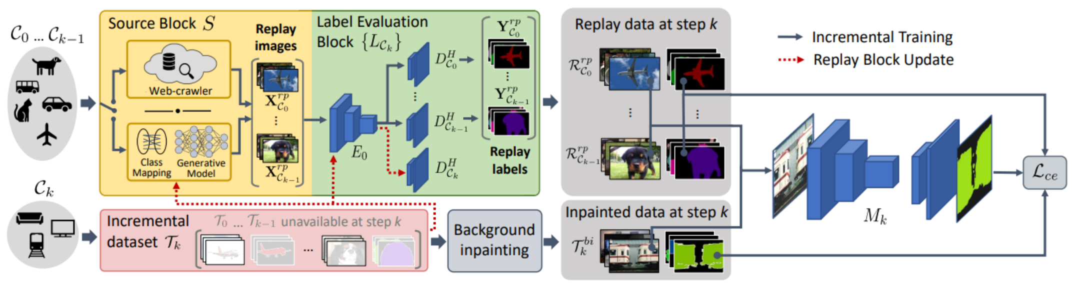

$\mathcal{C}_0$ ... $\mathcal{C}_{k-1}$
$\mathcal{C}_k$
Source Block S
Web-crawler
Class Mapping
Generative Model
Replay images
$\mathbf{X}^{rp}_{\mathcal{C}_0}$
$\mathbf{X}^{rp}_{\mathcal{C}_{k-1}}$
Label Evaluation Block $\{L_{\mathcal{C}_k}\}$
E_0
$D^H_{\mathcal{C}_0}$
$D^H_{\mathcal{C}_{k-1}}$
$D^H_{\mathcal{C}_k}$
$\mathbf{Y}^{rp}_{\mathcal{C}_0}$
$\mathbf{Y}^{rp}_{\mathcal{C}_{k-1}}$
Replay labels
Incremental dataset $\mathcal{T}_k$
$\mathcal{T}_0$ ... $\mathcal{T}_{k-1}$ unavailable at step k
Background inpainting
Replay data at step k
$\mathcal{R}^{rp}_{\mathcal{C}_0}$
$\mathcal{R}^{rp}_{\mathcal{C}_{k-1}}$
Inpainted data at step k
$\mathcal{T}^{bi}_k$
M_k
$\mathcal{L}_{ce}$
Incremental Training
Replay Block Update

Figure 4.2: RECALL architecture, from [5]

4.2 RECALL+: Adversarial Web-based Replay for Continual Learning in Semantic Segmentation

Two years later, the same research team developed an updated version of the RECALL algorithm. In this new work [6], they concentrated on devising a image selection strategy specifically designed for retrieving web images. Additionally, they introduced a background inpainting technique, which was applied to images during the current processing step. These two enhancements enabled them to achieve better results compared to the previous RECALL algorithm.

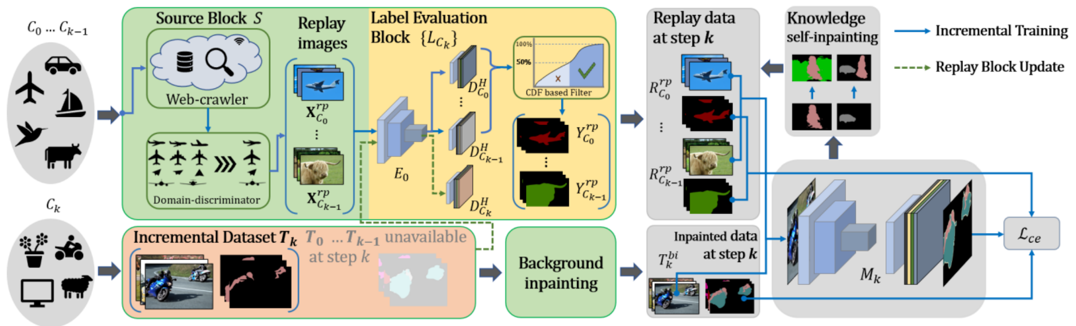


Figure 4.3: RECALL+ architecture, from [6]

4.2.1 Image selection strategies

They observed that the images retrieved using the Source Block designed in RECALL were somewhat uncontrolled. While some images were useful for training, others contained anomalies that rendered them ineffective or even misleading. For this reason, in the new version of the algorithm, they retained the main structure of the replay block described in 4.1.1 but added two new components before the creation of the replay dataset. Specifically, they introduced two image selection techniques that combine an adversarial learning method with a threshold-based selection mechanism, enabling the model to retain only the useful replay images.

Adversarial training strategy

At step 0, to effectively train the discriminator network, web images related to the class set C_0 are used as negative samples, while images associated with the dataset's class set C_0 serve as positive samples.

Consequently, at step 1, to retrieve images related to the class set $\mathcal{C}_0$, each replay image $\mathbf{X}$ generated by the Source Block is passed to the discriminator, which returns a confidence score $z = [z_p, z_{rp}] \in \mathbb{R}^2_{0+}$ for the dataset images (z_p) and the web-replay images (z_{rp}). At this point, we can define the set of *core* Replay Images $\mathcal{R}^{core}_{\mathcal{C}_0}$ as follows:

$$\mathcal{R}^{core}_{\mathcal{C}_0} = \{\mathbf{X}^{rp}_{\mathcal{C}_0} \mid z_p > \alpha \cdot z_{rp}\} \tag{4.10}$$

where α is a parameter used to control the ratio between the two scores, and it is set significantly larger than 1.
Subsequently, the domain discriminator is fine-tuned using the set $\mathcal{T}_1 \cup \mathcal{R}^{core}_{\mathcal{C}_0}$ as positive samples and $\mathbf{X}^{rp}_{\mathcal{C}_{0\to 1}}$ as negative samples. This step is crucial for effectively discriminating web images in step 2.

At step k, the available discriminator has been previously trained on the datasets $\mathcal{T}_{(k-1)} \cup \mathcal{R}^{core}_{\mathcal{C}_{(k-2)}}$ and $\mathbf{X}^{rp}_{\mathcal{C}_{0\to(k-1)}}$, using the first as positive samples while the second as negative. In the current step it is used to select the retrieved web-images:

$$\mathcal{R}^{core}_{\mathcal{C}_{0\to(k-1)}} = \{\mathbf{X}^{rp}_{\mathcal{C}_{0\to(k-1)}} \mid z_p > \alpha \cdot z_{rp}\} \tag{4.11}$$

As last stage of this step the domain discriminator is fine-tuned using the set $\mathcal{T}_k \cup \mathcal{R}^{core}_{\mathcal{C}_{0\to(k-1)}}$ as positive samples and $\mathbf{X}^{rp}_{\mathcal{C}_{0\to k}}$ as negative samples.

Image selection

Although the domain discriminator effectively reduces the shift between dataset and replay images, they noticed it is not able to differentiate between useful samples and less relevant ones. To address this issue, they developed an Image Selection method that retains only those images containing a significant number of pixels associated with the expected class.
In particular they examined the probability distribution functions of the pixel fractions for each class within the corresponding images to establish a reference for thresholding. Specifically, for each class c they calculated the Cumulative Distribution Function ($CDF, \mathcal{F}_c$) of the distribution $\mathcal{P}^c_{(\mathbf{X},\mathbf{Y})\sim\mathcal{T}_k \mid c \in \mathcal{C}_{0\to k}}[\mathbf{Y} = c]$. They then utilised this *CDF* to derive appropriate thresholds for object sizes using a quantile-based method.
To determine the threshold value t^{size}_c for the number of pixels belonging to class c in an image, they computed it as follows:

$$t_c^{size} = F_c^{-1}(0.5) \tag{4.12}$$

According to their strategy, a sample is deemed acceptable if the fraction of its pixels in the corresponding class falls within the range $[t_c^{size}, 1]$.

4.2.2 Self-teaching strategies

In their previous work, they developed the Background Self-Inpainting Block (4.1.1) to tackle the issue of background shift. While their previous work focused on addressing the problem of current images containing backgrounds associated with previous class sets, this work also explores the dual problem: the possibility that the background of replay images could contain objects related to the current classes.
For this reason, they developed *knowledge self-inpainting*, which allows the model to update the labelling of the replay images during training. In addition, they introduced a constraint term to prevent the expansion of old classes.
The knowledge self-inpainting mechanism, shown in Figure 4.4, is define as:

$$\mathbf{Y}^{ki}[h,w] = \begin{cases} \mathbf{Y}^{\text{rp}}[h,w] & \text{if} \quad \mathbf{Y}^{\text{rp}}[h,w] \in \mathcal{C}_{\mathcal{C}_{0\to(k-1)}} \\ \mathbf{Y}^{\text{max}}[h,w] & \text{if} \quad \mathbf{Y}^{\text{rp}}[h,w] \notin \mathcal{C}_{\mathcal{C}_{0\to(k-1)}} \ \wedge \ \mathbf{Y}^{\text{max}}[h,w] \in \mathcal{C}_k \\ b & \text{otherwise} \end{cases} \tag{4.13}$$

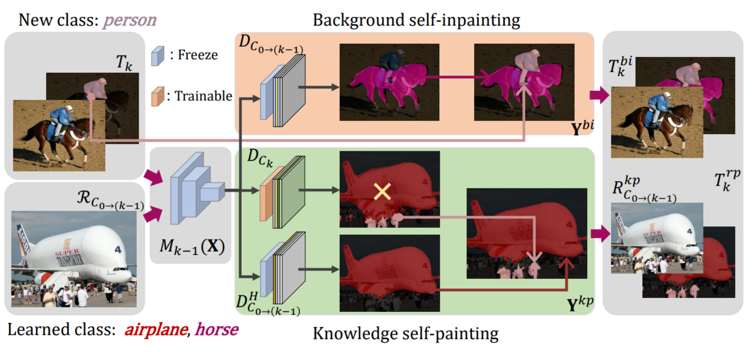


Figure 4.4: RECALL+ Self-teaching strategies, from [6]

4.3 Learning from the Web: Language Drives Weakly-Supervised Incremental Learning for Semantic Segmentation

In this third paper [7], the research team extended their web-crawler method to a more challenging scenario: Weakly-Supervised Incremental Learning for Semantic Segmentation (WILSS). Unlike the previously discussed Continual Incremental Learning for Semantic Segmentation (CILSS) (3.1), where images introduced at each step *t* are fully labelled at the pixel level, WILSS involves a weaker form of supervision. Specifically, in this scenario, the images introduced at step *t*, which correspond to the new set of classes $\mathcal{C}_t$, are only weakly labelled, meaning that the labels are provided at the image level rather than the pixel level.
The researchers assume that only the images from the initial step are fully supervised, while those introduced in all subsequent steps are weakly labelled.
In this new work, more sophisticated querying and image selection techniques have been developed to retain only the most meaningful web-retrieved images. We will focus on these advanced image selection techniques, particularly in a scenario where, at each step *t*, the current images from the dataset are accessible.
However, it is important to emphasise that they also examined an even more restrictive scenario. At step *t*, not only are the images from previous steps unavailable, but the images introduced at the current step are also absent. In this extreme case, the only information available is the names of the new classes. As a result, the system must retrieve images related to these new classes directly from the web.

Problem definition for WILSS task

The main difference compared to the convention used previously (see 4) lies in the shape of the label $\mathbf{y} \in \mathcal{Y}$. At step 0, the labels for the current images are at the pixel level, represented as $\mathbf{y} \subset \mathbb{R}^{H*W*|\mathcal{C}_0|}$. However, for all subsequent steps *k*, the labels shift to an image-level representation $y \subset \mathbb{R}^{|\mathcal{C}_k|}$.

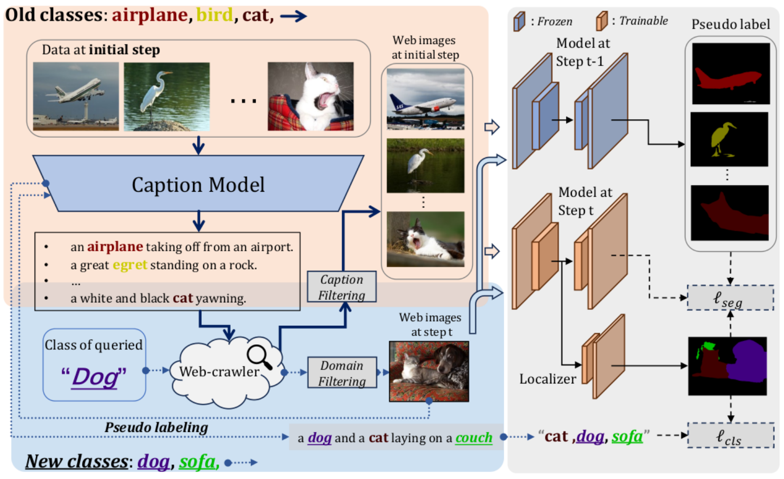


Figure 4.5: General architecture, from [7]

4.3.1 Labelling for weakly-labelled images

The method for treating weakly-labelled images builds on previous work [26]. At each step t, the architecture consists of a sheared encoder E^t, a segmentation decoder D^t that is incrementally extended to accommodate new classes, and a localiser L^t, which is trained from scratch at every step to provide pseudo-supervision for the segmentation model. Specifically, the localiser offers two types of guidance: one at the pixel level, represented as $\mathbf{y}_L^t \in \mathbb{R}^{H*W*|\mathcal{C}_{0\to t}|}$, and the other at the image level, denoted as $y_L^t \in \mathbb{R}^{|\mathcal{C}_{0\to t}|}$. Consistent with previous methods, the model from the preceding task is retained and is represented as $(D \circ E)^{t-1}$.

The image-level supervision is calculated using a multi-label soft-margin loss:

$$\mathcal{L}_{CLS}(y, y_L^t) = -\frac{1}{|\mathcal{C}_{0\to t}|} \sum_{c \in \mathcal{C}_{0\to t}} y^c \, log(y_L^c) + (1 - y^c) \, log(1 - y_L^c) \tag{4.14}$$

Considering the pixel-level predictions, and following the method outlined in [26], the initial pixel-level guidance $\mathbf{y}_L^t$ is smoothed to mitigate noise, resulting in $\tilde{\mathbf{y}}_L$ [27].This smoothed output is then combined with the predictions from the previous model:

$$\hat{\mathbf{y}} = \begin{cases} \min(\mathbf{y}_D^{c(t-1)}, \tilde{\mathbf{y}}_L^c) & \text{if} \quad c = b \\ \tilde{\mathbf{y}}_L^c & \text{if} \quad c \in \mathcal{C}^t \\ \mathbf{y}_D^{c(t-1)} & \text{otherwise} \end{cases} \tag{4.15}$$

where $\mathbf{y}_D^{t-1} \in \mathbb{R}^{H*W*|\mathcal{C}_{0\to(k-1)}|}$ is the prediction of the segmentation model $(D \circ E)^{t-1}$. Then the pixel-level loss is obtained as:

$$\mathcal{L}_{SEG}(\,\hat{\mathbf{y}},\,\mathbf{y}_D^t\,) = -\frac{1}{|\mathcal{I}|\;|\mathcal{C}_{0\to t}|}\sum_{i\in\mathcal{I}}\;\sum_{c\in\mathcal{C}_{0\to t}} \hat{y}_{(i)}^c \;log(y_{(i)\,D}^{c\,t}) + (1-\hat{y}_{(i)}^c)\;log(1-y_{(i)D}^{c\,t}) \quad (4.16)$$

where $\mathcal{I} = H * W$ denotes the pixel dimension of the image, and $\hat{y}_{(i)}$ represents the prediction for the i-th pixel.

To further enhance the segmentation capabilities of the model, two additional losses, $\mathcal{L}_{KDE}$ and $\mathcal{L}_{KDL}$, introduced in [26], are incorporated. The final objective function is as follows:

$$\mathcal{L} = \mathcal{L}_{SEG} + \mathcal{L}_{CLS} + \mathcal{L}_{KDE} + \mathcal{L}_{KDL} \quad (4.17)$$

4.3.2 Rehearsal strategies

In this paper, they explored the possibility of using the capabilities of a *vision-language model* (VLM) to generate text descriptions of images in order to develop a new strategy for querying and selecting images from the web. They worked under the assumption that it is possible to store only the captions of the images. In doing so, they succeeded in using the caption model both for querying relevant samples from the web and to provide additional supervision for verifying whether the retrieved images are useful for training. The strategy involves two key steps: querying based on captions and selecting based on captions.

Caption-based querying

In detail, this strategy uses a caption model, denoted as M_{CAP}, which takes an image $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^{H*W*3}$ as input.

At step 0, the M_{CAP} model generates captions for all the current images. Consequently, in step 1, we retrieve the replay images associated with step 0 as follows:

$$\mathcal{X}_{r,\,0}^{web} = \{\mathbf{X} = \mathcal{D}^{web}(q') \mid q' = M_{CAP}(\mathbf{x}) : \mathbf{x} \in \mathcal{T}_0\} \quad (4.18)$$

where $\mathcal{D}^{web}$ represents the distribution of the images available on the web, while $\mathcal{T}_0$ denotes

the training set utilised at step 0.
Similarly, at the generic step t, the replay images related to the sets of classes $\mathcal{C}_{0\to t}$, are retrieved as:

$$\mathcal{X}^{web}_{r,\,0\to(t-1)} = \{\mathbf{X} = \mathcal{D}^{web}(q') \mid q' = \mathcal{M}_{CAP}(\mathbf{x}) : \mathbf{x} \in \mathcal{T}_{0\to(t-1)}\} \tag{4.19}$$

Caption-based image selection

Although this new caption-based querying is more robust concerning the class name, there is no guarantee that the replay images are functional for training. For this reason, inspired by classic Natural Language Processing methods, they employed the Penn TreeBank [28] and the lexical database WordNet [29] to develop a new image selection strategy.
The idea behind this strategy is to recalculate the captions for the replay images and compare them to those used in the queries, verifying whether the words in the captions belong to the same semantic family.

More in detail, as precisely described in [7], assuming to be at step t the method is defined as follows:

1. feeding the replay image $\mathbf{x} \in \mathcal{X}^{web}_{r,\,0\to(t-1)}$ into the caption model, a second caption q'' is generated as: $q'' = \mathcal{M}_{CAP}(\mathbf{x})$;
2. they tokenized the two queries and identified two pairs of words (n'_1, n'_2) and (n''_1, n''_2), which represent the first two nouns from sentences q' and q'', respectively, based on their syntax tags as in [28];
3. for each noun n they extracted a set of hypernyms, i.e. its cognitive synonyms, from the WordNet Tree [29];
4. for each noun n they created a corresponding vectorized descriptor v (e.g., v'_1 corresponds to n'_1 , and so on) by considering the depth d of each of its hypernyms in the tree and updating $v[d]+=1$;
5. they computed the cosine distance between each vector (e.g., they compared v'_1 with both v_1 and v_2);
6. if at least one couple of descriptors is similar (e.g., v_1 with v_1 or v_2), then the image $\mathbf{x}$ is kept, meaning that it contains at least one concept that was contained in the dataset image. They assumed two vectors v_i and v_j to be similar if $1 - cos(v_i, v_j) > T$, where T represents the threshold value.

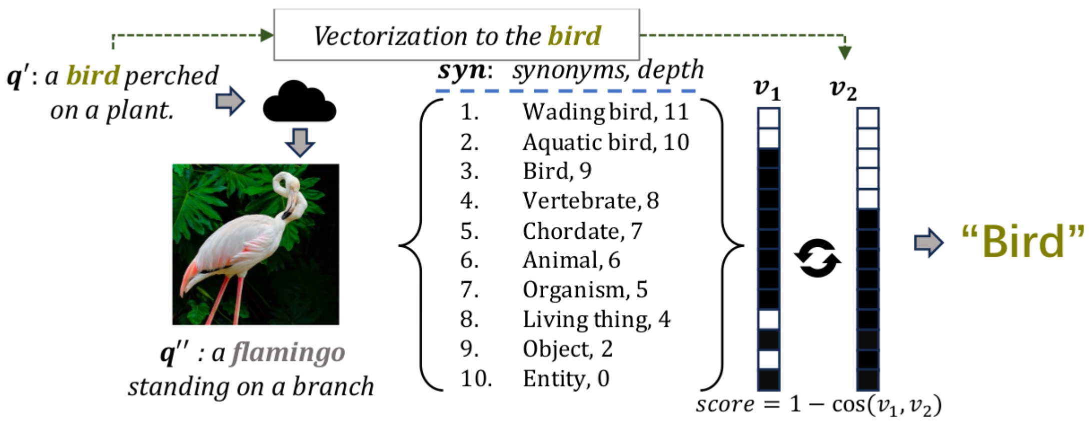


Figure 4.6: Illustration of caption-based image selection method, from [7]

5 Image Selection Techniques for Replay Images in Semantic Segmentation

In this thesis, I extended the work of [7], focusing on a scenario where, at each step after step 0, the current images from the dataset are available with image-level labels, and the replay images are retrieved using the captions of the dataset images as queries (Equation: 4.19). Specifically, I developed new image selection strategies that allow the model to retain only the most relevant replay images for the training, with the aim of improving overall performance. All the strategies developed in this work are based on the CLIP architecture [8] for feature extraction. As a naming convention, throughout this thesis, I will refer to the three image selection techniques as *strategies*, while the variations within each strategy will be termed *methods*.

CLIP architecture

In 2021, OpenAI developed a new neural network architecture called CLIP (Contrastive Language–Image Pretraining) which is efficiently able to learn visual concepts from natural language supervision. The primary objective of this architecture is to effectively correlate images with their corresponding text descriptions. It consists of two encoders: one for images and another for text. A crucial aspect of this design is that both encoders share the same embedding space. This method aims to ensure that images and their text counterparts are mapped to similar embedding vectors, or ideally, the same vector.

To achieve this goal, the model is trained on a dataset comprising millions of image-text pairs. During training, for each batch, embedding vectors are calculated for both images and their corresponding descriptive texts. These vectors are then organised into a distance matrix, as illustrated in Figure 5.1. In this matrix, the diagonal elements represent the distances between the embedding vectors of each image and its text counterpart, while the off-diagonal elements capture the distances for all other combinations. The objective of the training was to minimise the distances on the diagonal of the distance matrix, while simultaneously maximising the off-diagonal distances.

The resulting architecture efficiently extracts features from both images and texts, mapping images and texts with similar information into similar embedding vectors.

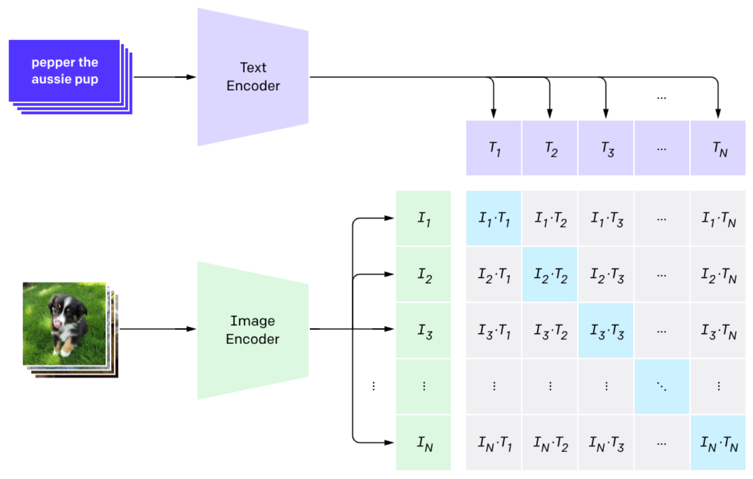

pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1{\cdot}T_1$ $I_1{\cdot}T_2$ $I_1{\cdot}T_3$ ... $I_1{\cdot}T_N$
$I_2{\cdot}T_1$ $I_2{\cdot}T_2$ $I_2{\cdot}T_3$ ... $I_2{\cdot}T_N$
$I_3{\cdot}T_1$ $I_3{\cdot}T_2$ $I_3{\cdot}T_3$ ... $I_3{\cdot}T_N$
$I_N{\cdot}T_1$ $I_N{\cdot}T_2$ $I_N{\cdot}T_3$ ... $I_N{\cdot}T_N$

Figure 5.1: Graphical representation of the batch training step in the CLIP model, from [8].

5.1 CLIP-Based Strategy for Image-to-Image Similarity Assessment

The initial strategy involved using CLIP to verify whether the retrieved images $\mathbf{X}^{web}$ were similar to the dataset one $\mathbf{X}^{orig}$ used in the queries. For both the dataset and replay images, I utilised the images as visual information and obtained the corresponding textual descriptions from the language model Flamingo [30] [31]. Given that CLIP can process both visual and linguistic information, all four combinations have been explored, as shown in the following table:

		Dataset Image	
		Text	Image
Replay Image	Text	Text-to-Text method	Image-to-Text method
	Image	Text-to-Image method	Image-to-Image method

It is important to note that the step in which the feature vector of the current image is calculated is different from the one where the feature vector of the replay image is computed. Specifically, the caption and feature vector of the dataset image are generated only once, during the single step in which the image is available. In contrast, the caption and feature vector for the replay image are recalculated during each subsequent step. This method requires a slight relaxation of the assumption, allowing not only the captions but also the feature vectors of the dataset images to be stored.
To clarify the subsequent explanations of the four methods, we will assume we are at step h, with $h > k$, and we will consider a dataset image $\mathbf{X}_c^{orig}$ associated with the class $c \in \mathcal{C}_k$.

5.1.1 Similarity by comparing text-based descriptors

In this first method, called Text-to-Text method, at step k, the dataset image $\mathbf{X}_c^{orig}$ is first processed through the caption model, and the resulting output q' is then used as input to the CLIP model to derive its feature vector $\mathbf{v}^{orig}$, as follows:

$$q' = M_{CAP}(\mathbf{X}_c^{orig}) \tag{5.1}$$

$$\mathbf{v}^{orig} = CLIP\left(q'\right) \tag{5.2}$$

At step h, the related replay image $\mathbf{X}^{web}$ is processed through the caption model to generate its corresponding text description q''. Similar to the dataset image, the text descriptor of the replay image is used to obtain its feature vector $\mathbf{v}^{web}$ as:

$$\mathbf{X}^{web} = \mathcal{D}^{web}(q') \tag{5.3}$$

$$q'' = M_{CAP}(\mathbf{X}^{web}) \tag{5.4}$$

$$\mathbf{v}^{web} = CLIP\,(q'') \tag{5.5}$$

Finally, at step h, the replay image is considered useful for the training only if the cosine similarity between the two feature vectors, $\mathbf{v}^{orig}$ and $\mathbf{v}^{web}$, exceed a specified threshold.

$$1 - cos(\,\mathbf{v}^{orig}, \mathbf{v}^{web}\,) = \begin{cases} accept & \text{if } \geq threshold \\ reject & \text{if } < threshold \end{cases} \tag{5.6}$$

The entire image selection process is graphically represented in Figure 5.2 and described by Algorithm 5.1.

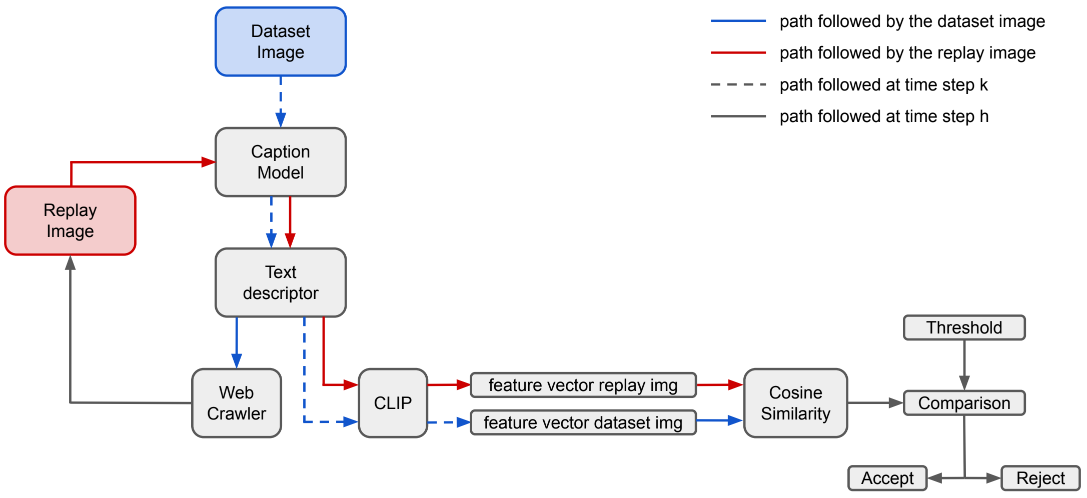

Dataset Image
Caption Model
Replay Image
Text descriptor
Web Crawler
CLIP
feature vector replay img
feature vector dataset img
Cosine Similarity
Threshold
Comparison
Accept
Reject
path followed by the dataset image
path followed by the replay image
path followed at time step k
path followed at time step h

Figure 5.2: Graphic representation of the image selection strategy applies on a replay image in the Text-to-Text method.

Algorithm 5.1 Text-to-Text method

```
**Step t**
$th$ = value of the threshold
$\mathcal{X}^{orig}_{0\to(t-1)}$ = dataset images related to the sets of classes $\mathcal{C}_{0\to(t-1)}$
$\mathcal{X}^{orig}_{t}$ = dataset images related to the sets of classes $\mathcal{C}_t$
for $\mathbf{X}^{orig}_t \in \mathcal{X}^{orig}_t$
    $q'_t = M_{CAP}(\mathbf{X}^{orig}_t)$
    $\mathbf{v}^{orig}_t = CLIP(q'_t)$
    store $\mathbf{q}'_t$
    store $\mathbf{v}^{orig}_t$
end for
for $\mathbf{X}^{orig} \in \mathcal{X}^{orig}_{0\to(t-1)}$
    $q'$ = previously saved caption corresponding to $\mathbf{X}^{orig}$
    $\mathbf{v}^{orig} = CLIP(\ q'\ )$ = previously saved feature vector corresponding to $\mathbf{X}^{orig}$
    $\mathcal{X}^{web} = \{\mathcal{D}^{web}(q') \mid q' = M_{CAP}(\mathbf{X}^{orig})\}$
    for $\mathbf{X}^{web} \in \mathcal{X}^{web}$
        $q'' = M_{CAP}(\mathbf{X}^{web})$
        $\mathbf{v}^{web} = CLIP(\ q''\ )$
        $sim = 1 - cos(\ \mathbf{v}^{orig}, \mathbf{v}^{web}\ )$
        if $sim \geq th$
            Image saved
            Break
        else
            Continue
        end if
    end for
end for
```

5.1.2 Comparison of text descriptors of dataset images and web images descriptors

In this second method, called Text-to-Image method, at step k, the feature vector $\mathbf{v}_c^{orig}$, corresponding to the dataset image $\mathbf{X}_c^{orig}$, is obtained in the same way as in the previous method (see Equation: 5.2):

$$q' = M_{CAP}(\mathbf{X}_c^{orig}) \tag{5.7}$$

$$\mathbf{v}^{orig} = CLIP\,(q') \tag{5.8}$$

The key distinction lies in the method used to compute the feature vector $\mathbf{v}^{web}$ for the replay image $\mathbf{X}^{web}$; in this case, at step h, the replay image is used directly as input to the CLIP architecture as:

$$\mathbf{X}^{web} = \mathcal{D}^{web}(q') \tag{5.9}$$

$$\mathbf{v}^{web} = CLIP\,(\mathbf{X}^{web}) \tag{5.10}$$

Finally, also in this method the replay image is considered useful for the training only if the cosine similarity between the two feature vectors, $\mathbf{v}^{orig}$ and $\mathbf{v}^{web}$, exceed a specified threshold.

$$1 - cos(\,\mathbf{v}^{orig}, \mathbf{v}^{web}\,) = \begin{cases} accept & \text{if} \geq threshold \\ reject & \text{if} < threshold \end{cases} \tag{5.11}$$

The complete image selection process is illustrated in Figure 5.3 and detailed in Algorithm 5.2.

Algorithm 5.2 Text-to-Image method

```
Step t
$th$ = value of the threshold
$\mathcal{X}^{orig}_{0\to(t-1)}$ = dataset images related to the sets of classes $\mathcal{C}_{0\to(t-1)}$
$\mathcal{X}^{orig}_t$ = dataset images related to the sets of classes $\mathcal{C}_t$
for $\mathbf{X}^{orig}_t \in \mathcal{X}^{orig}_t$
    $q'_t = M_{CAP}(\mathbf{X}^{orig}_t)$
    $\mathbf{v}^{orig}_t = CLIP(q'_t)$
    store $\mathbf{q}'_t$
    store $\mathbf{v}^{orig}_t$
end for
for $\mathbf{X}^{orig} \in \mathcal{X}^{orig}_{0\to(t-1)}$
    $q'$ = previously saved caption corresponding to $\mathbf{X}^{orig}$
    $\mathbf{v}^{orig} = CLIP(\ q'\ )$ = previously saved feature vector corresponding to $\mathbf{X}^{orig}$
    $\mathcal{X}^{web} = \{\mathcal{D}^{web}(q') \mid q' = M_{CAP}(\mathbf{X}^{orig})\}$
    for $\mathbf{X}^{web} \in \mathcal{X}^{web}$
        $\mathbf{v}^{web} = CLIP(\ \mathbf{X}^{web}\ )$
        $sim = 1 - cos(\ \mathbf{v}^{orig}, \mathbf{v}^{web}\ )$
        if $sim \geq th$
            Image saved
            Break
        else
            Continue
        end if
    end for
end for
```

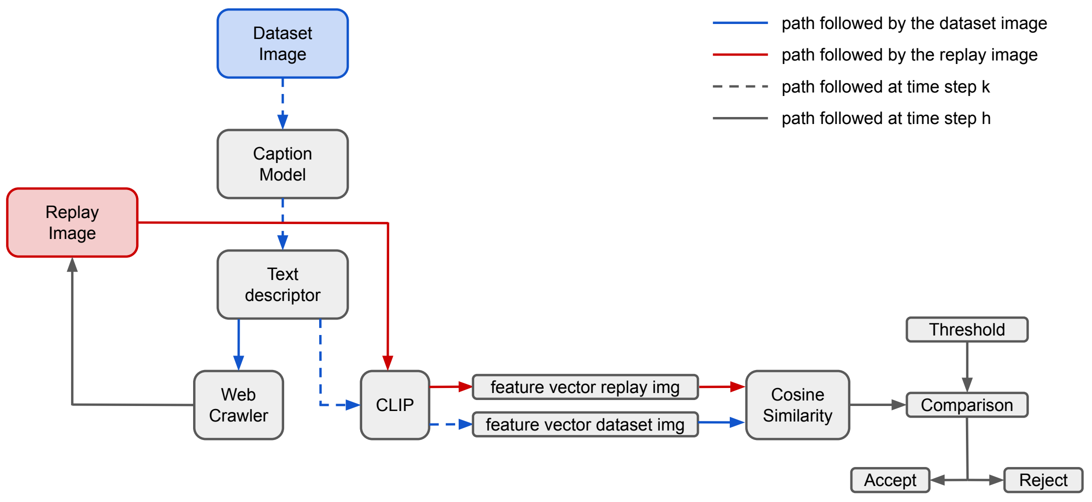

Dataset Image
Caption Model
Replay Image
Text descriptor
Web Crawler
CLIP
feature vector replay img
feature vector dataset img
Cosine Similarity
Threshold
Comparison
Accept
Reject
path followed by the dataset image
path followed by the replay image
path followed at time step k
path followed at time step h

Figure 5.3: Graphic representation of the image selection strategy applies on a replay image in the Text-to-Image method.

5.1.3 Comparison of dataset images descriptors and descriptors of the web images

The third method, called Image-to-Text method, can be considered the dual version of the second. In this case, the dataset image $\mathbf{X}_c^{orig}$ is input directly into the CLIP model to obtain its feature vector $\mathbf{v}^{orig}$. In contrast, the feature vector $\mathbf{v}^{web}$ for the replay image $\mathbf{X}^{web}$ is derived from its corresponding text descriptor q'', rather than from the image itself.
At step k the following quantities are computed:

$$q' = M_{CAP}(\mathbf{X}_c^{orig}) \tag{5.12}$$
$$\mathbf{v}^{orig} = CLIP\,(\mathbf{X}_c^{orig}) \tag{5.13}$$

while at step h:

$$\mathbf{X}^{web} = \mathcal{D}^{web}(q') \tag{5.14}$$
$$q'' = M_{CAP}(\mathbf{X}^{web}) \tag{5.15}$$
$$\mathbf{v}^{web} = CLIP\,(q'') \tag{5.16}$$

Finally, as in previous methods, the effectiveness of the replay image is evaluated through cosine similarity, as:

$$1 - cos(\mathbf{v}^{orig}, \mathbf{v}^{web}) = \begin{cases} accept & \text{if } \geq threshold \\ reject & \text{if } < threshold \end{cases} \tag{5.17}$$

The entire image selection process is depicted in Figure 5.4 and explained in Algorithm 5.3.

Algorithm 5.3 Image-to-Text method

```
Step t
$th$ = value of the threshold
$\mathcal{X}^{orig}_{0\to(t-1)}$ = dataset images related to the sets of classes $\mathcal{C}_{0\to(t-1)}$
$\mathcal{X}^{orig}_t$ = dataset images related to the sets of classes $\mathcal{C}_t$
for $\mathbf{X}^{orig}_t \in \mathcal{X}^{orig}_t$
    q'$_t = \mathcal{M}_{CAP}(\mathbf{X}^{orig}_t)$
    $\mathbf{v}^{orig}_t = CLIP(\mathbf{X}^{orig}_t)$
    store $\mathbf{q}'_t$
    store $\mathbf{v}^{orig}_t$
end for
for $\mathbf{X}^{orig} \in \mathcal{X}^{orig}_{0\to(t-1)}$
    $q'$ = previously saved caption corresponding to $\mathbf{X}^{orig}$
    $\mathbf{v}^{orig} = CLIP(\mathbf{X}^{orig})$ = previously saved feature vector corresponding to $\mathbf{X}^{orig}$
    $\mathcal{X}^{web} = \{\mathcal{D}^{web}(q') \mid q' = \mathcal{M}_{CAP}(\mathbf{X}^{orig})\}$
    for $\mathbf{X}^{web} \in \mathcal{X}^{web}$
        $q'' = \mathcal{M}_{CAP}(\mathbf{X}^{web})$
        $\mathbf{v}^{web} = CLIP(q'')$
        $sim = 1 - cos(\mathbf{v}^{orig}, \mathbf{v}^{web})$
        if $sim \geq th$
            Image saved
            Break
        else
            Continue
        end if
    end for
end for
```

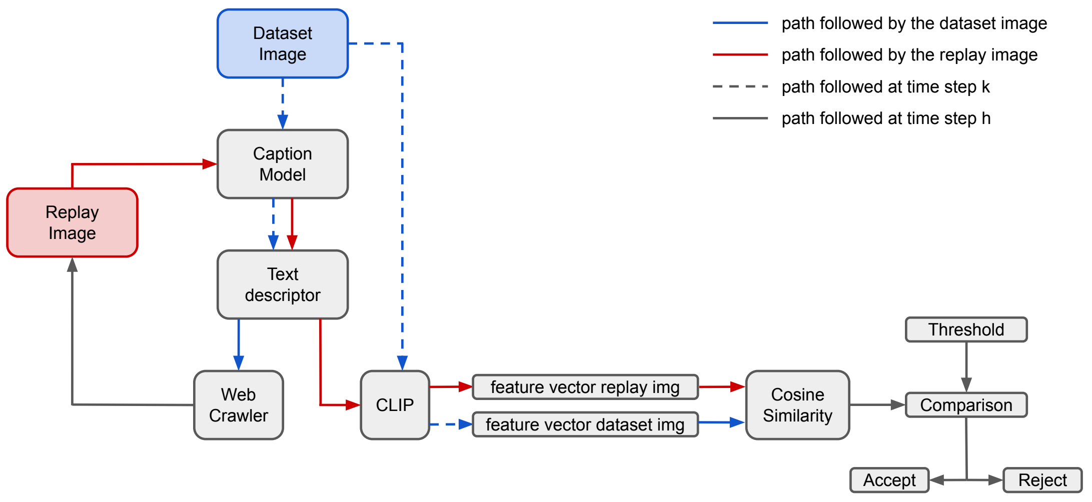

Dataset Image
Caption Model
Replay Image
Text descriptor
Web Crawler
CLIP
feature vector replay img
feature vector dataset img
Cosine Similarity
Threshold
Comparison
Accept
Reject
path followed by the dataset image
path followed by the replay image
path followed at time step k
path followed at time step h

Figure 5.4: Graphic representation of the image selection strategy applies on a replay image in the Image-to-Text method.

5.1.4 Similarity by comparing image-based descriptors

In this final method of the first strategy, called Image-to-Image method, the captioning model is utilised solely to retrieve the replay image $\mathbf{X}^{web}$. Specifically, the dataset image $\mathbf{X}_c^{orig}$ and the replay image $\mathbf{X}^{web}$ are used directly as inputs for the CLIP model to compute the two feature vectors $\mathbf{v}^{orig}$ and $\mathbf{v}^{web}$.
At step k, the caption and the feature vector of the dataset image are computed:

$$q' = M_{CAP}(\mathbf{X}_c^{orig}) \tag{5.18}$$

$$\mathbf{v}^{orig} = CLIP\,(\mathbf{X}_c^{orig}) \tag{5.19}$$

At step h,the replay image is retrieved, and its corresponding feature vector is calculated as:

$$\mathbf{X}^{web} = \mathcal{D}^{web}(q') \tag{5.20}$$

$$\mathbf{v}^{web} = CLIP\,(\mathbf{X}^{web}) \tag{5.21}$$

In this final method, the replay image's effectiveness is assessed using cosine similarity, as follows:

$$1 - cos(\,\mathbf{v}^{orig}, \mathbf{v}^{web}\,) = \begin{cases} accept & \text{if } \geq threshold \\ reject & \text{if } < threshold \end{cases} \tag{5.22}$$

The full image selection process is shown in Figure 5.5 and outlined in Algorithm 5.4.

Algorithm 5.4 Image-to-Image method

```
Step t
$th$ = value of the threshold
$\mathcal{X}^{orig}_{0\to(t-1)}$ = dataset images related to the sets of classes $\mathcal{C}_{0\to(t-1)}$
$\mathcal{X}^{orig}_t$ = dataset images related to the sets of classes $\mathcal{C}_t$
for $\mathbf{X}^{orig}_t \in \mathcal{X}^{orig}_t$
  $q'_t = \mathcal{M}_{CAP}(\mathbf{X}^{orig}_t)$
  $\mathbf{v}^{orig}_t = CLIP(\mathbf{X}^{orig}_t)$
  store $\mathbf{q}'_t$
  store $\mathbf{v}^{orig}_t$
end for
for $\mathbf{X}^{orig} \in \mathcal{X}^{orig}_{0\to(t-1)}$
  $q'$ = previously saved caption corresponding to $\mathbf{X}^{orig}$
  $\mathbf{v}^{orig} = CLIP(\,\mathbf{X}^{orig}\,)$ = previously saved feature vector corresponding to $\mathbf{X}^{orig}$
  $\mathcal{X}^{web} = \{\mathcal{D}^{web}(q') \mid q' = \mathcal{M}_{CAP}(\mathbf{X}^{orig})\}$
  for $\mathbf{X}^{web} \in \mathcal{X}^{web}$
    $\mathbf{v}^{web} = CLIP(\,\mathbf{X}^{web}\,)$
    $sim = 1 - cos(\,\mathbf{v}^{orig}, \mathbf{v}^{web}\,)$
    if $sim \geq th$
      Image saved
      Break
    else
      Continue
    end if
  end for
end for
```

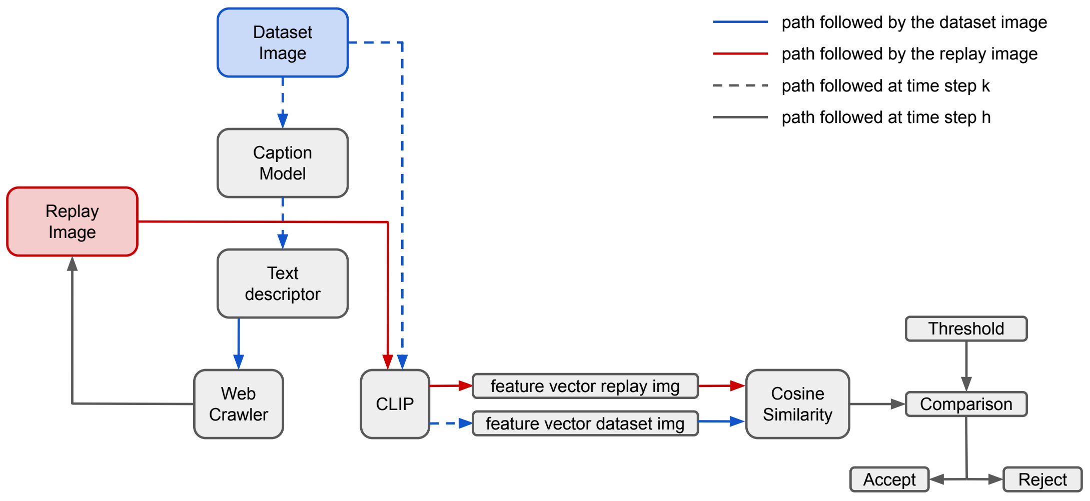

Dataset Image
Caption Model
Replay Image
Text descriptor
Web Crawler
CLIP
feature vector replay img
feature vector dataset img
Cosine Similarity
Threshold
Comparison
Accept
Reject
path followed by the dataset image
path followed by the replay image
path followed at time step k
path followed at time step h

Figure 5.5: Graphic representation of the image selection strategy applies on a replay image in the Image-to-Image method.

5.2 Image Selection Strategy Based on CLIP Class Prototypes

One potential limitation of the previous strategy, along with a possible way to enhance the image selection process, is that the replay images were closely tied to the corresponding dataset images used as queries. There may be instances where the dataset image, which contains the class $\tilde{c}$, represents a very specific situation. In such cases, a replay image could be rejected not due to its implausibility as an image containing the class $\tilde{c}$, but rather because the scenario it depicts differs significantly from that of the dataset image.
The idea behind this second strategy was to develop a method that allows the model to compare replay images, not with the specific images used as queries, but with an average representation of the classes that the replay images are expected to contain. To explain this concept in greater detail, the main objective is to build a prototype feature vector for each class that effectively captures the average information representing that class. These class-specific vectors are then leveraged to compare with the feature vectors of replay images, allowing the model to determine which images are most relevant. This method shifts the focus from direct comparison with dataset images to a more generalised, class-based comparison.
Using CLIP as a feature extractor, two possible methods have been explored: working with the image caption or directly using the image itself as input for the CLIP model.

The generic step k can be divided into two distinct stages: the first focuses on the current images, while the second deals with the replay images. The initial stage is particularly important, as it lays the foundation for subsequent steps. During this stage, although the two explored methods differ in terms of the input used by the CLIP model, the rest of the process remains identical.
In this initial phase, captions for the current images are generated and stored. Additionally, key quantities, essential for selecting images associated with the current set of classes in later stages, are computed and saved. Specifically, for each class $c \in \mathcal{C}_k$, the following quantities are crucial for the image selection strategy:

- *Class prototype* , $\bar{\mathbf{v}}_c$
- *Class mean similarity*, $\bar{s}_c$
- *Class mean standard deviation*, σ_c

The *class prototype* $\bar{\mathbf{v}}_c$ is obtained by computing the average of all the feature vectors corresponding to the dataset images that contain objects related to class c. Assuming that the values of the cosine similarity between these feature vectors and the class prototype are approximately distributed according to a Gaussian distribution, the *class mean similarity* $\bar{s}_c$ and the *class mean standard deviation* σ_c are calculated as the corresponding mean and standard deviation of the distribution.

This new image selection strategy, compared to the first one discussed in Section 5.1, offers greater stability and is also more memory-efficient. Specifically, this method eliminates the need to store a feature vector for each dataset image, requiring only one feature vector for each class instead.

5.2.1 Image-based image selection method with CLIP

In this section, we will explore in detail the method that leverages images directly as input for the CLIP model.

Assuming we are at step t, Figure 5.6 and the first part of Algorithm 5.5 represent the first stage of the step. The dataset images $\mathcal{X}_t^{orig}$ are fed to the caption model to obtain the corresponding textual descriptions. The related feature vectors $\mathbf{v}^{orig}$, obtained by passing the dataset images through the CLIP model, are used for the calculation of $\bar{\mathbf{v}}_c$, $\bar{s}_c$ and σ_c for all $c \in \mathcal{C}_k$.

In the second stage of the process, the new image selection strategy is implemented, as outlined in the second part of Algorithm 5.5 and illustrated in Figure 5.7. Once the web crawler retrieves replay images linked to the dataset images from previous steps using the saved captions, the image selection strategy is applied as follows:

1. through the CLIP model the feature vectors of the replay images are computed

2. the cosine similarities are calculated between the newly obtained feature vectors and the prototypes of all the classes present in the dataset images used as query. In particular, a replay image is considered useful for training if the cosine similarity between its feature vector and at least one prototype of the classes it is intended to represent exceeds a specific threshold. The threshold varies for each class and is defined as the class mean similarity, $\bar{s}_c$, plus a fixed coefficient, n, multiplied by the class's standard deviation σ_c. The parameter n is a hyperparameter that remains consistent across all classes. This method allows the threshold to adapt based on how the information is distributed within each class.

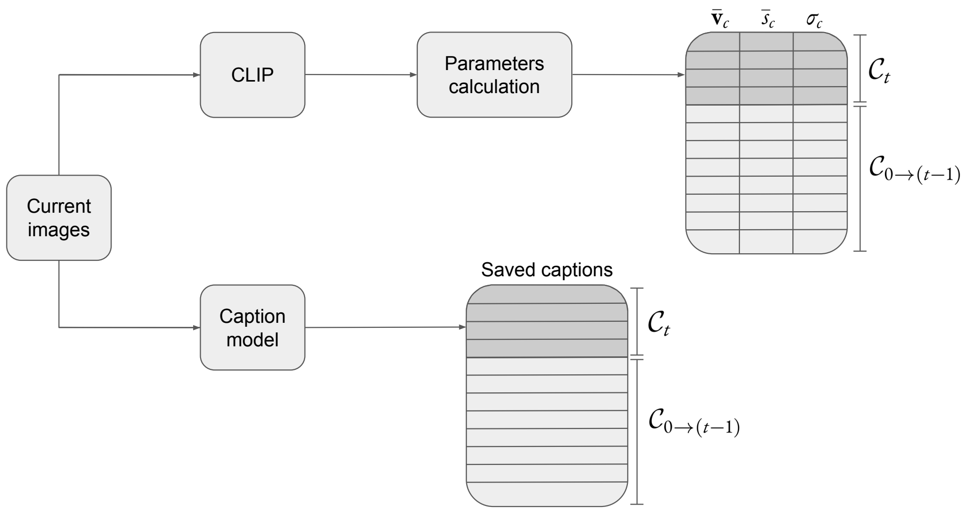


Figure 5.6: Graphical representation of the firsts stage of step t

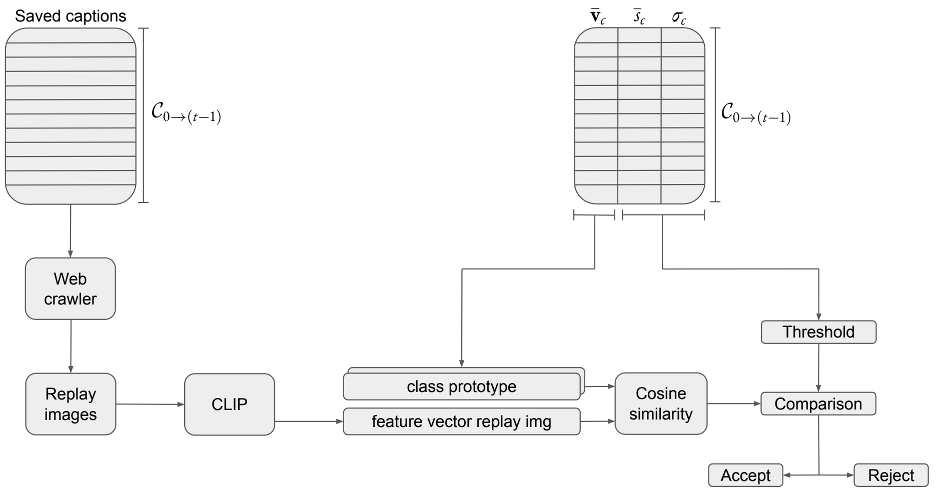


Figure 5.7: Graphical representation of the second stage of step t

Algorithm 5.5 Image-Based Image Selection Strategy using Class prototypes

```
Step t
$n$ = number of standard deviations
$\mathcal{X}^{orig}_{0\to(t-1)}$ = dataset images related to the sets of classes $\mathcal{C}_{0\to(t-1)}$
$\mathcal{X}^{orig}_{t}$ = dataset images related to the sets of classes $\mathcal{C}_t$
for $\mathbf{X}^{orig}_t \in \mathcal{X}^{orig}_t$
    $q'_t = M_{CAP}(\mathbf{X}^{orig}_t)$
    store $q'_t$
    $\mathbf{v}^{orig}_t = CLIP(\mathbf{X}^{orig}_t)$
end for
Use $\{\mathbf{v}^{orig}_t\}$ to compute and store $\bar{\mathbf{v}}_c$ , $\bar{s}_c$ and $\sigma_c$ for all $c \in \mathcal{C}_t$
for $\mathbf{X}^{orig} \in \mathcal{X}^{orig}_{0\to(t-1)}$
    $q'$ = saved caption corresponding to $\mathbf{X}^{orig}$
    c = classes belonging to $\mathbf{X}^{orig}$
    $\mathcal{X}^{web} = \{\mathcal{D}^{web}(q') \mid q' = M_{CAP}(\mathbf{X}^{orig})\}$
    for $\mathbf{X}^{web} \in \mathcal{X}^{web}$
        $\mathbf{v}^{web} = CLIP(\mathbf{X}^{web})$
        for $c \in$ c
            $\bar{\mathbf{v}}_c$ = prototype related to class $c$
            $\bar{s}_c$ = mean similarity related to class $c$
            $\sigma_c$ = standard deviation related to class $c$
            $sim = 1 - cos(\ \bar{\mathbf{v}}_c, \mathbf{v}^{web}\ )$
            if $sim \geq \bar{s}_c + n \cdot \sigma_c$
                Image saved
                Break
            else
                Continue
            end if
        end for
    end for
end for
```

5.2.2 Text-based image selection method with CLIP

This second method is similar to the one previously discussed, with the core strategy remaining unchanged.

The key difference lies in the preprocessing of images before they are fed into the CLIP model. Instead of using the dataset images and their replay versions directly, these images are first processed by a caption model. The captions generated from this processing serve as the input for the CLIP model.

Assuming we are at step t, Figure 5.8 and the first part of Algorithm 5.6 illustrate the path that the current images take during the first stage of this step. The images are fed into the caption model to generate their corresponding textual descriptions. These captions are then stored and used to compute the associated feature vectors through the CLIP model. These vectors are subsequently utilised to calculate $\bar{\mathbf{v}}_c$, $\bar{s}_c$ and σ_c for all $c \in \mathcal{C}_t$.
As outlined in the second part of Algorithm 5.6 and illustrated in Figure 5.9, the image selection strategy used in this second stage is nearly identical to that described in the first method. The only difference is that, in this case, the replay images are not fed directly into CLIP; instead, their textual captions are used as input.

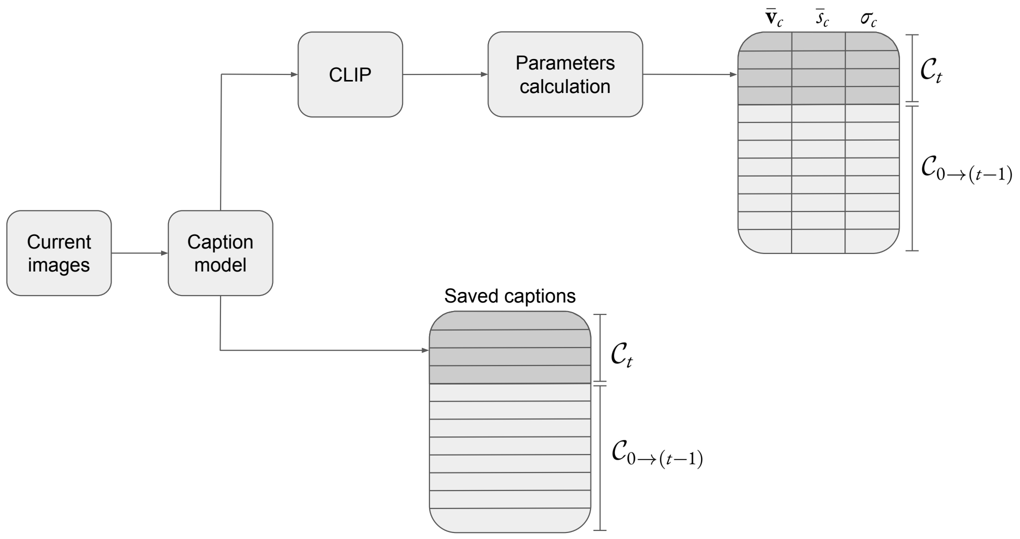


Figure 5.8: Graphical representation of the firsts stage of step t

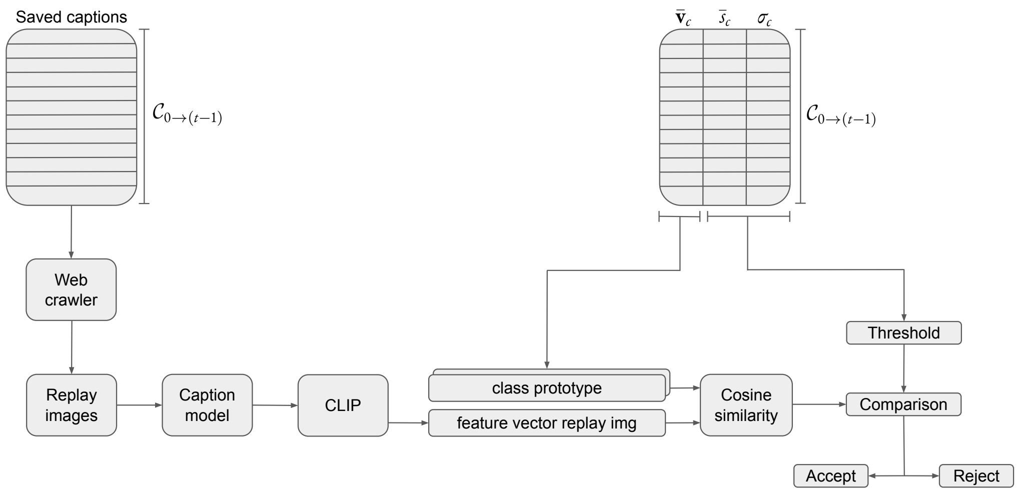


Figure 5.9: Graphical representation of the second stage of step t

Algorithm 5.6 Text-Based Image Selection Strategy using Class prototypes

```
Step t
$n$ = number of standard deviations
$\mathcal{X}^{orig}_{0\to(t-1)}$ = dataset images related to the sets of classes $\mathcal{C}_{0\to(t-1)}$
$\mathcal{X}^{orig}_{t}$ = dataset images related to the sets of classes $\mathcal{C}_t$
for $\mathbf{X}^{orig}_t \in \mathcal{X}^{orig}_t$
  $q'_t = M_{CAP}(\mathbf{X}^{orig}_t)$
  store $q'_t$
  $\mathbf{v}^{orig}_t = CLIP(\ q'_t\ )$
end for
Use $\{\mathbf{v}^{orig}_t\}$ to compute and store $\bar{\mathbf{v}}_c$ , $\bar{s}_c$ and $\sigma_c$ for all $c \in \mathcal{C}_t$
for $\mathbf{X}^{orig} \in \mathcal{X}^{orig}_{0\to(t-1)}$
  $q'$ = saved caption corresponding to $\mathbf{X}^{orig}$
  c = classes belonging to $\mathbf{X}^{orig}$
  $\mathcal{X}^{web} = \{\mathcal{D}^{web}(q') \mid q' = M_{CAP}(\mathbf{X}^{orig})\}$
  for $\mathbf{X}^{web} \in \mathcal{X}^{web}$
    $q'' = M_{CAP}(\mathbf{X}^{web})$
    $\mathbf{v}^{web} = CLIP(\ q''\ )$
    for $c \in \mathbf{c}$
      $\bar{\mathbf{v}}_c$ = prototype related to class $c$
      $\bar{s}_c$ = mean similarity related to class $c$
      $\sigma_c$ = standard deviation related to class $c$
      $sim = 1 - cos(\ \bar{\mathbf{v}}_c, \mathbf{v}^{web}\ )$
      if $sim \geq \bar{s}_c + n \cdot \sigma_c$
        Image saved
        Break
      else
        Continue
      end if
    end for
  end for
end for
```

5.3 Image Selection Strategy Based on Reduced CLIP Class Propotypes using PCA

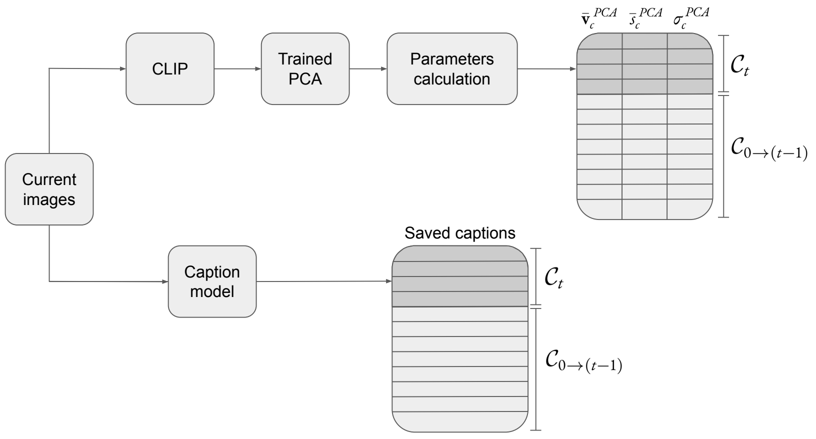
This third strategy builds on the previous idea of comparing feature vectors, not with those from individual dataset images, but with feature vectors that capture general information about specific classes. I extended this method further by comparing the feature vectors of the replay images with new class prototypes, which provide an even more generalised representation of the class. To achieve this broader representation, I reduced the dimensionality of the feature vectors using the Principal Component Analysis (PCA) algorithm [32]. The PCA model in the architecture is always inserted immediately after the CLIP model. Additionally, in this strategy, two possible methods have been developed: working with image captions or directly using the image itself as input for the CLIP model.

5.3.1 Image-based image selection method with CLIP and PCA

In this method, both the dataset and replay images are directly used as input to the CLIP model. The PCA model, which is subsequently applied for dimensionality reduction, is trained only once at step 0. To train the PCA, all feature vectors corresponding to the dataset images available at step 0 are used. Once the PCA model is trained, it is used to reduce the dimensionality of the output vectors from CLIP for all subsequent steps.
The following paragraphs will provide a more detailed explanation of the strategy.

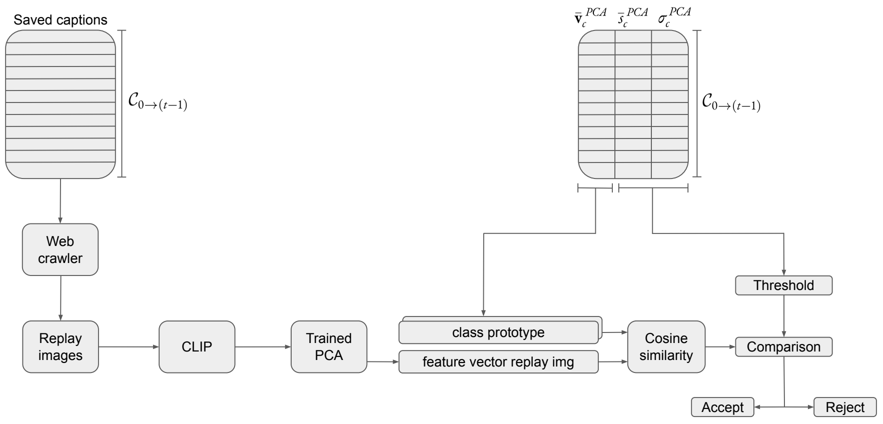
At step 0, all the dataset available images $\mathcal{X}_0$ are fed the the CLIP model and the resulting feature vectors are used to train the PCA.
Assume we are at step t and the trained PCA model is available. In the first stage of this step, illustrated in Figure 5.10 and in the first part of Algorithm 5.7, the current images are fed into the caption model, and the resulting textual captions are saved. Simultaneously, the images are directly input into the CLIP model. The resulting feature vectors are first reduced by passing through the trained PCA model and then used to calculate the quantities $\bar{\mathbf{v}}_c^{PCA}$, $\bar{s}_c^{PCA}$ and σ_c^{PCA} for all $c \in \mathcal{C}_t$.
In the second stage of the step, the replay images related to previous classes are selected, as shown in the Figure 5.11 and in the second part of the Algorithm 5.7. First, the replay images are fed into the CLIP model, and the resulting feature vectors are reduced using PCA. Next,

the cosine similarities between the reduced feature vectors and the reduced prototypes, for all classes present in the dataset query image, are calculated. A replay image is considered useful for training if its cosine similarity exceeds the threshold for at least one of the associated class prototypes. This threshold varies for each class and is defined as the class mean similarity, $\bar{s}_c^{PCA}$, plus a fixed coefficient, n, multiplied by the class's standard deviation, σ_c^{PCA}. The parameter n is a hyperparameter that remains consistent across all classes, allowing the threshold to adapt based on the distribution of information within each class.



$\bar{\mathbf{v}}_c^{PCA}$ $\bar{s}_c^{PCA}$ σ_c^{PCA}
CLIP
Trained PCA
Parameters calculation
$\mathcal{C}_t$
$\mathcal{C}_{0\to(t-1)}$
Current images
Saved captions
Caption model
$\mathcal{C}_t$
$\mathcal{C}_{0\to(t-1)}$

Figure 5.10: Graphical representation of the firsts stage of step t



Saved captions
$\mathcal{C}_{0\to(t-1)}$
$\bar{\mathbf{v}}_c^{PCA}$ $\bar{s}_c^{PCA}$ σ_c^{PCA}
$\mathcal{C}_{0\to(t-1)}$
Web crawler
Threshold
Replay images
CLIP
Trained PCA
class prototype
feature vector replay img
Cosine similarity
Comparison
Accept
Reject

Figure 5.11: Graphical representation of the second stage of step t

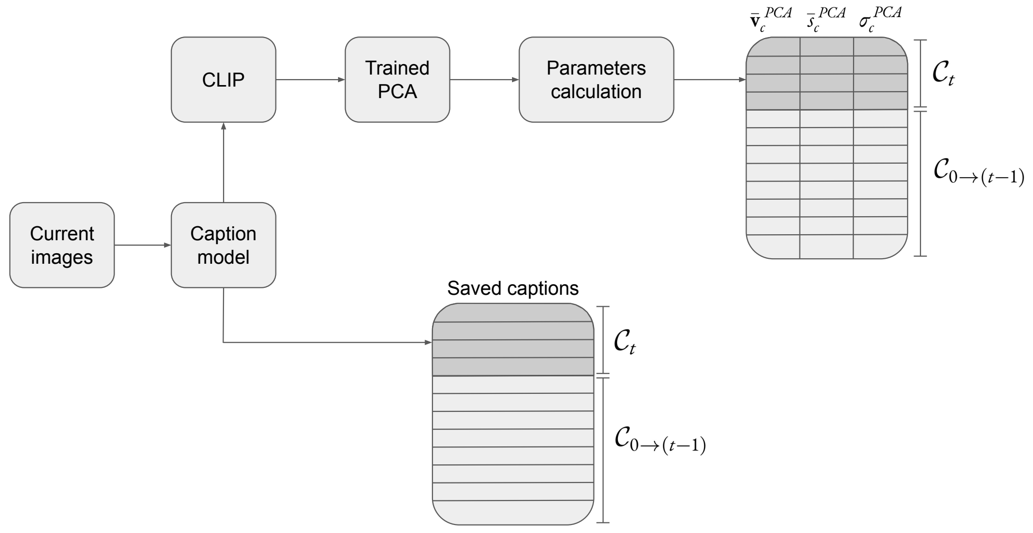
Algorithm 5.7 Image-Based Image Selection Strategy using Reduced Class Prototypes

```
**Step t**
$d$ = output dimension from PCA
$n$ = number of standard deviations
$\mathcal{X}^{orig}_{0\to(t-1)}$ = dataset images related to the sets of classes $\mathcal{C}_{0\to(t-1)}$
$\mathcal{X}^{orig}_t$ = dataset images related to the sets of classes $\mathcal{C}_t$
**for** $\mathbf{X}^{orig}_t \in \mathcal{X}^{orig}_t$
    $q'_t = M_{CAP}(\mathbf{X}^{orig}_t)$
    **store** $q'$
    $\mathbf{v}^{orig}_t = CLIP(\mathbf{X}^{orig}_t)$
    $\mathbf{v}^{orig\_PCA}_t = PCA(\mathbf{X}^{orig}_t, d)$
**end for**
**Use** $\{\mathbf{v}^{orig\_PCA}_t\}$ **to compute and store** $\bar{\mathbf{v}}^{PCA}_c$, $\bar{s}^{PCA}_c$ **and** $\sigma^{PCA}_c$ **for all** $c \in \mathcal{C}_t$
**for** $\mathbf{X}^{orig} \in \mathcal{X}^{orig}_{0\to(t-1)}$
    $q'$ = saved caption corresponding to $\mathbf{X}^{orig}$
    c = classes belonging to $\mathbf{X}^{orig}$
    $\mathcal{X}^{web} = \{\mathcal{D}^{web}(q') \mid q' = M_{CAP}(\mathbf{X}^{orig})\}$
    **for** $\mathbf{X}^{web} \in \mathcal{X}^{web}$
        $\mathbf{v}^{web} = CLIP(\mathbf{X}^{web})$
        $\mathbf{v}^{web}_{PCA} = PCA(\mathbf{v}^{web}, d)$
        **for** $c \in \mathbf{c}$
            $\bar{\mathbf{v}}^{PCA}_c$ = prototype related to class $c$
            $\bar{s}^{PCA}_c$ = mean similarity related to class $c$
            $\sigma^{PCA}_c$ = standard deviation related to class $c$
            $sim = 1 - cos(\bar{\mathbf{v}}^{PCA}_c, \mathbf{v}^{web}_{PCA})$
            **if** $sim \geq \bar{s}^{PCA}_c + n \cdot \sigma^{PCA}_c$
                Image saved
                **Break**
            **else**
                **Continue**
            **end if**
        **end for**
    **end for**
**end for**
```

5.3.2 Text-based image selection method with CLIP and PCA

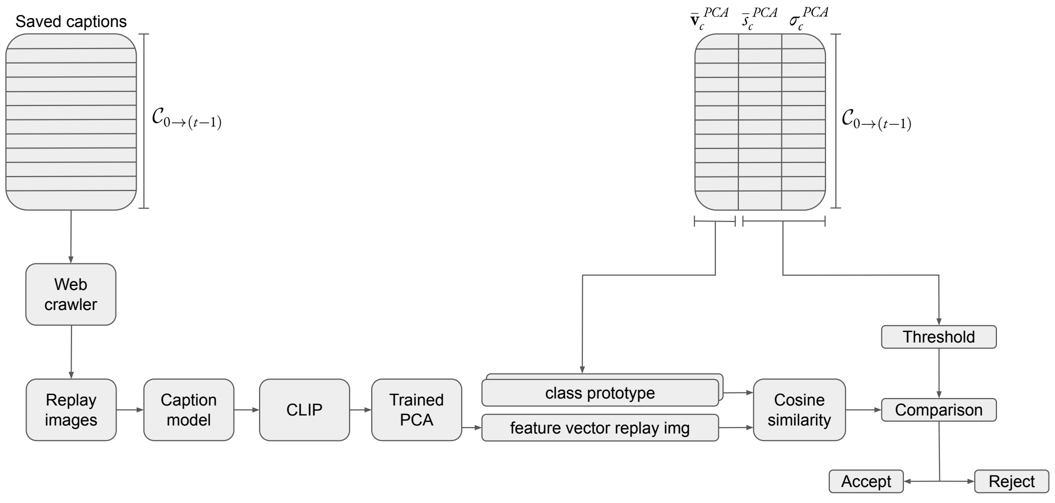
The second method of this strategy uses the captions associated with the dataset and replay images as input for the CLIP model. The core of the image selection strategy remains the same; the only difference is that the caption model has been inserted before the CLIP model each time.

At step 0, all the dataset available images $\mathcal{X}_0$ are sequentially fed into the caption model and the CLIP model. The resulting feature vectors are then used to train the PCA.
Assume we are at step t and the trained PCA model is available. In the first stage of this step, illustrated in Figure 5.12 and in the first part of Algorithm 5.8, the current images are fed into the caption model. The resulting textual descriptions are saved and also passed through the CLIP model. As in the previous method, the resulting feature vectors are first reduced by passing through the trained PCA model and then used to calculate the quantities $\bar{\mathbf{v}}_c^{PCA}$, $\bar{s}_c^{PCA}$ and σ_c^{PCA} for all $c \in \mathcal{C}_t$.
In the second stage of the step, the replay images related to previous classes are selected, as shown in Figure 5.13 and in the second part of Algorithm 5.8. The image selection procedure remains almost identical to the previously explained method, with the only difference being that the feature vectors for the replay images are obtained using their textual descriptions instead of the images themselves. For this reason, the replay images are not fed directly into the CLIP model; instead, they first pass through the caption model. The remaining processes of the strategy remain the same.



$\bar{\mathbf{v}}_c^{PCA}$ $\bar{s}_c^{PCA}$ σ_c^{PCA}
CLIP
Trained PCA
Parameters calculation
$\mathcal{C}_t$
$\mathcal{C}_{0\to(t-1)}$
Current images
Caption model
Saved captions
$\mathcal{C}_t$
$\mathcal{C}_{0\to(t-1)}$

Figure 5.12: Graphical representation of the firsts stage of step t



Saved captions
$\mathcal{C}_{0\to(t-1)}$
$\bar{\mathbf{v}}_c^{PCA}$ $\bar{s}_c^{PCA}$ σ_c^{PCA}
$\mathcal{C}_{0\to(t-1)}$
Web crawler
Threshold
Replay images
Caption model
CLIP
Trained PCA
class prototype
feature vector replay img
Cosine similarity
Comparison
Accept
Reject

Figure 5.13: Graphical representation of the second stage of step t

Algorithm 5.8 Text-Based Image Selection Strategy using Reduced Class Prototypes

```
Step t
$d$ = output dimension from PCA
$n$ = number of standard deviations
$\mathcal{X}^{orig}_{0\to(t-1)}$ = dataset images related to the sets of classes $\mathcal{C}_{0\to(t-1)}$
$\mathcal{X}^{orig}_{t}$ = dataset images related to the sets of classes $\mathcal{C}_t$
for $\mathbf{X}^{orig}_t \in \mathcal{X}^{orig}_t$
    $q'_t = M_{CAP}(\mathbf{X}^{orig}_t)$
    store $q'$
    $\mathbf{v}^{orig}_t = CLIP(q')$
    $\mathbf{v}^{orig\_PCA}_t = PCA(\mathbf{X}^{orig}_t, d)$
end for
Use $\{\mathbf{v}^{orig\_PCA}_t\}$ to compute and store $\bar{\mathbf{v}}^{PCA}_c$, $\bar{s}^{PCA}_c$ and $\sigma^{PCA}_c$ for all $c \in \mathcal{C}_t$
for $\mathbf{X}^{orig} \in \mathcal{X}^{orig}_{0\to(t-1)}$
    $q'$ = saved caption corresponding to $\mathbf{X}^{orig}$
    c = classes belonging to $\mathbf{X}^{orig}$
    $\mathcal{X}^{web} = \{\mathcal{D}^{web}(q') \mid q' = M_{CAP}(\mathbf{X}^{orig})\}$
    for $\mathbf{X}^{web} \in \mathcal{X}^{web}$
        $q'' = M_{CAP}(\mathbf{X}^{web})$
        $\mathbf{v}^{web} = CLIP(q'')$
        $\mathbf{v}^{web}_{PCA} = PCA(\mathbf{v}^{web}, d)$
        for $c \in \mathbf{c}$
            $\bar{\mathbf{v}}^{PCA}_c$ = prototype related to class $c$
            $\bar{s}^{PCA}_c$ = mean similarity related to class $c$
            $\sigma^{PCA}_c$ = standard deviation related to class $c$
            $sim = 1 - cos(\bar{\mathbf{v}}^{PCA}_c, \mathbf{v}^{web}_{PCA})$
            if $sim \geq \bar{s}^{PCA}_c + n \cdot \sigma^{PCA}_c$
                Image saved
                Break
            else
                Continue
            end if
        end for
    end for
end for
```

6
Dataset

The PASCAL VOC 2012 dataset is one of the most widely recognized and extensively used datasets in the field of computer vision, particularly for tasks such as object detection, image segmentation, and image classification. It contains a rich collection of images, each depicting objects from 20 different categories across various classes like animals, vehicles, and indoor objects. The images are annotated with ground-truth data, making them valuable for training, validating, and evaluating machine learning models. A key feature of the dataset is that it presents objects in realistic scenarios, adding to its utility in real-world applications.
The 20 object classes available in PASCAL VOC 2012 are divided into four main groups:

- ***Person***: person;
- ***Animal***: bird, cat, cow, dog, horse, sheep;
- ***Vehicle***: aeroplane, bicycle, boat, bus, car, motorbike, train;
- ***Indoor***: bottle, chair, table, plant, sofa, monitor

Figure 6.1: Aeroplane

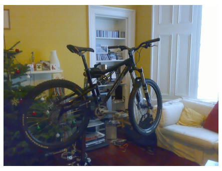


Figure 6.2: Bicycle

Figure 6.3: Bird

Figure 6.4: Boat

Figure 6.5: Bottle

Figure 6.6: Bus

Figure 6.7: Car

Figure 6.8: Cat

Figure 6.9: Chair

Figure 6.10: Cow

Figure 6.11: Table

Figure 6.12: Dog

Figure 6.13: Horse

Figure 6.14: Motorbike

Figure 6.15: Person

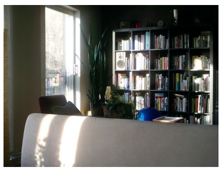


Figure 6.16: Plant

Figure 6.17: Sheep

Figure 6.18: Sofa

Figure 6.19: Train

Figure 6.20: Monitor

The dataset provides several types of annotations for each image. These include bounding boxes, segmentation masks, image-level labels, and action labels. These annotations support a variety of computer vision tasks, enabling the dataset's versatility across multiple applications.

PASCAL VOC 2012 is commonly used for tasks such as:

- Image classification, where models are trained to classify entire images based on the presence of objects.
- Object detection, where the goal is to localize and identify objects within images using bounding boxes.
- Image segmentation, which focuses on pixel-wise classification to segment objects within an image.
- Object segmentation, a more fine-grained task where individual objects are segmented.
- Action recognition, using action labels to identify activities or interactions involving objects or people.

7 Results

In this chapter, we analyse the performance of the segmentation model combined with the three developed image selection strategies. The results are primarily compared to those reported in [7], which we will refer to as *LfW*. Additionally, two other values are used for comparison: the *baseline* and the *upper bound*. The *baseline* is obtained by using the replay images without applying any image selection strategies, while the *upper bound* is achieved by the single shot training where all the dataset images are available at the initial step, without relying on replay images.

7.1 Experimental setting

7.1.1 Web dataset

Following the methodology outlined in [7], the Flickr website is used as primary source for web images due to its extensive collection of diverse content. For each class being learned, a set of $10,000$ web images has been gathered. For the old class images, 20 images for each dataset image caption were retrieved using the download caption equation (Eq. 4.19). However, to ensure comparability with previous results, I employed 100 rehearsal images for each previously learned class.

7.1.2 Implementation detail

The code for this project is implemented using PyTorch version 1.13.1 and Python version 3.10.14. The segmentation architecture is based on the method described in [7], utilising the DeepLabV3 model [12] with ResNet101 [13] as the backbone. Training of the model is conducted using Stochastic Gradient Descent (SGD) over 30 epochs in the initial step, followed by 40 epochs in subsequent incremental steps. The initial learning rate is set to $lr = 0.01$. The model is trained with pseudo annotations, integrating outputs from the localizer with predictions from the previous version of the model.

For image captioning, I employ the M_{CAP} model, in line with the methodology from [7]. Specifically, I use the pre-trained Visual Language Model (VLM) OpenFlamingo [30][31].

7.2 Setting

In this thesis, the segmentation architecture and image selection strategies are evaluated on the PASCAL VOC 2012 dataset using three distinct settings. These settings differ based on how new classes are introduced incrementally. Notably, pixel-level annotations are available only for the images in step 0, while in subsequent steps, new images are annotated only at the image-level. All evaluations are performed using an overlapping configuration. The three settings, ordered by increasing difficulty, are described as follows:

15-5-ov

In this setting, the first 15 classes are introduced at step 0, while the remaining 5 classes are added at step 1. The distribution of available classes at each step is outlined as follows:

Step	**Introduced classes**
0	Aeroplane Bicycle Bird Boat Bottle Bus Car Cat Chair Cow Table Dog Horse Motorbike Person
1	Plant Sheep Sofa Train Monitor

10-10-ov

In this configuration, the initial 10 classes are included at step 0, with the remaining 10 classes introduced at step 1. The class distribution for each step is as follows:

Step	Introduced classes
0	Aeroplane Bicycle Bird Boat Bottle Bus Car Cat Chair Cow
1	Table Dog Horse Motorbike Person Plant Sheep Sofa Train Monitor

10-10s-ov

In this setup, the first 10 classes are introduced at step 0, while the remaining 10 classes are added sequentially, one at a time, over the subsequent 10 steps. The distribution of classes across steps is detailed as follows:

Step	Introduced classes
0	Aeroplane Bicycle Bird Boat Bottle Bus Car Cat Chair Cow
1	Table
2	Dog
3	Horse
4	Motorbike
5	Person
6	Plant
7	Sheep
8	Sofa
9	Train
10	Monitor

The model's performance is evaluated using the mean Intersection over Union (*mIoU*) metric, which generalises the Intersection over Union (*IoU*) metric across multiple classes. As shown in Figure 7.1, given an image **X** and a class *c*, the *IoU* is calculated as the ratio between the intersection and union of the predicted and ground-truth masks, expressed by the following equation:

$$IoU = \frac{\text{(Groud Truth Mask)} \cap \text{(Predicted Mask)}}{\text{(Groud Truth Mask)} \cup \text{(Predicted Mask)}} = \frac{TP}{FN + TP + FP} \tag{7.1}$$

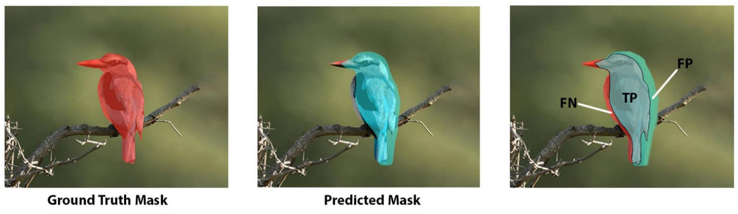


Figure 7.1: Graphical representation of IoU metric, from [9].

For tasks involving multiple classes, the $mIoU$ is defined as the average IoU across all classes:

$$mIoU = \frac{1}{N}\sum_{i=0}^{N} IoU_i \qquad N = \text{number of classes} \tag{7.2}$$

The $mIoU$ is a widely used metric in semantic segmentation because it provides a more robust assessment than pixel accuracy, as it accounts for both how well the model identifies the presence of a class and how much it incorrectly classifies other regions.

The table below summarises the $mIoU$ performances of the *baselin*e model, the *upper bound*, and the *LfW* under the three experimental settings:

Image selection strategy	15-5-ov	10-10-ov	10-10s-ov
LfW	73.4%	65.3%	52.2%
Baseline	72.6%	64.9%	50.8%
Upper bound	75.4%	75.4%	75.4%

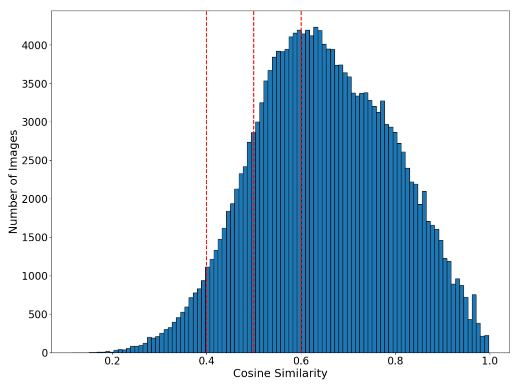
Table 7.1: $mIoU$ performances of the *baseline* model, the *upper bound*, and the *LfW* image selection strategy across the three experimental settings.

Table 7.1 shows that, as expected, reducing the number of available classes in the initial step while increasing the number of subsequent steps makes the task more challenging.

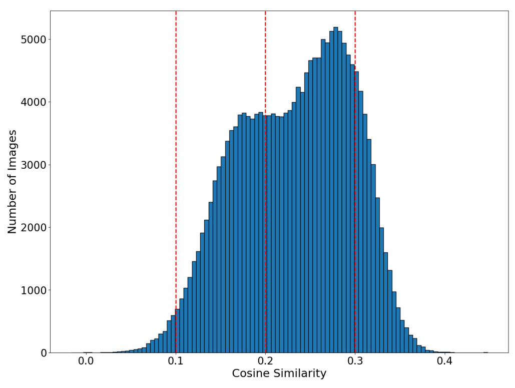
7.3 Results on CLIP-Based Strategy for Image-to-Image Similarity Assessment

In the image selection strategy outlined in Algorithms 5.1, 5.2, 5.3, and 5.4, a crucial distinction is the necessity of selecting an external threshold value manually. This requirement differentiates it from the other two strategies, where threshold values are automatically determined and tailored for each class.

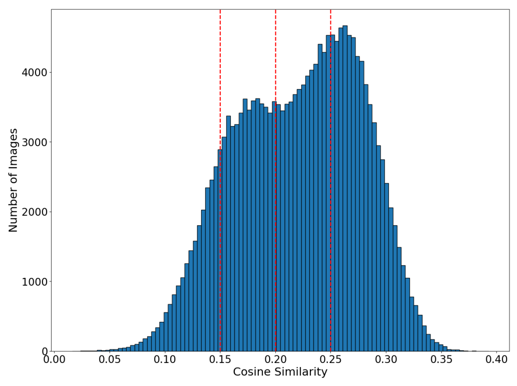
To identify a suitable threshold for each of the four methods, I examined the distribution of cosine similarities between the feature vectors of replay images and those of the dataset images.



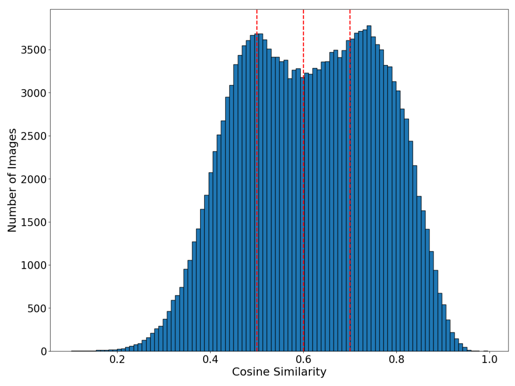
(a) Text-to-Text method



(b) Text-to-Image method



(c) Image-to-Text method



(d) Image-to-Image method

Figure 7.2: Distribution of cosine similarity between the feature vectors of replay images and those of the dataset images across the four methods in the first image selection strategy. The red dashed lines represent the values of the threshold shown in Table 7.2

Table 7.2 summarises the top three threshold selections for each of the four methods, based

on the testing of various threshold values informed by reference distributions.
The notation used in the following table consists of two types and a value. The two types represent how respectively the dataset and the replay images are used: *Image* indicates that the images are directly used as input for the CLIP model, while *text* indicates that the captions of the images are used as input. The number represents the value of the threshold used in that specific method.

Image selection strategy	**15-5-ov**	**Δ**	**10-10-ov**	**Δ**	**10-10s-ov**	**Δ**
LfW	73.40%		65.35%		52.29%	
Image-Image_0.5	73.30%	−0.10%	66.02%	+0.67%	51.73%	−0.56%
Image-Image_0.6	73.70%	+0.30%	65.94%	+0.59%	54.62%	+2.33%
Image-Image_0.7	73.27%	−0.13%	65.79%	+0.44%	51.58%	−0.71%
Image-Text_0.15	73.33%	−0.07%	65.51%	+0.16%	51.17%	−1.12%
Image-Text_0.2	73.81%	+0.41%	65.71%	+0.36%	51.52%	−0.77%
Image-Text_0.25	73.31%	−0.09%	65.70%	+0.35%	54.58%	+2.29%
Text-Image_0.1	73.22%	−0.18%	65.89%	+0.54%	53.29%	+1.00%
Text-Image_0.2	73.27%	−0.13%	65.55%	+0.20%	53.51%	+1.22%
Text-Image_0.3	73.31%	−0.09%	65.79%	+0.44%	53.35%	+1.06%
Text-Text_0.4	73.22%	−0.18%	65.83%	+0.48%	52.11%	−0.18%
Text-Text_0.5	73.61%	+0.21%	65.95%	+0.60%	52.49%	+0.20%
Text-Text_0.6	73.04%	−0.36%	64.44%	−0.91%	51.77%	−0.52%
Baseline	72.6%		64.9%		50.8%	
Upper bound	75.4%		75.4%		75.4%	

Table 7.2: Results of the three settings using the CLIP-Based Strategy for Image-to-Image Similarity Assessment. This table compares the image selection strategy against the baseline, the upper bound, and the method from the previous paper. The Δ values represent the differences between the obtained results and those of LfW.

From Table 7.2, we can notice that the intermediate values of the threshold across the four methods outperform the results obtained by LfW in nearly all settings, except for the 15-5-

ov setting in the Text-Image method and for the 10-10s-ov setting in the Image-Text method. The better performance of the intermediate threshold values can be attributed to the fact that a higher threshold on similarity between the replay and dataset images will lead to discard too much replay data; we aim to retrieve images representing the class, even if they differ from the query images. However, setting the threshold too low could result in retrieving images that lack useful information.

In the 15-5-ov setting, the method achieving the highest performance is Image-Text_0.2, which improves upon the LfW result by 0.41%. For the 10-10-ov setting, both Text-Text_0.5 and Image-Image_0.6 achieve notable improvements, with respective increases of 0.60% and 0.59%. Furthermore, Image-Image_0.6 stands out with the best performance in the 10-10s-ov setting, showing a significant increase of 2.33%.

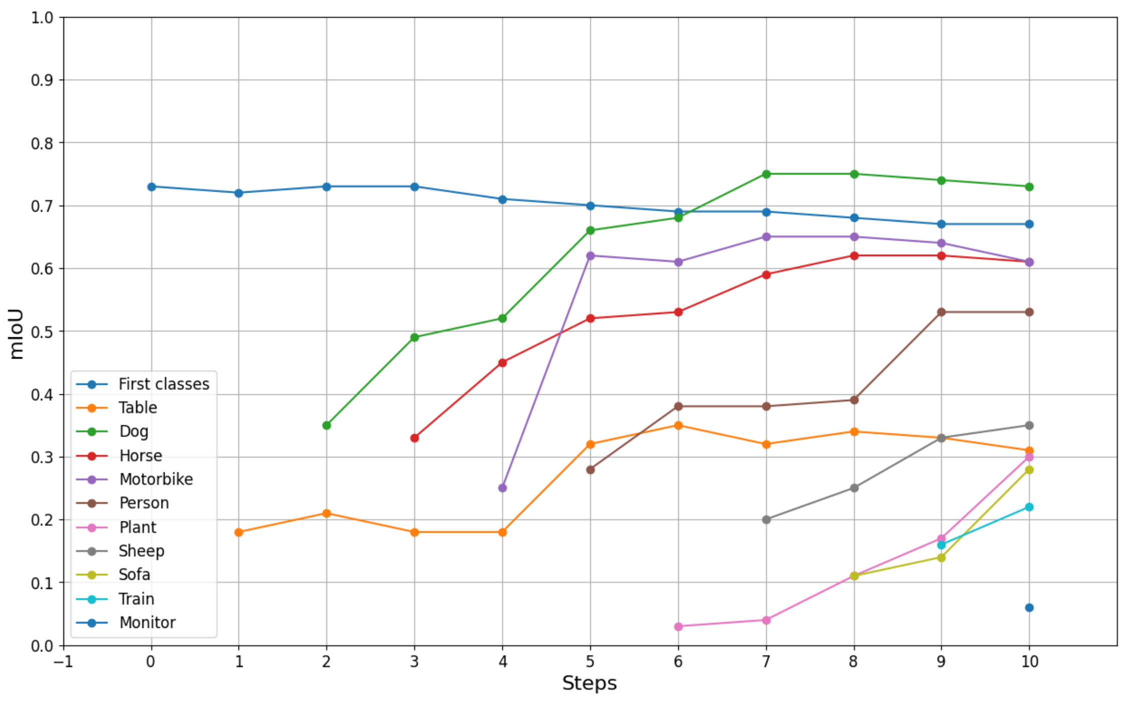
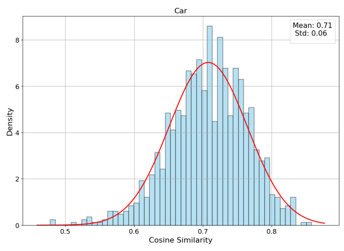


Figure 7.3: Behaviour of the $mIoU$ for classes across the steps in the 10-10s-ov setting using the Image-Image_0.6 method.

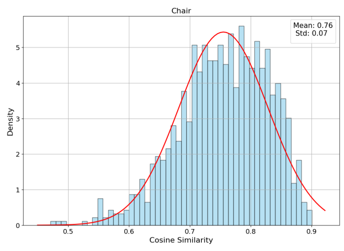
Given these observations, we can conclude that the Image-Image_0.6 method emerges as the overall most effective one within this image selection strategy.

7.4 Results on Image Selection Strategy Based on CLIP Class Prototypes

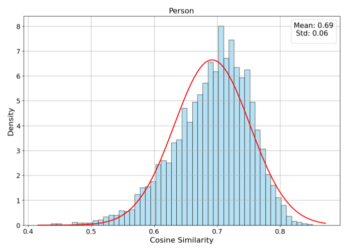
In this section, we analyse the performance achieved using the second strategy, as outlined in Algorithms 5.5 and 5.6. In this image selection strategy, the external hyperparameter that is manually selected is the scaling factor for the standard deviations, denoted by n. Once n is chosen, the threshold for each class is dynamically determined by the formula $th_c = \bar{s}_c + n \cdot \sigma_c$, where $\bar{s}_c$ and σ_c represent the mean and standard deviation, respectively, of the cosine similarity distribution for each class, under the assumption that the distribution follows a Gaussian model.



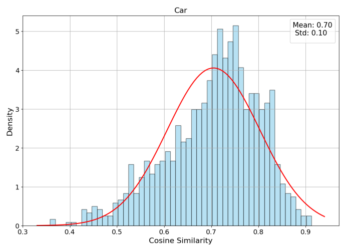
(a) Distribution of cosine similarity for images related to the car class in the image-driven method.



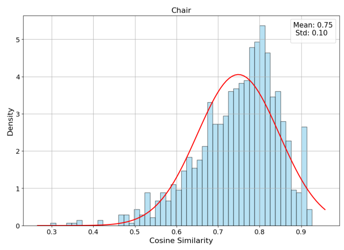
(b) Distribution of cosine similarity for images related to the chair class in the image-driven method.



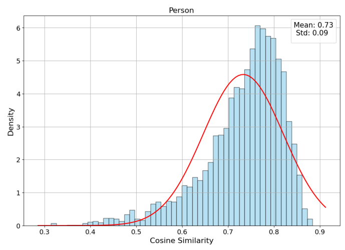
(c) Distribution of cosine similarity for images related to the person class in the image-driven method.



(d) Distribution of cosine similarity for images related to the car class in the text-driven method.



(e) Distribution of cosine similarity for images related to the chair class in the text-driven method.



(f) Distribution of cosine similarity for images related to the person class in the text-driven method.

Figure 7.4: Distribution of cosine similarity for images related to some class in the image-driven method (top) and in the text-driven method (bottom).

Table 7.3 presents the top three selections for the number of standard deviations across both methods.
The notation used in the table combines a type and a value. The type indicates how the dataset and replay images are processed: *Image* denotes that the images are directly used as input for

the CLIP model, while *Text* signifies that the captions of the images are used instead. The accompanying number represents the value of the hyperparameter n, which is used to compute the thresholds.

Image selection strategy	15-5-ov	Δ	10-10-ov	Δ	10-10s-ov	Δ
LfW	73.40%		65.35%		52.29%	
Image_-2	73.40%	+0.00%	66.17%	+0.82%	50.58%	−1.71%
Image_-1	73.55%	+0.15%	65.47%	+0.12%	50.92%	−1.37%
Image_0	73.40%	+0.00%	65.75%	+0.40%	55.31%	+3.02%
Text_-1	73.34%	−0.06%	65.42%	+0.07%	50.79%	−1.50%
Text_0	73.40%	+0.00%	65.82%	+0.47%	52.25%	−0.04%
Text_1	73.28%	−0.12%	65.65%	+0.30%	51.22%	−1.07%
Baseline	72.6%		64.9%		50.8%	
Upper bound	75.4%		75.4%		75.4%	

Table 7.3: Results of the three settings using the Image Selection Strategy Based on CLIP Class Prototypes. This table compares the image selection strategy against the baseline, the upper bound, and the method from the previous paper. The Δ values represent the differences between the obtained results and those of LfW.

From Table 7.3, we observe that using images directly as input to the CLIP model generally yields better performance than using image captions. Specifically, in the 15-5-ov setting, the Image_-1 method achieves the best performance with a 0.15% improvement. In the 10-10-ov setting, the Image_-2 method leads with an increase of 0.82%. Notably, the Image_0 method delivers the highest performance in the 10-10s-ov setting, with a significant improvement of 3.02%. Furthermore, it is the only method in this image selection strategy that consistently outperforms LfW across all three settings. The behaviour of the *mIoU* for classes over time in the 10-10s-ov setting, using the Image_0 method, is illustrated in Figure 7.5.

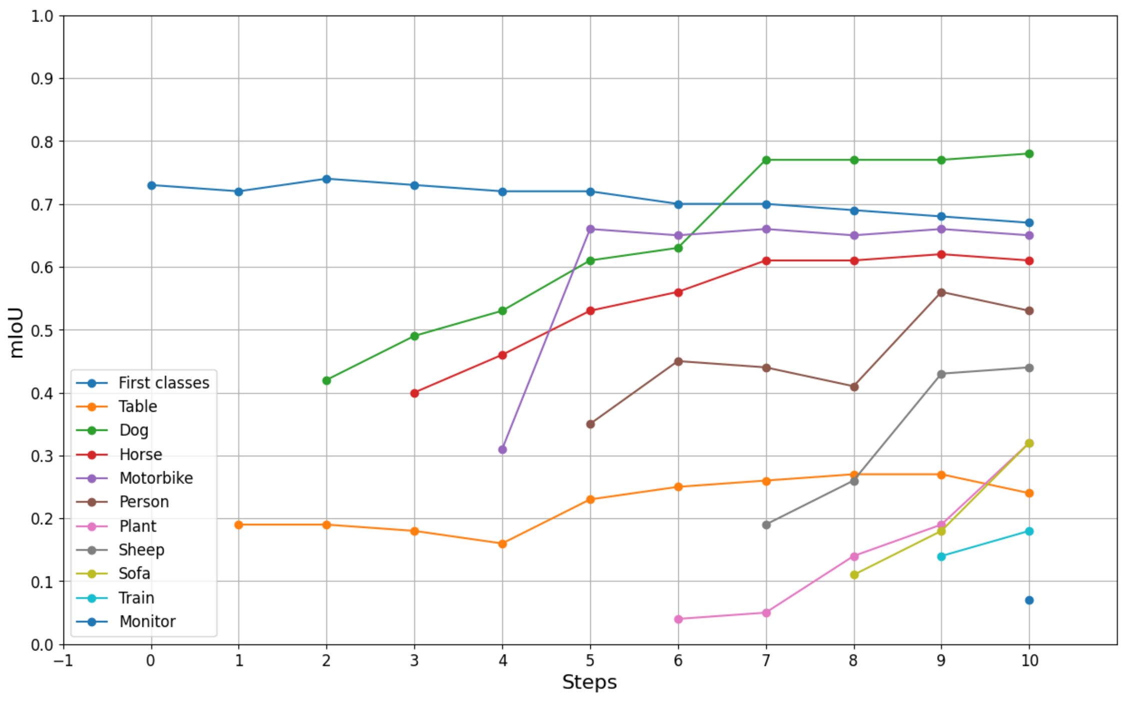


Figure 7.5: Behaviour of the $mIoU$ for classes across the steps in the 10-10s-ov setting using the Image_0 method.

Based on these findings, we conclude that the Image_0 method is the most effective overall within this image selection strategy.

7.5 Results on Image Selection Strategy Based on Reduced CLIP Class Prototypes using PCA

The results analysed in this section pertain to the third image selection strategy developed, as outlined in Algorithms 5.7 and 5.8. In this strategy, the output vectors from the CLIP model are reduced using PCA, with the vector dimensions set to 100 instead of the standard 512 dimensions used by CLIP.

Similar to the previous strategy, the notation used to define the different methods consists of a type, which refers to how the inputs are treated for CLIP, and a number that represents the hyperparameter n used to determine the threshold values. These thresholds are defined as $th_c = \bar{s}_c^{PCA} + n \cdot \sigma_c^{PCA}$.

Table 7.4 presents the top three selections for the number of standard deviations across both methods.

Image selection strategy	15-5-ov	Δ	10-10-ov	Δ	10-10s-ov	Δ
LfW	73.40%		65.35%		52.29%	
PCA_Image_-1.5	73.48%	+0.08%	65.64%	+0.29%	53.21%	+0.92%
PCA_Image_-1	73.46%	+0.06%	65.95%	+0.60%	51.29%	−1.00%
PCA_Image_1	73.54%	+0.14%	65.56%	+0.21%	51.89%	−0.40%
PCA_Text_-1	73.60%	+0.20%	65.62%	+0.27%	54.14%	+1.85%
PCA_Text_0	73.31%	−0.09%	65.85%	+0.50%	50.28%	−2.01%
PCA_Text_1	73.23%	−0.17%	65.44%	+0.09%	51.65%	−0.73%
Baseline	72.6%		64.9%		50.8%	
Upper bound	75.4%		75.4%		75.4%	

Table 7.4: Results of the three settings using the Image Selection Strategy Based on Reduced CLIP Class Prototypes using PCA. This table compares the image selection strategy against the baseline, the upper bound, and the method from the previous paper. The Δ values represent the differences between the obtained results and those of LfW.

From Table 7.4, we observe that both PCA-based methods, particularly PCA_Image_-1.5 and PCA_Text_-1, outperform LfW across all three settings. In the 15-5-ov and 10-10s-ov settings, PCA_Text_-1 achieves the best results, improving upon LfW by 0.20% and 1.85%, respectively. In the 10-10-ov setting, PCA_Image_-1 attains the highest performance, increasing the result by 0.60%.

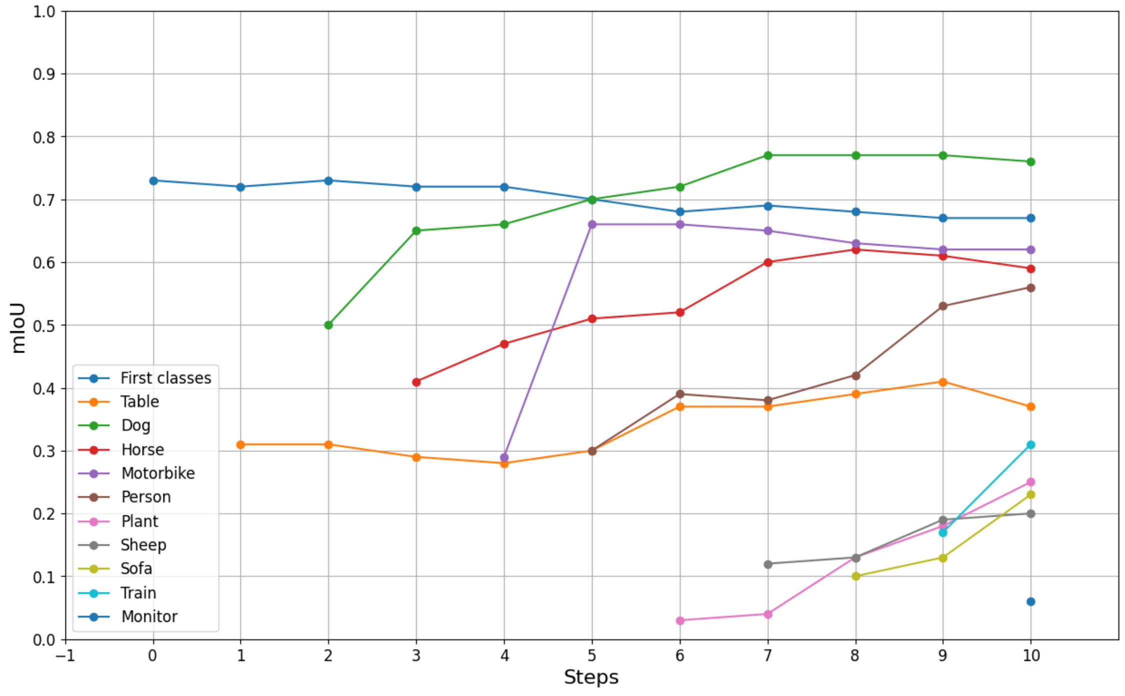


Figure 7.6: Behaviour of the $mIoU$ for classes across the steps in the 10-10s-ov setting using the PCA_Text_-1 method.

These findings indicate that the PCA_Text_-1 method stands out as the most effective method within this image selection strategy.

7.6 Final Performance Comparison of Image Selection Strategies

In this final section, we compare the three developed image selection strategies. A key observation across all strategies is that methods using the captions of the images as input for the CLIP model tend to result in slightly lower performance compared to those using the images directly.

The best configuration for each of the three strategies are summarised in Table 7.5.

Image selection strategy	15-5-ov	Δ	10-10-ov	Δ	10-10s-ov	Δ
LfW	73.40%		65.35%		52.29%	
Image-Image_0.6	73.70%	+0.30%	65.94%	+0.59%	54.62%	+2.33%
Image_0	73.40%	+0.00%	65.75%	+0.40%	55.31%	+3.02%
PCA_Text_-1	73.60%	+0.20%	65.62%	+0.27%	54.14%	+1.85%
Baseline	72.6%		64.9%		50.8%	
Upper bound	75.4%		75.4%		75.4%	

Table 7.5: Results of the best method for each image selection strategy. This table compares the image selection strategy against the baseline, the upper bound, and the method from the previous paper. The Δ values represent the differences between the obtained results and those of LfW.

From the results in Table 7.5, we can observe that the image selection strategy based on the reduced prototype, PCA_Text_-1, shows the lowest improvement compared to LfW. This could be due to two possible reasons: either PCA is not the optimal algorithm for reducing the dimensionality of the feature vectors, or the chosen size of 100 may be too small to capture sufficient information about the classes in the images.
When comparing the first two image selection strategies, it's not straightforward to determine which performs better. Image-Image_0.6 shows superior performance in both the 15-5-ov and 10-10-ov settings, outperforming the other strategy by 0.30% and 0.19%, respectively. However, Image_0 exceeds Image-Image_0.6 by 0.69% in the more challenging 10-10s-ov setting. It is important to highlight that Image_0 is less memory-intensive than Image-Image_0.6. While both strategies utilise captions for queries, Image_0 only requires the storage of a single feature

vector per class, whereas Image-Image_0.6 necessitates storing a feature vector for each available image.Furthermore, Image_0 performs better in the setting considered to be the most challenging and closest to real-world applications.

8 Conclusion

In this thesis, I developed three novel image selection strategies within the framework of a web-based rehearsal strategy to address the challenge of catastrophic forgetting in the continual learning task of semantic segmentation. Building upon a baseline architecture that retrieves replay images using dataset images as queries, my objective was to enhance the existing image selection methods to create a more efficient strategy that retains only those images containing meaningful information. These image selection strategies leverage feature vectors extracted from a visual language model, assessing the relevance of replay images by comparing their feature vectors to those of the dataset query images or to a generic feature vectors that encapsulate class-specific information.

I evaluated the proposed strategies on the Pascal VOC 2012 dataset, employing three different experimental settings with increasing levels of complexity. In all settings, pixel-level annotations were available for the images in the initial step, while in subsequent steps, only image-level annotations were provided for the newly introduced classes. The only variation among the settings was the way the dataset's 20 classes were introduced incrementally.

The results demonstrate that the developed image selection strategies outperform existing methods, achieving state-of-the-art performance across all tested settings. The most substantial improvements were observed in the most challenging and realistic scenario, underscoring the effectiveness of the proposed strategies. These findings highlight the viability of web-based rehearsal strategies for weakly-supervised incremental learning in semantic segmentation.

These results underscore the importance of ongoing research into solutions within the field of web-based rehearsal strategies, focusing on developing new techniques for retrieving and selecting replay images. In particular, given the slightly lower performance of the methods that utilized captions, a potential improvement for the image selection strategy could involve employing a more sophisticated captioning model. Additionally, exploring alternative techniques beyond PCA for reducing the dimensionality of the feature vector may yield further enhancements.
Additionally, the image selection strategies developed in this thesis can be applied beyond web-based replay. An intriguing alternative for future work would be to investigate their application within the framework of generative rehearsal strategies in semantic segmentation tasks related to continual learning. Such explorations could further expand the utility and impact of these image selection techniques in various contexts.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," 2016. [Online]. Available: https://arxiv.org/abs/1511.00561

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[4] U. Michieli, M. Toldo, and P. Zanuttigh, "Chapter 8 - domain adaptation and continual learning in semantic segmentation," in *Advanced Methods and Deep Learning in Computer Vision*, ser. Computer Vision and Pattern Recognition, E. Davies and M. A. Turk, Eds. Academic Press, 2022, pp. 275–303. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128221099000175

[5] A. Maracani, U. Michieli, M. Toldo, and P. Zanuttigh, "Recall: Replay-based continual learning in semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 7026–7035.

[6] C. Liu, G. Rizzoli, F. Barbato, U. Michieli, Y. Niu, and P. Zanuttigh, "Recall+: Adversarial web-based replay for continual learning in semantic segmentation," *arXiv preprint arXiv:2309.10479*, 2023.

[7] C. Liu, G. Rizzoli, P. Zanuttigh, F. Li, and Y. Niu, "Learning from the web: Language drives weakly-supervised incremental learning for semantic segmentation," *arXiv preprint arXiv:2407.13363*, 2024.

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[9] K. LearnOpenCV, "Intersection over union (iou) in object detection segmentation," 2022. [Online]. Available: https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/

[10] K. O'Shea, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[12] L.-C. Chen, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] U. Michieli and P. Zanuttigh, "Incremental learning techniques for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision workshops*, 2019, pp. 0–0.

[15] ——, "Knowledge distillation for incremental learning in semantic segmentation," *Computer Vision and Image Understanding*, vol. 205, p. 103167, 2021.

[16] F. Cermelli, M. Mancini, S. R. Bulo, E. Ricci, and B. Caputo, "Modeling the background for incremental learning in semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9233–9242.

[17] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3400–3409.

[18] U. Michieli and P. Zanuttigh, "Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 1114–1124.

[19] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[20] G. Hinton, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[21] F. Ozdemir and O. Goksel, "Extending pretrained segmentation networks with additional anatomical structures," *International journal of computer assisted radiology and surgery*, vol. 14, pp. 1187–1195, 2019.

[22] M. H. Phan, S. L. Phung, L. Tran-Thanh, A. Bouzerdoum *et al.*, "Class similarity weighted knowledge distillation for continual semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 866–16 875.

[23] G. Nguyen, S. Chen, T. Do, T. J. Jun, H.-J. Choi, and D. Kim, "Dissecting catastrophic forgetting in continual learning by deep visualization," *arXiv preprint arXiv:2001.01578*, 2020.

[24] D. Goswami, R. Schuster, J. van de Weijer, and D. Stricker, "Attribution-aware weight transfer: A warm-start initialization for class-incremental semantic segmentation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3195–3204.

[25] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *Advances in neural information processing systems*, vol. 30, 2017.

[26] F. Cermelli, D. Fontanel, A. Tavera, M. Ciccone, and B. Caputo, "Incremental learning in semantic segmentation from image labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4371–4381.

[27] M. Lukasik, S. Bhojanapalli, A. Menon, and S. Kumar, "Does label smoothing mitigate label noise?" in *International Conference on Machine Learning*. PMLR, 2020, pp. 6448–6458.

[28] A. Taylor, M. Marcus, and B. Santorini, “The penn treebank: an overview,” *Treebanks: Building and using parsed corpora*, pp. 5–22, 2003.

[29] C. Fellbaum, “Wordnet: An electronic lexical database,” *MIT Press google schola*, vol. 2, pp. 678–686, 1998.

[30] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. L. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. a. Bińkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan, “Flamingo: a visual language model for few-shot learning,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 23 716–23 736. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/960a172bc7fbf0177ccccbb411a7d800-Paper-Conference.pdf

[31] A. Awadalla, I. Gao, J. Gardner, J. Hessel, Y. Hanafy, W. Zhu, K. Marathe, Y. Bitton, S. Gadre, S. Sagawa, J. Jitsev, S. Kornblith, P. W. Koh, G. Ilharco, M. Wortsman, and L. Schmidt, “Openflamingo: An open-source framework for training large autoregressive vision-language models,” 2023. [Online]. Available: https://arxiv.org/abs/2308.01390

[32] J. Shlens, “A tutorial on principal component analysis,” 2014. [Online]. Available: https://arxiv.org/abs/1404.1100