Università degli Studi
di Modena e Reggio Emilia

Department of Engineering "Enzo Ferrari"

PhD Program in
Information and Communication Technologies (ICT)

Cycle XXXVIII

Advanced AI Techniques for Continual and Robust Learning on Structured Data

Supervisor:
Prof. Simone Calderara

Candidate:
Martin Menabue

Industrial Co-Tutor:
Dr. Cosimo Fiorini

Coordinator:
Prof. Luigi Rovati

Doctoral Thesis

Academic Year 2024 - 2025

This PhD thesis was funded by the European Union – Next Generation EU, Mission 4 “Education and Research”, Component 2 “From Research to Business”, Investment 3.3 “Introduction of innovative doctorates that respond to the innovation needs of enterprises and promote the recruitment of researchers by companies”.

Abstract

Artificial Intelligence methods have achieved remarkable success across domains, yet their effective application to dynamic and structured data remains a major challenge. This thesis investigates advanced AI techniques for continual and robust learning in scenarios where data evolve over time and exhibit complex dependencies. The research explores multiple complementary directions aimed at overcoming the limitations of current models in adaptability, stability, and resilience. The first part of the thesis focuses on Continual Learning, addressing the problem of learning from sequential streams of data without catastrophic forgetting. A prompt-learning strategy based on CLIP embeddings is first proposed, enabling the dynamic selection of task-specific prompts and improving performance on downstream tasks. Additionally, a distillation-based approach leveraging Vision Transformers is introduced, in which attention representations are transferred from teacher to student models to enhance knowledge retention and stability across tasks. The second part of the thesis investigates the robustness and security of Deep Learning models. It begins by analyzing adaptive backdoor attacks, highlighting their effectiveness and the significant risks they pose to industrial processes and critical infrastructures. Subsequently, the thesis addresses Federated Learning, a collaborative and distributed paradigm in which structured information naturally emerges from client interactions. A novel defense mechanism against backdoor attacks is proposed, exploiting the spectral properties of local data representations to identify and mitigate malicious participants through data synthesis and representation alignment. Overall, this work contributes to the development of AI models capable of continual adaptation, secure collaboration, and effective exploitation of structured information, with a particular focus on real-world and industrial applications.

Contents

Part I

Continual Learning

Chapter 1

The Forgetting Problem

1.1 Overview

Humans naturally acquire a wide variety of skills throughout their lives, learning in a continuous and adaptive manner as they face ever-changing situations. When previously encountered tasks reappear, people are typically able to recall how they were solved and to apply the strategies that proved effective in the past. In artificial intelligence, however, numerous studies have shown that neural networks do not inherently possess this human-like ability. When a model is trained sequentially on multiple tasks and is later evaluated on an earlier one, its performance often drops dramatically, even if it had previously mastered that task. This phenomenon is known as *catastrophic forgetting* [50, 65].

One of the root causes of this issue lies in the standard training paradigm of neural networks. When a model learns a new task via gradient descent, there is no mechanism that preserves the parameters important for previously learned tasks. As a result, the knowledge encoded in those parameters can be overwritten by the updates required to solve the new task. Separating the parameters used for different tasks can be an effective strategy; however, it does not fully resolve the problem. For instance, the outputs associated with one task may become disproportionately large, overshadowing the outputs of the correct task and ultimately leading to incorrect predictions. Moreover, sharing a subset of parameters across different tasks can be advantageous and memory-efficient, especially when those tasks exhibit similar characteristics.

It is therefore clear that addressing catastrophic forgetting is far from trivial, and a substantial body of research has explored strategies to mitigate or alleviate the effects of this phenomenon. This research area, known as ***Continual Learning*** (**CL**), focuses on designing neural networks capable

of learning from a continuous stream of data (a property called *plasticity*) without forgetting previously acquired knowledge (a property named *stability*). Continual Learning is particularly relevant for companies with multiple branches or offices, either within the same country or across different countries. In such scenarios, it is often desirable to train a neural network sequentially on the data collected at each location. A key challenge in this context is to ensure that the model retains knowledge acquired from previous sites while effectively learning from new data. By mitigating catastrophic forgetting, Continual Learning enables organizations to leverage knowledge accumulated over time, improving overall model performance and consistency across diverse datasets.

Most recent CL studies concentrate on image-classification scenarios, where the goal is to predict the class of an object given an input image. Moreover, the CL setting is typically categorized into different configurations depending on the problem assumptions:

- **Task-Incremental Learning** (**Task-IL**) → Each task contains a distinct set of classes, and during inference the neural network is informed of the task identity associated with the input image.
- **Domain-Incremental Learning** (**Domain-IL**) → All tasks share the same class set, but differ in the underlying data-generating distribution.
- **Class-Incremental Learning** (**Class-IL**) → Each task introduces new classes, and during inference the neural network is *not* given access to the task identity of the input image.

The Task-IL setting is the simplest, as knowing the task identity allows the model to activate only the subset of parameters assigned to that task while disabling the rest, thereby preventing interference. The Domain-IL setting requires more sophisticated mechanisms to preserve class consistency while adapting to distribution shifts. Finally, the Class-IL setting is the most challenging, and it is precisely the setting on which my research has focused.

The following sections provide an overview of several approaches that have been proposed over the years to address the forgetting problem.

1.2 Early Continual Learning approaches

Traditionally, Continual Learning methods are organized into the following categories:

- **Rehearsal-based** → The key idea is to store a subset of examples encountered during training and then replay them alongside samples from the current task.
- **Regularization-based** → Additional terms are incorporated into the loss function to mitigate forgetting in various ways.
- **Knowledge distillation-based** → The goal is to align the representations of the model in its current state with those of another model or with its own previous states.

Rehearsal-based methods. From the beginning, rehearsal-based approaches have demonstrated remarkable effectiveness. **Experience Replay** (**ER**) [67] is a straightforward technique that revisits previously seen examples alongside samples from the current task. However, in scenarios involving datasets with millions of images, storing all past examples is practically infeasible. Typically, it is assumed that the number of examples that can be retained is limited, and intuitively, the larger the memory, the better the performance. The presence of a memory constraint requires strategies to determine which examples to store: not all samples are equally informative or important, and storing certain examples may be more beneficial than others. For instance, **Dark Experience Replay** (DER) [11] employs a *reservoir sampling* strategy, allowing examples from the current task stream to be selected such that each has an equal probability of being retained, even without knowing the total stream size in advance. Additionally, DER performs replay of the neural network's logits (i.e., its outputs), a strategy that significantly improves model performance compared to plain experience replay.

Regularization-based methods. These methods address the stability-plasticity dilemma by introducing a penalty term to the objective function, thereby constraining the optimization trajectory to preserve representations learned from previous tasks. The most prominent family of approaches within this category focuses on *parameter regularization* (or *weight consolidation*). The seminal work of **Elastic Weight Consolidation** (**EWC**) [37] mitigates catastrophic forgetting by selectively slowing down learning on the weights that are critical for past tasks. EWC estimates the importance of each parameter using the diagonal of the Fisher Information Matrix, approximating the posterior distribution of the weights. Following EWC, several variations were proposed to refine the estimation of parameter importance. For instance, **Synaptic Intelligence** (**SI**) [91] accumulates the contribution of each parameter to the change in loss over the entire training trajectory,

allowing for an online update of importance weights without requiring access to the data of previous tasks at the end of training. Similarly, **Memory Aware Synapses** (**MAS**) [3] computes importance based on the sensitivity of the learned output function to perturbations in the parameters.

Knowledge distillation methods. Parallel to parameter-based constraints, *functional regularization* methods, often based on Knowledge Distillation (KD), were also widely explored. A pioneering method in this sub-domain is **Learning without Forgetting** (**LwF**) [45]. Unlike EWC, which constrains the weights, LwF preserves the input-output mapping of the old tasks. It employs the old model (the teacher) to generate soft targets for the current data, forcing the updated model (the student) to mimic the behavior of the previous model on the new data, thereby retaining the knowledge of past classes without storing old samples. This is achieved through a distillation loss that forces the student's outputs on the old heads to match the soft targets generated by the frozen teacher. While LwF demonstrated that forgetting could be mitigated without storing past data, its performance degrades significantly in Class-IL settings due to the lack of old class samples. To overcome this, **iCaRL** (**Incremental Classifier and Representation Learning**) [66] combined knowledge distillation with a rehearsal buffer. iCaRL stores a small subset of exemplars to represent previous class distributions. During training, the distillation loss is applied to the sigmoid outputs of the network, ensuring that the feature representation remains discriminative for past classes while adapting to new ones.

1.3 The advent of ViTs

With the increasing adoption of Vision Transformers (ViTs) across various Deep Learning tasks, researchers in Continual Learning have also explored methods to leverage the remarkable potential of these models. From early on, ViTs demonstrated strong performance on standard classification benchmarks, which has led most approaches to start from pre-trained ViT models. The main challenge thus becomes designing effective strategies to adapt these pre-trained ViTs to downstream tasks while mitigating forgetting.

One of the first methods proposed, **L2P** [85], employs a prompt learning approach. Specifically, L2P defines a pool of *prompts*, which are trainable token sequences that are pre-concatenated to the input tokens of the ViT. By training only the prompts and leaving the rest of the ViT unchanged, the model can be efficiently adapted to new tasks.

A key component of this approach is the prompt selection mechanism.

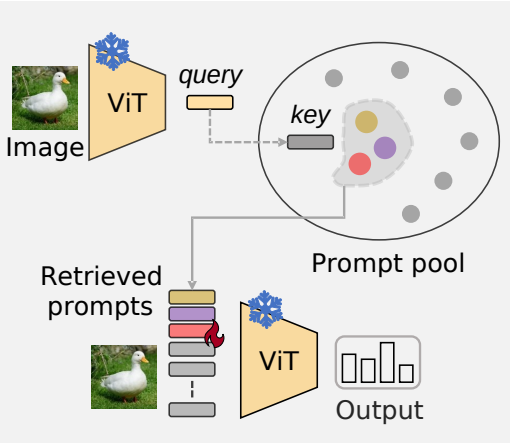


Figure 1.1: An illustration of the prompt retrieval mechanism used by L2P.

L2P uses a retrieval-based strategy (shown in Figure 1.1: a representation of an input image is extracted from a pre-trained ViT, typically the CLS token from the last block, which is referred to as the *query*. Since the ViT is pre-trained, the query encodes meaningful information about the input image. This query is then compared against a pool of *keys*, which are also learnable tokens. Each key is associated with a different prompt, which means that the number of prompts is equal to the number of keys. Various similarity measures can be used to compare the query and keys; L2P employs cosine similarity, as defined in Definition 1.3.1.

Definition 1.3.1: Cosine similarity

Given two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, the cosine similarity can be expressed as:

$$\cos\left(\mathbf{a}, \mathbf{b}\right) = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{a}\|_2 \, \|\mathbf{b}\|_2}. \tag{1.1}$$

The prompt corresponding to the key with the highest similarity to the query is selected and pre-concatenated to the ViT input tokens.

Building upon the success of L2P, **DualPrompt** [84] introduces a more structured approach to prompt learning by explicitly decoupling the learned knowledge into two distinct types: *task-invariant* and *task-specific* instructions. This method addresses a limitation in L2P, where a single prompt pool effectively mixes both general and specific features, potentially leading to suboptimal interference. DualPrompt attaches two complementary sets of learnable prompts to the frozen ViT backbone: (i) General Prompts (G-Prompt), which are designed to encode *task-invariant* knowledge that is

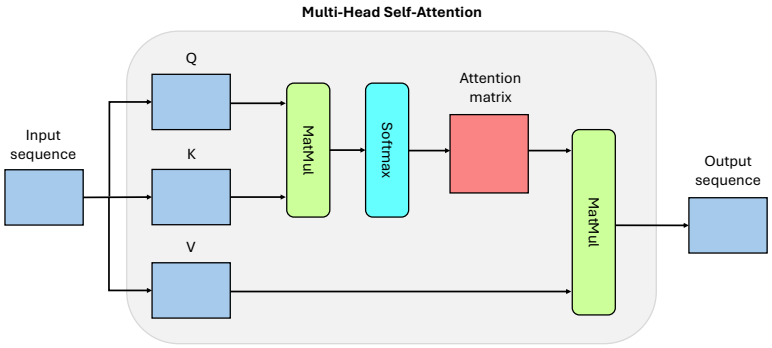
shared across all tasks. They are typically inserted into the shallower layers of the ViT. Since these features are universally applicable, G-Prompts are shared and updated throughout the entire continual learning process. (ii) Expert Prompts (E-Prompt), which capture *task-specific* knowledge. They are maintained in a pool similar to L2P and are inserted into the deeper layers of the ViT. A query-key retrieval mechanism, similar to L2P, selects the most relevant Expert Prompts for a given input image. This allows the model to adapt to specific tasks without interfering with the shared general knowledge.

In addition, DualPrompt is the first CL work to clearly distinguish two prompting strategies that later inspired a series of subsequent methods:

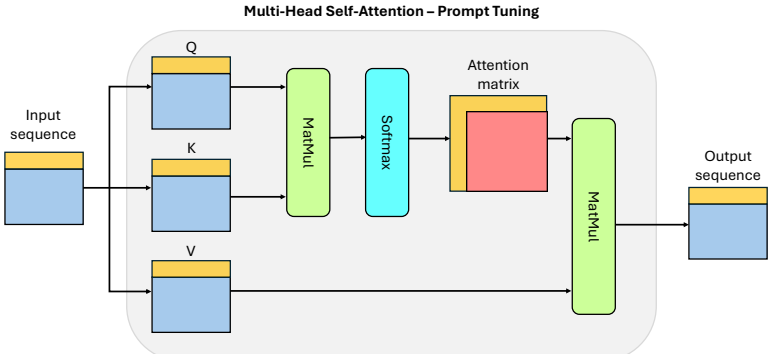
- **Prompt Tuning**: prompt tokens are prepended directly to the input token sequence of a ViT block.
- **Prefix Tuning**: each prompt is divided into two learnable components, $\mathbf{p}_k$ and $\mathbf{p}_v$. The former is prepended to the keys of the MSA layer, and the latter to the values, while the queries remain unchanged.

In the first case, the input sequence length of the ViT increases because additional tokens are introduced. As a result, the prompted ViT incurs a slight computational overhead compared to the original model. In contrast, prefix tuning preserves the original sequence length; the modifications occur only within the Multi-Head Self-Attention (MSA) operation. Specifically, the matrix product between queries and (augmented) keys produces an attention map with additional positions. This map is then multiplied by the (similarly augmented) values, ultimately yielding an output sequence whose length matches that of the original ViT. A small computational overhead is still present, but is significantly lower than the Prompt Tuning strategy. See Figure 1.2 for the visual representations of the original MSA operation and the two prompting strategies.

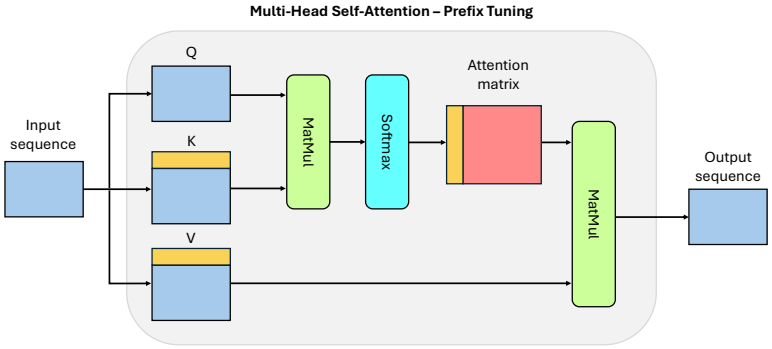
A subsequent prompting-based method is **CODA-Prompt** [71], which substantially redefines the prompt-retrieval mechanism. Specifically, CODA-Prompt constructs a pool of *prompt components* for each task. As in L2P and DualPrompt, a frozen and pretrained ViT is used to extract the query, and similarity scores with a set of learnable keys are computed. The key innovation lies in the selection step: instead of choosing the prompt associated with the highest similarity, the similarity scores are treated as weights to compute a weighted sum of the available prompt components. The resulting prompt is then applied using the Prefix Tuning strategy to adapt a second ViT to the downstream task. To mitigate forgetting, the prompt components associated with a task are frozen once training on that task is completed. The main



(a)



(b)



(c)

Figure 1.2: Representations of the MSA operation without/with prompting.

advantage of CODA-Prompt is that, in subsequent tasks, the weighted-sum mechanism allows the resulting prompt to incorporate useful information from earlier tasks, often leading to improved performance.

Not all approaches have used prompting strategies with Vision Transform-

ers. For instance, **SLCA** [92] performs fine-tuning on the entire pre-trained ViT, but follows a carefully designed strategy: the feature extracting part of the ViT, i.e., all layers before the classifier, is fine-tuned using a very low learning rate to adapt it to the current task without deviating significantly from the pre-trained weights. In contrast, the classifier is fine-tuned with a higher learning rate to allow more rapid adaptation. Furthermore, a key novelty of SLCA lies in its classifier alignment phase: at the end of a task, SLCA computes statistics from the features extracted by the network and uses them to estimate a Gaussian probability distribution. This distribution is then employed to generate new representations, which can subsequently be used to realign the classification heads, thereby mitigating interference between them.

1.4 CLIP-based strategies for Continual Learning

The advent of Vision-Language Models (VLMs), particularly **CLIP** (Contrastive Language-Image Pre-training) [62], has introduced a new paradigm in Continual Learning by enabling methods to leverage rich semantic associations rather than relying solely on visual features. **AttriCLIP** [81] exploits the zero-shot capabilities of a frozen CLIP backbone to function as a non-incremental learner. Instead of learning class-specific weights, AttriCLIP constructs an "attribute word bank" and learns to select and tune relevant textual attribute prompts that describe the visual concepts. By matching input images to these learnable text attributes in the shared CLIP embedding space, it achieves strong performance without a rehearsal buffer, as the shared semantic attributes bridge the gap between disjoint tasks.

Concurrently, **PromptFusion** [15] addresses the stability-plasticity dilemma by explicitly decoupling these two objectives into separate modules built upon a pre-trained backbone. It comprises a *Stabilizer* module, instantiated using **CoOp** (Context Optimization for CLIP) [93], which freezes learned prompts from previous tasks to preserve old knowledge (stability), and a *Booster* module, typically based on **Visual Prompt Tuning** [34], which maintains a shared set of learnable parameters to rapidly adapt to new data (plasticity). By fusing the predictions from the stable, CLIP-guided text-visual alignment of the Stabilizer and the plastic visual adaptation of the Booster, PromptFusion effectively balances long-term retention with short-term adaptation.

Chapter 2

STAR-Prompt: rethinking prompt retrieval

2.1 Motivation

As the starting point of our research, we investigated whether existing Continual Learning methods, despite reporting impressive performance, exhibit weaknesses that might hinder their effectiveness in practice. Given the recent dominance of prompting-based strategies with Vision Transformers (ViTs), our analysis focused primarily on this family of approaches.

For illustration, consider two tasks, A and B, on which a given model is trained sequentially. We immediately observed a fundamental limitation of L2P: when training on task B, the query extracted from an input image often shows higher similarity to keys associated with task A, causing some prompts from task A to be selected and fine-tuned. This typically happens because samples from task B, even if belonging to different classes, may share visual characteristics with samples from task A. As a result, the queries of task B tend to be closer to the previously learned keys of task A than to the randomly initialized keys of task B. However, fine-tuning prompts of earlier tasks inevitably leads to stronger forgetting of their knowledge, ultimately degrading overall performance.

This issue is partially alleviated by later methods such as DualPrompt and CODA-Prompt. In these frameworks, each task is assigned its own dedicated prompts, and both prompts and keys from previous tasks are frozen during the training of new tasks. This prevents overwriting previously acquired knowledge. Nonetheless, upon closer inspection, we identified another significant problem. Returning to our two-task example (A and B), after incremental training we frequently observe the following undesirable situa-

tions:

1. When testing on task A, some queries are more similar to keys from task B than to keys from task A.

2. When testing on task B, some queries are more similar to keys from task A than to keys from task B.

These effects are visually illustrated in Figure 2.1.

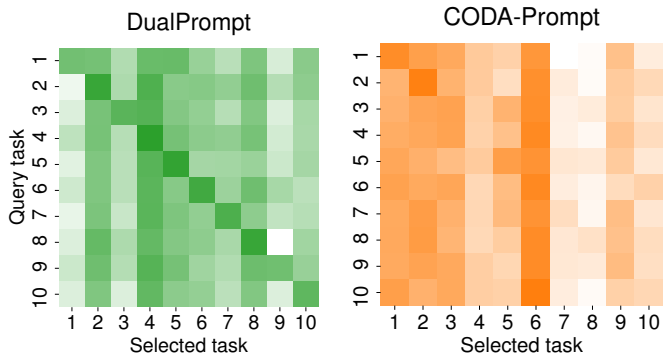


Figure 2.1: Prompt retrieval on ImageNet-R [28], performed at the end of the incremental training. In each confusion matrix, the y axis represents the task of the query sample, while the x axis shows the task of the corresponding selected key.

The reason behind scenario (1) is straightforward: when training on task B, the keys and prompts of task A are frozen, and no examples from task A are available. Hence, there is no way to make queries from task A dissimilar to the keys of task B. Scenario (2), however, is more counterintuitive, and its explanation differs between DualPrompt and CODA-Prompt:

- In DualPrompt, during each task, only that task's E-Prompts are trained. Yet, there is no constraint ensuring that, at test time, queries from task B are closer to keys from task B rather than keys from task A.

- CODA-Prompt computes a weighted combination of prompt components from all previously observed tasks, while freezing keys and prompts from earlier tasks. However, the model may still assign higher weights to past prompt components than to those of the current task. When this happens, gradients flowing through the current prompt components are reduced, slowing down training and making convergence harder to achieve.

Overall, these scenarios reveal a persistent *interference* between keys and prompts belonging to different tasks. This interference negatively impacts performance: in L2P it exacerbates forgetting, while in DualPrompt and CODA-Prompt it degrades final accuracy. These observations motivated us to design a new solution addressing these shortcomings. The resulting approach, which we call **STAR-Prompt** [53], rethinks the prompt-retrieval mechanism with the goal of mitigating cross-task interference. The full method is described in detail in the following sections. The paper of STAR-Prompt was published at ECCV 2024, and the code is available at `https://github.com/aimagelab/mammoth`.

2.2 Key Ideas

Our method, **STAR-Prompt**, aims to improve existing prompting-based approaches for Continual Learning along several dimensions. A first observation is that, in methods such as L2P, DualPrompt, and CODA-Prompt, the keys are simply learnable vectors. Once training is completed, it becomes extremely difficult to determine what these vectors actually represent. As a result, one cannot clearly explain why certain queries align more closely with particular keys than with others. In contrast, if the keys carried an interpretable semantic meaning, we could better understand why specific prompts are retrieved. Thus, our initial goal was to design a mechanism for obtaining keys with explicit and meaningful semantics.

To this end, we leverage the CLIP model. CLIP consists of two neural networks: a *visual encoder*, which extracts visual features from an input image, and a *text encoder*, which extracts textual features from natural-language prompts. Two projection layers map the outputs of the two encoders into a shared embedding space. CLIP is trained to pull visual features closer to the textual features of prompts that describe the input image, while pushing apart visual and textual features that are semantically unrelated. After training, this shared embedding space enables straightforward comparison between images and text.

This suggests an elegant way to endow our prompts' keys with semantic meaning: we generate keys using the textual features extracted by CLIP's text encoder, applied to prompt templates such as "a photo of CLS", where CLS is replaced by the name of a class (e.g., "duck", "dog", etc.). These keys, which we refer to as *class prototypes*, encode characteristic information about the classes. To extract the query, we simply use the visual encoder of the pre-trained CLIP model: it produces a feature vector for the input image, which serves as our query. In the shared embedding space, this vector should

lie close to the class prototype corresponding to the correct class, essentially exploiting CLIP's zero-shot capabilities.

While conceptually appealing, this approach encounters two challenges:

1. If two classes are visually similar, queries may mistakenly align with the prototype of a similar but incorrect class. More fine-grained keys would help mitigate this issue.

2. If the downstream task differs significantly from CLIP's pretraining dataset, the resulting prototypes may not be sufficiently informative.

To address these limitations, inspired by AttriCLIP and Prompt-Fusion, we adopt the **CoOp** method [93]. Instead of using fixed prompt templates, CoOp learns trainable textual prompts. Each prompt is composed of a start-of-sequence (SOS) token, a set of learnable vectors, the class-specific token CLS, and an end-of-sequence (EOS) token. In its class-specific configuration, CoOp instantiates one learnable prompt per class. Training proceeds by freezing the CLIP encoders and updating only the learnable prompts: for each input image, the goal is to pull the visual feature vector toward the textual feature vector of the correct class prompt while pushing it away from those of incorrect classes. The original CoOp paper shows that this strategy outperforms CLIP's zero-shot performance.

In our setting, an additional challenge arises: training occurs over a sequence of tasks. Nevertheless, we can simply assign one trainable textual prompt to each class and, for each task, update only the prompts corresponding to the classes within that task, while leaving all others frozen. However, as an attentive reader may notice, without further measures, our keys could suffer from the same types of interference issues previously identified in DualPrompt and CODA-Prompt. We observed that this interference is largely caused by the classifier, which may assign consistently higher scores to classes from one task than to those from another. In principle, this could be mitigated by "recalibrating" the classifier; however, we lack access to the examples from previous tasks. This is where we draw inspiration from the alignment mechanism used in SLCA [92]. Specifically, at the end of training on a given task, we extract the feature vectors of all examples in its training set (which is feasible because we have not yet moved on to the next task). SLCA assumes that these feature vectors follow a multivariate Gaussian distribution. Although this may not hold exactly in practice, they found it to be a reasonable approximation empirically. However, we argue that other probabilistic models could provide a better approximation: for this reason, in STAR-Prompt, we employ a Gaussian Mixture Model (GMM). The parameters of this distribution can be computed directly from the extracted features.

We estimate one GM per class of the current task. Storing these distribution parameters is acceptable within the constraints of Continual Learning, since we do not retain any raw data (thus avoiding issues related to privacy or data-storage limitations).

This probabilistic modeling grants us a powerful capability: we can sample new synthetic feature vectors from the estimated distributions, both for the current task and for previous ones. By feeding these sampled features into the classifier and enforcing that it predicts the correct class, we enable the classifier to "see" examples from both current and past tasks simultaneously. This allows it to recalibrate its decision boundaries to produce balanced predictions across tasks, something that incremental training alone cannot achieve.

This process also provides the class prototypes described earlier, which can directly be used for classification. Given an input example of an unknown class, we extract its CLIP visual embedding and compare it against the prototypes; the class associated with the most similar prototype becomes the prediction. While this approach —- which we call *Incremental CoOp* - — is already highly effective, we found experimentally that it may fail in scenarios where the downstream task contains classes that are very similar. In such cases, the visual encoder may produce nearly indistinguishable queries for those classes. Consequently, the similarity between a query and an incorrect class prototype may exceed that of the correct one, leading to misclassification.

For this reason, we opt to use a dedicated neural classifier, such as the standard ViT employed in L2P, DualPrompt, and CODA-Prompt. This introduces the second phase of STAR-Prompt. The idea is to use the class prototypes as keys within a prompt retrieval mechanism, similar to that of prior prompting-based methods. Concretely, we extract the query using CLIP's visual encoder and compute its similarities with all keys (i.e., the class prototypes). This yields a similarity score for each class. Following the approach of CODA-Prompt, we treat these similarity scores as weights and compute a weighted sum of the class-specific prompts. This produces a single final prompt, which is then fed into the ViT.

In this design, even if an incorrect class obtains a higher similarity score than the correct one, the weighted sum ensures that both the incorrect and correct prompts (the latter still having a non-negligible similarity score) contribute to the final prompt representation. The ViT can then leverage this combined prompt to infer the correct class. During this phase, the ViT is trained to map the final prompt to the correct class via cross-entropy. However, the ViT classifier may also exhibit the same interference issue identified earlier. For this reason, we apply an alignment step, analogous to the one

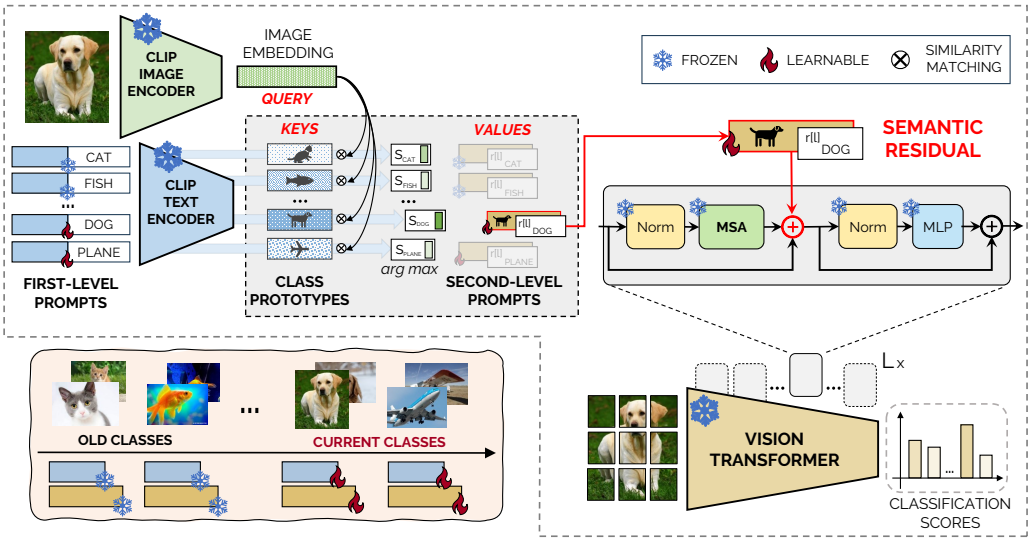


Figure 2.2: The architecture of **STAR-Prompt**. The bottom left box illustrates our CL setting, in which first- and second-level prompts of the old tasks are frozen.

used in the first phase, after completing training on each task.

With the high-level functioning of STAR-Prompt now outlined, the next section provides a formal description of the entire procedure.

2.3 Detailed explanation

Problem setting. Continual Learning considers a scenario where a model encounters a sequence of T tasks. Each task D_t, for $t = 1, \dots, T$, consists of input–label pairs (x_i, y_i) with $i = 1, \dots, |D_t|$. Throughout this work, we focus on the Class-IL paradigm, where the class set $\mathcal{Y}_t$ associated with task D_t does not overlap with $\mathcal{Y}_{t'}$ for any $t' \neq t$, and no information about the task index is available at test time. For ease of exposition, we further assume that every task introduces the same number of classes, i.e., $|\mathcal{Y}_1| = \cdots = |\mathcal{Y}_T| = N$. A crucial aspect of this setting is that, once the model proceeds to task D_t, the data from previously encountered tasks $D_{1:t-1}$ cannot be accessed.

Our approach, illustrated in fig. 2.2, builds upon two pre-trained components:

- the CLIP framework [62], comprising the visual encoder $E_{vis}(\cdot)$ and the textual encoder $E_{txt}(\cdot)$;
- a Vision Transformer (ViT) [38] pre-trained on ImageNet, denoted as

$f(\cdot)$.

As explained before, we employ a *two-level* strategy: the first level learns the prompts of the CLIP text encoder, which then produces class prototypes. The second level leverages these prototypes as keys for a prompt retrieval strategy. The produced final prompt is used to finetune the ViT on the downstream task.

First level. In detail, the first-level prompting step is inspired by CoOp and modifies the text encoder $E_{txt}(\cdot)$ by appending a learnable vector, specific to each class, to the input sequence. Therefore, each class $y_c \in \mathcal{Y}_t$ is assigned a learnable prompt p_c. This prompt is represented by a single learnable token, which is concatenated with the initial word embedding of the class name, denoted as [CLS-NAME]:

$$\boldsymbol{w}_c = E_{txt}([p_c; [\text{CLS-NAME}]]). \tag{2.1}$$

For example, if y_c corresponds to the class '*dog*', then [CLS-NAME] is the embedding of the word '*dog*'. The combined embeddings are then passed through $E_{txt}(\cdot)$ to produce the final textual representation $\boldsymbol{w}_c \in \mathbb{R}^d$.

Unlike CoOp, however, we do not use the resulting textual class representations directly for classification. Instead, these vectors are used as **class-specific keys** for the second level of STAR-Prompt.

Second level. The keys are used to retrieve a second set of learnable prompts $Q_c \in \mathbb{R}^{L \times d'}$, designed to adapt each layer of the ImageNet pretrained Vision Transformer $f(\cdot)$. Concretely, for the l-th layer of $f(\cdot)$ with $l < L$ and embedding dimension d', the vector $Q_c[l] \in \mathbb{R}^{d'}$ serves as the second-level class-specific prompt for adapting that particular layer.

In the second-level prompting stage, we make use of the CLIP multimodal embedding space. For an input image, let $\boldsymbol{z} = E_{vis}(x)$ denote its visual embedding, serving as the query. We then calculate the cosine similarity between $\boldsymbol{z}$ and each class-specific key $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{Nt}$ obtained from the first-level prompting stage: $\text{sim}_c = \langle \boldsymbol{z}, \boldsymbol{w}_c \rangle$. Each key corresponds to one of the Nt classes encountered up to the current task.

Next, we construct the final second-level prompt $\mathbf{R} \in \mathbb{R}^{L \times d'}$ by selecting the prompt Q_{c_k} corresponding to the class with the **highest** cosine similarity:

$$\mathbf{R} = \sum_{k=1}^{N_t} \text{sim}_{c_k} \cdot Q_{c_k}, \tag{2.2}$$

Algorithm 1 Training of STAR-Prompt on the current task

Require: Index of the current task D_t, $t \in 1, \ldots, T$, the orthogonality loss weighting coefficient λ, the number of M Gaussian components, the number of E_1 and E_2 training epochs with real and generated samples (respectively), learning rate lr.

First stage → *goal*: learning first-level prompts $p_c|_{c\in\{1,\ldots,N\}}$ for each class $c \in \mathcal{Y}_t$ of the current task t.

for $ep := 1, \ldots, E_1$ **do**
  $\mathcal{L}_{\text{CE}} \leftarrow$ use eqs. (2.4) and (2.5)
  $\mathcal{L}_{\text{OP}} \leftarrow$ use eq. (2.10)
  $p_c \leftarrow p_c - lr \cdot \nabla_{p_c|_{c\in\mathcal{Y}_t}} \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{OP}}$
end for
 # apply Generative Replay
$\mathcal{G}_c \leftarrow \text{EM}(\{E_{vis}(x)\}_{x,y=c\in D_t})\ \forall c \in \mathcal{Y}_t$
for $ep := 1, \ldots, E_2$ **do**
  # create samples from $\mathcal{G}_{c'}|_{c'\in\mathcal{Y}_1,\ldots,\mathcal{Y}_t}$
  $\mathcal{L}^P_{\text{GR}} \leftarrow$ use eq. (2.8)
  $p_c \leftarrow p_c - lr \cdot \nabla_{p_c|_{c\in\mathcal{Y}_t}} \mathcal{L}^P_{\text{GR}}$
end for

Second stage → *goal*: learning second-level prompts Q_c, query weights $A_c\ \forall c \in \mathcal{Y}_t$, and the classifiers $\theta_{t'}|_{t'\in\{1,\ldots,t\}}$.

$\texttt{param} \leftarrow \{Q_c, A_c\}_{c\in\mathcal{Y}_t} \cup \theta_t$
for $ep := 1, \ldots, E_1$ **do**
  $\mathcal{L}_{\text{CE}} \leftarrow$ use eqs. (2.4) and (2.6)
  $\mathcal{L}_{\text{OQ}} \leftarrow$ use eq. (2.11)
  $\texttt{param} \leftarrow \texttt{param} - lr \cdot \nabla_{\texttt{param}} \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{OQ}}$
end for
 # apply Generative Replay
$\mathcal{H}_c \leftarrow \text{EM}(\{f(x_i; \mathbf{R})\}_{x,y=c\in D_t})\ \forall c \in \mathcal{Y}_t$
for $ep := 1, \ldots, E_2$ **do**
  # create samples from $\mathcal{H}_{c'}|_{c'\in\mathcal{Y}_1,\ldots,\mathcal{Y}_t}$
  $\theta_{t'} \leftarrow \theta_{t'} - lr \cdot \nabla_{\theta_{t'}} \mathcal{L}^Q_{\text{GR}}\ \forall\ t' \in \{1, \ldots, t\}$
end for

In eq. (2.2), the factor sim_{c_k} acts as a weight, reflecting the confidence that CLIP assigns to the key $\boldsymbol{w}_{c_k}$.

We explored several different ways to use this second-level prompt with the ViT architecture. Eventually, we discovered that adding this prompt instead of, for example, concatenating it (as done in prompt/prefix tuning) was more effective. Since the prompt incorporates information about the classes related to the subject of the image, and is employed in a sum operation, we call it "***semantic residual***. More specifically, we inject this semantic information into each MLP layer of $f(x)$, thus we have:

$$\boldsymbol{e}' = \text{MSA}(\text{LN}(\boldsymbol{e}_l)) + \boldsymbol{e}_l + \mathbf{R}[l]. \tag{2.3}$$

Here, $\mathbf{R}[l]$ is a d'-dimensional vector applied uniformly to all tokens in the visual sequence. To ensure proper dimensionality for addition, this vector is broadcast across every token in the layer.

In conclusion, the final classification layer of the ViT provides the classification scores. The predicted label is then the label with the highest score.

Main training objective. For each task D_t, we expand the prompts by incorporating the class-specific prompts associated with the new task. Concretely, the first-level prompt set becomes $\mathcal{P}_t := \mathcal{P}_{t-1} \cup \{p_c \mid c \in \mathcal{Y}_t\}$, while

the second-level prompt set is updated as $\mathcal{Q}_t := \mathcal{Q}_{t-1} \cup \{Q_c \mid c \in \mathcal{Y}_t\}$. To mitigate forgetting, all prompts from previous tasks (those in $\mathcal{P}_{t-1}$ and $\mathcal{Q}_{t-1}$) are kept frozen, and only the prompts tied to the current task are optimized. Although both levels could be trained jointly in an end-to-end fashion, we opt for a simpler two-phase training pipeline. In both phases, the primary objective is the standard cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{|D_t|} \sum_{i=1}^{|D_t|} \log p(y_i = c \mid x_i), \tag{2.4}$$

where, during the first phase, the posterior probabilities are computed as:

$$p(y_i = c \mid x_i) = \frac{\exp(\langle \boldsymbol{z}_i, \boldsymbol{w}_c \rangle / \tau)}{\sum_{c' \in \mathcal{Y}_t} \exp(\langle \boldsymbol{z}_i, \boldsymbol{w}_{c'} \rangle / \tau)}, \tag{2.5}$$

with τ denoting CLIP's temperature parameter. In the second phase, the posteriors in eq. (2.4) are instead derived from the final-layer CLS token, $\boldsymbol{e}_L^{\text{CLS}} = f(x_i; \mathbf{R})$, of the Vision Transformer $f(\cdot)$ [38]:

$$p(y_i = c \mid x_i) = g_{\theta_t}(\boldsymbol{e}_L^{\text{CLS}}), \tag{2.6}$$

where $g_{\theta_t}(\cdot)$ denotes the classification head for task t. This head consists of a linear projection followed by a softmax layer. Its parameters θ_t are freshly initialized at the beginning of task t and updated during training, while the classification heads from earlier tasks remain frozen.

Generative replay. As previously mentioned, we employ a generative replay approach after both the first-level and the second-level prompt training. Concretely, at the end of each task, we extract the features of all input samples and we estimate a probabilistic model to represent them. In practice, we found that a Gaussian Mixture Model (GMM) provides a strong approximation of the true distribution of the features.

Formally, in the first stage, for each class c, we fit a GM $\mathcal{G}_c$ to the CLIP visual embeddings $\boldsymbol{z}_i = E_{vis}(x_i)$ of all samples $(x_i, y_i = c) \in D_t$:

$$\mathcal{G}_c = \mathcal{G}(\cdot; \boldsymbol{\phi_c}) = \sum_{m=1}^{M} \phi_m^c \, \mathcal{N}(\cdot; \mu_m^c, \Sigma_m^c). \tag{2.7}$$

The parameter vector $\boldsymbol{\phi_c}$ includes the M mixture weights as well as the mean and covariance of each Gaussian component, and is estimated via Expectation–Maximization (EM). Once learned, $\mathcal{G}_c$ is used in later tasks t' ($t' \geq t$)

to sample $n = 256$ synthetic features $\tilde{\boldsymbol{z}}$ for replay. These generated features allow us to define the generative-replay loss:

$$\mathcal{L}_{\mathrm{GR}}^{P} = -\frac{1}{nNt} \sum_{c=1}^{Nt} \sum_{i=1}^{n} \log p(y_i = c \mid \tilde{\boldsymbol{z}}_i), \tag{2.8}$$

where the posterior probability for a synthetic feature $\tilde{\boldsymbol{z}}_i \sim \mathcal{G}_c$ is given by:

$$p(y_i = c \mid \tilde{\boldsymbol{z}}_i) = \frac{\exp(\langle \tilde{\boldsymbol{z}}_i, \boldsymbol{w}_c \rangle / \tau)}{\sum_{c'=1}^{Nt} \exp(\langle \tilde{\boldsymbol{z}}_i, \boldsymbol{w}_{c'} \rangle / \tau)}. \tag{2.9}$$

Note that the denominator includes all Nt classes observed up to the current task.

In the second training stage, we apply the same idea to the ViT-based representations. Specifically, we use the features $\boldsymbol{e}_L^{\mathrm{CLS}} = f(x_i; \mathbf{R})$ from the final ViT layer to fit another GM, denoted as $\mathcal{H}_c$, following the formulation in eq. (2.7). During rehearsal, we replace $\boldsymbol{e}_L^{\mathrm{CLS}}$ in eq. (2.6) with n synthetic samples drawn from each $\mathcal{H}_c$, and – as in eqs. (2.8) and (2.9) – compute the corresponding generative-replay loss $\mathcal{L}_{\mathrm{GR}}^{Q}$.

Query weighting. In line with CODA-Prompt [72], we learn a class-dependent weighting vector $A_c \in \mathbb{R}^d$ that modulates the contribution of the visual features for class c. Each feature vector $\boldsymbol{z} = E_{vis}(x)$ is multiplied element-wise with A_c, and the similarity term in eq. (2.2) is updated by replacing $\mathrm{sim}_c = \langle \boldsymbol{z}, \boldsymbol{w}_c \rangle$ with $\langle \boldsymbol{z} \odot A_c, \boldsymbol{w}_c \rangle$.

Orthogonality. Following the strategy of CODA-Prompt, we further encourage new prompts to be decorrelated from those learned in earlier tasks. This is achieved by minimizing the following loss terms for the two prompt codebooks:

$$\mathcal{L}_{\mathrm{OP}} = \sum_{c \in \mathcal{Y}_t, c' \in \mathrm{past}(t)} \langle \boldsymbol{p}_{c'}, \boldsymbol{p}_c \rangle, \tag{2.10}$$

$$\mathcal{L}_{\mathrm{OQ}} = \frac{1}{L} \sum_{l=1}^{L} \sum_{c \in \mathcal{Y}_t, c' \in \mathrm{past}(t)} \langle Q_{c'}[l], Q_c[l] \rangle. \tag{2.11}$$

where $\mathrm{past}(t) = \{c' \mid c' \in \mathcal{Y}_{t'},\ t' < t\}$ denotes the set of all previously observed classes.

The entire algorithm of STAR-Prompt is shown in Algorithm 1.

2.4 Experiments

Benchmarks. We assess our method across a variety of datasets, following recent studies on pre-trained Continual Learning methods [84, 72, 92, 15]. First, we evaluate on conventional image benchmarks such as CIFAR-100 and Imagenet-R. To probe robustness to domain shift from the ImageNet pre-training [57, 20], we additionally experiment with datasets that exhibit progressively lower similarity to ImageNet. The tested domains (details in the supplementary material) are:

- **Natural images**: *Split CIFAR-100* [40] and *Split Imagenet-R* [28], containing 100 and 200 classes respectively, each split into 10 tasks.
- **Fine-grained / specialized**: following [92], we use *Split Cars-196* [39] and *Split CUB-200* [77], with classes partitioned into 10 tasks.
- **Aerial**: RGB satellite collections for land-use/land-cover classification: *Split EuroSAT* [26, 27] (organized as 5 binary tasks) and *Split RESISC45* [18] (45 classes split into 9 tasks).
- **Medical**: three settings covering plant and human disease images: *Split CropDiseases* [32] (7 tasks with 5 classes each), *Split ISIC* [19] (6 skin-disease classes across 3 tasks), and *Split ChestX* [83] (chest X-rays organized into 2 tasks of 3 classes each).

Evaluation metrics. We report the **Final Average Accuracy** (FAA), measured after the final task. Each configuration is run three times using different class orderings [92]; we therefore report the mean and standard deviation of FAA.

Implementation details. Training is performed with Adam [36] using a learning rate of 10^{-3}. The number of epochs is dataset-dependent; batch size is set to 16 for Split Imagenet-R and 128 for the remaining datasets. To ensure fairness, competing methods are trained for the same number of epochs and their hyperparameters (including optimizer and learning rate) are tuned to obtain their best performance. Following common practice in Continual Learning, inference is carried out in an *instance-wise* fashion: prompts are selected independently for each sample.

Compared methods. Our comparison focuses on state-of-the-art prompt tuning approaches such as L2P [85], DualPrompt [84], AttriCLIP [81], PromptFusion [15], and CODA-Prompt [72]. We also include full-network fine-tuning baselines such as SLCA [92], rehearsal-based methods like DER++ [11] and GDumb [61], and regularization-based LwF [45]. Consistently with prior

Table 2.1: The Final Avg. Accuracy and std dev. for **natural** and **specialized** domains. † denotes methods fine-tuning the whole model. * highlights rehearsal approaches. We take results (1) from [72], (2) from [81], (3) from [92], (4) from [15]. Due to the absence of a public codebase (at the time of this research work), we could not reproduce the results of PromptFusion on all datasets.

Model	Imagenet-R	CIFAR-100	Cars-196	CUB-200	*Avg.*
Joint (*STAR-Prompt*)	90.03 ±0.33	92.32 ±0.28	89.00 ±0.33	85.64 ±0.30	89.25
Joint (3) † (*ViT-B/16*)	79.60 ±0.87	93.22 ±0.16	80.31 ±0.13	88.00 ±0.34	85.28
Fine-tune † (*ViT-B/16*)	17.21 ±4.72	17.47 ±2.51	9.28 ±0.31	11.36 ±1.43	13.83
k-NN (*ViT-B/16*)	18.93	26.14	11.57	21.44	19.52
Zero-shot CLIP [62]	85.36	73.27	76.46	61.14	74.06
LwF † [45]	19.09 ±5.72	19.68 ±0.90	23.24 ±1.88	16.73 ±4.16	19.69
GDumb * † [61]	44.28 ±0.51	57.92 ±1.67	28.74 ±0.47	61.34 ±0.46	48.07
DER++ * † [11]	56.66 ±0.97	79.77 ±1.14	53.66 ±1.51	74.62 ±0.73	66.18
L2P (3) [85]	66.49 ±0.40	82.76 ±1.17	38.18 ±2.33	62.21 ±1.92	62.41
DualPrompt (3) [84]	68.50 ±0.52	85.56 ±0.33	40.14 ±2.36	66.00 ±0.57	65.05
CODA-Prompt [72]	75.45 ±0.56 (1)	86.25 ±0.74 (1)	31.99 ±3.39	67.30 ±3.19	65.25
AttriCLIP [81]	86.25 ±0.75	81.40 (2)	70.98 ±0.41	50.07 ±1.37	72.18
PromptFusion (4) * [15]	80.70 ±–	87.40 ±–	–	–	–
SLCA (3) † [92]	77.00 ±0.33	**91.53** ±0.28	67.73 ±0.85	**84.71** ±0.40	80.24
STAR-Prompt	**89.83** ±0.04	90.12 ±0.32	**87.62** ±0.20	84.10 ±0.28	**87.92**

works [84, 72, 92], all methods use the same backbone (ViT-B/16) as the main classifier (our notation: $f(\cdot)$), initialized with a fully supervised ImageNet-21K pre-training. Methods that leverage CLIP adopt ViT-L/14 for $E_{vis}(\cdot)$ and $E_{txt}(\cdot)$ as in [81].

Training-free baselines and bounds. We include the zero-shot CLIP classifier [62] (Zero-shot CLIP) that uses handcrafted textual templates (e.g., "A photo of a CLS") and computes predictions via the CLIP posterior in eq. (2.5). Comparing our method to this zero-shot baseline isolates the algorithmic gains beyond CLIP's innate capabilities. We also consider a simple k-NN baseline built on the frozen ImageNet backbone that memorizes latent training features and classifies by nearest neighbors.

For completeness, we report an upper bound obtained by training our method jointly on all tasks (Joint (*STAR-Prompt*)). We also show Joint † (*ViT-B/16*) and Fine-tune † (*ViT-B/16*), where the ViT-B/16 is fine-tuned on all tasks either jointly or incrementally without any anti-forgetting strategy.

Table 2.2: The Final Avg. Accuracy for the **aerial** and **medical** domains. For ease of presentation, we omit the standard deviation.

Model	EuroSAT	RESISC	CropDis.	ISIC	ChestX	*Avg.*
Joint (*STAR-Prompt*)	97.21	96.41	99.19	78.25	45.36	83.28
Joint † (*ViT-B/16*)	98.19	96.88	99.68	88.31	48.92	86.40
Fine-tune † (*ViT-B/16*)	19.91	14.96	13.24	30.30	30.92	21.87
k-NN (*ViT-B/16*)	28.18	24.86	30.74	19.09	11.51	22.88
Zero shot CLIP	54.50	63.15	27.58	29.14	26.27	40.13
LwF †	25.13	15.37	22.31	33.06	32.82	25.74
GDumb * †	90.99	60.07	83.61	61.64	32.33	65.73
DER++ * †	93.08	51.84	92.53	65.68	35.52	67.72
L2P	46.34	63.27	74.68	47.13	32.46	52.78
DualPrompt	71.39	76.21	81.41	49.99	35.70	62.94
CODA-Prompt	63.12	70.46	77.09	44.87	38.62	58.83
AttriCLIP	57.51	66.64	33.21	26.77	28.94	42.61
SLCA †	88.69	85.70	93.80	59.19	39.07	73.29
STAR-Prompt	**93.70**	**92.28**	**94.92**	**66.67**	**41.85**	**77.88**

2.5 Discussion

Comparison with other approaches. As illustrated in Tab. 2.1 and 2.2, STAR-Prompt achieves the strongest overall performance across the evaluated benchmarks. The advantage is especially clear when compared to other CLIP-driven approaches (e.g., AttriCLIP and PromptFusion). In the two fine-grained datasets — Cars-196 and CUB-200— our method exceeds the Zero-shot CLIP baseline by +11.16 and +22.96 points, respectively. These improvements highlight how subtle inter-class distinctions (such as telling a "red headed woodpecker" from a "red bellied woodpecker") are often poorly resolved by the original CLIP features, and thus benefit substantially from the additional flexibility provided by our second-level prompting mechanism. A similar pattern emerges in the aerial and medical benchmarks, where the average margin over Zero-shot CLIP reaches +37.75.

We also observe an average gain of +5.96 points over the strongest competing method (SLCA), despite STAR-Prompt using significantly fewer trainable parameters (see supplementary material). Furthermore, STAR-Prompt performs very close to its Joint upper bound, differing by only −3.59 points on average. This indicates that our approach experiences *minimal performance degradation* compared to a non-incremental training regime, even in domains such as aerial imagery and medical data where adaptability is crucial.

Finally, we note that most prompting-based strategies underperform in these more challenging domains. Replay-oriented methods generally perform better because they more aggressively adapt the backbone to domain shifts.

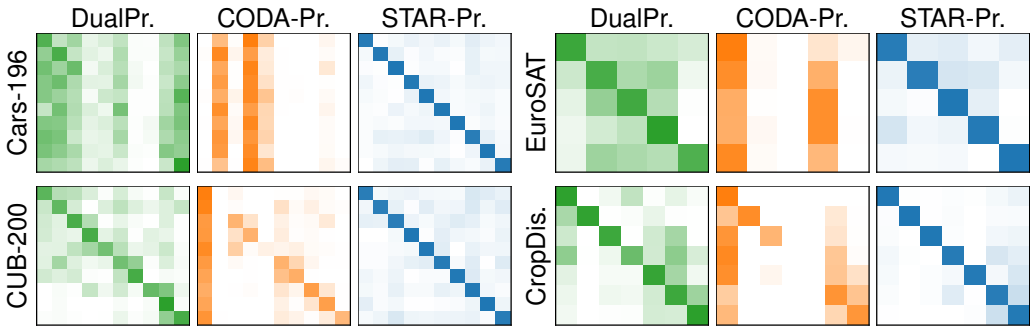


Figure 2.3: Analysis of prompt selection stability for Split Cars-196, Split EuroSAT, Split CUB-200, and Split CropDiseases. We assess various models at the end of the last task and report results as confusion matrices. The y axis indicates the task of the query sample, while the x axis shows the task of the corresponding selected key.

In contrast, STAR-Prompt offers a more balanced trade-off between stability and plasticity, emerging as the most robust prompting-based solution in these settings.

Prompt selection stability. We refer to a prompting mechanism as ***stable*** when, for a given task, its query-key selection procedure consistently retrieve the prompts that are appropriate for that task, rather than incorrectly activating prompts associated with future tasks. Figure 2.3 presents an overview of the prompt-selection behavior at the end of the final task, showing that STAR-Prompt retrieves task-relevant prompts with the highest accuracy. Figure 2.4 further refines this analysis by examining the first task in detail. While CODA-Prompt [72] may appear to match our retrieval performance on Split EuroSAT at first glance, we attribute this to its tendency to overemphasize the prompts learned during the initial task, a pattern that also emerges in Figure 2.3. These findings strengthen our conclusion that STAR-Prompt consistently achieves an effective balance between stability and adaptability across the entire training sequence.

Computational cost. The computational overhead of STAR-Prompt is on par with that of other prompt-based techniques (e.g., L2P, CODA-Prompt), as it similarly requires two forward passes. The main distinction lies in the architecture: while existing approaches use a single backbone, our method operates with two separate backbones, each with its own set of parameters.

Table 2.3: Ablative studies on STAR-Prompt (Final Avg. Acc. ± std dev).

Model	Imagenet-R	Cars-196	EuroSAT	ISIC
STAR-Prompt	**89.83** ±0.04	**87.62** ±0.20	**93.70** ±0.09	**66.67** ±1.45
Ablations on two-level prompting				
Classify with first-level keys $\boldsymbol{w}_c$	88.16 ±0.27	87.57 ±0.12	90.12 ±0.54	58.87 ±1.25
w/o first-level prompts	88.97 ±0.52	79.88 ±0.46	86.25 ±5.32	58.68 ±5.57
Other secondary ablations				
Prefix Tuning (no residuals)	71.34 ±0.34	60.03 ±4.14	90.92 ±0.44	62.46 ±0.88
w/o Generative Replay	88.55 ±0.06	87.17 ±0.07	83.78 ±1.82	50.76 ±3.36
w. Unimodal Generative Replay	89.62 ±0.12	87.28 ±0.12	92.58 ±0.40	60.79 ±0.68
w/o Confidence Modulation	88.73 ±0.16	87.29 ±0.13	93.68 ±0.38	63.53 ±0.75

2.6 Ablation studies

Table 2.3 analyzes the contribution of each component of STAR-Prompt. For brevity, we present results from one dataset per domain.

Two-level prompting. The first two comparisons evaluate the impact of the two-level strategy. In particular, *"Classify with first-level keys* $\boldsymbol{w}_c$*"* corresponds to an ablation that omits the second stage. Classification is performed using the learned keys $\boldsymbol{w}_c$ as class prototypes, similar to CoOp (posteriors are computed as in eq. (2.5)). As shown in Table 2.3, this approach achieves competitive performance, demonstrating the stability of the first-level keys. However, for tasks requiring greater plasticity (e.g., EuroSAT and ISIC), the performance gap with STAR-Prompt becomes more pronounced. Likewise, STAR-Prompt outperforms the variant *"w/o first-level prompts"*, which replaces learned first-level prompts with static CLIP textual templates [62],

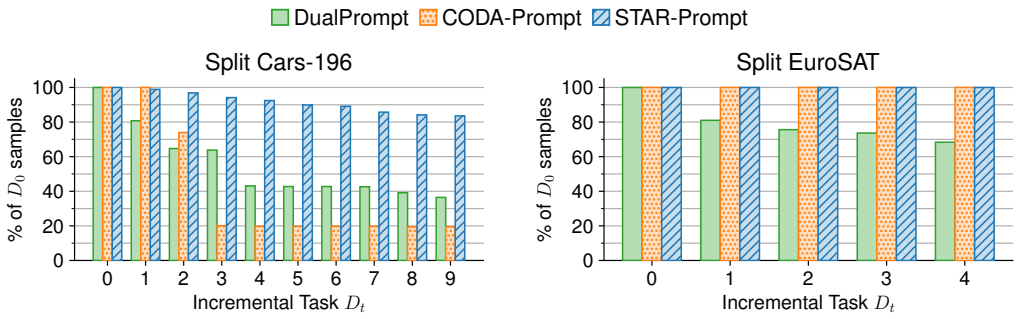


Figure 2.4: Prompt retrieval on the first task of different datasets.

using the resulting embeddings as class-prototype keys and learning only the second-level prompts Q_c.

Semantic residuals. Next, we replace our semantic residual mechanism (eq. (2.3)) with *Prefix Tuning* [84], where each ViT layer is augmented with 5 key and 5 value prompt tokens concatenated to the MSA keys and values. Results in Table 2.3 show the superior effectiveness of our additive residual design.

Generative Replay. Omitting the generative replay stage results in a substantial performance drop, particularly when task distributions diverge from the pre-training domain. The row *w. Unimodal Generative Replay* shows results using a single Gaussian instead of a MoG. While this is generally better than *w/o Generative Replay*, it still falls short of the full MoG-based approach.

Confidence modulation of the semantic residuals. Finally, eq. (2.2) incorporates the *confidence* of the CLIP encoders into the residuals. The last row shows that this *Confidence Modulation* consistently improves performance across all datasets.

2.7 Conclusion

In this work, we introduced STAR-Prompt, a prompting framework for Continual Learning that integrates three key innovations. First, we enhance the stability of prompt selection through the use of a foundation model combined with a two-level prompt tuning strategy. Second, we substitute conventional prompt concatenation with additive semantic residuals, enabling the transfer of class-specific information into the MLP layers. Finally, we incorporate a generative replay mechanism based on a multi-modal modeling of feature distributions. Each component of STAR-Prompt provides a meaningful contribution, collectively allowing it to surpass existing state-of-the-art methods.

Chapter 3

SCAD: Distilling Attention Representations

3.1 Overview

After investigating Continual Learning methods in the traditional single-label image classification setting, our research group explored other areas where these techniques could be applied. This led us to the relatively sparse literature on *Multi-Label Continual Learning* (MLCL). In this setting, the task is again image classification; however, each image can be associated with one or more classes, or labels. Multiple labels may arise for different reasons: an image may contain multiple objects (e.g., both a dog and a cat), or labels can represent attributes (e.g., rough, red), and the subject depicted may be described through a combination of attributes. Multi-label classification is widely recognized as significantly more challenging than single-label classification.

A common approach for multi-label learning is to have a neural network produce a score for each label. If the score exceeds a certain threshold, the label is assigned to the image. During training, researchers often use binary cross-entropy applied to the logits (i.e., the network's final outputs). In this case, the threshold is naturally set to 0.

MLCL introduces an additional challenge due to its combination with Continual Learning. Here, the model is trained on a sequence of tasks, with each task introducing new labels or attributes. In traditional Class-IL Continual Learning (as studied by our group), all examples in a new task belong exclusively to the task's labels, and previous task examples are no longer accessible. In MLCL, however, a crucial difference emerges: since examples can have multiple labels, a test image from a new task may contain not only

one or more labels from the current task, but also labels from previous tasks. This immediately raises concerns that existing single-label Continual Learning techniques may be less effective in this setting. For instance, strategies that freeze parameters or prompts from previous tasks may not fully prevent forgetting, while approaches that fine-tune all parameters could perform better, as the classifier can adjust all parameters jointly.

Given the lack of studies applying recent Continual Learning techniques based on Vision Transformers (ViTs) to MLCL, the first step in our research was to evaluate the performance of these models in this context. We quickly observed that many of these methods fail to achieve satisfactory results on established MLCL benchmarks, being particularly affected by catastrophic forgetting. This motivated the design and development of a new approach to address the shortcomings of existing methods. Our work resulted in the creation of **SCAD** [52], a new MLCL baseline that leverages the distillation of ViT-specific representations to mitigate forgetting, while also incorporating well-established replay strategies. The paper of SCAD was published at LOD 2024, and the code is available at `https://github.com/aimagelab/SCAD-LOD-2024`.

3.2 Context

Related works. One of the most representative and influential contributions to the field of MLCL is presented in [1]. In this work, the authors introduce a novel and comprehensive benchmark named **IIRC**, specifically designed to evaluate Continual Learning models under multi-label conditions. The benchmark is constructed by restructuring well-known datasets, such as CIFAR-100 and ImageNet, so that each sample is associated with both a superclass and a subclass label. This hierarchical structure enables a finer-grained and more realistic assessment of models, as it mirrors the fact that objects often belong simultaneously to multiple semantic categories.

The authors formalize two distinct evaluation settings. The first, referred to as *complete information*, assumes that all labels associated with each sample are available at training time. The second, called *incomplete information*, provides only the label relevant to the current task, leaving the remaining labels unseen until future tasks. This latter scenario is considerably more challenging because the model must not only learn incrementally but also infer and maintain relationships between partially observed labels over time.

Given its realism and difficulty, we adopt IIRC as one of the core benchmarks in our evaluation protocol and focus specifically on the incomplete-information setting. This choice allows us to assess whether the proposed

method can effectively handle situations where the semantic structure of the data is revealed progressively (a realistic constraint in many real-world applications). Moreover, the hierarchical and multi-label nature of IIRC offers a rigorous testing ground for measuring robustness against forgetting, adaptability to new tasks, and the capacity to maintain coherent label representations over long sequences of incremental updates.

A particularly compelling line of work is presented in [35], where the authors address two fundamental challenges simultaneously: catastrophic forgetting in continual learning and the long-tail distribution problem, in which a small number of classes (the *head*) contain most of the samples, while the majority of classes (the *tail*) are severely underrepresented. Through extensive experimentation, the authors demonstrate that tail classes are disproportionately prone to forgetting, largely because their limited sample availability provides the model with fewer opportunities to consolidate their representations over time.

To mitigate this issue, the authors introduce a novel sampling strategy called *Partitioned Reservoir Sampling*, specifically designed to decide which samples should be stored in the replay buffer. Unlike traditional reservoir sampling, this method maintains a more balanced representation of both head and tail classes by partitioning the buffer into dedicated segments and enforcing class-aware selection criteria. This ensures that rare classes are not overshadowed by more frequent ones during replay, ultimately leading to a more equitable contribution of all classes to the training process.

Beyond addressing the imbalance directly, the authors also highlight the broader implication that Continual Learning systems deployed in real-world environments (where data distributions are rarely uniform) must incorporate mechanisms that explicitly account for data scarcity and class imbalance. Their approach provides valuable insights into how buffer management strategies can be adapted to preserve performance across the entire label space, offering a meaningful step toward more robust and fair Continual Learning models.

Formal definition of MLCL. In the multi-label classification setting, a dataset $\mathcal{D}$ is composed of pairs $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{|\mathcal{D}|}$, where each image $\mathbf{x}_i \in \mathbb{R}^{3\times H\times W}$ is associated with a multi-hot vector $\mathbf{y}_i \in \mathbb{R}^{|\mathcal{Y}|}$. The entry $y_{i,j}$ equals 1 when the i-th sample contains the j-th label, and 0 otherwise; the set $\mathcal{Y}$ collects all admissible labels. Given a batch $\mathcal{B} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=0}^{B}$, we denote by $\mathbf{X}[s:e]$ the subset $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=s}^{e} \subseteq \mathcal{B}$.

Under the MLCL paradigm, a model f_θ is required to learn from a sequence of tasks $\{T_t\}_{t=1}^{N}$, where each task is defined as $T_t = \{(\mathbf{x}_j^t, \mathbf{y}_j^t)\}_{j=1}^{|T_t|}$.

The learning objective consists of minimizing the classification loss across all tasks:

$$\min_{\theta} \mathcal{L}_{\text{clf}} = \sum_{t=1}^{N} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim T_t} \left[\ell(\mathbf{y}, f_\theta(\mathbf{x})) \right], \tag{3.1}$$

where ℓ typically corresponds to the binary cross-entropy loss [63, 1]. Since, by assumption, data from past tasks cannot be revisited, optimizing eq. (3.1) directly is not possible. The challenge, therefore, lies in updating the model to learn the current task while avoiding the loss of information acquired during earlier stages of training.

Pretraining and forgetting. Previous works [64, 51] highlighted that large models pretrained on extensive datasets generally exhibit strong resistance to catastrophic forgetting. Nonetheless, as noted in [9], progressive task-wise fine-tuning can cause the parameters to drift away from their pretrained configuration, ultimately exacerbating forgetting. Motivated by this observation, we investigated mechanisms capable of maintaining a model's proximity to its pretrained state throughout Continual Learning.

Our solution builds upon ideas from the literature on inter-network knowledge transfer [79, 9, 46, 89]. SCAD uses two pretrained networks: the *teacher*, whose parameters remain fixed, and the *student*, which is updated incrementally. The following sections provide a detailed explanation of how this framework is employed to mitigate forgetting.

3.3 Selective Class Attention Distillation

Backbone. For both the teacher and student networks, we adopt a pretrained Vision Transformer [38], in line with recent Continual Learning methods that rely on this architecture. In a ViT, an input image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$ is partitioned into S non-overlapping patches. These patches are flattened and mapped into embedding vectors through a patch embedding layer, forming a sequence of tokens. A dedicated *class token*, used for prediction, is prepended to this sequence, yielding $\mathbf{x}_p \in \mathbb{R}^{(S+1)\times D}$, where D is the embedding dimension. This token sequence is then processed by a stack of transformer encoder blocks composed of Multi-Head Self-Attention [74] and feed-forward layers. The final class token representation is subsequently used for classification. The multi-head attention mechanism enables the architecture to capture diverse relationships among tokens, allowing different heads to specialize on complementary local and global structures within the image.

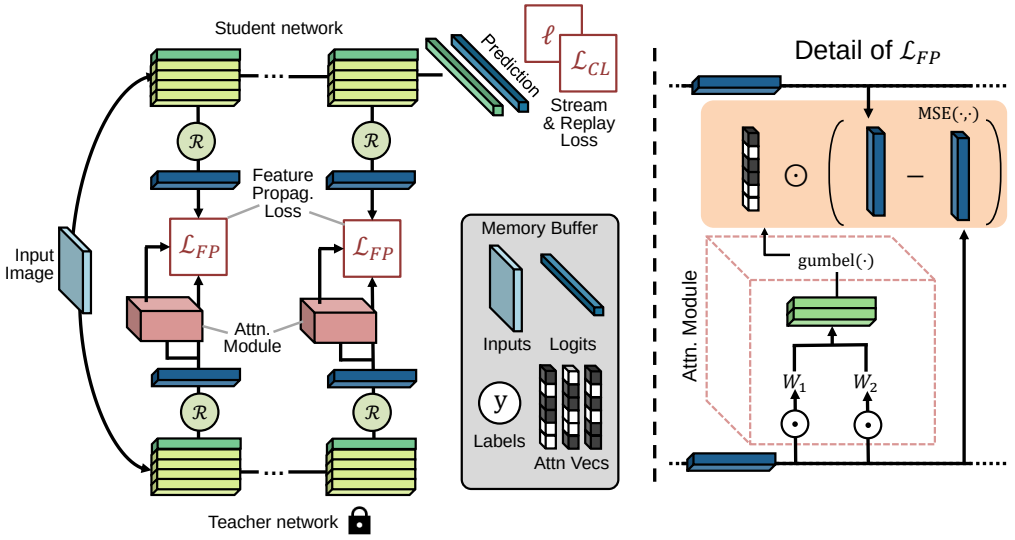


Figure 3.1: The approach relies on two pretrained Vision Transformers: a frozen *teacher* model and a trainable *student*. Each input image is processed by both networks, and intermediate features are extracted. Using eq. (3.2) together with the indexing scheme defined in eq. (3.3), an attention vector capturing the interactions between the class token and the remaining tokens is computed. These steps, summarized by the operator $\mathcal{R}$ in the figure, feed attention-based modules (*adapters*) that generate binary masks for selective knowledge transfer. A feature propagation loss, $\mathcal{L}_{FP}$, is employed to preserve the correspondence between the teacher's and student's attention vectors.

Knowledge transfer technique. Although directly aligning the intermediate features of teacher and student networks represents a simple form of knowledge transfer, alternative approaches have been explored, particularly for Vision Transformers [16, 89, 79]. Our method takes inspiration from the strategy proposed in [79]. When an input passes through a ViT, each transformer block produces an intermediate representation $F^l \in \mathbb{R}^{(S+1)\times D}$, with $l \in \{1, \ldots, |L|\}$ indicating the block index. From these representations, we derive token-to-token correlations after normalizing the feature maps:

$$\mathcal{R}^l = \frac{F^l}{\|F^l\|_2^2} \cdot \left(\frac{F^l}{\|F^l\|_2^2}\right)^T \tag{3.2}$$

which yields $\mathcal{R}^l \in \mathbb{R}^{(S+1)\times(S+1)}$. Our goal is to distill the attention vector $\mathcal{R}^l[0, 1 : S+1]$, encoding the interactions between the class token and the remaining tokens. To enforce alignment between the teacher and student, we

compute the difference between their corresponding attention vectors:

$$D^l = \mathcal{R}^l_{\mathcal{T}}[0, 1 : S+1] - \mathcal{R}^l_{\mathcal{S}}[0, 1 : S+1] \tag{3.3}$$

where the subscripts $\mathcal{T}$ and $\mathcal{S}$ refer to the teacher and student models, respectively. This formulation encourages the student to preserve the relational structure encoded by the teacher, maintaining consistent interactions between the class token and the other tokens.

Adapter Networks. While aligning teacher and student representations is effective for mitigating forgetting, it may overly constrain the student, limiting its adaptability to new tasks. To overcome this, we introduce learnable ***adapter*** modules designed to filter out irrelevant information from the teacher. The teacher's attention vectors $\mathcal{R}_{\mathcal{T}}$ are processed by these adapters, which output binary masks denoted by $\mathbb{M}(\cdot)$. Specifically, each attention vector is first multiplied by two sets of learnable weights to produce a vector $\mathbf{v} \in \mathbb{R}^{(S+1)\times 2}$, followed by binary Gumbel-Softmax sampling along the last dimension to obtain a binary vector. This vector is then element-wise multiplied with the distance vector $\mathcal{D}$ from equation 3.3, and the mean squared distance is averaged across all adapters. The resulting loss function is:

$$\mathcal{L}_{FP} = \frac{1}{|L|} \sum_{l \in L} \left\| \mathbb{M}(\mathcal{R}^l_{\mathcal{T}}) \odot \mathcal{D}^l \right\|_2^2, \tag{3.4}$$

where $\mathbb{M}^l(\mathcal{R}^l_{\mathcal{T}})$ denotes the binary mask produced by the l^{th} adapter and $\odot$ represents element-wise multiplication.

Experience Replay. Rehearsal strategies are well-known to effectively combat forgetting [22, 4]. In our approach, we maintain a fixed-size memory buffer $\mathcal{M}$ to store a subset of past samples, including their labels and logits from the student network [11, 65, 67]. Examples are selected using reservoir sampling [76], ensuring an unbiased and representative subset of the data stream. The model is then trained to correctly classify these stored examples by minimizing the binary cross-entropy on the buffered data:

$$\mathcal{L}_{er} = \mathbb{E}_{(\mathbf{x}_i, \mathbf{y}_i) \sim \mathcal{M}} \Big[\mathrm{BCE}(f_\theta(\mathbf{x}_i), \mathbf{y}_i) \Big]. \tag{3.5}$$

Additionally, to preserve the knowledge encoded in previous tasks, we encourage the student's current logits to match the stored logits $\mathbf{h}_i$ through a mean squared error loss:

$$\mathcal{L}_{der} = \mathbb{E}_{(\mathbf{x}_i, \mathbf{h}_i) \sim \mathcal{M}} \Big[\mathrm{MSE}(f_\theta(\mathbf{x}_i), \mathbf{h}_i) \Big], \tag{3.6}$$

where MSE is the mean squared error. This encourages the student to replicate the responses of past tasks, reinforcing retention. Finally, we incorporate the ACE method [12], which computes the classification loss only on the labels of the current batch, helping to smooth abrupt changes in the model's representations over time.

Adapter Mask Replay. Since adapter modules contain learnable parameters, they are vulnerable to forgetting over time. To mitigate this, we introduce a replay mechanism specifically for the adapter binary masks. Concretely, the binary masks produced by the adapters on the original, non-augmented training samples are stored in a memory buffer. During subsequent tasks, when experience replay is performed, the retrieved samples are passed through the adapters, and the newly generated masks are compared with the saved ones. This comparison is enforced through the following regularization loss:

$$\mathcal{L}_{FP\_rep} = \mathbb{E}_{\substack{(\mathbf{x},\mathbf{m}^l)\sim\mathcal{M} \\ l\in L}}\Big[\,\mathrm{BCE}\left(\mathbb{M}(\mathcal{R}^l_{\mathcal{T}}(\mathbf{x})), \mathbf{m}^l\right)\Big], \tag{3.7}$$

where BCE is the binary cross-entropy loss, and $(\mathbf{x}, \mathbf{m})$ denotes a pair consisting of a training sample and the corresponding stored attention mask from the memory buffer.

Final loss. The overall training objective combines all relevant terms as follows:

$$\mathcal{L} = \mathcal{L}_{clf} + \alpha\,\mathcal{L}_{der} + \beta\,\mathcal{L}_{er} + \lambda_{FP}\,\mathcal{L}_{FP} + \lambda_{FP\_rep}\,\mathcal{L}_{FP\_rep}, \tag{3.8}$$

where α, β, λ_{FP}, and λ_{FP_rep} are hyperparameters that balance the contribution of each loss component.

3.4 Experimental Setup

Datasets. We evaluate our approach on two benchmarks: IIRC CIFAR-100 and a new dataset that we derive from WebVision.

The **IIRC CIFAR-100** benchmark [1] assigns two labels to each image: one corresponding to a superclass and the other to a subclass, following the structure of CIFAR-100 [40]. The benchmark is organized into 22 tasks. In the first task, only superclass labels are available, while subsequent tasks progressively introduce the subclass labels. To increase the level of difficulty, we adopt the *incomplete information setting*, in which only the label relevant to the current task is revealed for each sample.

The **WebVision** dataset [43] contains more than 2.4 million web images accompanied by metadata such as titles, descriptions, and tags. Leveraging this metadata, we construct a challenging real-world multi-label continual benchmark, referred to as **Incremental WebVision**, following the pipeline below:

1. Starting from the original set of 816,814 tags, we filter out those appearing fewer than 1000 times to avoid overly rare labels.

2. Using feature similarity in the CLIP embedding space [62] as a proxy for semantic relatedness, we inspect clusters of tags that frequently co-occur and manually merge those that are semantically redundant. This yields a curated subset of roughly 350,000 images annotated with a set of 300 cleaned tags, which serve as multi-label targets.

3. For each task T_i ($i = 1, \ldots, 6$), we select 50 new labels $\mathcal{Y}_i$. We then assemble the task dataset by choosing approximately 58,000 images (i.e., 350,000/6) that contain at least one label belonging to $\mathcal{Y}_{1..i} \triangleq \bigcup_{j=1}^{i} \mathcal{Y}_j$. Our benchmark follows the *complete information* setting described in [1].

4. From each task, we set aside 5000 images for testing, while using the remaining $\sim$ 53,000 images for training.

The resulting dataset provides a challenging and realistic scenario for Continual Learning, especially given the substantial annotation noise known to be present in WebVision [43].

Metrics. As highlighted in [73], standard multi-class evaluation measures do not adequately reflect the nuances of partial correctness in multi-label prediction. Following the conventions established in the IIRC framework, we adopt the **Precision-Weighted Jaccard Similarity** (**PWJS**) as our primary evaluation metric, defined as:

$$R_{j,k} = \frac{1}{n_k} \sum_{i=1}^{n_k} \frac{|Y_{j,k,i} \cap \hat{Y}_{j,k,i}|}{|Y_{j,k,i} \cup \hat{Y}_{j,k,i}|} \times \frac{|Y_{j,k,i} \cap \hat{Y}_{j,k,i}|}{|\hat{Y}_{j,k,i}|}$$

Here, $R_{j,k}$ denotes the performance on task k after completing training on task j; $\hat{Y}_{j,k,i}$ represents the predicted label set for the i-th test instance of task k; $Y_{j,k,i}$ corresponds to the ground-truth labels; and n_k is the size of the test set associated with task k.

Given the sequential nature of Continual Learning, we represent performance using the average value of $R_{j,k}$ computed after finishing the final task. We refer to this summary metric as the **Final Average PWJS**:

$$AR_f = \frac{1}{N} \sum_{m=1}^{N} R_{N,m}, \tag{3.9}$$

where N is the total number of tasks in the benchmark.

All experiments are initialized from identical pretrained weights, repeated twice, and we report the mean and standard deviation of the resulting AR_f values for every compared method.

Competitors. To contextualize the performance of our method, we benchmark it against several prominent approaches in the Continual Learning literature:

- **Experience Replay (ER)** [65] implements a basic rehearsal mechanism in which a fixed-size memory stores previously encountered samples. During training, the model revisits this buffer together with the data from the current task. While conceptually simple, the method incurs substantial memory usage.

- **ER with Asymmetric Cross-Entropy (ER-ACE)** [12] extends ER by modifying the loss computation: for samples of the current task, the loss is restricted to the labels present in the batch, whereas buffered samples contribute a loss over all classes observed so far.

- **DER++-ACE** [12] integrates logit distillation, as introduced in [11], with the asymmetric-loss formulation of ER-ACE, yielding a more robust replay-based strategy.

- **Learning to Prompt (L2P)** [86] maintains a pool of trainable prompts, each paired with a learnable key. A frozen Vision Transformer provides feature embeddings used to query this pool via similarity matching; the most relevant prompts are then prepended to the ViT blocks. Only the prompts are updated during training, while the backbone remains frozen.

- **CODA-Prompt (CP)** [71] assigns a set of prompt components to each task. Instead of selecting discrete prompts as in L2P, it computes a weighted combination of them using similarity scores. The method also leverages learnable attention vectors to filter out irrelevant query components.

Table 3.1: Final Average PWJS (with standard deviation) across the full task sequence for both the IIRC and WebVision benchmarks. Methods relying on replay buffers are listed in the lower section of the table.

AR_f %	IIRC-CIFAR		Incr. WebVision	
Joint (UB)	86.83 (0.22)		25.12 (0.01)	
Finetuning-ACE	3.21 (0.01)		2.34 (0.36)	
Finetuning (LB)	0.24 (0.01)		18.81 (0.59)	
L2P [86]	3.76 (0.01)		0.31 (0.25)	
CP [71]	3.79 (0.01)		15.05 (1.45)	
Buffer Size	500	2000	2000	5000
ER [65, 67]	21.04 (1.81)	38.55 (0.91)	17.76 (0.18)	21.63 (0.18)
ER-ACE [12]	41.83 (1.75)	50.66 (0.87)	7.75 (1.78)	16.26 (0.34)
DER++-ACE [11]	41.66 (2.55)	61.82 (0.09)	8.97 (0.01)	17.03 (0.01)
SCAD (ours)	**45.23** (0.03)	**66.58** (0.02)	**20.02** (0.35)	**22.17** (0.60)

For completeness, we also report results for standard reference settings: **Finetuning**, representing a lower bound where no mechanism is used to mitigate forgetting; **Finetuning-ACE**, which incorporates the selective Cross-Entropy scheme from [12] as a simple baseline; and **Joint training**, an upper-bound scenario assuming access to all data simultaneously rather than incrementally.

3.5 Results

Table 3.1 highlights a clear contrast in difficulty between the two evaluation settings. All approaches experience a considerable drop in performance on WebVision, which is consistent with the substantial degree of annotation noise documented for this dataset. On the IIRC benchmark, prompt-based strategies yield limited gains, and interestingly, Coda-Prompt does not outperform L2P (an observation that differs from the outcomes typically reported on standard single-label Continual Learning benchmarks). This discrepancy underscores the need for methods specifically tailored to more difficult, multi-label scenarios.

Replay-based strategies generally achieve stronger results than prompt-based ones on both datasets. The asymmetric loss introduced by ACE proves

Table 3.2: Final PWJS on the IIRC CIFAR100 benchmark obtained by our approach, with and without the adapter-generated binary masks used to regulate knowledge transfer.

	Binary masks	
	✓	✗
Final PWJS	66.58 ±0.02	63.68 ±0.01

particularly beneficial, confirming that constraining the influence of each sample is essential to mitigate task interference in multi-label settings. The addition of logit distillation in DER++ further reinforces performance, showing the value of preserving richer target information during replay.

Across all experimental configurations and buffer sizes, our method consistently delivers the highest performance. These findings emphasize the advantage of effectively leveraging and preserving pretrained knowledge throughout the Continual Learning process.

3.6 Model analysis

Contribution of selective distillation To better understand the role played by the adapter modules in our framework, we performed an ablation study in which adapters were removed and distillation was carried out directly using full attention vectors. The comparison, summarized in Table 3.2, indicates that the binary masks produced by the adapters lead to a noticeable improvement in the final score. This confirms that the adapters effectively filter the teacher's attention vector, allowing only relevant information to be distilled into the student model.

Quantitatively measuring catastrophic forgetting. Although AR_f summarizes end-of-training performance, Continual Learning research also emphasizes the extent to which a model loses competence on previous tasks as training progresses. To capture this phenomenon in the multi-label setting, we adapt the standard Adjusted Forgetting metric [7, 13] by expressing it in terms of PWJS:

$$FG_f = \frac{1}{N-1} \sum_{m=1}^{N-1} \left[\frac{R_{*,m} - R_{N,m}}{R_{*,m}} \right]^+, \tag{3.10}$$

$$\text{where} \quad R_{*,m} = \max_{t \in \{m,\dots,N-1\}} R_{t,m}, \quad \forall m \in \{1,\dots,N-1\},$$
$$[\cdot]^+ = \max(\cdot, 0).$$

The value of FG_f ranges from 0 to 100: a score of 100 reflects complete forgetting of earlier tasks, while 0 corresponds to perfect retention.

Table 3.3 reports FG_f for the same experiments discussed in Table 3.1. On the IIRC benchmark, the ACE formulation significantly mitigates forgetting, showing that restricting the loss to the classes present in each batch helps preserve parameters encoding older knowledge. On WebVision, the differences among rehearsal methods diminish, likely due to the inherent difficulty of the dataset, which makes it challenging for any method to maintain high performance.

Prompt-based approaches exhibit comparable forgetting on IIRC, but L2P suffers a larger drop on WebVision. Overall, rehearsal-based strategies tend to retain knowledge more effectively in the multi-label setting, highlighting the benefit of maintaining a memory buffer of past samples.

Table 3.3: Multi-Label Adjusted Forgetting FG_f (lower is better) for all evaluated methods across our benchmarks.

$\boldsymbol{FG_f}$	**IIRC-CIFAR Incr.**		**WebVision**	
Finetuning-ACE	65.40		74.45	
Finetuning (LB)	98.67		37.39	
L2P	97.29		66.41	
CP	97.77		44.06	
Buffer Size	500	2000	2000	5000
ER	69.98	41.57	34.33	25.64
ER-ACE	45.25	40.06	57.52	32.80
DER++-ACE	25.82	17.71	48.95	31.87
SCAD (ours)	24.36	16.40	28.37	24.92

3.7 Conclusion

In this study, we recognize the effectiveness of recent Continual Learning methods in mitigating *catastrophic forgetting* on standard single-label benchmarks. Nevertheless, our experiments on a newly designed, more challenging

multi-label benchmark demonstrate that current state-of-the-art prompting-based CL methods struggle in this scenario, whereas classical rehearsal-based approaches continue to deliver reliable performance.

Motivated by this observation, we introduce SCAD, a baseline model specifically designed to address the complexities of the multi-label setting. Our approach integrates the strengths of conventional rehearsal strategies with mechanisms for knowledge transfer, leveraging the insight that large-scale pretrained architectures inherently resist forgetting. Accordingly, SCAD is designed to preserve alignment with the pretraining state while accommodating new tasks.

Through an extensive empirical evaluation, we show that SCAD consistently outperforms existing state-of-the-art methods in the Multi-Label Continual Learning context. Consequently, we propose SCAD as a robust baseline for researchers exploring this emerging and challenging branch of Continual Learning.

Part II

Backdoors in Neural Networks

Chapter 4

The Backdoor Threat Landscape

4.1 Overview

In the second part of my PhD, I shifted my research focus toward backdoor attacks in neural networks, driven by my longstanding passion for computer security. Within the broad field of AI security, the subfield most directly concerned with the robustness of learning systems against malicious interference is precisely that of backdoor attacks. Briefly, backdoors implanted within a neural network model can enable malicious actors to control or manipulate the model's behavior.

Given the growing use of neural networks in safety-critical applications and complex industrial processes, the possibility for an attacker to alter a model's predictive mechanisms makes the study of such threats and their implications increasingly crucial.

Since the discovery of the first BadNets [23] attack in 2017, numerous research groups have proposed and investigated defense techniques aimed at mitigating the risks posed by backdoor threats. These defenses, widely discussed in the deep learning literature, have demonstrated strong effectiveness in limiting attackers' influence on model behavior. However, as I have observed in my research, many of these defenses fail when attackers employ more sophisticated, adaptive strategies specifically designed to bypass existing protection mechanisms.

Currently, only a few research groups systematically study the robustness of backdoor defenses against adaptive attacks. My research aims to show that even relatively simple backdoor attacks (such as BadNets) can be enhanced to circumvent several existing defense methods. I hope that my work will

raise awareness of the risks associated with adaptive backdoor attacks and encourage the research community to take these threats into account when designing and evaluating future defense strategies.

4.2 Definitions

There are several scenarios in which malicious actors can implant backdoors into neural networks:

- A user or company has sufficient computational resources to train models but lacks an adequate dataset to achieve satisfactory accuracy. As a result, they download a public dataset and train their models on it.
- A user or company lacks the computational resources to train neural networks and outsources the training process to a third party.
- A user or company lacks training resources but finds a pre-trained model for their target task on a public repository (such as HuggingFace). Consequently, they download and directly use the model from the repository.

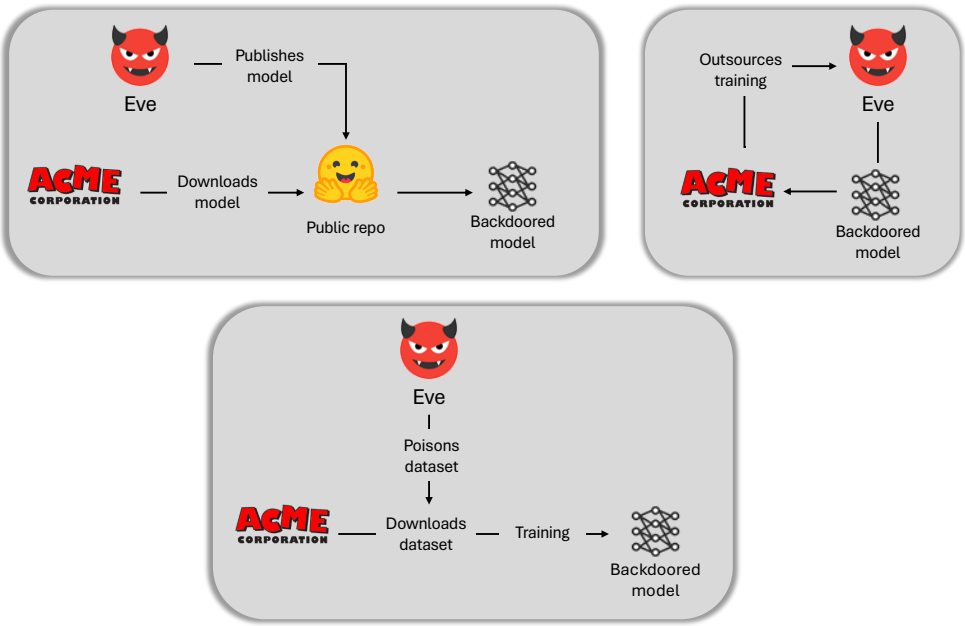


Figure 4.1: Examples of common scenarios in backdoor attacks.

It is easy to observe that these scenarios are very common in today's artificial intelligence landscape. Over the past few years, the number of parameters and overall complexity of neural networks have increased dramatically. While this growth has enabled models to achieve higher performance and representational power, it has also made training them significantly more expensive and resource-intensive. As a result, an increasing number of individuals and organizations rely on pre-trained models released by third parties or delegate model training to external providers.

However, this practice introduces a serious risk: the third party responsible for training the model may act maliciously and implant backdoors during the training process. We now introduce the definition of a backdoor in the context of artificial intelligence:

Definition 4.2.1: Backdoor

A neural network is said to **contain a backdoor** when the following two conditions hold simultaneously:

- The model provides accurate predictions when processing clean inputs.
- The model produces attacker-controlled predictions when processing inputs containing secret properties – known only to the attacker – referred to as backdoor triggers.

This definition highlights why backdoor attacks are particularly difficult to detect without dedicated countermeasures. The end user typically has access only to the trained model and a limited set of clean validation data to verify its accuracy. Since the backdoor triggers are known exclusively to the attacker, there is no apparent way to generate poisoned examples to test for the presence of a backdoor.

Traditionally, backdoor attacks have been studied primarily in the context of image classification [23, 17, 48]. In this task, the neural network is required to assign the correct class label to each input image.

Different types of backdoor attacks exist, depending on the attacker's objective. Among them, the most commonly studied — and often the most effective — are the so-called **poisoned-label** attacks, in which the attacker aims to make the neural network predict a specific target label for inputs containing the trigger. In this research, I focused on this class of attacks.

In this context, the most common metrics used to assess the effectiveness of backdoor attacks are:

- **Clean Accuracy** (**CA**): the ratio between the number of clean samples

correctly classified and the total number of clean samples;

- **Attack Success Rate** (**ASR**): the ratio between the number of poisoned samples classified with the target label and the total number of poisoned samples.

Poisoned-label attacks can be classified into two main types:

- **Source-specific**: only samples belonging to a particular source class are poisoned.
- **Source-agnostic**: samples from all classes are poisoned (potentially including the target class itself).

It is important to note that, in poisoned-label attacks, the labels assigned to poisoned images are intentionally incorrect relative to the visual content. Consequently, a manual inspection of the dataset may reveal mismatches between images and their labels. Likewise, if the backdoor triggers are visually salient, their presence might be noticed during visual inspection.

Nevertheless, manual detection is typically impractical for two main reasons. First, image classification datasets usually contain very large numbers of samples, so exhaustive manual review is unrealistic. Second, as will be shown in later sections, attackers can often achieve effective backdoors with a relatively small number of poisoned samples; this makes locating the few compromised examples among thousands or millions of images extremely challenging.

4.3 A look at the literature

Early works. The first backdoor attack, known as **BadNets** [23], was introduced in 2017. Among the classification tasks considered by the authors is traffic sign recognition for an autonomous driving system. This is undeniably a safety-critical task: a failure in the recognition module can lead to severe traffic accidents with potential human victims. Therefore, studying backdoor attacks in the context of vision systems for autonomous driving is essential, as it reveals how deliberately injected perturbations during training can turn seemingly correct decisions into dangerous behaviors in real-world scenarios.

The authors investigate several types of backdoor triggers, which share the property of being visually salient elements that are easily detected by a vision model. Typical examples include colored stickers or stickers depicting recognizable objects—such as a flower or an attention-grabbing icon—applied to traffic signs. In the attack model, the adversary's goal is for the model to

predict a target label when the trigger is present, while producing the correct label when the trigger is absent.

To simulate a realistic threat, the authors consider a source-specific attack: in this scenario, poisoned examples are applied only to certain source classes (for instance, stop signs) so that they are misclassified as a different target class (for example, speed limit signs). This attack model reflects practical cases in which an adversary aims to alter model behavior for particular objects, rather than causing a universal mapping from any input to the same target label.

Figure 4.2: Examples of backdoor triggers used in BadNets.

For BadNets-style attacks, the injection of a backdoor trigger into an image can be formally expressed with this definition:

Definition 4.3.1: Trigger injection

For an image $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$, the corresponding poisoned image $\hat{\mathbf{x}}$ is obtained as

$$\hat{\mathbf{x}} = \mathbf{x} \odot m + \boldsymbol{\delta} \odot (1 - m), \tag{4.1}$$

where $m \in \{0,1\}^{C\times H\times W}$ is a binary mask of the same dimensions as $\mathbf{x}$ (with $m_i = 1$ at positions preserving the original content and $m_i = 0$ where the trigger is applied), $\boldsymbol{\delta}$ is the trigger pattern, and $\odot$ denotes element-wise (Hadamard) multiplication.

The mask and trigger are typically defined across all channels (e.g., RGB), so $m, \boldsymbol{\delta} \in \{0,1\}^{C\times H\times W}$ or $\mathbb{R}^{C\times H\times W}$, depending on the chosen representation. In many practical scenarios, the trigger occupies only a local region of the image (a patch), corresponding to the subset of positions where $m = 0$.

Another contemporaneous work, commonly referred to as **Blended** [17], uses entire images as triggers by blending them with the original inputs. An interesting application examined by the authors concerns a face recognition

system used for access control to restricted areas. In this setting, the authors poison the model so that images of users wearing a particular type of glasses are classified as a different user with elevated privileges. The attack is implemented by overlaying the image of the glasses — which thus acts as the backdoor trigger — onto the original facial image. Like BadNets, the triggers in this approach are visually salient elements that are easily detectable by a neural network.

Early defenses. One of the earliest defenses against backdoors is **Neural Cleanse** [78]. Although developed several years ago, this technique remains a standard baseline in contemporary backdoor evaluation frameworks. Neural Cleanse pursues two primary goals: (i) detect whether a model contains a backdoor and (ii) identify the corresponding target label.

The core idea is to cast the search for an unknown trigger as an optimization problem. Referring to Definition 4.3.1, the mask m and the trigger pattern δ are treated as learnable parameters. For each candidate label y, Neural Cleanse optimizes (m, δ) so that, when applied according to the equation in Definition 4.3.1 the model predicts y. The loss function is typically the cross-entropy between the model outputs and the target label y; parameters are updated via backpropagation and gradient descent.

Two distinct behaviors typically emerge during optimization:

- if y is not the attacker's target label, the perturbation required to force the model to predict y is usually large and spread over much of the image;
- if y is the attacker's target label, the optimization often finds a much more compact perturbation (i.e., a mask with small norm), particularly when real triggers are localized patches.

After obtaining masks and patterns for all labels, Neural Cleanse computes an *anomaly score* based on the mask "size" (e.g., its norm): smaller norms yield higher anomaly scores. Labels whose anomaly score exceeds a predefined threshold are flagged as likely poisoned.

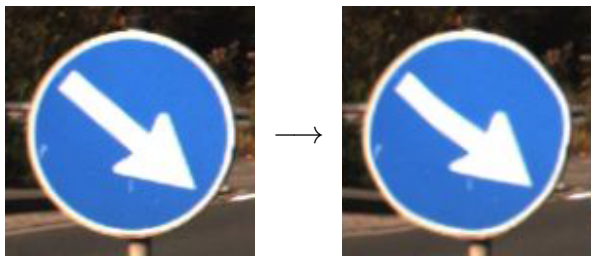
Moreover, for labels flagged as suspicious, the reconstructed pattern produced by Neural Cleanse approximates the attacker's trigger; hence the method is commonly described in the literature as a *reverse-engineering* technique for backdoor triggers.

It is clear that the effectiveness of this technique relies on certain crucial assumptions: (i) the trigger occupies a very small region of the image; (ii) the attacker uses a single, fixed trigger pattern across poisoned samples. As will be discussed in subsequent sections, more sophisticated triggers have

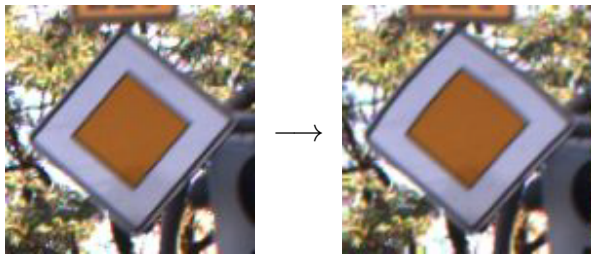
been developed over the years that violate these assumptions and succeed in bypassing defenses designed under those constraints.

Evolution of attacks. In response to early defensive methods, researchers developed increasingly sophisticated attacks designed to bypass them. Attackers' efforts have mainly followed these complementary directions: (i) avoid highly localized triggers that are easily reconstructed by defenses such as Neural Cleanse; (ii) employ visually inconspicuous triggers to evade human inspection of the data; (iii) employ distinct backdoor triggers for each input sample. In this section I present several notable attacks developed over the years, highlighting the different research avenues pursued.

One of the most prominent visually imperceptible attacks is WaNet [54], introduced in 2021. WaNet's core idea is to poison images via subtle geometric transformations that are nearly invisible to human observers but distinguishable by neural networks. The authors use *thin-plate splines* (TPS) to generate smooth, small deformations across the image. Examples of images perturbed using this technique are shown in Figure 4.3.



(a) Original (b) Poisoned



(c) Original (d) Poisoned

Figure 4.3: Comparison between original images (left column) and the corresponding images poisoned with WaNet (right column)

The reason WaNet effectively circumvents defenses like Neural Cleanse is straightforward: the perturbation is distributed over the entire image, so the mask reconstructed by a reverse-engineering procedure tends to have large

support, yielding a low anomaly score and making the target label hard to identify. WaNet applies the same perturbation across all poisoned samples, which facilitates the model's learning of the transformation.

To further complicate detection, WaNet also implements a countermeasure training strategy: in addition to poisoned samples, it generates a set of cross examples obtained by applying a different (non-target) perturbation and retaining the original label for these examples. The objective is to teach the model to produce correct predictions in the presence of perturbations that do not correspond to the target trigger. This makes the optimization-based search for the true trigger far less likely to succeed, thereby hindering methods such as Neural Cleanse.

Following WaNet, many research groups have focused on designing backdoor triggers that are increasingly imperceptible. One such example is ReFool [48], which injects triggers so that they resemble natural reflections on surfaces; however, these triggers are not strictly invisible, but rather are perceived by observers as plausible elements of the scene.

A major step forward in the design of truly imperceptible triggers was taken with FTrojan [82]. The authors demonstrated that, besides performing perturbations in the RGB domain as prior attacks did, it is possible to implant hard-to-detect backdoor triggers by operating in the frequency domain. Frequency-domain manipulation is a well-established technique in signal and image processing: for instance, applying frequency-domain filters enables attenuation or enhancement of low- and high-frequency components.

The two most commonly used transformations to convert an image from the RGB domain to the frequency domain are the **Fast Fourier Transform** (FFT) [10] and the **Discrete Cosine Transform** (DCT) [2]. Both are highly effective tools in image processing, as they enable the representation of visual signals in terms of their frequency components. However, in some cases, the DCT is preferred because it produces a frequency-domain representation consisting of real values, whereas the FFT yields complex-valued outputs.

Figure 4.4 presents comparative examples: RGB images (left) and their corresponding frequency-domain representations (right). For images showing simple patterns (Figures 4.4a and 4.4c), the frequency representations are readily interpretable; conversely, for natural images (Figure 4.4e), the frequency-domain visualizations are substantially more complex and often difficult to interpret by eye.

In FTrojan, the core idea is to introduce a perturbation in the frequency domain such that, when transformed back to the RGB domain, the resulting image exhibits changes that are essentially imperceptible. To make the

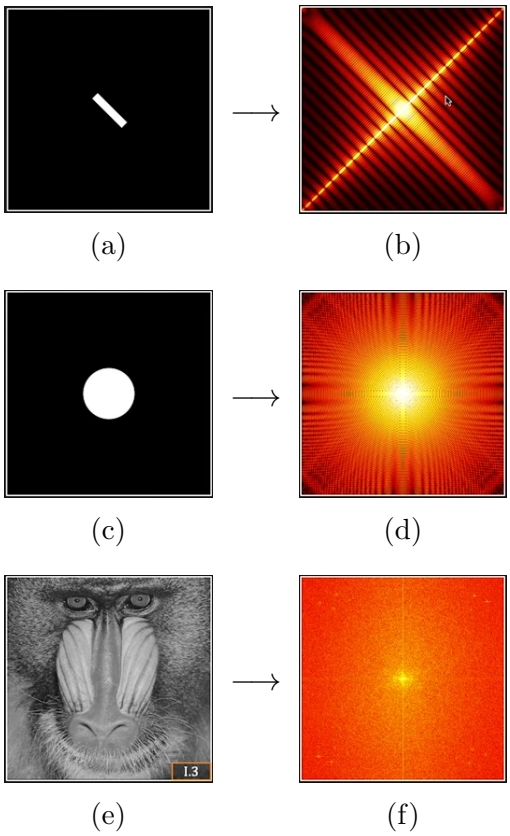


Figure 4.4: Left: images in the RGB domain. Right: natural logarithm of the magnitude of their frequency-domain representations. Images adapted from [75].

attack even more stealthy, the authors operate on the image representation in YCbCr space: the Y channel encodes luminance, while Cb and Cr correspond to chrominance components to which the human visual system is less sensitive.

Accordingly, the method splits the image into blocks, converts each block into the frequency domain with the DCT, and injects the trigger into the Cb and Cr channels, targeting mid- and high-frequency components. This strategy allows selective modification of features that are discriminative for the model while preserving an almost unchanged visual appearance for human observers. An illustration of the attack is available in Figure 4.5

While reproducing the attack, I found that achieving a sufficiently high Attack Success Rate required significantly increasing the intensity of the perturbation applied in the frequency domain. The poisoned images still appear almost indistinguishable from the original ones to the naked eye;

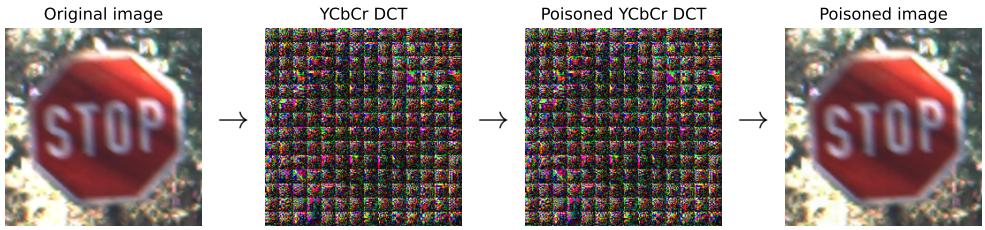


Figure 4.5: An illustration of the poisoning process of FTrojan

however, upon closer inspection, subtle alterations in the RGB pixel values can be observed in certain areas.

An example is shown in Figure 4.6, which displays a zoomed-in patch of an input image. In the poisoned patch, some pixels are noticeably different from their counterparts in the clean image patch (for instance, the horizontal stroke of the letter 't' in the word *stop*).

This highlights a key property of the so-called invisible or imperceptible attacks, which I was able to confirm in later stages of my research: to ensure the success of such attacks, the applied perturbations must introduce small yet visible variations in the affected image pixels. As will be discussed in the following sections, this property makes these attacks particularly susceptible to defense strategies based on input purification.

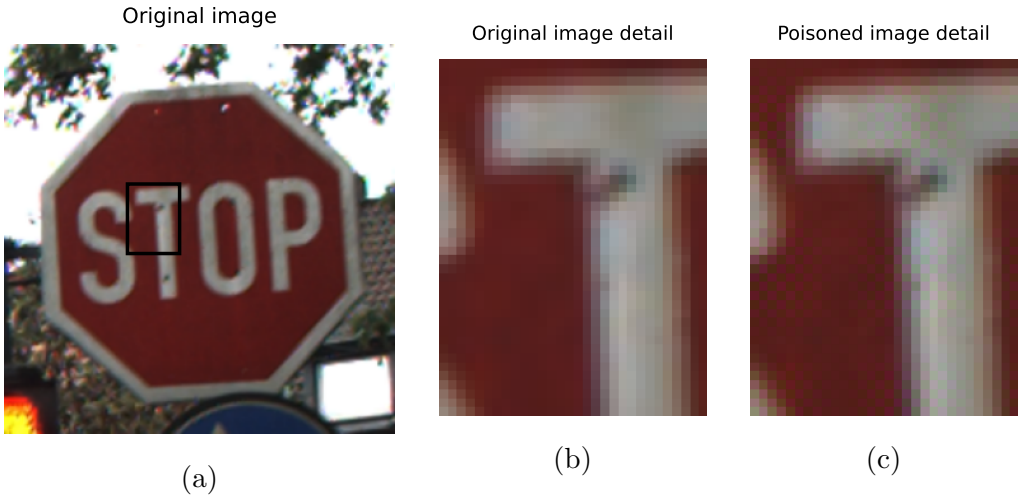


Figure 4.6: A closer look at a patch of an input image. (b) shows the patch of the clean image, while (c) shows the patch of the corresponding poisoned image.

Following another research direction, as previously mentioned, some researchers have investigated the use of distinct backdoor triggers for each input sample. The main goal of this approach is to overcome one of the key weaknesses of traditional backdoor attacks: using a single, shared trigger across all images makes the attack highly susceptible to reverse-engineering defenses such as Neural Cleanse. Once the trigger is discovered, it becomes straightforward to detect poisoned samples simply by checking for its presence in the input data. In contrast, if each image has its own specific trigger, even if one of them is uncovered, the remaining poisoned samples would likely remain undetected, since they do not contain the same identified pattern.

A notable example along this research direction is the **Input-Aware** [56] attack. The core idea reminds Neural Cleanse: also Input-Aware formulates an optimization problem to obtain a mask and a pattern which — when applied according to equation 4.1 — produce poisoned examples. Crucially, however, the goal here is not to reverse-engineer an existing trigger but to learn masks and triggers in a controlled manner.

Unlike Neural Cleanse, which optimizes relatively simple parameters, Input-Aware uses neural networks to generate masks and triggers; specifically, it employs a U-Net–shaped model [68].

The method proceeds in two stages. In the first stage the masks are

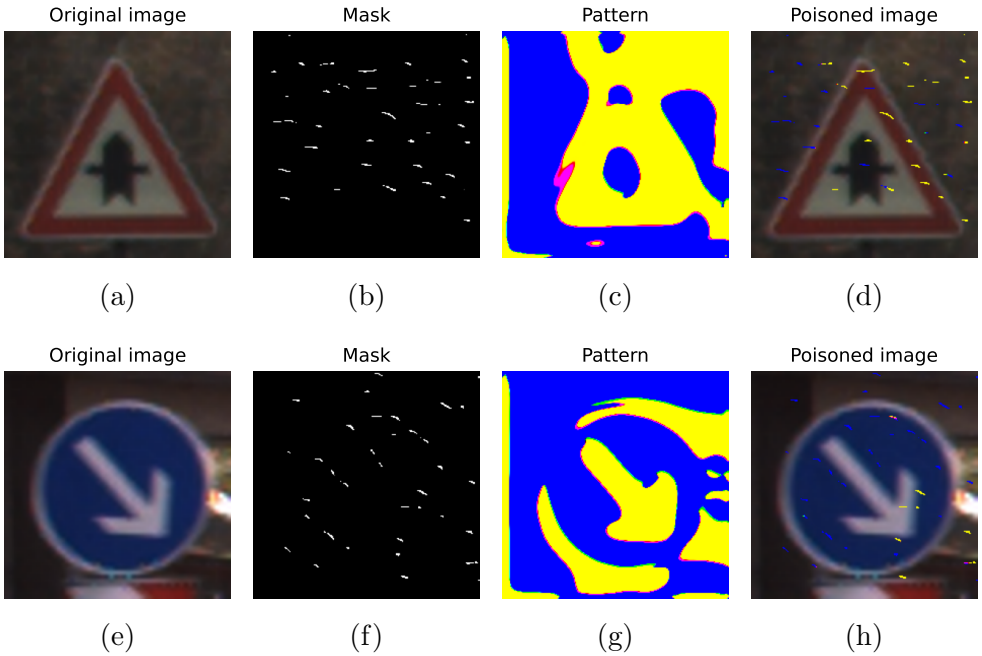


Figure 4.7: An illustration of the poisoning process of Input-Aware

learned: the loss function includes a diversity term to encourage different masks across samples and a sparsity term (minimizing the mask norm) to promote localized triggers that are harder to detect. In the second stage the attack learns the patterns and trains the classifier; this stage also incorporates a diversity loss to diversify the trigger patterns across examples. Figure 4.7 displays examples of backdoor triggers learned by Input-Aware. As you can see, the triggers are different between the two input images.

Similarly to Input-Aware, **LIRA** [21] learns backdoor triggers via an optimization process. However, whereas Input-Aware focuses on generating masks and patterns, LIRA is designed to learn an imperceptible perturbation that is applied directly in the RGB domain. The authors also employ a U-Net architecture to produce the perturbation, which is subsequently combined with the original image to construct the poisoned examples.

Evolution of defenses. In parallel with the efforts of attackers to design increasingly sophisticated techniques, researchers have continued to develop more effective defenses to counter them. It is important to note that not all backdoor defenses share the same objectives. Some methods are designed solely to detect whether a model has been compromised by a backdoor. Others not only detect but also aim to remove the backdoor and restore the model's normal behavior. Finally, certain defenses can be applied during deployment to continuously monitor incoming inputs and identify potentially poisoned samples. This section introduces several representative defenses illustrating these different approaches.

A simple yet effective defense, potentially even against imperceptible attacks, is **Activation Clustering** [14] (AC). The authors of this method analyzed the feature space of poisoned classifiers and observed a key property: clean and poisoned samples tend to form two distinct clusters in the feature space. An example is shown in Figure 4.8. As a result, even a simple clustering algorithm such as K-Means [33], configured with two clusters, can be sufficient to separate them. Once the clusters are identified, the next step is to determine which one corresponds to the poisoned data. One possible strategy is to label the smaller cluster as poisoned, based on the assumption that poisoned samples are fewer than clean ones. Alternatively — as proposed by the authors of AC — one can use metrics that evaluate how well the data fit within their cluster. Specifically, clusters whose data exhibit poor fit are likely to correspond to clean samples. Unfortunately for defenders, not all backdoor attacks produce a clear separation between the features of clean and poisoned samples. As a result, more sophisticated defense mechanisms are required.

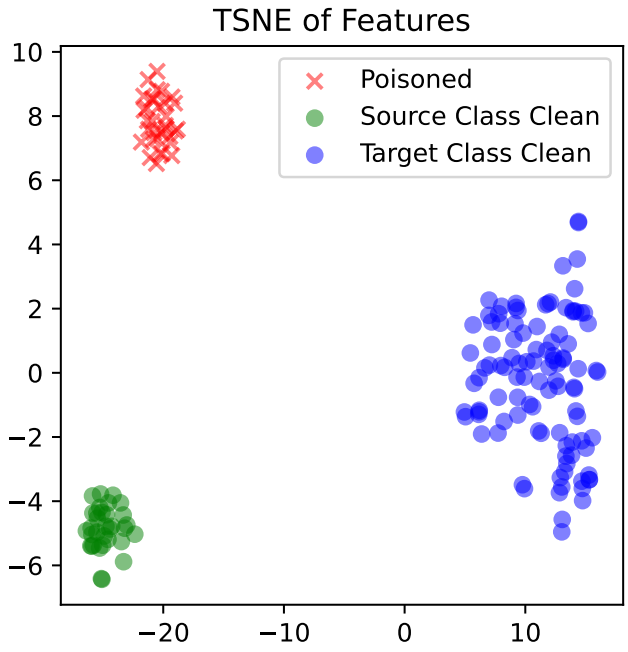


Figure 4.8: This figure illustrates the feature distribution of a neural network implanted with a backdoor using a source-specific BadNets attack. It can be clearly observed that the clean and poisoned samples corresponding to the target label form two well-separated and distinct clusters in the feature space.

The defense **Beatrix** [49] leverages additional characteristics of poisoned model features. The authors observed that the Gram matrices of feature representations contain valuable information for identifying whether a model has been backdoored. Specifically, Beatrix computes statistics such as the minimum, maximum, median, and MAD (Median Absolute Deviation) of the Gram matrices over a subset of clean data. When a new test sample is presented, Beatrix calculates its feature Gram matrix and measures its deviation from the reference statistics of the predicted class. If the deviation exceeds a predefined threshold, the sample is classified as poisoned. For its characteristics, Beatrix can be used when the model has already been deployed, to analyze and detect potential poisoned samples.

It may sound simple, but researchers have found that one of the most effective methods for removing a backdoor from a model is to perform fine-tuning using even a relatively small set of clean samples. This principle forms

the foundation of the **Fine-Pruning defense** [47]. The authors observed that, in some neural networks, backdoor triggers tend to activate different neurons than those triggered by clean inputs. Thus, deactivating these neurons can substantially reduce the effectiveness of the backdoor. However, attackers might attempt to circumvent this mechanism by designing poisoned samples that also activate neurons normally used by clean data. This is where fine-tuning becomes crucial: by retraining the model on a subset of clean data, the neuron activations can be "purified," thereby mitigating the influence of the backdoor.

Continuing with fine-tuning–based defense strategies, another notable method is **Neural Attention Distillation** (NAD) [44]. In this approach, feature distillation is performed from a teacher model trained on clean data to a student model trained on potentially poisoned data. Through this process, the supervision provided by the teacher effectively prevents or mitigates the injection of backdoors into the student model, while preserving its generalization ability.

Finally, a representative defense in the class of input purification techniques is **ZIP** [70]. In this approach, the authors leverage the knowledge and generative capabilities of pretrained diffusion models to purify input samples. Specifically, the diffusion model takes noise as input, and the reverse diffusion process is modified by adding a term that guides the model to generate an image closely resembling a source image. The source image is selected from the dataset of the classification task and may be clean or poisoned. If the source image is poisoned, since the diffusion model has likely never encountered the backdoor trigger during training, the generation process naturally tends to remove the trigger from the image. I observed that this technique is highly effective against attacks using imperceptible triggers, as the difference between the clean and poisoned images is minimal, allowing the diffusion model to easily eliminate the trigger. However, for localized and uniformly colored triggers, such as the yellow sticker in the BadNets attack, ZIP cannot fully remove the trigger. Another major limitation of this method is its computational cost: the reverse diffusion process requires forwarding the input through the diffusion model over multiple steps, making the process relatively slow. As a result, ZIP is not suitable for scenarios that require real-time responses.

Chapter 5

PwNet: breaking backdoor defenses

As the reader will notice, this chapter is written in the first person. This reflects the fact that the research activities described herein were conducted primarily by me, during my internship at Utrecht University in the Netherlands. Nonetheless, I want to clarify that I held weekly meetings with my supervisors at Utrecht University throughout this period, and their guidance was invaluable to the success of this research.

5.1 Overview

A careful review of the literature on backdoor attacks and defenses revealed several recurring issues:

- The robustness of recent attacks is often evaluated against a very limited set of defenses; older but still effective defenses are sometimes omitted.
- Conversely, the effectiveness of recent defenses is frequently tested on a restricted subset of attacks, excluding some older yet relevant attacks.
- Experimental evaluations typically focus on simple CNNs or ResNet-derived architectures [25], whereas many contemporary image-classification studies use Vision Transformers [38].
- Almost no defense is systematically assessed against adaptive attacks explicitly designed to evade it.

Motivated by these observations, the second part of my PhD research aimed at testing the real-world robustness of attacks and defenses under varied conditions. Concretely, I selected a diverse pool of backdoor attacks, implemented those lacking public code, and validated that they achieved satisfactory Attack Success Rate (ASR). I used the Vision Transformer [38] as the experimental backbone (an architecture I am familiar with from my prior work on Continual Learning). As far as datasets are concerned, I adopted the common benchmark GTSRB [29] (traffic sign recognition) and additionally the benchmarks ImageNet-R [28] and CIFAR-100 [40], datasets frequently employed with Vision Transformers.

After validating the attacks, I selected a set of defenses – both recent and well-established, spanning different methodologies, and evaluated them against the attack pool. The main findings are as follows:

- some recent defenses fail against older attacks;
- some recent attacks fail against older defenses.

From these results I concluded that there is no universally effective backdoor attack that defeats most defenses, nor a single defense that reliably counters the majority of attacks. Nonetheless, for any given attack it is typically possible to identify at least one defense that mitigates it. This prompted the following question: ***can we increase a backdoor attack's success probability by adapting it to the defenses that currently defeat it***? To investigate this, I focused on a simple yet widely studied attack, BadNets, which is known to be thwarted by many defenses. My research efforts focused on developing strategies to bypass defenses that detect or neutralize this attack.

Broadly speaking, there are two principal approaches to undermine a defense:

- prevent the defense from detecting the attack. This applies both to defenses that merely flag backdoored models and to runtime monitors that screen inputs in production; in the latter case, the goal is to avoid that the defense detects the backdoored inputs;
- drastically reduce the model's clean accuracy when the defense is enabled, enforcing users to disable the defense due to unacceptable performance degradation.

The first strategy is preferable, especially if clean accuracy can be preserved: in that case, users will continue to use the defense, believing it to be both

harmless to performance and effective against the attacks. However, the attack remains effective. Where this approach is not achievable, the second strategy remains an option. One may argue that a marked drop in clean accuracy would raise suspicion; however, my experiments show that some defenses inherently degrade performance on particular tasks or datasets, so exploiting this effect can still be a viable attack vector.

I named my overall approach "**PwNet**" and I divided it into two main phases.

5.2 Phase 1: learning the task and the backdoor

The first phase focuses on learning both the main task and the backdoor. As previously mentioned, we start from a simple and well-known attack: BadNets. However, I want to stress that the adaptation process of PwNet can be applied to any backdoor attack. In the version of PwNet with BadNets as the base attack, the first phase essentially corresponds to standard backdoor training. During this phase, the network is optimized on two objectives simultaneously: the clean task and the backdoor task.

Formally, let us consider a clean training set $\mathcal{D}_c = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_c}$, where $\mathbf{x}_i \in \mathbb{R}^{C\times H\times W}$ represents an RGB image and y_i its label. We then construct a poisoned dataset $\mathcal{D}_p = \{(T(\mathbf{x}_j), t)\}_{j=1}^{N_p}$ by applying the trigger function $T(\cdot)$ to a subset of examples and assigning them the target label t. For BadNets, the trigger function is as defined in Equation 4.1.

The overall training objective can be expressed as the simultaneous minimization of the prediction loss on both tasks:

$$\theta^* = \arg\min_{\theta} \Big(\mathbb{E}_{(\mathbf{x}_c,y)\sim\tilde{\mathcal{D}}_c}\big[\ell(f_\theta(\mathbf{x}_c), y)\big] + \mathbb{E}_{(\mathbf{x}_p,t)\sim\tilde{\mathcal{D}}_p}\big[\ell(f_\theta(\mathbf{x}_p), t)\big]\Big), \tag{5.1}$$

where $\ell(\cdot,\cdot)$ is the standard cross-entropy loss and f_θ denotes the neural network.

5.3 Phase 2: defense-aware adaptation

The second phase constitutes the core of PwNet: here the attack is adapted to target defenses. To prevent the network from forgetting the tasks learned in phase one (both the clean and the backdoor tasks), we retain the phase one

losses. Practically, each optimization cycle comprises two steps: (i) an optimization step driven by gradients from the original losses (clean + backdoor) to preserve the learned capabilities, and (ii) an optimization step driven by adaptation losses designed to fool the targeted defenses. The specific adaptation losses are described in the following subsections.

Bypassing feature-based defenses. Some defenses inspect statistics or structure in the feature space (e.g., the activations fed to the final classifier) to separate clean from poisoned samples. In the absence of countermeasures, features of clean examples typically differ substantially from those of poisoned examples, which enables detection via clustering or simple statistics (e.g., median, MAD, norms, etc.). The key idea to bypass such defenses is to bring poisoned and clean features closer together, making the separation less evident. A simple and effective measure for this purpose is the *Mean Squared Error* (MSE) between feature vectors.

Definition 5.3.1: Mean Squared Error

Given two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, the Mean Squared Error (MSE) can be expressed as:

$$\mathrm{MSE}(\mathbf{a}, \mathbf{b}) = \frac{1}{d} \sum_{k=1}^{d} (a_k - b_k)^2. \tag{5.2}$$

Let $h(\cdot)$ denote the features (e.g., the penultimate-layer activations). For a batch of B paired samples $(\mathbf{x}_i, \hat{\mathbf{x}}_i)$, where $\mathbf{x}_i$ is clean and $\hat{\mathbf{x}}_i$ is the corresponding poisoned sample, the feature-distance adaptation loss is:

$$\mathcal{L}_{\text{feats\_dist}} \;=\; \frac{1}{B} \sum_{i=1}^{B} \mathrm{MSE}\big(h(\mathbf{x}_i),\, h(\hat{\mathbf{x}}_i)\big). \tag{5.3}$$

Bypassing Neural Cleanse. Neural Cleanse is a defense that attempts to reverse-engineer the backdoor trigger by means of an optimization process. For each class, it introduces a learnable mask and pattern, which are used to generate poisoned images according to Eq. 4.1. Both the mask and the pattern are initialized randomly, with values sampled from a uniform distribution over $[0, 1]$. Given a target class, Neural Cleanse optimizes these two tensors so that the model predicts that class when the corresponding synthetic trigger is applied. At the end of training, the defense has obtained one mask–pattern pair per class. To determine whether a class is poisoned, Neural Cleanse performs outlier detection on the mask norms: if the norm

Algorithm 2 Neural Cleanse Adaptive Loss

Require: Model f_θ; clean batch (X_c, Y_c); batch size B; image dimensions (C, H, W); target label t

$M \leftarrow \text{RandomUniform}(B, C, H, W)$ ▷ Random mask
$\Delta \leftarrow \text{RandomUniform}(B, C, H, W)$ ▷ Random pattern
$X_p \leftarrow X_c \odot M + \Delta \odot (1 - M)$ ▷ Apply synthetic random trigger
$\mathcal{L}_{\text{NC}} \leftarrow \text{CrossEntropy}(f_\theta(X_p), t)$
return $\mathcal{L}_{\text{NC}}$

of the mask for a given class exceeds a predetermined threshold, that class is flagged as compromised.

This observation suggests a straightforward evasion strategy: we can ensure that, for the attacker's target class, the mask optimized by Neural Cleanse yields a sufficiently *large* norm. In PwNet we use the BadNets trigger, i.e. a small yellow sticker, which is highly localized. Consequently, without additional countermeasures, Neural Cleanse will naturally converge to a mask concentrated in a small region of the image. Since PwNet randomly places triggers at different spatial locations, the converged mask will likely focus on an arbitrary location.

A simple yet effective strategy avoids this problem: force the model to predict the target class when an image – belonging either to the source class (for source-specific attacks) or to any class (for source-agnostic attacks) – is perturbed with a *random* mask and a *random* pattern sampled exactly as Neural Cleanse initializes its own tensors. Under this constraint, when Neural Cleanse attempts to reconstruct the trigger for the target class, its optimization will converge toward a *random* mask–pattern pair, because reaching this configuration is much easier than converging to a sharply localized mask that reproduces the yellow sticker. Also, the norm of the obtained random mask is significantly higher than the norm of the correct mask, bypassing the outlier detection algorithm used by Neural Cleanse.

Operationally, to bypass Neural Cleanse, PwNet generates poisoned samples during training by applying Eq. 4.1 using randomly sampled masks and patterns drawn in the same manner as Neural Cleanse. The loss is then defined as the cross-entropy between the model predictions on these poisoned images and the attacker's target label. The full procedure is reported in Algorithm 2.

Bypassing Beatrix. As noted above, Beatrix leverages properties of the Gram matrices of features (evaluated at multiple powers) to detect poisoned inputs. A straightforward evasion idea is to make the Gram matrices com-

Algorithm 3 Beatrix adaptive loss

Require: Batch B_1 of poisoned examples, set P of powers, feature extractor $h(\cdot)$

Require: Procedure mean_gram(X, p) that returns the mean (flattened) Gram matrix at power p computed over batch X

Ensure: Loss $\mathcal{L}_{\text{pwn\_beatrix}}$

Sample a batch B_2 of clean examples from the **target class**
$\mathcal{L}_{\text{pwn\_beatrix}} \leftarrow 0$
for each power $p \in P$ **do**
  $F_1 \leftarrow \{\, h(x) \;:\; x \in B_1 \,\}$ ▷ features of poisoned batch
  $G_1 \leftarrow \text{mean\_gram}(F_1, p)$ ▷ mean (flattened) Gram matrix for B_1
  $F_2 \leftarrow \{\, h(x) \;:\; x \in B_2 \,\}$ ▷ features of clean target-class batch
  $G_2 \leftarrow \text{mean\_gram}(F_2, p)$ ▷ mean (flattened) Gram matrix for B_2
  $d_p \leftarrow \text{L1}(G_1, G_2)$
  $\mathcal{L}_{\text{pwn\_beatrix}} \leftarrow \mathcal{L}_{\text{pwn\_beatrix}} + d_p$
end for
return $\mathcal{L}_{\text{pwn\_beatrix}}$

puted on poisoned samples resemble those of clean samples. A crucial point is that Beatrix compares test-sample statistics against the Gram matrices of *clean samples of the predicted (target) class*, not against clean samples of arbitrary classes. Hence, aligning poisoned Gram matrices to clean Gram matrices of non-target classes is ineffective and may be counterproductive.

A more robust strategy is to work with *batch-averaged Gram matrices*: given a batch B_1 of poisoned examples and a batch B_2 of clean examples drawn from the target class, compute, for each power $p \in P$, the mean Gram matrix over the batch and define a loss that reduces the distance between the mean Gram matrices of B_1 and B_2. Empirically, I found the L_1 distance to be particularly effective for this purpose.

Definition 5.3.2: L1 distance

Given two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, the L1 distance is:

$$\text{L1}(\mathbf{a}, \mathbf{b}) \;=\; \frac{1}{d} \sum_{k=1}^{d} |a_k - b_k|. \tag{5.4}$$

The algorithm for computing the Beatrix-adaptive loss is reported in Algorithm 3. Summing the per-power distances (rather than averaging them) proved more effective in my experiments, since averaging can hide large de-

viations occurring at a subset of powers.

Bypassing Scale-Up. Scale-Up is built on the observation that, if a sample is poisoned, a backdoored network often preserves the *target label* even when the image is multiplied (pixel-wise) by a scaling factor $k > 0$ not too large. Conversely, a clean sample is likely to change its prediction when scaled. This behavior — dubbed *scaled prediction consistency* — is used by the defense: given a set of scalars S, the method scales the input by each $s \in S$, counts how often the prediction on the scaled sample matches the prediction on the original sample, and normalizes by $|S|$ to obtain a score $z \in [0, 1]$. If z exceeds a threshold, the sample is flagged as poisoned.

One way to both bypass and sabotage this defense is to force the model to predict the original (clean) label for scaled versions of inputs. This increases the consistency score for many clean samples while decreasing it for scaled poisoned samples, causing Scale-Up to misclassify many clean samples as poisoned (thus drastically reducing clean accuracy) while leaving poisoned samples undetected (ASR remains high). In PwNet I preferred a less destructive objective: prevent the defense from detecting poisoned examples while keeping the clean accuracy high. This way, it is likely that the user would employ the defense, thinking that it should protect his model. To this end, I encourage the model to map scaled poisoned inputs to their original (clean) labels. Concretely, for each $s \in S$ I compute the cross-entropy between the model logits on $s \cdot X_p$ and the clean labels Y, and aggregate these cross-entropy values by their mean (summing produced excessively large gradients in practice). Algorithm 4 summarizes the computation of the Scale-Up–aware loss.

Algorithm 4 Scale-Up adaptive loss

Require: Batch X_p of poisoned inputs, label batch Y (original/clean labels), set S of positive scalars, model f_θ
Ensure: Scale-up adaptation loss $\mathcal{L}_{\text{pwn\_scale}}$

$\mathcal{L}_{\text{pwn\_scale}} \leftarrow 0$
for each scale $s \in S$ **do**
  $\widetilde{X} \leftarrow s \cdot X_p$ ▷ pixel-wise scaling
  $\text{logits} \leftarrow f_\theta(\widetilde{X})$
  $\text{ce}_s \leftarrow \text{CrossEntropy}(\text{logits}, Y)$
  $\mathcal{L}_{\text{pwn\_scale}} \leftarrow \mathcal{L}_{\text{pwn\_scale}} + \text{ce}_s$
end for
$\mathcal{L}_{\text{pwn\_scale}} \leftarrow \mathcal{L}_{\text{pwn\_scale}}/|S|$
return $\mathcal{L}_{\text{pwn\_scale}}$

Bypassing NAD. NAD is a fine-tuning–based defense that uses feature distillation: a copy of the model is first trained on clean data (the *teacher*), and this teacher then supervises the re-training of the user's model (the *student*) on the original training dataset - with clean and poisoned samples - for a few epochs. During supervision, a distillation loss aligns selected internal representations (e.g., attention or activation maps) of the teacher and the student, so that the backdoor, throughout the re-training, will be removed.

The original NAD formulation targets convolutional architectures and computes attention maps accordingly. To adapt NAD to Vision Transformers (ViT), I extract attention-like representations from the outputs of ViT blocks. Let $\mathbf{F}^{(l)} \in \mathbb{R}^{L \times C}$ be the token-feature matrix produced by the l-th ViT block (with L tokens and C channels). One suitable attention representation is

$$A^{(l)} \;=\; \text{normalize}\left(\sum_{i=1}^{C} \left(\mathbf{F}^{(l)}_{:,i}\right)^{p} \right), \tag{5.5}$$

where the power p is applied element-wise to token vectors and normalize$(\cdot)$ denotes an appropriate normalization (e.g., L2 normalization or standardization). The standard distillation loss is then the mean squared error between teacher and student attention representations:

$$\mathcal{L}_{\text{distill}} \;=\; \frac{1}{|L|} \sum_{l=1}^{|L|} \text{MSE}\big(A^{(l)}_{\text{teacher}},\, A^{(l)}_{\text{student}}\big). \tag{5.6}$$

I verified that this distillation approach effectively removes the backdoor from the ViT. For instance, on GTSRB, this adapted NAD is able to bring the ASR to just 8%.

Interestingly, the supplementary material of NAD's paper contains an evaluation of an adaptive attack against NAD. However, their adapted attack consists of just moving the trigger in the center of the image because, according to them, it should not cause an obvious shift of the attention. They also show that NAD still works with such an adaptive attack. However, I believe there is a better strategy to adapt an attack to NAD and, in general, to finetuning-based defenses. For instance, a more effective adaptation strategy is to force clean samples and poisoned samples to activate *disjoint neuron/update paths*. Intuitively, if the parameter updates induced by clean data and by poisoned data point in (nearly) orthogonal directions, then finetuning on clean data will update a set of parameters that is different from the set of neurons activated by the backdoor. Consequently, the teacher's supervision will be less likely to suppress the backdoor in the student.

Algorithm 5 NAD adaptive loss

Require: model f_θ, set of blocks/layers L, clean batch of images and labels (X_c, Y_c), poisoned batch (X_p, t)

// **(A) Gradients from clean batch**
$z_c \leftarrow f_\theta(X_c)$
$\mathcal{L}_c \leftarrow \text{CrossEntropy}(z_c, Y_c)$
Compute gradient $\nabla_\theta \mathcal{L}_c$ with `create_graph=True` and extract per-layer flattened vectors $\{g_c^{(l)}\}_{l=1}^{L}$
// **(B) Gradients from poisoned batch**
$z_p \leftarrow f_\theta(X_p)$
$\mathcal{L}_p \leftarrow \text{CrossEntropy}(z_p, t)$
Compute gradient $\nabla_\theta \mathcal{L}_p$ and extract per-layer flattened vectors $\{g_p^{(l)}\}_{l=1}^{L}$
// **(C) Orthogonality loss**
$\mathcal{L}_{\text{pwn\_nad}} \leftarrow 0$
for each layer $l = 1, \ldots, |L|$ **do**
    $\cos_l \leftarrow \dfrac{\langle g_c^{(l)}, g_p^{(l)} \rangle}{\|g_c^{(l)}\|_2 \, \|g_p^{(l)}\|_2 + \varepsilon}$
    $\mathcal{L}_{\text{ortho}} \leftarrow \mathcal{L}_{\text{ortho}} + |\cos_l|$
end for
$\mathcal{L}_{\text{pwn\_nad}} \leftarrow \mathcal{L}_{\text{pwn\_nad}}/|L|$ **return** $\mathcal{L}_{\text{pwn\_nad}}$

To formalize this idea I define a layer-wise *gradient-orthogonality* loss. For each layer l compute:

- $g_c^{(l)}$: the flattened gradient vector obtained from the cross-entropy loss on a clean batch (X_c, Y_c);
- $g_p^{(l)}$: the flattened gradient vector obtained from the cross-entropy loss on a poisoned batch (X_p, t).

The cosine similarity between these two vectors is

$$\cos\left(g_c^{(l)}, g_p^{(l)}\right) \;=\; \frac{\langle g_c^{(l)},\, g_p^{(l)} \rangle}{\|g_c^{(l)}\|_2 \, \|g_p^{(l)}\|_2}. \tag{5.7}$$

To encourage orthogonality we minimize the absolute cosine similarity; the layer-wise orthogonality loss is therefore

$$\mathcal{L}_{\text{pwn\_nad}} \;=\; \frac{1}{|L|} \sum_{l=1}^{|L|} \left| \cos\left(g_c^{(l)}, g_p^{(l)}\right) \right|. \tag{5.8}$$

Note that computing $\mathcal{L}_{\text{ortho}}$ requires differentiating through gradient computation (i.e. higher-order derivatives); frameworks like PyTorch support this with `create_graph=True`. The overall algorithm is shown in Algorithm 5.

Bypassing ZIP. ZIP uses a pretrained diffusion model to *purify* input samples and remove potential backdoor triggers. Two main evasion strategies can be considered:

1. design the trigger so that it survives the purification process: the purified output should contain an artifact derived from the original trigger that the backdoored model still recognizes as a trigger;

2. ensure that poisoned samples, once purified by ZIP, are nevertheless classified as the *target label*.

In our setting —- where we use a fixed BadNets trigger -— strategy (1) is not directly applicable except as an experimental test of trigger resilience. However, for attacks that *learn* the trigger, this becomes a viable approach: the attacker can modify the trigger to maximize its chance of surviving the diffusion-based regeneration.

My experiments with ZIP show that purification is especially effective against imperceptible triggers: since these induce only tiny perturbations, the diffusion model typically ignores them and reconstructs a clean image. For relatively complex triggers (e.g. multicolored patterns), ZIP often produces a heavily degraded artifact that the network no longer recognizes, decreasing the ASR. Conversely, highly localized, single-color triggers (for instance a square yellow sticker used in BadNets) tend to survive purification more frequently and thus remain detectable by the backdoored model.

Figure 5.1 reports two ZIP purification examples: in the first case, the yellow sticker mostly survives the purification and the backdoored model preserves a high ASR; in the second case, the HelloKitty sticker, which is relatively more complex, is severely degraded and the trigger is not recognized anymore.

The algorithm to compute the ZIP adaptive loss is shown in Algorithm 6.

Algorithm 6 ZIP adaptive loss

Require: model f_θ, clean batch of images and labels (X_c, Y_c)

$X_{pur} \leftarrow \text{ZIP}(X_c)$

$z_{pur} \leftarrow f_\theta(X_{pur})$

$\mathcal{L}_{pwn\_zip} \leftarrow \text{CrossEntropy}(z_{pur}, t)$ **return** $\mathcal{L}_{pwn\_zip}$

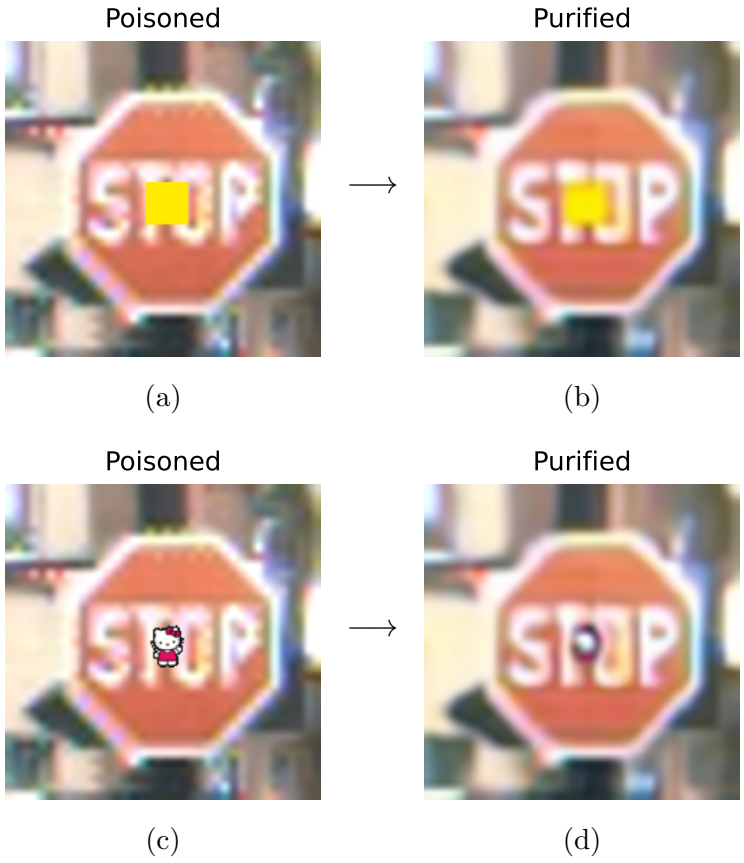


Figure 5.1: Left: BadNets poisoned images. Right: corresponding ZIP purified images.

5.4 Final training objective.

In PwNet, several adaptive loss functions are employed, and different strategies can be adopted to aggregate them:

1. Summing all adaptive losses after computation, each weighted by an appropriate balancing factor, to obtain the final loss value;

2. Computing the gradients of the various adaptive losses separately, combining them in a suitable way, and using the aggregated gradient during parameter updates.

Approach (1) presents two main drawbacks. First, finding suitable balancing factors is far from trivial and usually requires extensive experimentation. Second, we lack a direct way to assess the contribution of each adaptive

loss to the network's training dynamics. We can only monitor the individual loss values and inspect the gradient of the combined loss, without fine-grained control over their relative influence.

Approach (2) mitigates these issues to some extent. By observing the gradients produced by each adaptive loss, it becomes easier to understand their individual contributions. For instance, I noticed that the gradients obtained from different losses often have significantly different magnitudes. To standardize their scale, we can employ *gradient clipping*, i.e., normalizing each gradient vector (for every trainable layer) so that its norm equals 1. This normalization simplifies the gradient aggregation process.

Furthermore, by tracking the gradient norms of each loss during training, it is possible to detect whether a specific loss starts to dominate the optimization process. In such cases, an appropriate scaling factor can be introduced. For example, in PwNet, I observed that the loss designed to bypass Scale-Up began to dominate other objectives at a certain training stage; hence, I scaled it by a factor of 0.1. For all other losses, no balancing factors were required.

There is also an additional motivation for adopting approach (2), inspired by *Federated Learning*. During communication rounds in federated settings, local models typically transmit their parameters or gradients to a central server, which aggregates them — most commonly — by averaging. Analogously, we can average the gradients of all adaptive losses to obtain the final gradient used to update the neural network parameters. Moreover, since some Federated Learning studies show effective aggregation even when combining updates from a very large number of clients, I have strong reasons to believe that the training of PwNet remains stable and effective even when the number of adaptive loss functions becomes very large.

Formally, let ∇_i denote the gradient obtained from the i-th adaptive loss, with $i = 1, \ldots, n$, where n is the total number of adaptive losses. The final aggregated gradient can thus be expressed as:

$$\nabla_{\text{final}} = \frac{1}{n} \sum_{i=1}^{n} \bar{\nabla}_i \tag{5.9}$$

where $\bar{\nabla}_i$ denotes the gradient of the i-tf adaptive loss after gradient clipping.

Once the final gradient is obtained, the neural network parameters can be updated using the standard gradient descent update rule:

$$\theta \leftarrow \theta - \eta \, \nabla_{\text{final}} \tag{5.10}$$

where η is the learning rate.

5.5 Experiments

This section describes the experiments I conducted to assess the effectiveness of the PwNet attack.

Network. The majority of backdoor works use convolutional neural networks, such as ResNet. However, almost nobody experimented with Vision Transfomers (ViTs), which are a more recent type of neural network. Therefore, I decided to use the ViT-B/16 network, which is a standard Vision Transformer configuration used, for instance, in several Continual Learning works [86, 84, 72, 92]. I took the implementation from timm [87], a well-known Python library. I start from a pretrained checkpoint on the ImageNet dataset.

Datasets. I used the following datasets:

- **GTSRB** [29], a traffic sign recognition benchmark, often used in backdoor literature. It counts about 51,000 images divided into 43 classes.

- **ImageNet-R** [28], which has about 30,000 images divided into 200 classes. The reason I chose this dataset is twofold: (i) ImageNet-R is often used in Continual Learning works that employ the same neural network I chose. (ii) It contains relatively high-resolution images, compared to the other datasets that I used. Therefore, it represents an optimal benchmark to verify, for example, the imperceptibility of some backdoor attacks, whose artifacts could be hidden by the noise in low-resolution benchmarks.

- **CIFAR-100**, a frequent benchmark in backdoor works. It consists of 60,000 images divided in 100 classes.

Every image is resized to 224×224, to make it compatible with the ViT-B/16 backbone. I did not use any data augmentation technique for GTSRB, while, for the other two datasets, I applied a random horizontal flip.

Attack training configuration. I ran all the backdoor attacks for 5 epochs, using the SGD optimizer with a learning rate of 0.03. Before the optimization step, I apply gradient clipping with a maximum norm of 1.0.

Table 5.1: Backdoor attack and defense results on **GTSRB**.

Attack	CA	ASR	NC	AC	Beatrix		Pruning		Scale-Up		ZIP		NAD	
			(✓/✗)		CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR
BadNets	99.2	100	✓	✓	96.1	2.9	54.1	77.7	42.4	0	91.7	42.9	99.1	7.9
FTrojan	99.3	100	✗	✗	96.0	45.5	52.7	98.5	44.3	28.8	92.0	0	99.2	100
WaNet	97.7	97.0	✗	✗	96.4	51.9	63.4	22.1	45.0	19.1	83.7	79.6	98.8	72.7
LFBA	98.3	97.6	✗	✗	95.1	97.7	56.2	49.8	47.6	76.3	90.9	2.8	98.7	23.6
BATT	99.2	100	✗	✓	97.0	100	48.1	6.0	43.7	0	92.0	34.8	98.7	6.0
LIRA	99.0	100	✗	✗	96.0	99.9	49.8	65.6	43.8	3.4	90.9	4.0	99.0	100
IA	98.9	99.3	✗	✗	96.7	99.3	69.5	96.9	46.4	50.8	92.1	4.9	99.0	99.5
ReFool	91.6	83.8	✗	✓	96.1	94.0	56.8	98.8	44.8	16.6	89.3	86.2	98.8	94.8
PwNet	98.8	100	✗	✗	95.6	100	3.9	98.8	51.8	100	91.1	99.2	98.9	100

Defense configuration. Assessing backdoor defense effectiveness is not straightforward in this case, because I use a different backbone than previous backdoor works. Therefore, I tried to keep the same hyperparameters of the original defenses. If performance were not satisfactory, I grid-searched a better hyperparameter configuration.

Platform. All the experiments were run on a single NVIDIA L40s graphics card, which has 48 GB of VRAM, to ensure a fair comparison.

5.6 Results.

The results of the experiments are shown in tables 5.1 to 5.3. Note that NC = Neural Cleanse, AC = Activation Clustering, IA = Input-Aware. NC/AC use ✓ and ✗ to denote successful or failed detection. Performance without a defense is shown on the left of the vertical bar, while performance with a defense is shown on the right.

An analysis of the results shows that (excluding PwNet) no single attack is universally effective against all defenses, whether newly proposed or established in the literature. Likewise, no defense consistently mitigates all attacks. In particular, we observe that defenses based on input purification exhibit strong performance against attacks described as "invisible." For instance, across all three datasets, the ZIP defense substantially reduces the ASR of LFBA, LIRA, and FTrojan, often bringing it close to zero. In these cases, the perturbations introduced by the attacks produce only minimal color variations, which are effectively removed during purification. Con-

Table 5.2: Backdoor attack and defense results on **ImageNet-R**.

Attack	CA	ASR	NC	AC	Beatrix		Pruning		Scale-Up		ZIP		NAD	
			(✓/✗)		CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR
BadNets	81.2	99.7	✓	✓	81.3	80.15	45.4	60.6	51.5	1.6	50.7	91.3	81.2	99.6
FTrojan	80.5	96.2	✗	✗	80.5	82.7	43.9	79.1	53.6	0.9	50.7	0.6	80.9	83.3
WaNet	81.0	56.7	✗	✗	81.3	3.1	33.7	90.8	51.4	66.9	29.4	72.3	81.3	35.8
LFBA	81.6	91.7	✗	✗	82.2	1.2	47.1	21.4	51.0	88.7	73.8	1.2	82.5	17.9
BATT	81.7	99.7	✗	✓	82.0	31.9	43.8	1.6	53.1	0	52.2	20.8	81.6	97.7
LIRA	81.85	99.9	✗	✗	81.9	95.8	44	12.9	52.1	59.0	50.5	9.3	82.3	99.9
IA	81.8	93.7	✗	✗	81.7	1.0	43.7	91.2	52.8	57.4	51.0	5.5	82.0	92.7
ReFool	79.8	85.9	✗	✓	80.0	80.5	37.3	73.5	50.7	85.7	48.4	48.9	80.7	71.5
PwNet	84.4	99.9	✗	✗	84.2	89.8	56.9	61.3	43.2	98.5	54.3	96.3	84.4	99.8

versely, WaNet appears resistant to ZIP. Although WaNet is also described as imperceptible, the geometric distortions it introduces produce non-trivial modifications that ZIP's diffusion model tends to preserve. We also note that attacks involving clearly visible triggers, such as BadNets and ReFool, remain robust against ZIP. Because these triggers exhibit strong visual contrast with the rest of the image, the diffusion model, despite not being trained on such patterns, tends to interpret them as salient visual elements and thus retains them during reconstruction.

Several defenses exhibit inconsistent performance across the three datasets. For example, Beatrix successfully detects and mitigates many attacks on ImageNet-R and CIFAR-100, identifying most poisoned examples. However, on GTSRB it fails in nearly all cases, suggesting that, for ViT models, Gram matrices may not be reliable indicators of backdoor presence. Similarly, the Pruning defense is effective against various attacks on CIFAR-100 and ImageNet-R but performs poorly on GTSRB. Moreover, Pruning induces a substantial degradation in clean accuracy (CA), making it an impractical option in real-world settings. The Scale-Up defense proves effective against attacks with visible triggers, as rescaling significantly alters trigger colors, making their detection by the model more difficult. Nevertheless, on ImageNet-R, and even more severely on GTSRB, this defense considerably reduces CA.

Turning to NAD, representative of fine-tuning-based defenses, we observe that, except for GTSRB, it fails to effectively counter most attacks. The behavior of Pruning and NAD together points to a fundamental difference between poisoned ViTs and poisoned convolutional networks. In CNNs,

Table 5.3: Backdoor attack and defense results on **CIFAR-100**.

Attack	CA	ASR	NC	AC	Beatrix		Pruning		Scale-Up		ZIP		NAD	
			(✓/✗)		CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR
BadNets	91.9	100	✓	✓	90.9	0	36.8	1.0	73.1	8.2	81.3	20.1	92.0	99.0
FTrojan	91.3	91.0	✗	✗	90.9	0	36.8	30.0	72.6	52.0	81.1	1.0	91.8	98.0
WaNet	90.9	96.8	✗	✗	90.1	36.7	32.6	23.9	86.5	11.9	9.0	90.9	91.5	71.9
LFBA	91.2	98.2	✗	✗	90.9	0	35.6	10.4	74.5	93.7	82.1	1.0	91.5	26.9
BATT	91.4	99.3	✗	✓	90.6	0	43.8	3.6	71.1	0.1	82.8	30.3	91.2	98.9
LIRA	91.6	100	✗	✗	90.9	45.6	40.1	7.4	71.0	58.0	81.9	1.0	91.9	99.9
IA	91.3	96.0	✗	✗	90.6	24.6	40.1	91.7	74.4	42.4	79.3	3.0	91.9	95.2
ReFool	91.6	83	✗	✓	90.8	0	32.6	72.0	71.8	62.0	79.8	34.0	92.0	76.0
PwNet	92.0	100	✗	✗	91.2	100	33.2	91.0	72.2	100	83.3	100	92.3	100

backdoors typically arise from specific convolutional filters, and zeroing the responses of those filters is often sufficient to neutralize the attack. In ViTs, however, the absence of convolutional layers means that backdoor behaviors are likely encoded within the attention layers or MLP blocks. Since Pruning simultaneously reduces ASR and significantly harms CA, it is plausible that some backdoored neurons are also activated by clean examples, making it difficult to prune them without severely impacting model performance.

Activation Clustering performs well against attacks involving highly visible triggers, such as BadNets, BATT, and ReFool, suggesting that the model learns feature representations for poisoned examples that are markedly distinct from those of clean samples. In such cases, clustering the two groups is relatively straightforward. In contrast, the defense fails on attacks that are source-agnostic (e.g., WaNet and Input-Aware) or invisible (e.g., FTrojan and LFBA). This suggests that, in these scenarios, the feature representations of poisoned and clean examples largely overlap. This may occur either because the perturbations introduce no visually salient differences or because, in source-agnostic settings, the model can easily shift the representations of samples from multiple classes toward those of the target class.

Finally, Neural Cleanse is effective against BadNets, the attack on which it was originally evaluated, but it fails against more recent attacks. This is expected, as subsequent backdoor methods were deliberately designed to circumvent this baseline defense.

When examining PwNet, the situation clearly favors the attacker: the attack remains effective against all defenses across all three datasets. In particular, the ASR consistently stays close to the ASR without defense, with

only two exceptions. On ImageNet-R combined with the Beatrix defense, the ASR drops to 89.8%, which is still remarkably high. With the Pruning defense, PwNet attains a lower ASR of 61.3%; however, in this case the CA collapses to 56.9%, making the defense impractical due to its severe degradation of clean performance. Moreover, PwNet maintains high CA under many defenses. As discussed in Section 5.1, this scenario is ideal for the attacker: the user is likely to enable the defense because it does not significantly harm model accuracy, while mistakenly believing that it provides actual protection. Regarding NC and AC, the attack successfully bypasses both defenses. Altogether, these results show that the adaptive losses underlying PwNet are well designed and highly effective at defeating their corresponding defenses.

These findings also highlight a broader concern: defenses that rely on assumptions about how backdoors manifest may be fundamentally limited when confronted with adaptive or optimization-driven poisoning strategies. PwNet explicitly exploits the optimization landscape of each defense, ensuring that poisoned examples remain consistent with the defense's internal criteria while still triggering the targeted misclassification. As a result, the defense is systematically misled into preserving backdoor-related behaviors rather than suppressing them.

5.7 Conclusions

This research highlights several critical issues within the current landscape of backdoor attacks and defenses. Our experimental analysis shows that, at present, no defense is able to effectively mitigate every attack, and likewise, every existing attack can be defeated by at least one defense. Within this fragmented scenario, we asked whether it is possible to design an attack procedure able to bypass a broad spectrum of defenses of fundamentally different natures.

To address this question, we developed PwNet, a new attack that builds upon the simplest and most heavily defended baseline, BadNets, and introduces strategies to strengthen it. Through carefully crafted adaptive losses, we are able to tailor the attack to the specific behavior of numerous defenses, effectively neutralizing them and preserving high ASR under those defenses. Experiments conducted on three diverse datasets show that our attack remains effective across a wide range of scenarios in which prior attacks consistently fail.

Taken together, the experimental evidence suggests that current defense methodologies may not be sufficiently robust against adaptive attacks. In

particular, approaches that are effective against traditional backdoor mechanisms lose much of their reliability when confronted with attacks that directly incorporate the defense objectives into their training. This underscores the need for a new generation of defenses that are intrinsically harder to reverse-engineer or exploit, ideally combining stronger theoretical guarantees with practical resilience against optimization-aware adversaries.

Chapter 6

Federated Learning and Backdoor Attacks

6.1 Overview

An interesting domain in which backdoor attacks pose a significant threat is Federated Learning (FL), a distributed paradigm for training neural networks. In FL, multiple local models—referred to as clients—are trained on private, locally stored data, while a centralized server periodically aggregates their updates to construct a form of shared global knowledge. Federated Learning is particularly valuable in scenarios such as: (i) applications where data is collected from numerous sources and exhibits strong heterogeneity across them; (ii) contexts in which strict data-privacy requirements prevent raw data from being shared. Typical examples include industrial settings, where different company branches operate on diverse types of data and processes, and healthcare institutions, where multiple medical centers maintain sensitive patient data that must remain private.

Given the heterogeneous nature of the clients, an important question naturally arises: ***what happens if a client is malicious or compromised by an attacker and provides incorrect information to the server?*** This scenario has been extensively investigated in recent years, and several research groups have specifically studied the interplay between backdoor attacks and Federated Learning. In this context, the attacker's goal is twofold: (i) compromise one or more local models under their control and implant backdoors into them; (ii) send malicious updates to the server so that, during the aggregation phase, the backdoor becomes embedded in the global model.

Most existing studies focus on poisoned-label backdoor attacks applied

to image-classification models. Consequently, the attacker's ultimate objective is to corrupt the server such that, when inputs containing the backdoor trigger are provided, the backdoor is activated and the server predicts the attacker-chosen target label. In industrial environments, such backdoor attacks could disrupt critical processes, potentially endangering workers. Similarly, in healthcare settings, they could compromise diagnostic systems, preventing the detection of serious medical conditions. For these reasons, our research group decided to conduct an in-depth study of this threat landscape, aiming to assess the current state of research about backdoor defense in FL. We found that existing defense approaches unfortunately focus on overly simplistic attacks, predominantly consisting of BadNets variants. However, as previously discussed, numerous more sophisticated attacks have been developed to overcome BadNets' limitations and evade defenses. Furthermore, we observed that several defense techniques fail when confronted with these attacks. Consequently, our efforts concentrated on designing a defense technique capable of overcoming the limitations of existing approaches and detecting and mitigating even more advanced attacks. Our work led to the development of **SVCC**, a new defense technique built to withstand attacks more advanced than the classic BadNets. All details will be presented in the following sections.

6.2 Defenses landscape

In the literature, defensive strategies against backdoor attacks in Federated Learning are typically designed from the perspective of the centralized server. Such defenses may pursue several goals:

1. Detecting a backdoor on a client and finding a way to neutralize it, for instance by excluding the client from aggregation or restoring its correct behavior.

2. Identifying and removing a backdoor on the server, should the server itself be compromised.

3. Detecting poisoned clients and mitigating their influence on the server-side aggregation.

Approach (1) is rarely considered, as it would require the ability to inspect clients directly. Yet, in the FL and backdoor literature, it is commonly assumed that the server lacks direct control over client devices. Moreover, if such control were available, one could simply apply standard backdoor defense techniques developed for the non-federated setting directly on the

client. Similarly, case (2) could often be addressed by employing existing defense mechanisms known from centralized learning. The most relevant scenario is instead (3), which represents the main focus of backdoor defenses in Federated Learning. Here, the goal is to identify poisoned clients based on the information they send to the server, and subsequently mitigate their impact (typically by excluding them from aggregation). This family of methods is generally referred to as *robust aggregation*.

In what follows, we provide an overview of existing backdoor defenses in the Federated Learning literature.

Early approaches, such as **Krum** [6] and **Bulyan** [24], focus on robust aggregation techniques aimed at identifying and excluding anomalous client updates. Krum selects the update most consistent with the majority by computing pairwise distances, while Bulyan enhances robustness by combining Krum-like selection with coordinate-wise trimmed averaging. However, both methods rely on strict upper bounds on the number of attackers, which are often unrealistic in practical deployments.

More advanced defenses leverage richer statistical properties. **RLR** [58] analyzes the sign patterns of model updates to detect malicious behavior and adjusts server-side learning rates based on directional consistency. **CRFL** [88] clips the norm of aggregated gradients and injects Gaussian noise to limit the influence of poisoned updates. **DnC** [69] performs PCA on randomly selected gradient coordinates to detect clients with large projections, discarding a fixed proportion of clients regardless of their actual behavior.

Another research direction employs small sets of auxiliary data to improve detection and reduce attack success rates. **SageFlow** [59] and **FLARE** [80] rely on clean public samples to evaluate client behavior. SageFlow assigns weights to client contributions based on prediction entropy and public-data loss, whereas FLARE computes trust scores using Maximum Mean Discrepancy (MMD). These strategies, however, assume the availability of clean server-side data, which is often unrealistic.

More recent techniques model similarity and divergence across client updates. **RFA** [60] computes a robust mean via the Weiszfeld algorithm; **FLAME** [55] clusters clients using cosine distance between update vectors and employs clipping and noise to enhance robustness. **MMA** [30] highlights the weaknesses of relying on a single distance metric in high-dimensional spaces, combining multiple metrics—cosine, Manhattan, and Euclidean distances—together with whitening to produce a unified score. **FDCR** [31] incorporates the Fisher Information Matrix to weight gradients by parameter importance and uses clustering to filter out malicious updates.

Despite their advantages, these defenses often require public data, assume strict attacker-ratio bounds, or depend on manually tuned statistical thresh-

olds. In contrast, our method offers a data-agnostic defense that makes no assumptions beyond the standard requirement that fewer than 50% of clients are compromised [55, 31]. Furthermore, instead of discarding suspicious updates entirely, we introduce a cure mechanism that recovers useful information from potentially poisoned clients. This enables the integration of both benign and compromised updates without jeopardizing the global model, achieving a balanced trade-off between robustness and performance.

As we have seen previously, backdoor attacks have evolved significantly, moving away from easily identifiable trigger patterns. Therefore, in our experiments, in addition to BadNets, we also consider the attack WaNet, which generates backdoors via imperceptible spatial transformations, providing a more realistic and challenging evaluation setting.

Chapter 7

SVCC: curing bad clients

7.1 Core ideas

The starting point of our work was to evaluate the effectiveness of existing defenses when facing a more sophisticated backdoor attack, which in our experiments is WaNet [54]. We observed, for instance, that the client clustering performed by some defenses fails to reliably identify compromised clients. Consequently, we searched for alternative client properties that could more effectively distinguish benign clients from malicious ones.

One promising direction was to analyze the Gram matrices of client features after local training. This idea is motivated by the fact that some centralized defenses, such as Beatrix, actively exploit Gram matrices for backdoor detection. We discovered that the spectral representations of the Gram matrices differ significantly between benign and malicious clients. More precisely, we define the spectral representation of a Gram matrix as the vector containing the square roots of its eigenvalues. This choice is not arbitrary; it is justified by the following observation.

Let F denote the feature matrix extracted from a client. The corresponding Gram matrix G can be written as:

$$G = F^T F. \tag{7.1}$$

Now consider the Singular Value Decomposition (SVD) of the feature matrix F:

$$F = U \Sigma V^T, \tag{7.2}$$

where U is the matrix of left singular vectors, Σ is the diagonal matrix containing the singular values, and V is the matrix of right singular vectors.

By definition of the SVD, the singular values of F are obtained by computing the eigenvalues of $F^T F$, which is precisely the Gram matrix G. More

specifically, the eigenvalues of G correspond to the squared singular values of F. Therefore, we can recover the singular values of the feature matrix simply by taking the square roots of the eigenvalues of the Gram matrix.

We found that the singular-value vectors extracted from clean examples differ remarkably from those derived from poisoned examples. Recall that the SVD decomposes the transformation represented by a matrix into three sequential transformations performed by V^T, Σ, and U. Since Σ is diagonal, it acts as a scaling operator on the axes of the space. Apparently, this scaling behaves very differently for clean versus poisoned examples.

These observations led us to cluster the clients using their singular-value vectors as input data. Even a simple two-cluster approach is sufficient; accordingly, we employ the K-Means [33] algorithm with $k = 2$. Since we adopt the standard assumption that fewer than 50% of the clients are compromised, the smaller cluster is expected to correspond to the poisoned clients. Existing methods typically discard these clients entirely. However, we argue that this strategy is suboptimal, as malicious clients still acquire "clean" knowledge in addition to backdoored behavior. For this reason, we instead apply a client sanitization procedure aimed at removing the backdoor while preserving the useful knowledge learned by the compromised models.

7.2 Cure phase

The SVCC cure phase is formulated as an optimization procedure. For each poisoned client, we have access to its local model state S_k, obtained at the end of the local training, and to the global "clean" reference state S_g, computed as the arithmetic mean of the states of all benign clients. Each state is represented as a vector obtained by concatenating all the parameters of the neural network after flattening each layer.

Our goal is to construct a sanitized state $\hat{S}_k$ that preserves the benign knowledge of the client while removing the backdoor. To this end, we introduce a learnable gating vector $\lambda \in [0, 1]^d$, where d is the dimensionality of the model state. The sanitized state is defined as:

$$\hat{S}_k = \lambda \odot S_g + (1 - \lambda) \odot S_k, \tag{7.3}$$

where $\odot$ denotes elementwise multiplication.

To optimize λ, we leverage the Gram matrices of the poisoned clients. For each batch of synthetic samples, we extract the feature representations using the model parameterized by $\hat{S}_k$, compute their Gram matrix, and align it with the original Gram matrix of the client. The loss function is the Mean Squared Error (MSE) between the two Gram matrices. We perform gradient

descent on λ for a fixed number of epochs. At convergence, we obtain a sanitized state $\hat{S}_k$ in which the backdoor has been effectively removed while the benign knowledge of the client is preserved. Finally, the new server state is computed by aggregating the states of the benign clients together with the sanitized states of the cured clients. The algorithm of our curing approach is shown in Algorithm 7.

Algorithm 7 SVCC Client Cure Procedure

Require: Clean reference state S_g; poisoned client state S_k; number of training epochs E; batch generator GenBatch()

Initialize learnable gating vector
for $e = 1$ to E **do**
  $X \leftarrow$ GenBatch() ▷ Generate synthetic batch
  Construct sanitized state: $\hat{S}_k = \lambda \odot S_g + (1 - \lambda) \odot S_k$
  Load model parameters from $\hat{S}_k$
  Extract features F from X
  Compute Gram matrix: $G = F^\top F$
  Take the original Gram matrix G_k (precomputed from client k)
  Compute loss: $\mathcal{L} = \text{MSE}(G, G_k)$
  Update λ via gradient descent on $\mathcal{L}$
end for
return Sanitized state $\hat{S}_k = \lambda \odot S_g + (1 - \lambda) \odot S_k$

7.3 Experiments

Datasets. We assess our methodology using three widely adopted benchmark datasets: CIFAR-10 [40], CIFAR-100 [40], and TinyImageNet [41]. Following common practice in Federated Learning, each dataset is split among 10 clients according to a Dirichlet distribution with concentration parameter $\beta \in \{0.5, 0.3, 0.1\}$, inducing varying levels of label imbalance and thus different degrees of non-IIDness. Unless otherwise stated, all reported results correspond to the performance of the centralized model evaluated on the global test set after training.

Backdoor Attacks. To test the resilience of SVCC under diverse adversarial conditions, we evaluate it against two representative backdoor attacks: BadNets [23] and WaNet [54]. For BadNets, we use the same trigger as FDCR [31], which employs a visible 7×3 white-pixel trigger placed in the top-left corner of the input. As previously explained, WaNet produces

highly stealthy triggers by applying subtle spatial warping through Thin-Plate Spline (TPS) transformations [8]. For BadNets, we poison 50% of the training samples so that the model fully learns the trigger. For WaNet, we poison 10% of each mini-batch, adhering strictly to the procedure defined in the original work.

Metrics. We consider three metrics to evaluate model behavior:

- **Clean Accuracy (ACC):** the proportion of unaltered test samples classified correctly.
- **Backdoor Failure Rate (BFR):** the percentage of triggered inputs for which the attack does *not* induce the adversarial target label; formally, $\text{BFR} = 1 - \text{ASR}$.
- **V-Score:** the average between ACC and BFR, providing a single scalar that balances standard accuracy and defensive robustness:

$$V = \frac{\text{ACC} + \text{BFR}}{2}.$$

Evaluated Approaches. We compare our defense, SVCC, against *nine* state-of-the-art baselines. Krum, originally proposed for Byzantine-robust aggregation, and RFA, which reduces the influence of anomalous updates, are both commonly used against backdoor attacks. The remaining methods fall into three broad families:

1. *Gradient-filtering and projection-based strategies*: these approaches detect and discard poisoned updates by examining their geometric structure (DnC, RLR, FLAME).
2. *Noise- and clipping-based defenses*: these methods suppress malicious contributions by bounding update magnitudes and injecting noise (CRFL).
3. *Auxiliary-signal-based defenses*: these rely on additional statistics or model-derived signals to judge client trustworthiness (SageFlow, MMA, FDCR).

Model Architectures. For CIFAR-10, we adopt the convolutional architecture used in FDCR [31], consisting of a lightweight stack of convolutional, pooling, and fully connected layers suitable for efficiency-oriented experiments. For CIFAR-100 and TinyImageNet, we use the PyTorch implementation of ResNet-18 [5], modifying the first convolution to a 3×3 kernel with stride 1 to better accommodate the small input resolution.

Table 7.1: **CIFAR-10**. Results in terms of V-score [↑]. Best results in bold, second-best underlined.

	BadNets						WaNet					
	20%			40%			20%			40%		
β	0.5	0.3	0.1	0.5	0.3	0.1	0.5	0.3	0.1	0.5	0.3	0.1
FedAvg	53.33	38.87	47.68	34.06	30.98	34.89	51.44	48.07	44.54	40.37	39.47	34.63
RLR	72.39	54.91	53.40	48.71	33.97	18.93	68.11	52.91	52.65	53.59	29.79	31.13
Krum	51.92	30.92	48.38	−−	−−	−−	56.22	26.99	55.75	−−	−−	−−
RFA	50.02	36.07	45.54	33.29	30.79	31.59	51.04	46.42	43.65	39.55	38.10	33.88
DnC	77.54	61.68	47.48	26.48	25.37	17.50	63.26	59.91	43.12	40.05	29.58	27.27
MMA	46.83	37.20	28.34	30.82	30.38	31.44	39.88	39.59	45.11	38.44	37.20	22.94
FLAME	62.24	62.11	37.47	27.80	24.22	34.08	47.53	60.66	43.72	40.31	29.37	37.18
CRFL	61.34	54.34	40.02	31.67	28.59	23.13	59.43	46.25	39.23	42.12	33.83	25.07
Sageflow	59.11	44.80	36.08	37.67	30.63	26.65	58.41	53.98	43.12	43.45	41.31	33.49
FDCR	75.94	76.68	**64.18**	31.82	30.00	38.91	63.44	57.71	50.03	36.40	42.36	47.12
SVCC	**77.81**	**77.49**	63.80	**60.91**	**56.76**	**40.04**	**78.04**	**75.69**	**72.47**	**73.50**	**69.17**	**54.23**

Federated Learning Setup. We conduct experiments in a Federated Learning environment with 10 clients. Data heterogeneity is introduced through label imbalance sampled from a Dirichlet distribution with parameters $\beta \in \{0.5, 0.3, 0.1\}$ [42, 90]. Each client performs 10 epochs of local optimization per communication round using SGD with learning rate 0.01, batch size 64, and training continues for 50 rounds in total. We evaluate two threat scenarios in which either 20% or 40% of the clients are compromised.

Platform. All experiments are executed on a single NVIDIA A100 GPU with 64 GB of VRAM.

7.4 Results

Tables 7.1, 7.2 and 7.3 report the V-score, which jointly captures accuracy and backdoor robustness.

SVCC consistently ranks first or second across all settings, highlighting its strong resilience to backdoor attacks while effectively preserving clean accuracy. Such robustness is especially noticeable under challenging conditions, such as high data heterogeneity ($\beta = 0.1$) and severe threat scenarios (40% poisoned clients), where most defense methods suffer substantial performance degradation. In particular, RLR, DNC, and MMA exhibit sharp declines due to their susceptibility to outlier updates, clustering instability, and optimization challenges in non-i.i.d. environments, respectively. In contrast, SVCC maintains stable performance, underscoring its robustness against both data imbalance and sophisticated poisoning strategies.

Table 7.2: **CIFAR-100**. Results in terms of V-score [↑]. Best results in bold, second-best underlined.

	BadNets						WaNet					
	20%			40%			20%			40%		
β	0.5	0.3	0.1	0.5	0.3	0.1	0.5	0.3	0.1	0.5	0.3	0.1
FedAvg	27.30	27.14	26.35	26.55	26.30	25.52	33.60	33.02	30.75	30.30	30.13	28.70
RLR	23.65	21.23	18.82	22.04	19.42	17.82	71.43	67.85	49.61	51.79	47.10	38.18
Krum	61.66	41.38	23.05	−−	−−	−−	45.14	58.92	42.08	−−	−−	−−
RFA	26.56	34.39	23.20	25.72	24.99	22.69	36.73	43.14	28.30	29.93	30.22	25.42
DnC	48.74	40.95	41.83	23.46	17.58	22.43	<u>72.06</u>	<u>70.14</u>	59.03	36.69	29.90	21.95
MMA	50.46	50.50	45.37	50.30	33.69	15.64	48.50	43.17	33.76	31.78	27.15	18.06
FLAME	22.04	20.71	18.29	20.01	19.21	17.17	35.65	33.67	38.91	25.41	25.00	21.23
CRFL	65.78	65.11	<u>62.32</u>	<u>64.94</u>	<u>63.21</u>	<u>61.27</u>	68.44	67.81	<u>63.58</u>	<u>67.15</u>	<u>66.29</u>	<u>62.15</u>
Sageflow	24.98	24.39	22.33	24.97	24.15	21.89	54.19	44.23	26.86	32.30	29.49	23.64
FDCR	<u>76.15</u>	<u>75.85</u>	40.24	25.37	24.67	22.34	31.41	28.92	27.17	28.84	27.00	25.54
SVCC	**76.80**	**76.45**	**75.41**	**75.22**	**74.35**	**72.57**	**76.94**	**76.55**	**74.85**	**75.63**	**74.74**	**72.99**

On CIFAR-10, SVCC achieves the best results under the WaNet attack at 40% poisoning across all β settings, with a V-score of 73.50, 69.17, and 54.23 for $\beta = 0.5, 0.3, 0.1$, respectively. These represent significant improvements over FDCR, the second-best methodology under the WaNet attack. This highlights SVCC's ability to detect visually imperceptible attacks using spectral properties of client features. For CIFAR-100, the trend persists. SVCC remains competitive in both BadNets and WaNet settings. Notably, in the 40% WaNet case with $\beta = 0.1$, SVCC achieves the highest V-score of 72.99, significantly outperforming the other approaches, which struggle under this setup. Although some defenses perform well in low-poisoning regimes or for specific attacks (e.g., DNC against 20% WaNet), they fail to generalize across both attack types and threat intensities. On TinyImageNet, which presents increased visual complexity and class diversity, SVCC often achieves top-tier performance, particularly under the BadNets attack, where its margin over other methods widens. This result suggests that the core insight of SVCC is highly effective even in high-resolution, high-diversity settings. In summary, SVCC offers state-of-the-art defense across datasets, attacks, and threat models, combining robustness and generality. Its performance on TinyImageNet, in particular, demonstrates strong scalability and confirms its potential for deployment in complex Federated Learning environments.

7.5 Additional Results

For completeness, we report the full numerical results (clean accuracy A and backdoor failure rate R, both ↑) for all methods, datasets, and attack

Table 7.3: **Tiny-ImageNet**. Results in terms of V-score [↑]. Best results in bold, second-best underlined.

	BadNets						WaNet					
	20%			40%			20%			40%		
β	0.5	0.3	0.1	0.5	0.3	0.1	0.5	0.3	0.1	0.5	0.3	0.1
FedAvg	20.92	20.76	19.25	20.14	19.76	18.48	37.07	37.51	31.31	28.64	28.10	23.93
RLR	16.05	14.79	12.55	15.63	14.56	11.59	68.25	<u>65.66</u>	<u>61.77</u>	<u>66.60</u>	<u>64.11</u>	**58.50**
Krum	47.52	40.91	38.46	−−	−−	−−	43.13	58.03	55.70	−−	−−	−−
RFA	20.03	29.38	17.97	19.32	18.94	17.16	31.66	44.83	29.54	25.34	26.96	22.30
DnC	21.51	31.55	22.39	16.33	17.57	11.20	<u>69.23</u>	65.44	**66.27**	37.13	32.98	21.66
MMA	31.45	50.23	44.66	16.92	27.71	4.93	14.64	34.87	17.04	23.02	18.25	16.18
FLAME	18.09	17.83	14.28	14.51	13.74	12.36	34.99	33.53	33.76	24.93	21.70	18.12
CRFL	60.35	53.70	<u>51.61</u>	<u>52.80</u>	<u>52.65</u>	<u>51.49</u>	52.17	52.43	51.04	50.82	50.51	50.32
Sageflow	20.41	19.82	17.52	20.10	19.32	17.34	50.34	47.72	36.03	33.73	30.99	24.32
FDCR	<u>69.83</u>	<u>69.50</u>	33.89	19.24	18.47	16.57	44.04	38.68	27.09	25.38	25.09	21.20
SVCC	**70.24**	**70.15**	**68.54**	**68.98**	**68.50**	**66.44**	**70.17**	**69.63**	59.50	**69.13**	**68.92**	<u>57.34</u>

settings considered in our experiments. This detailed reporting is intended to address the limitation of aggregated metrics (e.g., V-score), which may obscure important trade-offs between robustness and utility. By presenting (A, R) side-by-side, we make these trade-offs explicit, enabling a more transparent and fine-grained comparison across methods.

Tables 7.4 to 7.9 are organized by dataset (CIFAR-10, CIFAR-100, Tiny-ImageNet) and, within each dataset, by attack type (BadNets, WaNet).

7.6 Conclusions

Our research shows that existing backdoor defenses tend to fail when facing more sophisticated and stealthy attack strategies. This observation motivated us to design a more advanced defense mechanism capable of handling these stronger threats. The outcome of this effort is SVCC, a novel defense approach that leverages the spectral characteristics of client feature representations to identify compromised participants. Unlike prior methods, which simply discard poisoned clients, our strategy attempts to *cure* them by removing the implanted backdoor while preserving the benign knowledge acquired during local training.

Extensive experiments conducted on three benchmark datasets show that our method consistently outperforms existing defenses, even under severe label imbalance conditions. Nonetheless, our approach is not without limitations: because it relies on accurate client clustering, mis-clustering may allow some malicious clients to influence the global model, potentially propagating the backdoor.

We hope that this work will inspire future research efforts, including the development of more robust clustering procedures and an exploration of purification-based strategies for mitigating backdoor threats. Ultimately, we believe that SVCC provides a solid foundation for advancing the study of reliable and robust defenses in federated learning.

Table 7.4: **Cifar-10, BadNets**. Results in terms of ACC (A) and Backdoor Failure Rate (R).

	20%						40%					
β	0.5		0.3		0.1		0.5		0.3		0.1	
Metric	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*
Fedavg	65.76	40.89	61.37	16.37	56.92	38.44	65.51	2.60	60.21	1.74	56.21	13.58
RLR	54.16	90.61	41.22	68.59	30.05	76.74	53.81	43.61	45.72	22.21	26.01	11.86
Krum	37.31	66.53	25.44	36.41	14.49	82.27	−−	−−	−−	−−	−−	−−
RFA	65.51	34.52	60.83	11.32	56.1	34.97	65.13	1.45	60.5	1.07	55.28	7.9
DnC	64.23	90.85	58.88	64.47	49.64	45.32	51.65	1.32	50.38	0.36	34.97	0.02
MMA	53.19	40.47	36.16	38.24	19.66	37.03	55.3	6.33	45.61	15.16	41.08	21.81
FLAME	60.99	63.50	47.2	77.01	40.76	34.17	55.37	0.23	48.38	0.06	38.56	29.59
CRFL	56.62	66.05	51.51	57.18	40.86	39.18	56.23	7.12	50.69	6.49	42.9	3.36
Sageflow	64.1	54.12	60.34	29.26	48.73	23.43	63.44	11.91	59.21	2.05	48.48	4.82
FDCR	59.53	92.35	57.35	96.01	49.29	79.07	62.51	1.14	55.23	4.78	48.59	29.23
SVCC	64.97	62.07	56.07	66.90	51.12	55.87	62.65	59.17	58.99	54.53	47.64	32.45

Table 7.5: **Cifar-10, WaNet**. Results in terms of ACC (A) and Backdoor Failure Rate (R).

	20%						40%					
β	0.5		0.3		0.1		0.5		0.3		0.1	
Metric	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*
Fedavg	66.54	36.34	62.28	33.85	56.64	32.45	66.95	13.79	62.01	16.94	57.37	11.90
RLR	53.57	82.66	43.35	62.46	30.13	75.18	53.72	53.46	42.28	17.29	26.76	35.50
Krum	41.26	71.19	27.21	26.78	14.97	96.52	−−	−−	−−	−−	−−	−−
RFA	66.62	35.46	61.97	30.87	56.87	30.43	66.58	12.51	61.92	14.28	57.02	10.75
DnC	64.29	62.24	60.26	59.57	47.33	38.92	55.55	24.55	50.42	8.74	35.10	19.43
MMA	54.82	24.94	37.17	42.02	23.64	66.58	57.97	18.91	52.18	22.22	39.32	6.56
FLAME	59.05	36.02	50.39	70.92	41.80	45.64	58.72	21.90	50.71	8.04	41.46	32.90
CRFL	57.19	61.67	52.33	40.17	42.12	36.35	56.79	27.45	50.55	17.12	44.06	6.07
Sageflow	66.27	50.56	61.61	46.35	53.21	33.02	66.11	20.78	61.68	20.93	50.37	16.61
FDCR	63.66	63.22	54.77	60.65	45.27	54.80	59.87	12.93	57.98	26.74	49.51	44.73
SVCC	64.43	85.38	58.52	76.72	50.28	66.56	65.08	80.51	59.26	79.08	48.29	60.16

Table 7.6: **Cifar-100, BadNets**. Results in terms of ACC (A) and Backdoor Failure Rate (R).

	20%						40%					
β	0.5		0.3		0.1		0.5		0.3		0.1	
Metric	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*
Fedavg	54.59	0.01	54.27	0.01	52.65	0.05	53.1	0.01	52.59	0.01	51.03	0.02
RLR	47.3	0.00	42.45	0.00	37.63	0.00	44.07	0.00	38.84	0.00	35.64	0.00
Krum	24.95	98.38	17.80	64.96	12.77	33.34	−−	−−	−−	−−	−−	−−
RFA	52.75	0.36	51.66	17.11	46.38	0.02	51.4	0.04	49.94	0.04	45.38	0.01
DnC	51.42	46.06	51.15	30.74	45.37	38.28	46.90	0.02	35.11	0.04	44.85	0.00
MMA	1.00	99.93	1.00	100.00	1.07	89.67	0.95	99.64	0.99	66.39	1.01	30.26
FLAME	43.3	0.78	40.92	0.50	36.38	0.19	40.00	0.03	38.41	0.01	34.31	0.03
CRFL	36.00	99.34	32.51	99.57	23.25	98.03	33.65	99.06	30.96	96.07	22.27	92.58
Sageflow	49.91	0.06	48.76	0.01	44.63	0.04	49.92	0.02	48.29	0.01	43.75	0.02
FDCR	53.15	99.15	52.97	98.73	47.21	33.28	50.73	0.01	49.33	0.00	44.68	0.01
SVCC	55.08	99.23	54.37	99.22	52.55	99.29	51.08	99.37	49.44	99.26	45.79	99.34

Table 7.7: **Cifar-100, WaNet**. Results in terms of ACC (A) and Backdoor Failure Rate (R).

	20%						40%					
β	0.5		0.3		0.1		0.5		0.3		0.1	
Metric	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*
Fedavg	56.34	10.86	55.81	10.24	54.12	7.37	56.82	3.77	56.23	4.02	54.84	2.55
RLR	49.84	93.03	44.15	91.56	37.32	61.91	49.84	53.74	44.53	49.67	39.16	37.2
Krum	23.52	66.77	20.56	97.28	15.12	69.05	−−	−−	−−	−−	−−	−−
RFA	54.35	19.1	52.82	33.46	48.39	8.22	54.3	5.55	53.2	7.24	48.41	2.42
DnC	53.62	90.51	53.57	86.71	46.62	71.45	51.87	21.51	49.03	10.76	29.41	14.5
MMA	1.00	0.00	1.00	33.33	1.00	66.51	0.98	1.22	1.01	0.08	1.00	0.00
FLAME	45.82	25.48	44.18	23.15	39.91	37.92	46.47	4.35	44.58	5.41	39.86	2.59
CRFL	37.25	99.92	33.43	99.71	24.89	99.46	38.17	99.65	34.18	99.62	25.48	98.45
Sageflow	51.98	56.41	50.03	38.43	43.15	10.57	52.02	12.58	49.10	9.88	45.06	2.23
FDCR	55.23	7.60	51.87	5.97	50.23	4.11	54.90	2.77	52.61	1.39	48.95	2.12
SVCC	54.65	99.13	53.4	99.03	52.13	98.85	52.17	98.97	50.62	98.86	46.98	99.01

Table 7.8: **Tiny-ImageNet, BadNets**. Results in terms of ACC (A) and Backdoor Failure Rate (R).

	20%						40%					
β	0.5		0.3		0.1		0.5		0.3		0.1	
Metric	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*
Fedavg	41.24	0.59	41.00	0.52	38.18	0.32	40.23	0.04	39.49	0.03	36.96	0.01
RLR	32.11	0.00	29.59	0.00	23.74	1.36	31.26	0.00	29.12	0.00	23.18	0.0
Krum	18.69	76.34	15.35	66.48	10.66	66.26	−−	−−	−−	−−	−−	−−
RFA	39.74	0.32	39.01	19.76	35.66	0.28	38.63	0.01	37.84	0.03	34.28	0.03
DnC	39.62	3.41	38.22	24.88	35.33	9.45	32.00	0.65	34.72	0.41	22.37	0.03
MMA	0.52	62.38	0.47	100.00	0.53	88.79	0.50	33.33	0.45	54.97	0.50	9.36
FLAME	30.69	5.49	30.00	5.67	26.11	2.44	28.72	0.29	27.39	0.08	24.69	0.03
CRFL	1.48	100.00	1.25	100.00	1.11	100.00	1.25	100.00	1.46	100.00	1.29	100.00
Sageflow	40.44	0.39	39.57	0.07	35.02	0.01	40.16	0.05	38.63	0.01	34.68	0.00
FDCR	40.17	99.5	39.61	99.39	35.02	32.76	38.48	0.00	36.93	0.00	33.14	0.00
SVCC	41.29	99.56	40.92	99.47	38.22	99.43	38.32	99.65	37.43	99.57	33.62	99.26

Table 7.9: **Tiny-ImageNet, WaNet**. Results in terms of ACC (A) and Backdoor Failure Rate (R).

	20%						40%					
β	0.5		0.3		0.1		0.5		0.3		0.1	
Metric	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*	*A*	*R*
Fedavg	42.52	31.62	42.17	32.84	39.95	22.66	42.99	14.29	42.65	13.56	40.19	7.67
RLR	36.85	99.65	32.13	99.20	24.53	99.00	35.82	97.38	32.06	96.16	24.36	92.63
Krum	20.28	65.98	17.06	98.99	11.66	99.74	−−	−−	−−	−−	−−	−−
RFA	41.12	22.19	40.11	49.55	36.82	22.27	41.42	9.25	40.43	13.49	37.31	7.28
DnC	40.61	97.85	39.84	91.04	35.37	97.17	35.06	39.20	35.43	30.53	23.89	19.44
MMA	0.52	28.75	0.56	69.19	0.53	33.55	0.51	25.52	0.50	0.00	0.50	31.86
FLAME	32.69	37.30	31.71	35.35	29.40	38.11	33.03	16.82	31.44	11.96	29.02	7.22
CRFL	1.27	100.00	1.27	100.00	1.39	100.00	1.4	100.00	1.48	100.00	1.20	100.00
Sageflow	41.95	58.74	40.58	54.87	36.22	35.85	41.91	25.55	40.25	21.73	36.16	12.47
FDCR	41.06	47.03	39.23	38.13	37.05	17.13	40.79	9.98	40.65	9.52	36.96	5.45
SVCC	41.69	99.4	41.72	98.78	38.51	87.21	39.74	98.51	38.46	99.38	37.07	77.62

Chapter 8

Conclusions and Future Work

This research has investigated the application of artificial intelligence systems to real-world scenarios characterized by dynamic and structured data, with particular emphasis on the ability of models to learn continually without forgetting previous knowledge and to remain robust against malicious attacks.

In the first part of the thesis, we focused on Continual Learning methods. Within prompt-based approaches using Vision Transformer architectures, we identified the issue of interference between different tasks in the prompt selection mechanism, which negatively impacts the stability of current methods. To address this challenge, we proposed STAR-Prompt, an approach that enables neural networks to maintain high performance across sequences of tasks while mitigating forgetting, outperforming existing state-of-the-art methods.

We then extended the study to the more complex scenario of Multi-Label Continual Learning, where it is necessary to model intricate relationships between classes belonging to different tasks. This setting has received limited attention in prior literature. Our experiments showed that standard Continual Learning methods fail to maintain satisfactory performance in multi-label settings, despite achieving high accuracy in traditional single-label benchmarks. To address this gap, we developed SCAD, a technique for distilling attention representations that preserves the knowledge embedded in the pretraining state. Experimental results indicate that consistency with the pretraining state significantly enhances the robustness of models against forgetting in multi-label scenarios.

The second part of the thesis focused on the security of AI systems, with particular attention to backdoor attacks, which represent a critical threat in industrial and real-world applications. Our analysis revealed that recently proposed defenses are not universally effective against all types of attacks. In particular, adaptive attacks enable adversaries to optimize their strategies in

response to existing defenses, creating a substantial risk of false confidence for users of AI systems. Within this context, our proposed adaptive attack, PwNet, shows the ability to circumvent several existing defense mechanisms, highlighting the serious threat of this type of attack.

Our study was then extended to the Federated Learning setting, where research on backdoor attacks is still limited. In this context, we proposed SVCC, a defense strategy based on the spectral analysis of Gram matrices. SVCC shows increased robustness in identifying compromised clients and, through a cleansing phase, is able to recover useful information even from clients initially flagged as malicious, overcoming limitations of current defense methods.

Overall, this work has contributed to advancing both the continual adaptability and the security of AI models in realistic scenarios, proposing methods that combine effectiveness, robustness, and adaptability. The results show that it is possible to design AI systems capable of continuous adaptation and secure collaboration, even in the presence of sophisticated threats.

Beyond the specific contributions, this research provides several insights and directions for future work. From a practical standpoint, the methods developed could be applied to industrial systems, predictive maintenance, collaborative robotics, and distributed data management in sensitive environments. Nonetheless, certain limitations remain: the computational and memory requirements of the proposed methods, the generalization to domains beyond the considered benchmarks, and the need for additional empirical validation on large-scale real-world datasets. Future research in Continual Learning could focus on developing more computationally and memory-efficient methods. Meanwhile, studies on backdoor attacks should investigate adaptive attack strategies and, within Federated Learning, examine attacks and defenses in more heterogeneous and realistic settings.

In conclusion, this thesis highlights the importance of jointly considering adaptability and security when designing AI systems for dynamic and critical environments. By addressing both Continual Learning and robustness to malicious interventions, this work lays the groundwork for AI models that are not only high-performing but also reliable and resilient in practical, real-world applications.

Bibliography

[1] Mohamed Abdelsalam, Mojtaba Faramarzi, Shagun Sodhani, and Sarath Chandar. Iirc: Incremental implicitly-refined classification. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2021.

[2] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.

[3] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision*, 2018.

[4] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient Based Sample Selection for Online Continual Learning. In *Advances in Neural Information Processing Systems*, 2019.

[5] Jason Ansel et al. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *ASPLOS*. ACM, 2024.

[6] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 2017.

[7] Lorenzo Bonicelli, Matteo Boschini, Emanuele Frascaroli, Angelo Porrello, Matteo Pennisi, Giovanni Bellitto, Simone Palazzo, Concetto Spampinato, and Simone Calderara. On the effectiveness of equivariant regularization for robust online continual learning. *arXiv preprint arXiv:2305.03648*, 2023.

[8] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585, 1989.

[9] Matteo Boschini, Lorenzo Bonicelli, Angelo Porrello, Giovanni Bellitto, Matteo Pennisi, Simone Palazzo, Concetto Spampinato, and Simone Calderara. Transfer without forgetting. In *Proceedings of the European Conference on Computer Vision*, 2022.

[10] E. O. Brigham and R. E. Morrow. The fast fourier transform. *IEEE Spectrum*, 4(12):63–70, 1967.

[11] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in Neural Information Processing Systems*, 2020.

[12] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022.

[13] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision*, 2018.

[14] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[15] Haoran Chen, Zuxuan Wu, Xintong Han, Menglin Jia, and Yu-Gang Jiang. PromptFusion: decoupling stability and plasticity for continual learning. *arXiv preprint arXiv:2303.07223*, 2023.

[16] Xianing Chen, Qiong Cao, Yujie Zhong, Jing Zhang, Shenghua Gao, and Dacheng Tao. Dearkd: data-efficient early knowledge distillation for vision transformers. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2022.

[17] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[18] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[19] Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 168–172. IEEE, 2018.

[20] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018.

[21] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11966–11976, 2021.

[22] Sebastian Farquhar and Yarin Gal. Towards Robust Evaluations of Continual Learning. In *International Conference on Learning Representations Workshop*, 2018.

[23] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[24] Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, 2018.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2018.

[27] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

[28] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021.

[29] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, 2013.

[30] Siquan Huang, Yijiang Li, Chong Chen, Leyu Shi, and Ying Gao. Multi-metrics adaptively identifies backdoors in federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[31] Wenke Huang, Mang Ye, Zekun Shi, Guancheng Wan, He Li, and Bo Du. Parameter disparities dissection for backdoor defense in heterogeneous federated learning. In *NeurIPS*, 2024.

[32] David Hughes, Marcel Salathé, et al. An open access repository of images on plant health to enable the development of mobile disease diagnostics. *arXiv preprint arXiv:1511.08060*, 2015.

[33] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, 2023.

[34] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *Proceedings of the European Conference on Computer Vision*, pages 709–727. Springer, 2022.

[35] Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. In *Proceedings of the European Conference on Computer Vision*, 2020.

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

[37] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017.

[38] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[39] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops*, 2013.

[40] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[41] Yann Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[42] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *IEEE International Conference on Data Engineering*, 2022.

[43] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017.

[44] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv preprint arXiv:2101.05930*, 2021.

[45] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[46] Sihao Lin, Hongwei Xie, Bing Wang, Kaicheng Yu, Xiaojun Chang, Xiaodan Liang, and Gang Wang. Knowledge distillation via the target-aware transformer. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2022.

[47] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*, pages 273–294. Springer, 2018.

[48] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *Computer Vision – ECCV 2020*, Cham, 2020. Springer International Publishing.

[49] Wanlun Ma, Derui Wang, Ruoxi Sun, Minhui Xue, Sheng Wen, and Yang Xiang. The" beatrix"resurrections: Robust backdoor detection via gram matrices. *arXiv preprint arXiv:2209.11715*, 2022.

[50] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, 1989.

[51] Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. In *International Conference on Learning Representations*, 2022.

[52] Martin Menabue, Emanuele Frascaroli, Matteo Boschini, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. An attention-based representation distillation baseline for multi-label continual learning. In *International Conference on Machine Learning, Optimization, and Data Science*, 2024.

[53] Martin Menabue, Emanuele Frascaroli, Matteo Boschini, Enver Sangineto, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. Semantic residual prompts for continual learning. In *European Conference on Computer Vision*, 2024.

[54] Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. *International Conference on Learning Representations*, 2021.

[55] Thien Duc Nguyen, Phillip Rieger, Roberta De Viti, Huili Chen, Björn B Brandenburg, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, et al. FLAME: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[56] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464, 2020.

[57] Jaehoon Oh, Sungnyun Kim, Namgyu Ho, Jin-Hwa Kim, Hwanjun Song, and Se-Young Yun. Understanding cross-domain few-shot learning based on domain similarity and few-shot difficulty. *Advances in Neural Information Processing Systems*, 2022.

[58] Mustafa Safa Ozdayi, Murat Kantarcioglu, and Yulia R Gel. Defending against backdoors in federated learning with robust learning rate. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[59] Jungwuk Park, Dong-Jun Han, Minseok Choi, and Jaekyun Moon. Sageflow: Robust federated learning against both stragglers and adversaries. *Advances in Neural Information Processing Systems*, 2021.

[60] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 2022.

[61] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Proceedings of the European Conference on Computer Vision*, 2020.

[62] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[63] Sai Rajeswar, Pau Rodriguez, Soumye Singhal, David Vazquez, and Aaron Courville. Multi-label iterated learning for image classification with label ambiguity. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2022.

[64] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*, 2022.

[65] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, 1990.

[66] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[67] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 1995.

[68] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[69] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *Network and Distributed System Security*, 2021.

[70] Yucheng Shi, Mengnan Du, Xuansheng Wu, Zihan Guan, Jin Sun, and Ninghao Liu. Black-box backdoor defense via zero-shot image purification. *Advances in Neural Information Processing Systems*, 36:57336–57366, 2023.

[71] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023.

[72] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11909–11919, 2023.

[73] Mohammad S Sorower. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18(1):25, 2010.

[74] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[75] First Principles Of Computer Vision. Image filtering in frequency domain - image processing ii, 2021. Accessed: 2025-11-04.

[76] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 1985.

[77] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[78] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE symposium on security and privacy (SP)*, pages 707–723. IEEE, 2019.

[79] Jiahao Wang, Mingdeng Cao, Shuwei Shi, Baoyuan Wu, and Yujiu Yang. Attention probe: Vision transformer distillation in the wild. In *ICASSP*, 2022.

[80] Ning Wang, Yang Xiao, Yimin Chen, Yang Hu, Wenjing Lou, and Y Thomas Hou. Flare: defending federated learning against model poisoning attacks via latent space representations. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022.

[81] Runqi Wang, Xiaoyue Duan, Guoliang Kang, Jianzhuang Liu, Shaohui Lin, Songcen Xu, Jinhu Lü, and Baochang Zhang. Attriclip: A non-incremental learner for incremental knowledge learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023.

[82] Tong Wang, Yuan Yao, Feng Xu, Shengwei An, Hanghang Tong, and Ting Wang. An invisible black-box backdoor attack through frequency domain. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 396–413, Cham, 2022. Springer Nature Switzerland.

[83] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017.

[84] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European conference on computer vision*, pages 631–648. Springer, 2022.

[85] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2022.

[86] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 139–149, 2022.

[87] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[88] Chulin Xie, Minghao Chen, Pin-Yu Chen, and Bo Li. Crfl: Certifiably robust federated learning against backdoor attacks. In *International Conference on Machine Learning*, 2021.

[89] Zhendong Yang, Zhe Li, Ailing Zeng, Zexian Li, Chun Yuan, and Yu Li. Vitkd: Practical guidelines for vit feature knowledge distillation. *International Conference on Learning Representations*, 2023.

[90] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, 2019.

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

[92] Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19148–19158, 2023.

[93] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 2022.