Localized Learning and Generalization in Artificial Neural Networks with Properties of the Global Workspace

(グローバルワークスペースの特性を持つ人工ニューラルネットワークにおける局所学習と汎化)

Sun Yuwei

孫 昱偉

Department of Information and Communication Engineering
The University of Tokyo

Doctoral Dissertation
博士論文

A dissertation submitted to the University of Tokyo
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

December 2023

ABSTRACT

This thesis investigates localized and contextual learning to tackle the out-of-distribution (OOD) problem, which is central to a model's capability to adapt to previously unseen samples for lifelong learning. This thesis aims to understand the knowledge reuse capability and the emergent generality within a collection of neural modules. This study is closely related to brain mechanisms of attention and memory, by incorporating the properties of Global Workspace (GW) in the prefrontal cortex and long-term memory (LTM) in the Hippocampus. The objective is to construct intelligent systems that possess concise representations of the world, enabling them to acquire new tasks more efficiently, with reduced processing time and minimized interference among various areas of knowledge.

To this end, I first study knowledge transfer among a collection of expert models that can only observe partial environments in a federated learning setting. I demonstrate that generalization can be achieved through the coordination of localized models without global objectives. Building upon this observation, I propose a novel federated domain generalization method for learning a global model by distilling domain-invariant knowledge from various localized models.

Another approach to localized learning I proposed is the Markov chain-based Homogeneous Learning, where a meta-observer aims to learn an efficient communication policy of individual models. To determine whether such localized learning can also enhance the generality of foundational models like Transformers, I introduce a novel Associative Transformer that learns distinct priors to guide selective attentions and reuse knowledge from previous observations based on associative memory-based replay. Importantly, the sparse attention with a bottleneck and the memory replay can find resonance with the working memory in the GW and the LTM in the Hippocampus, respectively. The consolidated implementation of the GW and LTM based on neural networks has demonstrated improved model performance and interpretability across a wide spectrum of tasks, compared to various existing Transformer architectures. Finally, I investigate and reveal potential risks associated with the localized learning methods that achieve emergent behavior of generalization through inter-module representation learning. Overall, this thesis proposes a novel approach using modular and reusable neural knowledge to tackle the OOD problem based on sparse attention and memory study in cognitive neuroscience. I deeply believe this study will make a substantial contribution to our comprehension of general intelligence in humans and mammals, as well as the development of intelligent machines with the potential for lifelong learning and long-term memory in the near future.

Acknowledgements

I still remember the moment when I finished reading the book "How to Create a Mind" by Ray Kurzweil in 2015; I realized that Artificial Intelligence was what I was meant to dedicate my life to. And I am now writing a thesis on how to build intelligent machines based on the principles of the Global Workspace Theory of the brain. This journey has been incredibly meaningful, filled with exciting learning and research, as I pursue my passion. I've learned that research is a humble endeavor, even when we aim to make a dent in the universe. Back in 2015, I was still using the n-gram algorithm to program a chatbot, which, in retrospect, seemed far from achieving any intelligence. In just seven years, we've witnessed the emergence of Large Language Models that have been incredibly versatile. The progress in AI is encouraging, and with the new skills I've gained through my PhD, I believe I am prepared to embark on the next journey towards creating intelligent machines that resemble a mind.

The scientific journey to pursue AI would not have been the same without the guidance that my supervisor, Hideya Ochiai, has given me. His support allowed me to pursue my passion for science and machine learning without limitations. He has provided motivations and encourages for most of the work presented in this thesis. In particular, I appreciate his guidance in the several projects regarding decentralized neural networks described in Chapter 3, Chapter 4, and Chapter 8.

Much of the work presented in this thesis has been done in collaboration with my mentors and colleagues. I appreciate the collaboration with Ng Chong, who guided the development and helped with the revision of the work described in Chapter 3. I would like to thank Zhirong Wu and Steve Lin for their suggestions in experiment designs and several rounds of paper revisions for the work described in Chapter 5. I would like to thank Ryota Kanai for introducing the studies of consciousness and neuroscience to me. He spent time developing ideas and encouraging me to pursue the work described in Chapter 5. I appreciate the guidance of Jun Sakuma. We collaborated on the studies of AI security presented in Chapter 6 and Chapter 7, where he guided me how to formulate good research questions and scientific writing.

I also want to thank Nagul Cooharojananone, Hitoshi Matsubara, Kaoru Sezaki, Esaki Hiroshi for their support and advice. I also cherish those moments attending workshops and spending time with my colleagues and friends from RIKEN, Wei Huang, Chao Li, Gu Lin, Jingfeng Zhang, Atsushi Nitanda, Vo Nguyen Le Duy, Tomasz M. Rutkowski, Minh Ha Quang, Taiji Suzuki, Qibin Zhao, Masashi Sugiyama, and Ryuichiro Hataya. I would like to thank all my friends and folks I've met at conferences, Felipe Meneguzzi, Marcus Liwicki, Bing Xue, Jun Yan, Xavier Boix, Rasmus Hoier, Samir Belhaouari, and many others. I want to thank JSPS, Heiwa Nakajima Foundation, Microsoft Research Asia, and Whole Brain Architecture Initiative for supporting my research. I also want to thank the members of my thesis dissertation committee, Hitoshi Matsubara, Hitoshi Iba, Yoichi Sato, Jun Sakuma, and Hideya Ochiai for their constructive

feedback in shaping and refining this thesis.

Finally, I must acknowledge the importance of the support of my family. I am grateful to my parents, Jianhua and Ping, who gave me the desire to pursue science and instilled in me an appreciation for life beyond the time I spent on research (that is, most of the time...)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Deep Neural Networks

Artificial Neural Networks (ANNs) are a fundamental component of machine learning, inspired by the structure and functioning of the human brain. Neurons in the brain create a network where each neuron possesses a singular axon that branches out and connects with the dendrites of other neurons at synapses. At these synapses, the coordinated firing patterns of extensive cell populations carry information [1]. The mathematical model of such neural networks called the perceptron was first proposed back in 1958 [2], which is a probabilistic model for information storage and organization. The multi-layer perceptron [3] then adds to the practicability of neural networks, as a useful alternative to traditional statistical modeling techniques. Lecun et al. [4] presented deep neural networks that allow computational models composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

Deep learning or deep neural networks (DNNs) consist of interconnected neurons organized in three or more layers. Given an input x, DNNs forward the input layer by layer and output a prediction $\hat{y}$. We can think of there are many knobs in DNNs represented as the strength of connections between every two neurons i and j, known as weights $w_{i,j}$. The optimization processing or model training is to find the optimum combination of these knobs through the process of the *back-propagation*. The loss $\mathcal{L}$ between the model prediction $\hat{y}$ and the ground truth y such as a numerical label for an input sample is computed. The loss is then propagated backward through the DNNs, allowing for the computation of gradients $\delta w_{i,j} = \frac{\partial \mathcal{L}(y,\hat{y})}{\partial w_{i,j}}$ for each weight with respect to the loss. The gradients are then used to update the weights. The gradient computation and weight update continue until the network converges to a desired performance, usually measured by its performance in a held-out test dataset. There are many other important factors to successfully training a model, such as designing the activation function for each layer's output, selecting the suitable weight update or optimization functions, and so on. We refer to [5] for detailed explanations of DNNs and tuning of architecture and parameters for different applications. We now introduce several important architectures that are essential for the discussion of the projects in this thesis. These include Convolutional Neural Networks, Vision Transformers, and Hopfield networks.

1.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [4] represent a specialized class of neural networks designed for processing data with a 2-dimensional structure. Besides images with natural 2 dimensions, other data such as 1-D time-series data can

also be represented as 2-D data. In CNNs, we apply a mathematical operation called the convolution that involves the element-wise multiplication of a filter (or a kernel) with local regions of the input data, followed by the summation of the results to produce feature maps. CNNs usually comprise multiple convolutional layers followed by a pooling layer. The pooling layer reduces the spatial dimensions of the convolutional layer output by selecting the maximum or average values from local regions. When the input data has a large dimension such a high-resolution image, using the pooling layer can improve computational efficiency. Additionally, several fully connected layers are typically employed at the end of neural networks to process and transform the extracted features into one-dimensional vectors for final predictions. This can be for either object recognition or regression tasks.

1.1.2 Vision Transformers

Transformers [6, 7, 8, 9] leverage self-attention mechanisms to model long-term dependencies among input sequence elements. Through self-attention, each element in the sequence can consider all others, enabling the model to capture intricate relationships even between distant elements. Unlike models such as CNNs with inherent inductive biases (the kernels in CNNs that process values from local regions), Transformers do not possess preconceived assumptions about the training data. Their flexibility lies in learning patterns exclusively from input data, allowing them to excel in capturing complex relationships. However, this flexibility necessitates a substantial amount of training data for effective generalization, posing challenges in low-resource settings or when confronted with out-of-distribution examples.

In computer vision, Vision Transformers (ViT) [9] tackle image classification tasks by processing sequences of image patches. The pre-processing layer partitions an image into non-overlapping patches, followed by a learnable linear projection layer. Let $x \in \mathbb{R}^{H\times W\times C}$ be an input, where (H, W) is the resolution of the image and C is the number of channels. x is separated into a sequence of patches $x_p \in \mathbb{R}^{N\times(P^2\cdot C)}$, where (P, P) is the resolution of each image patch and $N = \frac{HW}{P^2}$ is the number of patches. These patches are mapped to embeddings $v_p \in \mathbb{R}^{N\times E}$ with the linear projection. A learnable 1D position embedding $v_0 \in \mathbb{R}^E$ is prepended to the patch embeddings to retain positional information, which results in $v \in \mathbb{R}^{(N+1)\times E}$. The position embedding in ViT is crucial for preserving the sequential order of input data, maintaining the spatial relationships for tasks such as image recognition or classification.

ViT leverages the self-attention mechanism as the essential building blocks, where each head maps a query and a set of key-value pairs to an output. The patch embeddings are used to obtain the query, key, and value based on linear transformations $W^Q \in \mathbb{R}^{E\times D}$, $W^K \in \mathbb{R}^{E\times D}$, and, $W^V \in \mathbb{R}^{E\times D}$. We first compute an attention mask using the query and key as follows $\text{softmax}(\frac{W_i^Q v(W_i^K v)^T}{\sqrt{D}})$. Here, D represents the dimension of the projected query and key. The choice of the denominator might influence the computed attention maps, with $\sqrt{D}$ being a commonly suggested default. Finally, the attention mask is applied to the value to obtain the final output. To improve its performance, we can use A multiple heads and concatenate all the outputs by the following

$$\mathrm{h}^i(v) = \text{softmax}(\frac{W_i^Q v(W_i^K v)^T}{\sqrt{D}})\, W_i^V v, \tag{1.1}$$

$$\text{Multi-head}(v) = \text{Concat}(\mathrm{h}^1, \ldots, \mathrm{h}^A)\, W^O, \tag{1.2}$$

where W^O is a linear transformation for outputs.

ViT has demonstrated superior performance in image classification tasks with large-scale datasets. However, it is important to acknowledge that when trained on mid-sized datasets such as ImageNet [10], without strong regularization, ViT still shows a modest accuracy compared to CNNs models such as ResNets [11].

1.1.3 Hopfield Networks

Associative memory or content-addressable memory is a storage and retrieval system where information is organized based on its relationships with other stored data. The Hopfield network [12] serves as one type of associative memory model, comprising interconnected neurons with symmetric weights. To store patterns, this network employs one-shot learning by adjusting its weights according to input patterns until a stable state is achieved. These patterns then settle into attractors within the network's dynamics, each with its basin of attraction. When a test pattern falls within an attractor's attraction region, the Hopfield network adjusts its dynamics to recall the corresponding attractor using an energy function. During this process, the network aims to minimize the overall energy, with lower energy signifying a more stable state and enabling the retrieval of stored patterns.

Moreover, recent studies have explored modern variants of Hopfield networks [13] and showed their close relation to the attention mechanism in Transformers. The modern Hopfield networks employ an energy function as follows

$$E = -\text{lse}(\beta, X^T\xi) + \frac{1}{2}\xi^T\xi + \beta^{-1}\text{log}N + \frac{1}{2}\zeta^2, \tag{1.3}$$

$$\zeta = \max_i |x_i|, \tag{1.4}$$

where lse is the log-sum-exp function, $X = (x_1, \ldots, x_N)$, N is the number of total stored patterns in the Hopfield networks, β is an inverse temperature variable, and ζ is the largest norm of all stored patterns. Then, the update equation to recall a pattern from the Hopfield network can be derived as follows $\hat{\xi} = X \cdot \text{softmax}(\beta \cdot (X^T\xi))$. The update equation shows similarity to the attention mechanism in Transformer models. Depending on the value of β, the reconstructed input $\hat{\xi}$ can either represent a metastable state, combining various stored patterns, or a single stored pattern. A large β reduces the likelihood of metastable states, while a small β increases this likelihood.

1.1.4 Decentralized Neural Networks

Decentralized neural networks [14] are composed of independent neural network modules. These models, in contrast to centralized counterparts relying on large-scale training, aim to enhance performance by sharing local model weights, gradients, and representations. One prominent example is Federated Learning (FL) [15], where local models train on independent data distributions. By aggregating these local model updates, FL facilitates the learning of a global model over the entire data distribution without requiring access to local data.

In FL, the Parameter Server (PS) randomly selects k out of m clients at each time step and broadcasts the current global model to these selected clients C_k. Subsequently, each client $i \in C_k$ updates its received local copy of the global model using its local dataset. To improve the performance of the global model, all updates from these local models are collected and aggregated each round.

Specifically, Federated Averaging (FedAvg) [16] computes a weighted average of the local updates as follows

$$W_{t+1}^g = W_t^g + \sum_{i \in C_k} \frac{N_i}{N} (W_t^i - W_t^g), \tag{1.5}$$

where W_t^g represents the weights of the global model at time step t, W_{t+1}^g represents the weights of the updated global model, W_t^i represents the weights of client i's local model, and N_i and N represent the amount of training data for client i and the total training data for all the selected clients, respectively.

1.2 Global Workspace Theory of the Mind

1.2.1 Working Memory

Studies in cognitive science and neuroscience have shown the pivotal role of working memory in the architecture of general cognitive abilities. Working memory serves as a hub for the storage and manipulation of information essential for complex cognitive tasks, such as attention, reasoning, and learning. It operates as a sketchpad, allowing the brain to temporarily retain and manipulate relevant information. An intriguing aspect of working memory is its constraint that we usually can only hold a limited number of pieces of information simultaneously. However, this "bottleneck" property is rather than a defect but a critical feature for the cognitive system to manipulate information in real-time for tackling novel situations. Generality, a hallmark of both natural and artificial intelligence, demonstrates a close relationship with the functioning of working memory, highlighting its role in adaptability across diverse environments and challenges.

1.2.2 Properties of the Global Workspace

One of the long-standing theories in cognitive science aimed at explaining working memory and the conscious sensation of manipulating information within working memory is the Global Workspace Theory (GWT) [17, 18, 19, 20]. GWT introduces a fundamental cognitive architecture for information processing within the working memory of the human brain, where various specialized modules compete to write information into a shared workspace through a communication bottleneck. The bottleneck facilitates the processing of content-addressable information through attention [21, 22].

The four properties of the GWT [23] are 1) multiple specialised systems capable of operating in parallel (modules); 2) limited capacity workspace entailing a bottleneck in information flow and a selective attention mechanism; 3) global broadcast: availability of information in the workspace to all modules; 4) state-dependent attention, giving rise to the capacity to use the workspace to query modules in succession to perform complex tasks.

Studies in machine learning research, whether by happenstance or by design, have been contributing to building an *Artificial Global Workspace*. The diverse properties of the GWT have become a source of inspiration for investigations in machine learning, particularly in areas such as modular neural networks [24, 25, 26, 27], context-dependent localized learning [28, 29, 30, 31, 32], sparse attention mechanisms [33, 34, 35], and memory-augmented neural networks [36, 37, 38, 39, 40]. For instance, the Perceiver model [28] employed iterative cross-attention with a latent array as priors, along with a latent transformation applied to these priors, to effectively capture dependencies across input data. Another noteworthy

approach is Coordination [30], which utilized a bottleneck to encourage greater flexibility and generalization by fostering competition among specialized modules.

1.3 Localized Learning

Localized Learning is a methodology enabling the acquisition of modules within various contextual inputs, with the goal of functional specialisation, reusability, and compositionality. The modular property encourages the emergence of generalized behavior across these modules to achieve global objectives. This methodology diverges from approaches centered on global optimization techniques, which seek a single, overarching solution to a problem. Instead, localized learning concentrates on developing local, context-dependent solutions. This distinction provides several advantages. Firstly, it avoids the need for centralized computation, rendering it well-suited for real-world deployment on a large scale. Secondly, the modular and sparse characteristics of localized learning parallel the local and asynchronous updates observed in biological synapses within the brain. Thirdly, localized learning facilitates swift adaptation to new tasks by repurposing knowledge components from prior experiences. Lastly, when augmented with long-term memory, localized learning holds the promise of constructing a lifelong learning intelligence by seamlessly integrating both rapid and gradual learning mechanisms.

Localized Learning includes the study of inductive biases. The inductive bias of a learning algorithm is the set of assumptions that the learner uses to predict outputs of given inputs that it has not encountered [41]. Inductive biases encourage the learning algorithm to prioritize solutions with certain properties, since algorithms that excel on certain distributions might perform less effectively on others. When presented with a dataset, multiple subnetworks ("winning tickets") in a dense, randomly-initialized, feed-forward network with comparable performance on training points are possible, called the Lottery Ticket Hypothesis [42]. Given a finite training set, to generalize to new input data relies on some assumptions or inductive biases about the solution we are looking for [43]. Several of the most promising inductive biases from studies of neuroscience are the Global Workspace. For instance, utilizing various kinds of metadata about individual neural network modules, such as measured performance and learned representations, has the potential to learn, select, or combine different neural network modules efficiently to solve a new task. The acquired knowledge from different neural modules are leveraged for reasoning and planning in a more efficient way.

Localized Learning encompasses a range of topics, such as inductive biases, ensemble learning, decentralized learning, modular neural networks, scalability through increased parallelism, knowledge plasticity, and reusability. In the subsequent sections of this thesis, I will delve into various instances of Localized Learning within both decentralized neural networks and Transformer models. This exploration will include addressing challenges related to non-interference knowledge transfer, optimizing knowledge reuse, understanding inductive biases in Transformer training, and ensuring trustworthiness in the context of Localized Learning. Finally, I conclude this thesis by discussing several future directions of Localized Learning research for improving few-shot generalization and lifelong memory formation capabilities of intelligent machines.

Chapter 2

Central Hypotheses and Motivation

In this chapter, we review several critical hypotheses on tackling systematic generalization problems and discuss how localized learning in artificial neural networks with the properties of the Global Workspace (GW) is related to these hypotheses. Systematic generalization refers to the ability of a learning system to apply knowledge and skills systematically to new, unseen situations that share structural similarities with those encountered during training. Several important hypotheses on approaches to systematic generalization include the capability of learning experience transfer (GW3: global broadcast: availability of information in the workspace to all modules), inducing compositionality (GW1: multiple specialized systems capable of operating in parallel), leveraging inductive biases (GW2: limited capacity workspace entailing a bottleneck in information flow and a selective attention mechanism; GW4: state-dependent attention, giving rise to the capacity to use the workspace to query modules in succession to perform complex tasks), and maintaining adversarial robustness.

2.1 Learning Experience Transfer

In complex and non-stable environments, one-size-fits-all solutions are often insufficient. When the data distribution changes, the model usually suffers from poor generality to the new data. Models should be capable of learning and adapting to new tasks quickly by leveraging knowledge gained from previous learning experiences. Knowledge gained from one domain can be transferred to another, provided there are shared underlying principles.

Such changes in the underlying statistical distribution of data are usually referred to as distributional shifts, including covariate shift, prior probability shift, concept shift, and so on. To investigate the hypothesis that learning experience transfer can facilitate model generalization, in Chapter 3, we explore a case of multi-domain learning wherein data samples from various local models originate from distinct domains. Each domain is characterized by some domain-specific features.

To illustrate, consider a home robot undergoing training to navigate a new house. The robot may utilize operating data from other houses acquired by different robots to improve its performance in this new environment. However, given the variations in room views and light conditions among different houses, attempting to directly train a new model based on the knowledge gained from other models becomes challenging. Knowledge transfer without regulation will introduce negative transfer, diminishing a model's adaptability to unseen tasks and degrading its generality. This problem involving domain-specific features is referred as domain shift problems.

To be more concrete, the domain shift problem or the covariate shift problem occurs when the relationships between input x and output y variables remain consistent, but the input data itself has undergone a shift in its distribution. Given a source domain S and a target domain T, we assume knowledge is transferred from the source domain to the target domain. Then, the domain shit problem is defined by $P_S(y|x) = P_T(y|x)$, $P_S(x) \neq P_T(x)$, where $P_S(\cdot)$ is the source domain distribution, and $P_T(\cdot)$ is the target domain distribution.

2.2 Compositionality

Models should be able to generalize by combining basic elements in novel ways to form more complex structures. The ability to understand and manipulate complex structures relies on the compositionality of simpler, interpretable components. Compositionality is frequently linked with constructing intelligent systems through modular components or "building blocks" that can be flexibly combined to execute a diverse array of tasks. A pivotal advantage of compositionality is its ability to enhance the efficiency of learning and adaptation.

To investigate the hypothesis that compositionality can facilitate model generalization, in Chapter 4, we delve into a specific learning framework comprising multiple neural networks organized in the structure of an acyclic graph. In particular, we explored the prior probability shift problem or the non-independent and identically distributed (non-IID) data problem. The prior probability shift takes place when the mapping of input variables to output variables remains consistent, while the prevalence or prior probability of specific output classes changes, i.e., $P_i(y|x) = P_j(y|x)$, $P_i(y) \neq P_j(y)$. We assume that every pair of neural networks in the learning framework satisfies this condition. Therefore, different neural networks are trained on data from distinct class distributions.

Within this context, we introduce a novel methodology known as Homogeneous Learning [44]. In this approach, a local model is designated as the meta for each training round, employing reinforcement learning to iteratively update a global learning policy. The meta observes the states of other local models and its surrounding environment, computing expected rewards for various actions based on these observations. Then, the meta determines the optimal action by drawing upon past experiences. The objective is to learn a global learning policy, enabling the learning framework to efficiently tackle different problems through the planning and combination of various local neural network models.

2.3 Inductive Bias

Models with an inductive bias exhibit a preference toward certain types of generalizations based on the structure of the training data. A well-designed inductive bias can guide models to generalize in a manner consistent with the underlying data distribution. Conventional deep neural networks usually incorporate inductive biases into their architectures, such as convolutional layers focusing on different image areas. This often results in superior performance in tasks that align with the underlying assumptions or biases encoded in training data, allowing the model to generalize more effectively to new, unseen examples.

Nevertheless, Transformer models do not possess built-in inductive biases that enable them to attend to different segments of the input based on inherent data structures. This necessitates large-scale training of Transformer models to achieve competitive performance with deep neural networks that have a built-in inductive bias. To investigate the hypothesis that inductive biases can facilitate model

generalization, in Chapter 5, we demonstrate the functional and architectural nuances of the Associative Transformer (AiT), a novel Transformer with implemented inductive biases for contextual and localized learning through the acquisition of a set of priors. AiT is a sparse representation learner, leveraging sparse bottleneck attention enhanced by an attention balance loss to acquire naturally emerging specialized priors. These priors guide attention over the input samples and serve as attractors within the associative memory of a Hopfield network, enabling information broadcast from the workspace. The inductive biases induced by these priors contribute to the model's superior performance and generalization in relational reasoning tasks, outperforming conventional Transformer models.

2.4 Adversarial Robustness

Models that demonstrate robust systematic generalization might be vulnerable to adversarial attacks. In a system built on local contextual knowledge through transfer learning and compositionality, weaknesses could potentially extend across the entire system. An adversary might intrude into such a decentralized system by exploiting a compromised local model as a backdoor, either by manipulating its local training data or model weights. These injected poisonings could trigger behaviors desired by the attacker and propagate their influence to other models within the system through knowledge sharing and information broadcast.

To investigate the hypothesis that maintaining adversarial robustness is crucial for artificial neural networks with the properties of the Global Workspace, we delve into the trustworthiness of different localized learning systems in Chapters 6, 7, and 8. We investigate two types of poisoning attacks that can cause the misbehavior of these systems. Moreover, we propose a self-supervised decentralized learning framework to enhance robustness to attacks in localized learning. This framework also benefits from the learning experience transfer, compositionality, and inductive biases. In this approach, no global supervision is required, and all learning occurs with local supervision. The framework facilitates knowledge sharing and can leverage large-scale training over distributed data sources. Data from various modalities are processed by neural networks with specific inductive biases, enabling the learning of contextual representations and enhancing overall model performance.

Chapter 3

Generalization and Transfer Learning

This chapter consolidates my work on multi-source domain adaptation in Federated Learning [45].

Multi-source domain adaptation has been intensively studied. The distribution shift in features inherent to specific domains causes the negative transfer problem, degrading a model's generality to unseen tasks. In Federated Learning (FL), learned model parameters are shared to train a global model that leverages the underlying knowledge across client models trained on separate data domains. Nonetheless, the data confidentiality of FL hinders the effectiveness of traditional domain adaptation methods that require prior knowledge of different domain data. We propose a new federated domain generalization method called Federated Knowledge Alignment (FedKA). FedKA leverages feature distribution matching in a global workspace such that the global model can learn domain-invariant client features under the constraint of unknown client data. FedKA employs a federated voting mechanism that generates target domain pseudo-labels based on the consensus from clients to facilitate global model fine-tuning. We performed extensive experiments, including an ablation study, to evaluate the effectiveness of the proposed method in both image and text classification tasks using different model architectures. The empirical results show that FedKA achieves performance gains of 8.8% and 3.5% in Digit-Five and Office-Caltech10, respectively, and a gain of 0.7% in Amazon Review with extremely limited training data. Moreover, we studied the effectiveness of FedKA in alleviating the negative transfer of FL based on a new criterion called Group Effect. The results show that FedKA can reduce negative transfer, improving the performance gain via model aggregation by 4 times.

3.1 Domain Transfer in Federated Learning

Federated learning (FL) [16, 46] has been accelerating the collaboration among different institutions with a shared interest in machine learning applications such as privacy-preserving diagnosis of hospitals [47] and decentralized network intrusion detection [48]. One of the most challenging problems in FL is to improve the generality in tackling client data from different domains. These different domains are usually used for the same classification task but with particular sample features under varying data collection conditions of clients. A naive averaging of all clients' model updates cannot guarantee the global model's performance in different tasks due to the problem of negative transfer[49]. In this regard, the learned knowledge from a client might not facilitate the learning of others. The effectiveness of model sharing in FL regarding knowledge transferability to an unseen task is of great importance to real-life application. For example, in

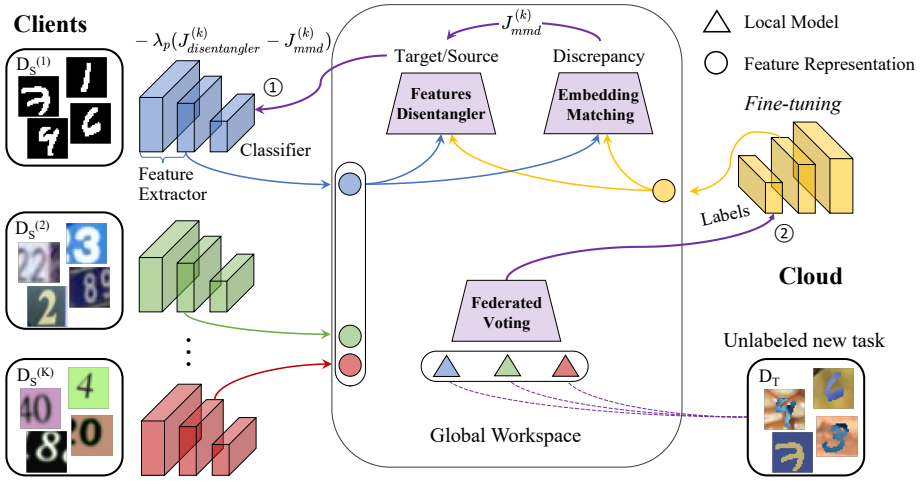


Figure 3.1: The framework of Federated Knowledge Alignment (FedKA) consists of three building blocks, i.e., Global Feature Disentangler, Embedding Matching, and Federated Voting. FedKA leverages distributed data domains based on feature distribution matching in a global workspace. The negative transfer is alleviated by ① local model representation learning on client domains and ② global model fine-tuning on the unlabeled cloud domain.

medical diagnosis, images collected by different medical machines can vary in sample quality. Client models learned on such diverse samples can diverge in the parameter space. Simply aggregating these models will not guarantee a better global model. Another example is connected autonomous vehicles based on FL. The multi-agent systems learn to tackle different driving situations in different cities and FL allows these agents to share the experience of driving in a new city.

Feature disentanglement is a common approach to alleviating problems of domain shift and negative transfer when encountering different domains, by separating domain-invariant features and domain-specific features from training samples [5, 50, 51, 52]. Nevertheless, such a practice necessitates that different domain data are centrally located at the same place for computation. In the above hospital and vehicle cases, feature disentanglement is unfeasible or impractical due to either privacy concerns or communication overheads in data sharing. The difficulty in federated domain generalization is that the source domain data of clients and the target domain data of a new task are usually separately located, which hinders effective knowledge sharing in FL. Moreover, the traditional model aggregation in approaches such as Federated Averaging (FedAvg)[16] cannot guarantee the improvement in the global model's performance by sharing local models trained on various client domains.

To this end, we propose Federated Knowledge Alignment (FedKA) (see Fig. 3.1) that alleviates negative transfer in FL improving the global model's generality to unseen tasks.

Overall, our main contributions are three-fold:

1) We proposed a novel domain generalization method FedKA in federated learning under the constraint of unknown client data, mainly due to data confidentiality. FedKA learns to reduce feature discrepancy between clients improving the global model's generality to tackle unseen tasks. (Section 3.3.2).

2) This work studied a new criterion for measuring negative transfer in federated learning (FL) called Group Effect, which throws light on the ineffectiveness of model aggregation in FL when training on different client domains. This work

provided detailed formulations (Section 3.3.3) and evaluation of Group Effect in FL (Section 3.4.5).

3) We performed extensive experiments on three different datasets, i.e., DigitFive, Office-Caltech10, and Amazon Review. We compared two different neural network architectures for model sharing including a lightweight two-layer model and a Resnet18 model. We demonstrated our method's effectiveness in improving the global model's prediction for various target tasks. (Section 3.4).

The remainder of this paper is structured as follows. Section 2 provides an overview of federated learning and domain adaptation and presents relevant work on the intersection of the two fields. Section 3 presents essential definitions and technical underpinnings of the proposed method. Section 4 presents the results of the empirical evaluation and the discussion of our main findings. Section 5 concludes the paper and provides future directions.

3.2 Related Work

Federated Learning

A distributed framework for machine learning (ML) [53] was introduced due to the proliferation of ML applications in academia and industry. The parameter server framework was further extended to a versatile and high-performance implementation for distributed ML based on local training data [54]. Moreover, Federated Learning (FL) [16] aims to train a model that learns a global probability distribution leveraging local model training on distributed data sources and trained model parameter sharing. Nevertheless, it usually bears a degraded global model performance when training on diversified client data [14]. There have been many works studying imbalanced Non-IID data in FL [55, 56]. For instance, federated group knowledge transfer (FedGKT)[57] leveraged Kullback Leibler (KL) Divergence to measure the prediction loss between an edge model and a cloud model, thus aligning knowledge of client models trained on Non-IID samples and the global model. Unfortunately, there are still not many efforts on the domain shift problem in FL, where each client owns data with domain-specific features due to different data collection environments.

Domain Adaptation

Domain adaptation [58, 59, 60] is one type of transfer learning to perform knowledge transfer from the source domain to the target domain. In this regard, a reconstruction-based method with an encoder-decoder architecture aims to learn a discriminative mapping of target samples to the source feature space, thus improving generalization performance [61, 62]. However, the generative approach is usually resource-consuming relying on computational capability. It is incompatible with resource-constrained clients in FL, such as mobile devices. In contrast, the method of feature disentanglement aims to distill features consistent across different domains thus improving the transferability of learned features. Therefore, the output of a model will remain unaffected despite several domain-specific feature changes. Deep Adaptation Networks (DAN)[63] trains two neural network models on the source and target domains, respectively. Then, DAN applies the multi-kernel Maximum Mean Discrepancy (MK-MMD) loss [64] to align features extracted from different layers of the two models. A variant of this method [65] aligns the joint distributions of multiple domain-specific layers across domains using a Joint Maximum Mean Discrepancy (JMMD) criterion. Furthermore, Do-

main Adversarial Neural Network (DANN)[66] leverages the domain confusion loss and the classification loss. DANN trains a classifier that distinguishes between source domain features and target domain features with encoders that distills representations indistinguishable by the domain classifier.

Domain Generalization in Federated Learning

The line of work in domain generalization for Federated Learning [58, 67] has been studied recently. For example, Federated Adversarial Domain Adaptation (FADA)[68] aims to tackle domain shift in FL through adversarial learning of domain-invariant features. Moreover, [69] presented a reversed scenario of FADA, where they tackled a multi-target domain adaptation problem for transferring knowledge learned from a labeled cloud dataset to different client tasks.

Unlike the studies mentioned above, FedKA leverages interactive learning between clients and the cloud thus overcoming the challenge of data confidentiality in FL. To improve the representation transferability, a client's encoder learns to align its output with the global embedding provided by the cloud. Simultaneously, the global model learns a better representation of the target task via fine-tuning based on the strategy of federated voting. Furthermore, to our best knowledge, there are no existing efforts to measure negative transfer in FL. Therefore, we propose Group Effect as an effective criterion.

3.3 Method

In this section, we first define the multi-source domain adaptation problem in Federated Learning (FL). Then, we present the technical underpinnings of Federated Knowledge Alignment (FedKA). Finally, we introduce our criteria for measuring the effectiveness of domain generalization methods in FL.

3.3.1 Multi-Source Domain Adaptation

We specifically consider a classification task with C categories. Let $x \in \mathbb{R}^V = X$ be a sample and $y \in \{1, 2, ..., C\} = Y$ be a label. D consists of a collection of N samples as $D = \{(x_i, y_i)\}_{i=1}^{N}$. In unsupervised domain adaptation[70, 71], given a source domain $D_S = (X_S, Y_S)$ and a target domain $D_T = (X_T, Y_T)$ where the labels Y_T are not provided, the goal is to learn the target conditional probability distribution $P(Y_T|X_T)$ in D_T with the information gained from D_S. The source domain D_S and target domain D_T usually share the same support of the input and output space $X \times Y$, but their data have domain discrepancies with specified styles, i.e., $P(X_S) \neq P(X_T)$.

A federated learning (FL) framework consists of the parameter server (PS) and K clients. We suppose that each client k has a different source domain $D_S^{(k)} = (X_S^{(k)}, Y_S^{(k)})$ and the PS has the unlabeled target domain $D_T = (X_T)$. Let f be a neural network classifier that takes an input x_i and outputs a C-dimensional probability vector where the jth element of the vector represents the probability that x_i is recognized as class j. Then, the prediction is given by $\hat{y} = \arg\max_j f(x)_j$ where $f(x)_j$ denotes the jth element of $f(x)$.

A client usually cannot share $D_S^{(k)}$ with the PS nor other clients mainly due to data confidentiality. Instead, FL learns a global probability distribution by updating a global model G_t based on the local models $L_t^{(k)}$ shared by different clients where t denotes the time step. The model aggregation allows FL to train over the entire data without disclosing distributed training samples. Notably,

FL proceeds by iterating the following steps: (1) the PS that controls the entire process of FL, initializes the global model G_0 and delivers it to all clients, (2) each client k updates the model using $N^{(k)}$ samples from the local data $D_S^{(k)}$, and sends back its model update $L_{t+1}^{(k)} - G_t$ to the PS, (3) then, PS aggregates all local model updates based on methods such as Federated Averaging (FedAvg), updates the global model, and sends the global model to all clients. Then, the model aggregation based on FedAvg can be formulated by the following

$$G_{t+1} = G_t + \sum_{k \in K} \frac{N^{(k)}}{\sum_{k \in K} N^{(k)}} (L_{t+1}^{(k)} - G_t). \tag{3.1}$$

The goal of the multi-source domain adaptation in FL is to learn a global model that predicts the target conditional probability distribution $P(Y_T|X_T)$ of D_T using the knowledge from the K client models learned on different source domains $\{D_S^{(1)}, D_S^{(2)}, ..., D_S^{(K)}\}$.

3.3.2 Federated Knowledge Alignment

Motivation

Effective knowledge transfer in multi-source domain adaptation of Federated Learning (FL) is critical to the success of distributed machine learning via model sharing. The challenge is to alleviate the negative transfer of FL such that the global model's generality to unseen tasks can be improved.

We demonstrate a global workspace where different latent representations and encoder models of clients are organized in a way that they can be leveraged to perform various tasks improving the global model's generality (Figure 3.1). This is inspired by Global Workspace Theory[17, 72] which enables multiple network models to cooperate and compete in solving problems via a shared feature space for common knowledge sharing. To this end, we propose Federated Knowledge Alignment (FedKA) that leverages the representation learning of client local models and the global model by feature distribution matching in the global workspace, facilitating effective knowledge transfer between clients.

Global Feature Disentangler

Let $f_e^{(k)} : \mathbb{R}^V \to \mathbb{R}^U$ be the encoder of a client model $L^{(k)}$. Let $f_c^{(k)} : \mathbb{R}^U \to \mathbb{R}^C$ be the class classifier of $L^{(k)}$. Then, given an input sample x, the client model outputs $\hat{y} = f_c^{(k)} f_e^{(k)}(x)$. Similarly, the global model consists of an encoder f_e^G and a class classifier f_c^G.

To learn an encoder $f_e^{(k)}$ that disentangles the domain-invariant features from $X_S^{(k)}$, we devise the global features disentangler by introducing a domain classifier in the PS. Notably, let f_d be the domain classifier that takes the feature representations H as the input and outputs a binary variable (domain label) q for each input sample h, which indicates whether h comes from the client k ($h \in H^{(k)} = f_e^{(k)}(X_S^{(k)})$ if $q = 0$) or from the target domain in the PS ($h \in H^G = f_e^G(X_T)$ if $q = 1$). The goal of the features disentangler is to learn a neural network that distinguishes between $H^{(k)}$ and H^G for different clients $k \in K$. The model learning of the features disentangler with respect to client k's source domain $D_S^{(k)}$ can be formulated by the following

$$J_{\text{disentangler}}^{(k)} = J(0, f_d f_e^{(k)}(X_S^{(k)})) + J(1, f_d f_e^G(X_T)), \tag{3.2}$$

$$\hat{f}_d = \arg\min_{f_d} J^{(k)}_{\text{disentangler}}(f_d, \hat{f_e^{(k)}}, \hat{f_e^G}), \tag{3.3}$$

where J is the negative log likelihood loss for the domain classification to identify between the representations of the source domain and the target domain.

Moreover, to learn an encoder $f_e^{(k)}$ that extracts domain-invariant features from client k's data, $f_e^{(k)}$ is updated by maximizing the above classification loss $J^{(k)}_{\text{disentangler}}$ of the features disentangler. When the features disentangler cannot distinguish whether an input representation is from the client domain or the cloud domain, $f_e^{(k)}$ outputs feature vectors that are close to the ones from the target domain. Then, each client k sends the feature representations $f^{(k)}_{e,t}(X_S^{(k)})$ to the PS every round t. In particular, the update of client k's encoder based on the features disentangler's classification loss can be formulated by

$$\hat{f_e^{(k)}} = \arg\max_{f_e^{(k)}} J^{(k)}_{\text{disentangler}}(\hat{f}_d, f_e^{(k)}, \hat{f_e^G}). \tag{3.4}$$

Embedding Matching

The global disentangler encourages a local model to learn features that are domain-invariant. We further enhance the disentanglement of features by measuring the high-dimensional distribution difference between feature representations from a client and the target domain in the parameter server (PS). In particular, we employ the Multiple Kernel variant of Maximum Mean Discrepancy (MK-MMD) to perform embedding matching between the two distributions $H_S^{(k)} = f_e^{(k)}(X_S^{(k)})$ and $H_T = f_e^G(X_T)$, using different Gaussian kernel $e_r, r \in \{1, 2, ..., R\}$ where R is the number of kernels. Then, for each kernel e_r:

$$\text{MMD}^2_{e_r}(H_S^{(k)}, H_T) = \left\| \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} \Phi(h_i) - \frac{1}{N^T} \sum_{j=1}^{N^T} \Phi(h_j) \right\|^2_{\mathcal{H}_{e_r}} \tag{3.5}$$

$$= \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} e_r(h_i, h_i') + \frac{1}{N^T} \sum_{j=1}^{N^T} e_r(h_j, h_j') - 2\frac{1}{N^{(k)}} \frac{1}{N^T} \sum_{i=1}^{N^{(k)}} \sum_{j=1}^{N^T} e_r(h_i, h_j),$$

where $\mathcal{H}$ is the reproducing kernel Hilbert space (RKHS) and Φ is a feature map $H \to \mathcal{H}$.

Furthermore, we consider as the embedding matching loss the distance between the mean embeddings of $\Phi_{e_r}(H_S^{(k)})$ and $\Phi_{e_r}(H_T)$ with five different Gaussian kernels (a bandwidth μ of two). Then, the local model of client k can be updated based on the embedding matching loss by the following

$$J^{(k)}_{\text{mmd}} = \frac{1}{5} \sum_{r=1}^{5} \text{MMD}^2_{e_r}(f_e^{(k)}(X_S^{(k)}), f_e^G(X_T)), \tag{3.6}$$

$$\hat{f_e^{(k)}} = \arg\min_{f_e^{(k)}} J^{(k)}_{\text{mmd}}(f_e^{(k)}, \hat{f_e^G}). \tag{3.7}$$

Local Model Representation Learning

For each round in FL, the PS sends back the computed gradients from the global feature disentangler $\frac{\partial J^{(k)}_{\text{disentangler}}}{\partial f_e^{(k)}}$ and the MK-MMD loss $J^{(k)}_{\text{mmd}}$ to client k to update the local encoder $f_e^{(k)}$. Then, for each client k, the local model is updated based on three different losses, i.e., the empirical loss $J^{(k)} = J(Y_S^{(k)}, f_c^{(k)} f_e^{(k)}(X_S^{(k)}))$, the features disentangler loss $J^{(k)}_{\text{disentangler}}(\hat{f}_d, f_e^{(k)}, \hat{f_e^G})$, and the MK-MMD loss $J^{(k)}_{\text{mmd}}(f_e^{(k)}, \hat{f_e^G})$. To alleviate the effect of noisy representations at early stages of the training, we adopt a coefficient λ_p that gradually changes from 0 to 1 with the learning progress of FL. Let b be the batch number, B be the number of total batches, r be the round number, and R be the number of total rounds in FL. λ_p is defined by $\lambda_p = \frac{2}{1+exp(-\gamma \cdot p)} - 1$, where $p = \frac{b+r\times B}{R\times B}$ and γ is set as five. Notably, we devise the local model representation learning as follows

$$E(f_c^{(k)}, f_e^{(k)}) = J^{(k)} - \lambda_p(J^{(k)}_{\text{disentangler}}(\hat{f}_d, f_e^{(k)}, \hat{f_e^G}) - J^{(k)}_{\text{mmd}}(f_e^{(k)}, \hat{f_e^G})), \quad (3.8)$$

$$(\hat{f_c^{(k)}}, \hat{f_e^{(k)}}) = \arg\min_{f_c^{(k)}, f_e^{(k)}} E(f_c^{(k)}, f_e^{(k)}). \quad (3.9)$$

Then, each client $k \in K$ with the source domain $D_S^{(k)}$ performs model learning every round t by the following $L_{t+1}^{(k)} = L_t^{(k)} - \eta \cdot \nabla E(f_c^{(k)}, f_e^{(k)})$, where η denotes the learning rate.

Global Model Fine-Tuning Based on Federated Voting

We propose a fine-tuning method called federated voting to update the global model every round without ground truth labels of the target domain samples. In particular, Federated voting fine-tunes the global model based on the pseudo-labels generated by the consensus from learned client local models. This strategy allows the global model to learn representations of the target domain data without ground truth labels thus improving the effectiveness of the feature distribution matching.

Let $y_i^{(k)} = \arg\max_j f_k(x_i)_j$ represents the prediction class of client k's local model $L^{(k)}$ with the input x_i. In FL, at each time step t, all client model updates $\{L_t^{(1)} - G_{t-1}, L_t^{(2)} - G_{t-1}, ..., L_t^{(K)} - G_{t-1}\}$ are uploaded to the PS. Given an unlabeled input sample x_i from the target domain D_T, the federated voting method aims to attain the optimized classification label y_i^* by the following

$$y_i^* = \arg\max_{c\in\{1,2,...,C\}} \sum_{k=1}^{K} \mathbb{1}\{y_i^{(k)} = c\}. \quad (3.10)$$

Note that this method could result in multiple y_i^* candidates that receive the same number of votes, especially when the total client number K is low while the total class number C is high. In such a case, we randomly select one class from the candidate pool as the label y_i^* of x_i.

Furthermore, the fine-tuning of the global model G_{t+1} is performed every round after the model aggregation with N^T samples from the target domain data X_T and the generated labels $Y^* = \{y_i^*\}_{i=1}^{N^T}$ using Eq. 3.10. We devise the global model fine-tuning as follows $G_{t+1}^* = G_{t+1} - \lambda_p \eta \cdot \nabla J(Y^*, G_{t+1}(\{x_i\}_{i=1}^{N^T}))$, where λ_p is the coefficient to alleviate noisy voting results at early stages of FL.

3.3.3 Criteria

We present two different criteria for measuring the effectiveness of a multi-source domain adaptation method, i.e., target task accuracy (TTA) and Group Effect (GE).

The performance of the global model G_t at time step t in the target task is measured by the target task accuracy (TTA) defined in the following

$$\mathrm{TTA}_f(G_t) = \frac{\sum_{(x,y)\in D_T} \mathbb{1}\{\arg\max_j f(x; G_t)_j = y\}}{|D_T|}, \tag{3.11}$$

where $|\cdot|$ denotes the size of the target domain dataset.

Moreover, we define a novel criterion for measuring negative transfer in FL, called Group Effect (GE). In particular, GE throws light on negative transfer of FL, due to inefficient model aggregation in the PS, which has not yet been studied to our best knowledge. We aim to measure to what extent the difference in clients' training data causes diverse local model updates that eventually cancel out in the parameter space leading to negative transfer. Intuitively, FedKA matches feature distributions of separate client domains with the target domain in the parameter server, such that we can alleviate the information loss from the model aggregation in FL. Given K local updates at the time step t, the group effect exists if $\mathrm{TTA}_f(G_{t+1}) < \frac{1}{K}\sum_{k\in\{1,2,\ldots,K\}} \mathrm{TTA}_f(G_t + \Delta_t^{(k)})$ where G_{t+1} is attained by Eq. 3.1. To this end, we propose GE based on TTA_f in Eq. 3.11 as follows

$$\mathrm{GE}_t = \frac{1}{K} \sum_{k\in\{1,2,\ldots,K\}} \mathrm{TTA}_f(G_t + \Delta_t^{(k)}) - \mathrm{TTA}_f(G_{t+1}). \tag{3.12}$$

3.4 Experiments

In this section, we first describe three benchmark datasets and detailed experiment settings. Next, we demonstrate the empirical evaluation results of our method using the metric of $\mathrm{TTA}_f(G_t)$ in Eq.3.11, followed by a discussion. Then, we compare the evaluation results based on different backbone encoder models. Furthermore, we show the effectiveness of our method in alleviating the Group Effect in Eq.3.12 of federated learning with both numerical results and the visualization of t-SNE features. We use PyTorch [73] to implement the models in this study.

3.4.1 Dataset

We employed three domain adaptation datasets, i.e., Digit-Five and Office-Caltech10 for the image classification tasks and Amazon Review for the text classification task, respectively.

Digit-Five is a collection of five most popular digit datasets, MNIST (mt) [74] (55000 samples), MNIST-M (mm) (55000 samples), Synthetic Digits (syn) [66] (25000 samples), SVHN (sv)(73257 samples), and USPS (up) (7438 samples). Each digit dataset includes a different style of 0-9 digit images.

Office-Caltech10 [75] contains images of 10 categories in four domains: Caltech (C) (1123 samples), Amazon (A) (958 samples), Webcam (W) (295 samples), and DSLR (D) (157 samples). The 10 categories in the dataset consist of objects in office settings, such as keyboards, monitors, and headphones.

Amazon Review [76] tackles the task of identifying the sentiment of a product review (positive or negative). This dataset includes reviews from four different merchandise categories: Books (B) (2834 samples), DVDs (D) (1199 samples), Electronics (E) (1883 samples), and Kitchen & housewares (K) (1755 samples).

3.4.2 Model Architecture and Hyperparameters

We consider different transfer learning tasks in the aforementioned datasets. Notably, we adopt each data domain in the applied dataset as a client domain. In this regard, besides the target domain of the cloud, there are four different client domains in Digit-Five and three different client domains in both Office-Caltech10 and Amazon Review, respectively. We conducted experiments using the following model architectures and hyperparameters.

Image Classification Tasks

Images in Digit-Five and Office-Caltech10 are converted to three-channel color images with a size of 28×28. Then, as the backbone model, we adopt a two-layer convolutional neural network (64 and 50 channels for each layer) with batch normalization and max pooling as the encoder and two independent two-layer fully connected neural networks (100 hidden units) with batch normalization as the class classifier and the domain classifier, respectively. Moreover, to perform the local model representation learning, we apply as a learning function Adam with a learning rate of 0.0003 and a batch size of 16 based on the grid search. Every round of FL, a client performs training for one epoch using 512 random samples (32 batches) drawn from its source domain. Furthermore, to compute the features disentangler loss, 512 random samples from the target domain are applied every round. The learning of the domain classifier neural network is performed during the client model representation learning based on a gradient reversal layer.

To attain a more accurate measurement of differences in feature representation distributions using embedding matching, we apply a lager batch size of 128 with the same samples in the client model representation learning. In Digit-Five, we employ two variants of the federated voting strategy, i.e., Voting-S and Voting-L, using 512 and 2048 random target domain samples, respectively. Moreover, since Office-Caltech10 is a relatively small dataset, we use all available samples in the target domain for federated voting. The learning hyperparameters of the global model are the same with the local models.

Text Sentiment Classification Task

To process text data of product reviews, we apply the pretrained Bidirectional Encoder Representations from Transformers (BERT) [77] to convert the reviews into 768-dimensional embeddings. We set the longest embedding length to 256, cutting the excess and padding with zero vectors. Moreover, we apply a two-layer fully connected neural network (500 hidden units) with batch normalization as the encoder using the flatten embeddings as the input. The features disentangler and class classifier share the same architectures as those in the image classification tasks, but with different input and output shapes (binary classification). We apply Adam with a learning rate of 0.0003, a batch size of 16, and 128 random training samples every round. A lower training sample number is because Amazon Review has much fewer samples compared to Digit-Five. Similarly, we employ

Table 3.1: $max_{G_t}\text{TTA}_f(G_t)$ (%) on the Digit-Five dataset based on different methods. The highest reported accuracy under each task is in bold.

Models/Tasks	→mt	→mm	→up	→sv	→sy	Avg
FedAvg	93.5±0.15	62.5±0.72	90.2±0.37	12.6±0.31	40.9±0.50	59.9
f-DANN	89.7±0.23	70.4±0.69	88.0±0.23	11.9±0.50	43.8±1.04	60.8
f-DAN	93.5±0.26	62.1±0.45	90.2±0.13	12.1±0.56	41.5±0.76	59.9
Voting-S	**93.7**±0.18	63.4±0.28	**92.6**±0.25	14.2±0.99	45.3±0.34	61.8
Voting-L	93.5±0.18	64.8±1.01	92.3±0.21	14.3±0.42	45.6±0.57	62.1
Disentangler+Voting-S	91.8±0.20	71.2±0.40	91.0±0.58	14.4±1.09	48.7±1.19	63.4
Disentangler+Voting-L	92.1±0.16	71.8±0.48	90.9±0.36	15.1±0.91	49.1±1.03	63.8
Disentangler+MK-MMD	90.0±0.49	70.4±0.86	87.5±0.25	12.2±0.70	44.3±1.18	60.9
FedKA-S	91.8±0.19	72.5±0.91	90.6±0.14	**15.2**±0.46	48.9±0.48	63.8
FedKA-L	92.0±0.26	**72.6**±1.03	91.1±0.24	14.8±0.41	**49.2**±0.78	**63.9**

128 random target domain samples every round for federated voting. In addition, in embedding matching, we apply a batch size of 16.

3.4.3 Ablation Study

To understand different components' effectiveness in FedKA, we performed an ablation study by evaluating $max_{G_t}\text{TTA}_f(G_t)$ during 200 rounds of FL. We considered different combinations of the three building blocks of FedKA and evaluated their effectiveness in different datasets. As a comparison model, the FedAvg method applies the averaged local updates to update the global model. Moreover, the f-DANN method extends Domain Adversarial Neural Network (DANN) to the specific task of federated learning, where each client has an individual DANN model for training. Similarly, for the f-DAN method, we adapted Deep Adaptation Network (DAN) to the specific task of federated learning. We discuss the evaluation results of the ablation study in the following.

Table 3.1 demonstrates the evaluation results in Digit-Five. Though f-DANN improved the model performance in the tasks of $mt, sv, sy, up \rightarrow mm$ and $mt, mm, sv, up \rightarrow sy$, f-DANN resulted in a decreased $max_{G_t}\text{TTA}_f(G_t)$ in the other three tasks. Furthermore, for the two variants of federated voting, the results suggest that federated voting can improve the model performance, especially, combined with the global feature disentangler and embedding matching. In the experiment on the Digit-Five dataset, FedKA achieves the best accuracy improving model performance by 6.7% on average.

Table 3.2 demonstrates the evaluation results in Office-Caltech10. The proposed method also outperforms the other comparison models. In addition, the effectiveness of federated voting appears to smaller compared to the case of Digit-Five. This is due to the less available target data in Office-Caltech10 for fine-tuning the global model.

Table 3.3 demonstrates the evaluation results in Amazon Review. FedKA improved the global model's performance in all target tasks outperforming the other approaches, by leveraging the global feature disentangler, embedding matching, and federated voting.

3.4.4 Effectiveness of Model Architecture Complexity

To further study the effectiveness of Federated Knowledge Alignment (FedKA) when applying different encoder models, we employed Resnet18 [11] without pre-training to perform feature extraction. Based on the same hyperparameter set-

Table 3.2: $max_{G_t}\text{TTA}_f(G_t)$ (%) on the Office-Caltech10 dataset based on different methods. The highest reported accuracy under each task is in bold.

Models/Tasks	C,D,W→A	A,D,W→C	C,A,W→D	C,D,A→W	Avg
FedAvg	52.9 ±0.56	**37.5** ±0.50	28.7±1.80	22.4±1.38	35.4
f-DANN	52.8 ±0.40	37.3 ±0.84	28.8±2.07	23.3±0.51	35.5
f-DAN	52.7±0.64	36.8±0.49	28.4±1.43	22.9±0.76	35.2
Voting	**53.3** ±0.80	37.3 ±0.58	27.8±2.37	23.3±1.92	35.4
Disentangler+Voting	52.5±0.65	37.3 ±0.84	29.9 ±2.70	23.4 ±1.72	35.8
Disentangler+MK-MMD	52.7±0.41	36.4±0.93	**31.1** ±1.91	**24.3** ±1.69	**36.1**
FedKA	52.8 ±0.57	37.2±0.29	29.3 ±1.51	23.7 ±1.15	35.8

Table 3.3: $max_{G_t}\text{TTA}_f(G_t)$ (%) on the Amazon Review dataset based on different methods. The highest reported accuracy under each task is in bold.

Models/Tasks	D,E,K→B	B,E,K→D	B,D,K→E	B,D,E→K	Avg
FedAvg	62.6±0.58	75.1±0.53	78.0±0.39	80.3±0.34	74
f-DANN	62.7±0.35	75.3±0.34	78.7±0.29	80.4±0.21	74.3
f-DAN	62.3±0.55	73.8±0.29	77.8±0.29	80.1±0.38	73.5
Voting	62.1±0.20	74.6±0.58	77.8±0.70	79.6±0.33	73.5
Disentangler+Voting	62.7±0.37	75.1±0.60	78.4±0.29	**80.8**±0.52	74.3
Disentangler+MK-MMD	**62.9**±0.32	75.3±0.33	78.5±0.23	80.2±0.12	74.2
FedKA	62.8±0.23	**75.8**±0.52	**78.8**±0.65	80.7±0.28	**74.5**

Table 3.4: $max_{G_t}\text{TTA}_f(G_t)$ (%) on the Digit-Five dataset using Resnet18 as the backbone. The highest reported accuracy under each task is in bold.

Models/Tasks	→mt	→mm	→up	→sv	→sy	Avg
FedAvg	**97.9**±0.07	71.3±0.79	**96.9**±0.05	11.9±0.62	55.8±1.60	66.8
f-DANN	97.5±0.07	77.1±0.29	96.8±0.38	12.1±1.01	79.5±0.37	72.6
f-DAN	**97.9**±0.09	71.7±1.22	96.7±0.18	11.3±0.68	55.5±1.00	66.5
Voting	96.5±0.20	72.1±1.24	**96.9**±0.17	**14.0**±0.74	61.1±0.28	68.1
Disentangler + Voting	96.5±0.21	76.5±0.53	96.8±0.42	13.7±0.45	79.4±0.61	72.6
Disentangler + MK-MMD	97.5±0.03	76.7±0.69	**96.9**±0.15	11.0±0.53	**80.1**±0.49	72.4
FedKA	96.4±0.23	**77.3**±1.01	96.6±0.38	13.8±0.81	79.5±0.68	**72.7**

Table 3.5: $max_{G_t}\text{TTA}_f(G_t)$ (%) on the Office-Caltech10 dataset using Resnet18 as the backbone. The highest reported accuracy under each task is in bold.

Models/Tasks	C,D,W→A	A,D,W→C	C,A,W→D	C,D,A→W	Avg
FedAvg	56.4 ±1.23	40.2 ±0.69	28.7±1.21	22.7±1.85	37.0
f-DANN	58.3 ±1.53	40.0 ±1.50	30.7 ±3.59	22.3±1.29	37.8
f-DAN	56.7±0.71	38.7±0.75	30.2±1.64	23.9 ±1.70	37.4
Voting	56.5 ±1.88	40.2 ±0.58	29.8±1.45	**24.1** ±0.69	37.7
Disentangler+Voting	**61.4** ±2.51	**40.4** ±1.01	31.5 ±3.11	23.9 ±1.89	**39.3**
Disentangler+MK-MMD	59.5 ±0.41	37.8±0.93	**32.2** ±3.21	22.3 ±1.00	38.0
FedKA	59.9 ±1.44	39.7±0.81	30.2 ±1.71	23.4 ±1.45	38.3

ting, we evaluate the model performance in Digit-Five (Table 3.4) and Office-Caltech10 (Table 3.5). As a result, FedKA achieved performance gains of 8.8% and 3.5% in Digit-Five and Office-Caltech10 with the Resnet18 backbone, respectively. Moreover, as shown in Figure 3.2, a more complex model could contribute to a larger gain in the global model performance improvement.

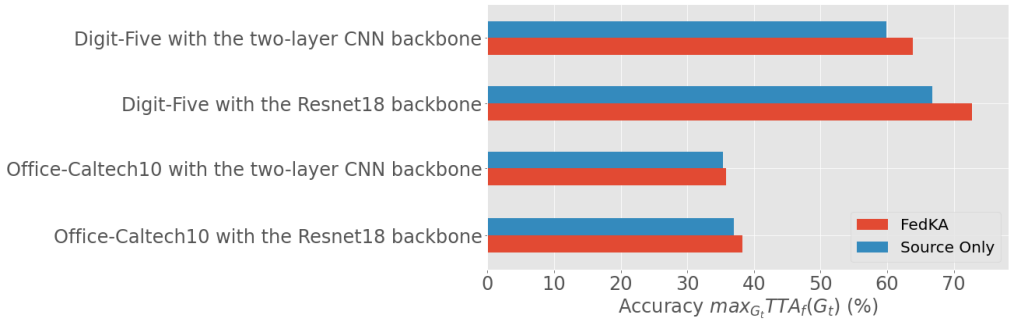
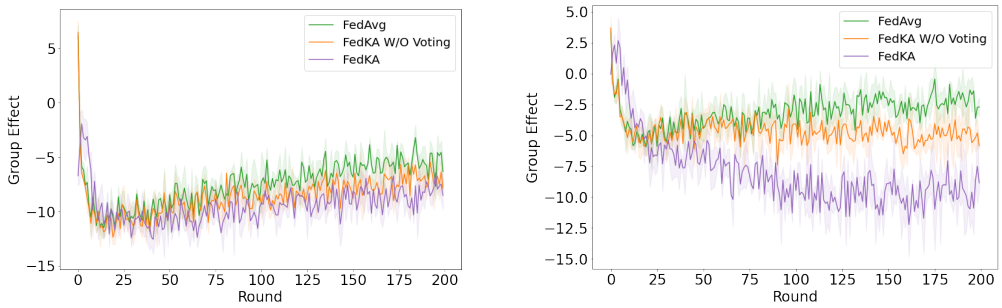


Figure 3.2: Model performance comparison between the Source Only (FedAvg) method and FedKA using different backbone encoder models. The result shows that FedKA can better benefit the improvement in global model performance with a more complex encoder model.



(a) mt,sv,sy,up→mm

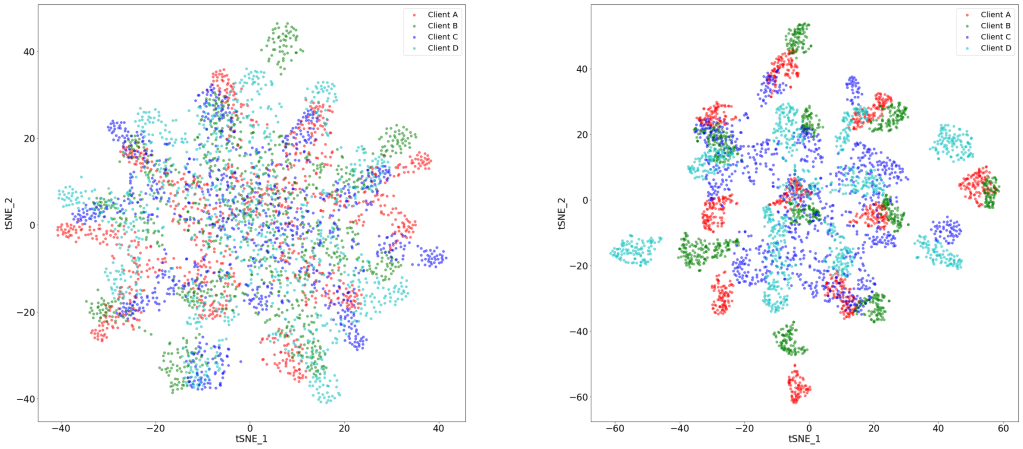
(b) mt,mm,sv,up→sy

Figure 3.3: Group Effect GE_t during the 200 rounds of FL. Lower is better. In Figure (a), though we observed a bounce-back behavior of GE after 20 rounds, the negative transfer in each round was decreased by FedKA. Moreover, in Figure (b), FedKA kept reducing GE even after FedAvg started to bounce back, successfully alleviating the negative transfer of model aggregation by 1 time without voting and 4 times with voting, respectively. Compared to the usual model aggregation of FedAvg, FedKA showed great performance in alleviating Group Effect by the global model fine-tuning with federated voting.

3.4.5 Effectiveness in Alleviating the Group Effect

To understand the effectiveness of FedKA in alleviating the negative transfer in Federated Learning (FL), we evaluated Group Effect GE_t in Eq. 3.12 during the 200 rounds of FL with the Digit-Five dataset (Figure 3.3). The GE value represents the amount of negative transfer occurring in FL during model aggregation, where a higher GE value reflects more information loss from the aggregation and a negative value represents a performance gain via the aggregation. In particular, as shown in the graphs, the learning progress had high GE values at the early stages, implicating that the model aggregation results in information loss and degraded model performance. As learning progresses, the GE values keep decreasing implicating the gradual convergence of client models towards the target domain distribution. The results show that FedKA can greatly alleviate the negative transfer in the model aggregation of FL increasing the global model's performance in unseen tasks.

To further verify the effectiveness of FedKA in alleviating negative transfer, we employ t-SNE [78] to visualize the extracted feature distributions from



(a) FedAvg (Resnet18) (b) FedKA (Resnet18)

Figure 3.4: T-SNE visualization of different client domain feature distributions in Digit-Five when applying Resnet18 as the encoder backbone.

different client data domains based on the learned global model (Figure 3.4). Apparently, the global model based on FedKA learns better representations for the classification tasks.

3.5 Discussion

Federated Learning (FL) has been adopted in various walks of life to facilitate machine learning on distributed client data. Nevertheless, the data discrepancy between clients usually hinders the effectiveness of model transfer in FL. Traditional domain adaptation methods usually require prior knowledge of different domain data and cannot benefit FL under the constraint of data confidentiality. In this work, we proposed Federated Knowledge Alignment (FedKA) to allow domain feature matching in the global workspace. FedKA improves the transferability of learned domain knowledge alleviating negative transfer in FL. The extensive experiments showed that FedKA could improve the global model's generality to unseen image and text classification tasks. In future work, we aim to employ self-supervised learning methods such as contrastive learning [79, 80, 81] to further improve the model's performance. Moreover, we will also consider the security of the proposed framework encountered with adversarial attacks such as information stealing [82, 83] in our future study.

Chapter 4

Learning to Learn with Reusable Neural Modules

This chapter consolidates my work on reinforcement learning-based learning policy optimization of neural modules [44].

Federated learning (FL) has been facilitating privacy-preserving deep learning in many walks of life such as medical image classification, network intrusion detection, and so forth. Whereas it necessitates a central parameter server for model aggregation, which brings about delayed model communication and vulnerability to adversarial attacks. A fully decentralized architecture like Swarm Learning allows peer-to-peer communication among distributed nodes, without the central server. One of the most challenging issues in decentralized deep learning is that data owned by each node are usually non-independent and identically distributed (non-IID), causing time-consuming convergence of model training. To this end, we propose a decentralized learning model called Homogeneous Learning (HL) for tackling non-IID data with a self-attention mechanism. In HL, training performs on each round's selected node, and the trained model of a node is sent to the next selected node at the end of each round. Notably, for the selection, the self-attention mechanism leverages reinforcement learning to observe a node's inner state and its surrounding environment's state, and find out which node should be selected to optimize the training. We evaluate our method with various scenarios for two different image classification tasks. The result suggests that HL can achieve a better performance compared with standalone learning and greatly reduce both the total training rounds by 50.8% and the communication cost by 74.6% for decentralized learning with non-IID data.

4.1 Module Selection with Reinforcement Learning

Centralized deep learning in high performance computing (HPC) environments has been facilitating the advancement in various areas such as drug discovery, disease diagnosis, cybersecurity, and so on. Despite its broad applications in many walks of life, the associated potential data exposure of training sources and privacy regulation violation have greatly decreased the practicality of such centralized learning architecture. In particular, with the promotion of GDPR [84], data collection for centralized model training has become more and more difficult.

Decentralized deep learning (DDL) is a concept to bring together distributed data sources and computing resources while taking the full advantage of deep learning models. Nowadays, DDL such as Federated Learning (FL) [15] has been offering promising solutions to social issues of data privacy, especially in large-scale multi-agent learning. These massively distributed nodes can facilitate

diverse use cases, such as industrial IoT [85], environment monitoring with smart sensors [86], human behavior recognition with surveillance cameras [87], connected autonomous vehicles control [88, 89], network intrusion detection [90, 91], and so forth.

Though FL has been attracting great attention due to the privacy-preserving architecture, recent years' upticks in adversarial attacks cause its hardly guaranteed trustworthiness. FL encounters various threats, such as backdoor attacks [92, 93, 94], information stealing attacks [95], and so on. On the contrast, fully decentralized architectures like Swarm Learning (SL) [96] leverages the blockchain, smart contract, and other state-of-the-art decentralization technologies to offer a more practical solution. Whereas, a great challenge of it has been deteriorated performance in model training with non-independent identically distributed (non-IID) data, leading to extremely increased time of model convergence.

Our contributions We propose a self-attention decentralized deep learning model called Homogeneous Learning (HL). HL leverages a shared communication policy for adaptive model sharing among nodes. A starter node initiates a training task and by iteratively sending the trained model and performing training on each round's selected node its model is updated for achieving the training goal. Notably, a node selection decision is made by reinforcement learning agents based on the current selected node's inner state and outer state of its surrounding environment to maximize a reward for moving towards the training goal. Finally, comprehensive experiments and evaluation results suggest that HL can accelerate the model training on non-IID data with 50.8% fewer training rounds and reduce the communication cost by 74.6%.

Paper outline This paper is organized as follows. Section 4.2 reviews the most recent work about DDL and methodologies for tackling data heterogeneity problems in model training. Section 4.3 discusses assumptions and definitions used in this research. Section 4.4 presents the technical underpinnings of Homogeneous Learning, including the local machine learning (ML) task model, the reinforcement learning model, and the self-attention mechanism to learn an optimized communication policy. Section 4.5 demonstrates experimental evaluations for tackling various image classification tasks with three baseline models applied. Section 4.6 concludes the paper and gives out a future direction of this work.

4.2 Related Work

In a real-life application, usually data owned by different clients in such a decentralized system are skewed. For this reason, the model training is slow and even diverges. Methodologies for tackling such data heterogeneity such as FL, have been studied for a long time. For example, Sener et al.[97] presented the K-Center clustering algorithm which aims to find a representative subset of data from a very large collection such that the performance of the model based on the small subset and that based on the whole collection will be as close as possible. Moreover, Wang et al.[98] demonstrated reinforcement learning-based client selection in FL, which counterbalances the bias introduced by non-IID data thus speeding up the global model's convergence. Sun et al.[90] proposed the Segmented-FL to tackle heterogeneity in massively distributed network intrusion traffic data, where clients with highly skewed training data are dynamically divided into different groups for model aggregation respectively at each round. Furthermore, Zhao et

al.[99] presented a data-sharing strategy in FL by creating a small data subset globally shared between all the clients. Likewise, Jeong et al.[100] proposed the federated augmentation where each client augments its local training data using a generative neural network. Different from the aforementioned approaches, HL leverages a self-attention mechanism that optimizes the communication policy in DDL using reinforcement learning models. It is aimed to reduces computational and communication cost of decentralized training on skewed data.

4.3 Preliminaries

4.3.1 Classification Task

We specifically consider supervised learning with C categories in the entire dataset D. Let $x \in \mathbb{R}^D$ be a sample and $y \in \{1, 2, ..., C\} = Y$ a label. D consists of a collection of N samples as $D = \{(x_i, y_i)\}_{i=1}^{N}$. Suppose that f denotes a neural network classifier taking an input x_i and outputting a C-dimensional real-valued vector where the jth element of the output vector represents the probability that x_i is recognized as class j. Given $f(x)$, the prediction is given by $\hat{y} = \arg\max_j f(x)_j$ where $f(x)_j$ denotes the jth element of $f(x)$. The training of neural network is attained by minimizing the following loss function with respect to the model parameter θ

$$J(\theta, D) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i; \theta)). \tag{4.1}$$

4.3.2 Decentralized Learning

We assume there are K clients. The kth client has its own dataset $D^{(k)} := \{(x_i, y_i)\}_{i=1}^{N^{(k)}}$ where $N^{(k)}$ is the sample size of dataset $D^{(k)}$. Here, $\cup_{i=1}^{K} D^{(k)} = D$ and $N = \sum_{k=1}^{K} N^{(k)}$. We also suppose that each client cannot share data each other due to some reason, mainly due to data confidentiality. Decentralized deep learning (DDL) is a framework to obtain a global model that is trained over the entire data without sharing distributed samples. For instance, federated learning (FL)[15] consists of the parameter server (PS) and lots of clients. Let G_t be the global model of the PS and $L_t^{(k)}$ be the local model of the client$k$ at the round t. For each training round t, a subset of clients $K_{selected}$ is selected for model training with the latest global model parameters G_t based on their own dataset $D^{(k \in K_{selected})}$. Then, the trained models $L_{t+1}^{(k \in K_{selected})}$ are sent back to the PS for aggregation thus improving the joint global model G_{t+1}.

Moreover, a peer-to-peer DDL system consists of distributed nodes functioning as both the server and the client based on decentralization technologies such as blockchain[96, 101, 102], token-exchange[103], and so on. For example, the token-exchange validates and issues security tokens to enable nodes to obtain appropriate access credentials for exchanging resources without the central server. This is different from FL where the parameter server plays the key role in learning process control of model sharing.

4.3.3 Data Heterogeneity

The challenges related to heterogeneity of nodes in DDL refer to two categories, i.e., data heterogeneity and hardware heterogeneity. Notably, data heterogeneity results in time-consuming convergence or divergence of model learning. Let

$p(x|y)$ be the common data distribution of the entire data D. We assume the common distribution $p(x|y)$ is shared by all nodes. Then, Node$k$ has $p_k(y)$. We first consider an independent and identically distributed (IID) setting, i.e., $p_i(x,y) = p(x|y)p_i(y)$ s.t. $p_i(y) = p_j(y)$ for all $i \neq j$. Under this assumption, the data distribution of the entire dataset can be represented by a node's local data distribution. Unfortunately, in real-life application, samples held by clients are usually skewed with various data distributions, i.e., $p_i(x,y) = p(x|y)p_i(y)$ s.t. $p_i(y) \neq p_j(y)$ for all $i \neq j$. Node1 follows $p1(x,y)$ and Node2 follows $p2(x,y)$. We further define and clarify such data heterogeneity as follows: given samples $\{(x_i, y_i)\}_{i=1}^{N^{(k)}}$ in node$k$'s local dataset $D^{(k)}$, when α samples are from a single main data class $c^{(k)}$ subject to $\alpha > \frac{N^{(k)}}{C}$ and the remaining samples are randomly drawn from the other $C-1$ data classes, the heterogeneity level $H^{(k)}$of node$k$ is formulated as $H^{(k)} = -p(y_i = c^{(k)}) * log(p(y_i \neq c^{(k)}))$ subject to $y_i \in \{y_i\}_{i=1}^{N^{(k)}}$. Moreover, we assign a main data class $c^{(k)} = k\%C$ to node$k$.

4.3.4 Communication Overhead

Communication overhead in DDL usually refers to the payload of shared local model parameters [57, 104] and communication distances between nodes that share a model with each other. We mainly discuss the later case here. In particular, let $d_{i,j}$ be the communication distance from node$i$ to node$j$. $\text{Dis}_{i\times j}$ is a symmetrical matrix where the bidirectional distances between two nodes are equal and the distance to a node itself $d_{i,j|i=j}$ is zero. In addition, each distance $d_{i,j|i\neq j}$ in the matrix is a random numerical value taken between 0 and β, where β denotes the upper bound of the relative distance (Equation 4.2).

$$\text{Dis}_{i\times j} = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,j} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ d_{i,1} & d_{i,2} & \cdots & d_{i,j} \end{pmatrix}, \tag{4.2}$$

$$\text{subject to: } d_{i,j|i=j} = 0,\ d_{i,j} = d_{j,i},\ d_{i,j|i\neq j} \in (0, \beta].$$

4.4 Methodology

We propose a novel decentralized deep learning architecture called Homogeneous Learning (HL) (Fig. 4.1). HL leverages reinforcement learning (RL) agents to learn a shared communication policy of node selection, thus contributing to fast convergence of model training and reducing communication cost as well. In HL, each node has two machine learning (ML) models, i.e., a local ML task model $L^{(k)}$ for the multi-classification task and an RL model $L^{\text{DQN}}$for the node selection in peer-to-peer communications.

4.4.1 Specialized Networks with Specialized Processing

We assume the K nodes in HL share the same model architecture for a classification task, which we call a local ML task model. Let y_i be the layer$i$'s output of $L^{(k)}$. $y_i = f_i(W_i y_{i-1}), i = 1, ..., p, y_0 = x$, where f_i is the activation function, W_i is the weight matrix of layer$i$, y_{i-1} represents the output of the previous layer, and p is the number of layers in $L^{(k)}$. Notably, we employ a three-layer convolutional neural network (CNN) with an architecture as follows: the first convolutional layer of the CNN model has a convolution kernel of size 5×5 with

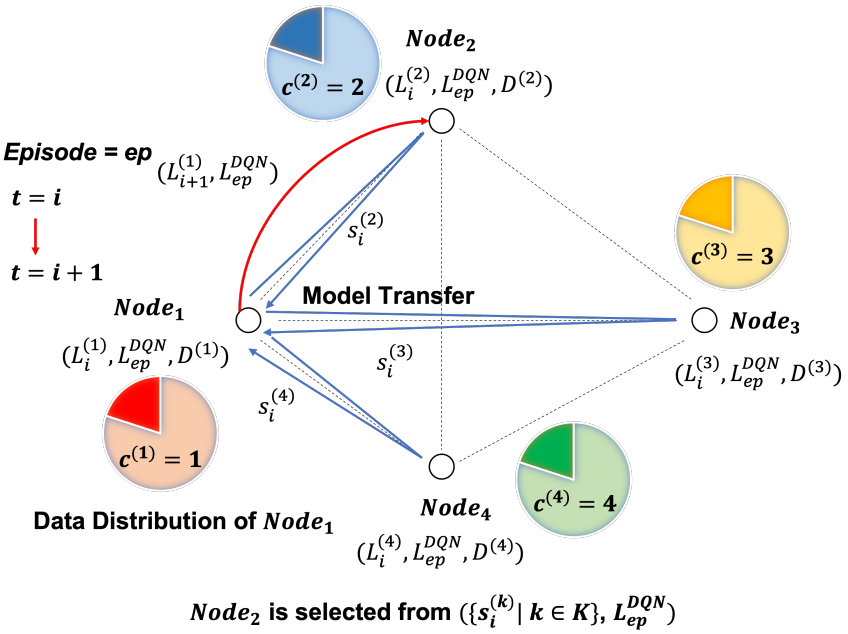


Figure 4.1: The framework of Homogeneous learning.

a stride of 1 and it takes one input plane and it produces 20 output planes, followed by a ReLU activation function; the second convolutional layer takes 20 input planes and produces 50 output planes and it has a convolution kernel of size 5×5 with a stride of 1, followed by ReLU; the output is flattened followed by a linear transformation of a fully connected layer, which takes as input the tensor and outputs a tensor of size C representing the C categories. Moreover, the categorical cross-entropy is employed to compute a loss $J(L_t^{(k)}, D^{(k)})$. After that, we apply as a learning function the Adam to update the model.

4.4.2 Learning Policy Optimization with Markov Models

In addition to the local ML task model, each node$k$ in HL is also associated with a reinforcement learning (RL) model L^{DQN}. The goal of the RL model is to learn a communication policy for the node selection in decentralized learning. There are three main components of the RL model, the state s, the action a, and the reward r. Based on the input state s, the RL model outputs an action a for the next node selection, and at the same time, updates itself by correlating the attained reward r with the performed action a. As a result, the recursive self-improvement of the RL model allows a node to constantly explore the relation between the system's performance and the selection policy (i.e., the self-attention mechanism in HL), contributing to faster convergence of model learning.

Every round t, a RL model observes the state s_t from two different sources, i.e., model parameters $s_t^{(k)}$ of the selected node$k$ and parameters of models in the surrounding environment $\{s_t^{(i)} | i \in K, i \neq k\}$. In particular, we employ a deep Q-network (DQN), which approximates a state-value function in a Q-learning framework with a neural network. Let y_i^{DQN} be the layer$i$'s output of L^{DQN}. $y_i^{DQN} = f_i^{\text{DQN}}(W_i^{DQN} y_{i-1}^{DQN}), i = 1, ..., q, y_0^{DQN} = s$, where f_i^{DQN} is the activation function of layer$i$, W_i^{DQN} is the weight matrix of layer$i$, y_{i-1}^{DQN} represents the output of the previous layer, and q is the number of layers in L^{DQN}. Notably, a DQN model consisting of three fully connected layers is applied (Fig. 4.2).

The two hidden layers consist of 500 and 200 neurons respectively, using as an activation function the ReLU. The output layer with a linear activation function consists of K neurons that output the rewards for selecting each node$k$ respectively, $k \in \{1, 2, ..., K\}$. Furthermore, at each round t, the node with the largest reward will be selected. $\hat{a}_t = \arg\max_j f^{\text{DQN}}(s_t)_j$. Consequently, the RL model selects and sends the trained local model $L_{t+1}^{(k)}$ of node$k$ to the next node a_t. As such, the local ML task model of nodea_t is then updated to $L_{t+1}^{(k)}$.

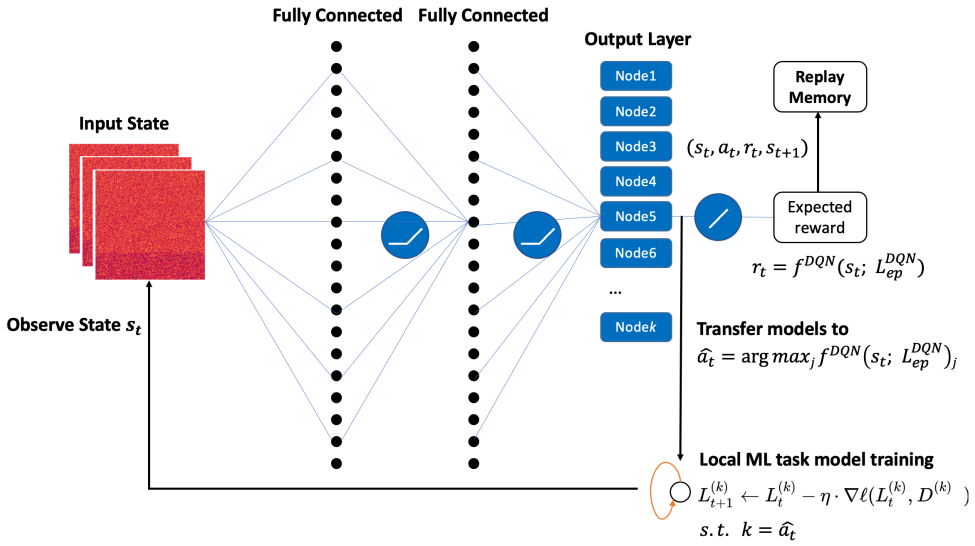


Figure 4.2: Module selection based on reinforcement learning

To understand the training of the RL model, we first define the input state s_t. The state s_t is a concatenated vector of the flattened model parameters of all nodes in the systems. $s_t = \{s_t^{(k)} | k \in K\}$. To efficiently represent the state and compute the RL model prediction, we adopt the principal component analysis (PCA) to reduce the dimension of the state s_t from an extremely large number (e.g., 33580 dimensions for the model parameters used in an MNIST classification task with an input size of 28×28) to K, where K is the number of nodes. K is adopted due to the minimum possible dimension of a PCA-based output vector is the number of input samples. Then, we define the output reward r_t. Every round t, a trained ML task model is evaluated on a hold-out validation set D_{val}, and the reward r_t can be computed from the validation accuracy ValAcc_t, the communication distance between the current node k and the next selected node a_t, and a penalty of minus one for taking each training step. $r_t = 32^{(\text{ValAcc}_t - \text{GoalAcc})} - d_{a_{t-1},a_t} - 1$, where GoalAcc denotes the desired performance on the validation set and d_{a_{t-1},a_t} is the communication distance drawn from the distance matrix $\text{Dis}_{i \times j}$. We employ an exponentially increasing function $32^{(\cdot)}$ to distinguish between different validation results when the ML task model is close to convergence when only small variance is observed in the results. In addition, an episode reward R is the accumulative reward of the current reward and discounted future rewards in the whole training process of HL. $R = \sum_{t=1}^{T} \gamma^{t-1} r_t$, where T is the total training rounds of HL in one episode.

With DQN, we often use experience replay during training. A RL model's experience at each time step t is stored in a data set called the replay memory. Let e_t be the model's experience at time t. $e_t = (s_t, a_t, r_t, s_{t+1})$, where r_t is the reward given the current state-action pair (s_t, a_t) and s_{t+1} is the state of the ML task models after training. We assume a finite size limit M of the replay

memory, and it will only store the last M experiences. Moreover, to facilitate constant exploration of a RL model, epsilon is a factor to control the probability of the next node being selected by the RL model. In particular, for each round, a random numerical value between 0 and 1 is obtained and compared with the current epsilon value Epsilon_{ep} where ep denotes the current episode. Then if the randomly picked value is greater than Epsilon_{ep}, the next node will be selected by the RL model. Otherwise, a random action of node selection will be performed. For either case, an experience sample $e_t = (s_t, a_t, r_t, s_{t+1})$ will be stored in the replay memory. The decentralized learning terminates when either the model achieves the desired performance on the validation set or exceeds a maximum number of rounds T_{max}, the learning progress of which is called an episode of HL. For each episode, we apply the epsilon decay ρ to gradually increase the possibility of the RL model's decision-making. $\text{Epsilon}_{ep+1} = \text{Epsilon}_{ep} \cdot e^{-\rho}$, where Epsilon_{ep+1} is the computed epsilon for the next episode and e is the Euler's number that is approximately equal to 2.718.

Furthermore, at the end of each episode ep, the RL model is trained on a small subset b of samples randomly drawn from the replay memory. We adopt as a learning function the Adam. Then, the optimization of the DQN model is formulated in (4.3). The updated DQN model is shared with the next selected node. As such, the RL model performs better and better in predicting the expected rewards of selecting each node for the next round, which results in the increase of the episode reward R by selecting the node with the largest expected reward at each round t.

$$\begin{aligned} \hat{r_{t+1}} &= \max_{a_i} f^{\text{DQN}}(s_{i+1}; L_{ep}^{\text{DQN}})_{a_i}, \\ \hat{r_t} &= f^{\text{DQN}}(s_i; L_{ep}^{\text{DQN}})_{a_i}, \\ Q(L_{ep}^{\text{DQN}}) &= \sum_{i=1}^{B} \ell(r_t + \gamma\, \hat{r_{t+1}}, \hat{r_t}), \\ \theta^* &= \arg\min_{\theta} Q(\theta), \\ \text{subject to: } \theta &= L_{ep}^{\text{DQN}}, \end{aligned} \tag{4.3}$$

where a_i denotes the predicted next step's action that maximizes the future reward, γ denotes the discount factor of the future reward, B denotes the number of samples in the subset b, and Q is the mean squared error loss function.

Finally, the model training of HL is formulated as Algorithm 1. Algorithm 2 demonstrates the application phase of HL after obtaining the optimized communication policy of node selection.

4.5 Experiments

4.5.1 Datasets and Architecture

We evaluated the proposed method based on two different image classification tasks of MNIST and Fashion-MNIST. MNIST [74] is a handwritten digit image dataset containing 50,000 training samples and 10,000 test samples labeled as 0-9, and Fashion-MNIST[105] is an image collection of 10 types of clothing containing 50,000 training samples and 10,000 test samples labeled as shoes, t-shirts, dresses, and so on. The image data in these two datasets are grayscale with a size of 28×28. Moreover, we considered both a 10-node scenario and a 100-node scenario of HL for tackling the two classification tasks respectively. The machine learning library we used to build the system is Tensorflow. All experiments were conducted

Algorithm 1 Model Training of Homogeneous Learning

initialize L_1^{DQN}
for each episode $ep = 1, 2, ...$ **do**
  initialize $L_0^{(a_0)}$ ▷ a_0 is the starter node
  for each step $t = 1, 2, ...$ **do**
    while $\text{ValAcc}_t < \text{GoalAcc}$ and $t < T_{max}$ **do**
      $\text{ValAcc}_{t+1}, a_t, L_{t+1}^{(a_{t-1})} = HL(L_t^{(a_{t-1})}, L_{ep}^{DQN})$
      Send $\{L_{t+1}^{(a_{t-1})}, L_{ep}^{DQN}\}$ to a_t for the next step's model training
    end while
  end for
  $\hat{r_{t+1}} = \max_{a_i} f^{\text{DQN}}(s_{i+1}; L_{ep+1}^{\text{DQN}})_{a_i}$
  $\hat{r_t} = f^{\text{DQN}}(s_i; L_{ep+1}^{\text{DQN}})_{a_i}$
  $L_{ep+1}^{DQN} = \arg\min_{L_{ep+1}^{\text{DQN}}} \sum_{i=1}^{B} \ell(r_t + \gamma \hat{r_{t+1}}, \hat{r_t})$
  $\text{Epsilon}_{ep+1} = \text{Epsilon}_{ep} \cdot e^{-\rho}$
end for

function $\text{HL}(L_t^{(a_{t-1})}, L_{ep}^{DQN})$
  $L_{t+1}^{(a_{t-1})} = Train(L_t^{(a_{t-1})}, D^{(a_{t-1})})$
  $\text{ValAcc}_{t+1} = Acc(D_{val}; L_{t+1}^{(a_{t-1})})$
  $s_t^{(a_{t-1})} = L_{t+1}^{(a_{t-1})}$
  $s_t^{(i)} = L_t^{(i)}$ subject to $i \in K, i \neq a_{t-1}$
  $s_t = \{s_t^{(a_{t-1})}, s_t^{(i)} | \, i \in K, \, i \neq a_{t-1}\}$
  $\hat{a_t} = \arg\max_j f^{\text{DQN}}(s_t; L_{ep}^{DQN})_j$
  $r_t = 32^{(\text{ValAcc}_t - \text{GoalAcc})} - d_{a_{t-1}, \hat{a_t}} - 1$
  Add $\{s_{t-1}, a_{t-1}, r_{t-1}, s_t\}$ to the replay memory

  return $\text{ValAcc}_{t+1}, \hat{a_t}, L_{t+1}^{(a_{t-1})}$
end function

Algorithm 2 Application of Homogeneous Learning

initialize $L_0^{(a_0)}$
obtain L^{DQN}
for each step $t = 1, 2, ...$ **do**
  while $\text{ValAcc}_t < \text{GoalAcc}$ **do**
    $L_{t+1}^{(a_{t-1})} = Train(L_t^{(a_{t-1})}, D^{(a_{t-1})})$
    $s_t^{(a_{t-1})} = L_{t+1}^{(a_{t-1})}$
    $s_t^{(i)} = L_t^{(i)}$ subject to $i \in K, i \neq a_{t-1}$
    $s_t = \{s_t^{(a_{t-1})}, s_t^{(i)} | \, i \in K, \, i \neq a_{t-1}\}$
    $\hat{a_t} = \arg\max_j f^{\text{DQN}}(s_t; L^{\text{DQN}})_j$
    Send $\{L_{t+1}^{(a_{t-1})}, L^{\text{DQN}}\}$ to $\hat{a_t}$ for the next step's model update
  end while
end for

on a GPU server with 60 AMD Ryzen Threadripper CPUs, two NVidia Titan RTX GPUs with 24 GB RAM each, and Ubuntu 18.04.5 LTS OS.

To compare the performance, we adopted three different baseline models, which are a centralized learning model based on the data collection of all nodes, a decentralized learning model based on a random communication policy, and a standalone learning model based on a node's local data without communication. For each type of model, we used the same architecture of the ML task model and the same training hyperparameters. We assigned the training goal of a model validation accuracy of 0.80 for the MNIST classification task and 0.70 for the Fashion-MNIST classification task respectively, using the hold-out test set in the corresponding dataset.

In addition, for the standalone learning, we adopted the early stopping to monitor the validation loss of the model at each epoch with a patience of five, which automatically terminated the training process when there appeared no further decrease in the validation loss of the model for the last five epochs. In both the centralized learning and the standalone learning, evaluation was performed at the end of each training epoch. On the other hand, in the two decentralized learning cases, due to multiple models existing in the system, evaluation was performed on the trained local model of each step's selected node with the same hold-out test set above.

Furthermore, for the decentralized learning, each node$k$ owned a total of 500 skewed local training data that have a heterogeneity level $H = 0.56$ ($p(y_i = c^{(k)}) = 0.8$) subject to $y_i \in \{y_i\}_{i=1}^{N^{(k)}}$. The discussion on various heterogeneity levels is in Section 4.5.3. In HL, to generate the distance matrix, the relative communication cost represented by the distance between two different nodes $d_{i,j|i\neq j}$ takes a random numerical value between 0 and 0.1. A random seed of 0 was adopted for the reproducibility of the distance matrix (See Appendix 4.5.3). For the local ML task model training, we adopted an epoch of one with a batch size of 32. A further discussion on the selection of these two hyperparameters can be found in Appendix 4.5.3. The Adam was applied as an optimization function with a learning rate of 0.001.

4.5.2 Coordination Efficiency

Common Communication Policy Learning As aforementioned, each node$k$ has a specific main data class $c^{(k)}$. We considered a starter node that had a main data class of digit '0' for MNIST and a main class of T-shirt for Fashion-MNIST. Then, starting from the starter node, a local ML task model was trained on the current node's local data and sent to the next step's node decided by either the RL model or a random action every step, depending on the epsilon of the current episode (we adopted an initial epsilon of one and a decay rate of 0.02). For each episode, we applied a maximum step of 35 for MNIST and 100 for Fashion-MNIST. Moreover, the ML task model and the RL model were updated using the hyperparameters in Table. 4.1. In addition, we applied a maximum replay memory size of 50,000 and a minimum size of 128, where the training of the DQN model started only when there were more than 128 samples in the replay memory and the oldest samples would be removed when samples were more than the maximum capacity.

For each episode, we computed the step rewards and the episode reward for the model training to achieve the desired performance. With the advancement of episodes, the communication policy evolved to improve the episode reward thus benefiting better decision-making of the next-node selection. Fig. 4.3 illustrates

Table 4.1: Hyperparameters for Homogeneous Learning

ML Task Model		RL Model	
Epoch	1	Episode	120
Batch size	32	Future reward discount	0.9
Learning rate	0.001	Epsilon decay	0.02
Optimization function	Adam	Epoch	1
Maximum step	35 (MNIST)/ 100 (Fashion-MNIST)	Batch size	16
		Learning rate	0.001

the episode reward and the mean reward over the last 10 episodes of HL in the 10-node and 100-node scenarios for MNIST and Fashion-MNIST respectively.

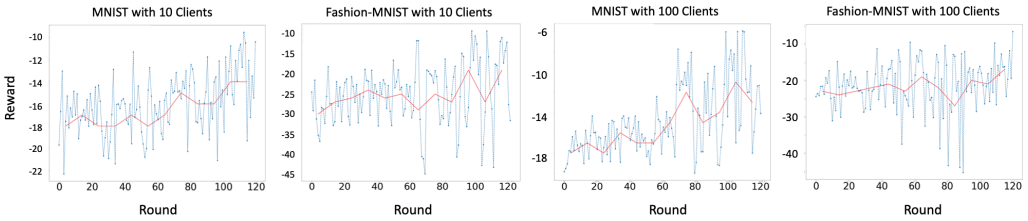


Figure 4.3: With the increase of training episodes, the mean reward over last 10 episodes is gradually increasing. The DQN model learned a better communication policy by training on samples from the replay memory, contributing to a faster convergence of model training.

4.5.3 Computational and Communication Cost

Computational cost refers to the required total rounds for a system to achieve the desired performance and was evaluated for all methods. Communication cost refers to the total communication distance of model sharing from the starter node to the last selected node and was evaluated for the two decentralized learning methods. Notably, to evaluate the computational and communication cost, we conducted 10 individual experiments using different random seeds for each method and adopted as final results the best cases of node selection over the last five episodes when the learned communication policy was prone to settling. The experiments were performed in the 10-node scenario for the MNIST task.

As shown in Fig. 4.5.a, due to limited local training data, the standalone learning appeared to be extremely slow after the validation accuracy reached 0.70. It terminated with a final accuracy of around 0.75 with the early-stopping strategy. Moreover, by comparing the decentralized learning methods with and without the self-attention mechanism, the result suggests that our proposed method of HL can greatly reduce the total training rounds facilitating the model convergence. In addition, though centralized learning shows the fastest convergence, it suffers from problems of data privacy.

As shown in Fig. 4.4.b, the bottom and top of the error bars represent the 25_{th} and 75_{th} percentiles respectively, the line inside the box shows the median value, and outliers are shown as open circles. As a result, it shows that HL can greatly reduce the total training rounds by 50.8% and the communication cost by 74.6% in decentralized learning of the 10-node scenario for the MNIST task.

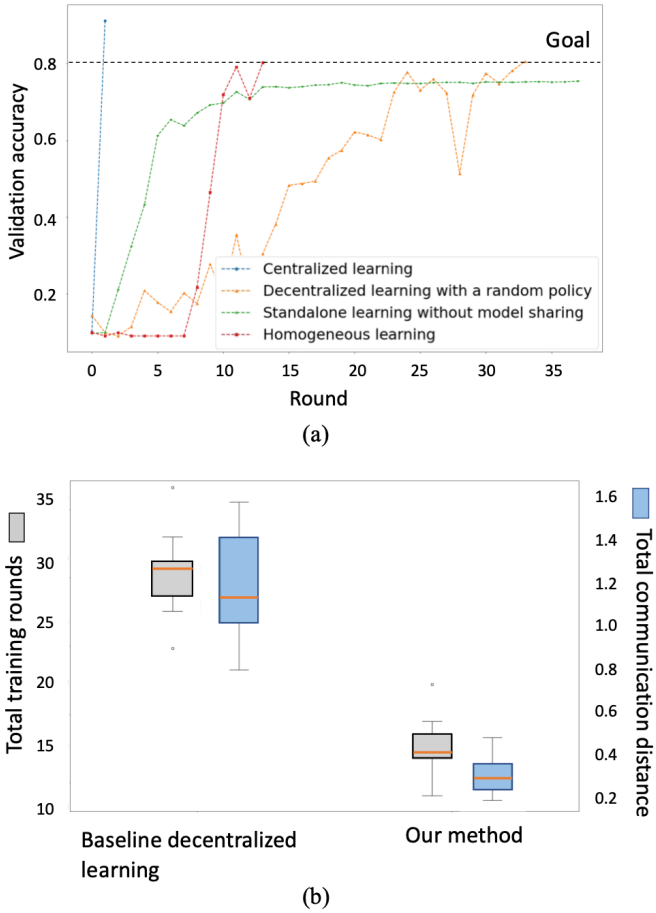


Figure 4.4: (a) Total training rounds based on different methods. (b) Cost comparison between the random policy-based decentralized learning and our method HL. Each error bar illustrates 10 individual experiments' results.

HL with Various Heterogeneity Levels We further studied the performance of the proposed method with different heterogeneity levels $H = \{0.24, 0.56, 0.90\}$ ($p(y_i = c^{(k)}) = \{0.6, 0.8, 0.9\}$ subject to $y_i \in \{y_i\}_{i=1}^{N^{(k)}}$). We evaluated the model performance in the 10-node scenario for the MNIST task. For the cases of $H = \{0.24, 0.56\}$, we applied a maximum training step of 35 as defined above. For the case of $H = 0.90$, we applied a maximum training step of 80 instead due to a challenging convergence of the ML task model using the highly skewed local training data. Fig. 5 illustrates the comparison of computational cost between HL and the random policy-based decentralized learning.

Communication Distance Matrix

Fig. 4.6 illustrates the generated distance matrix $D_{i\times j}$ in the 10-node scenario when applying a β of 0.1 and a random seed of 0.

Model Distribution Representation Optimization

Under the assumption of data heterogeneity, to allow a reinforcement learning (RL) agent to efficiently learn a communication policy by observing model states in the systems, a trade-off between the batch size and the epoch of local foundation model training was discussed. Fig. 4.7 illustrates the trained models' weights distribution in the 10-node scenario after applying the principal compo-

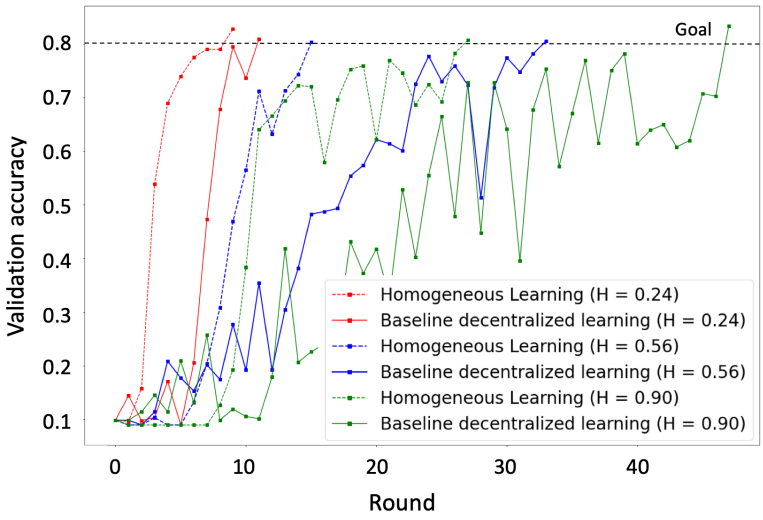


Figure 4.5: Total training rounds when applying local training data with various heterogeneity levels. The dash lines are the results of HL and the solid lines are the results of the random policy-based decentralized learning. Different colors represent different heterogeneity levels $H = \{0.24, 0.56, 0.90\}$. As we can see, HL becomes more efficient when training on distributed data with a higher heterogeneity level, contributing to a larger ratio of reduced total training rounds.

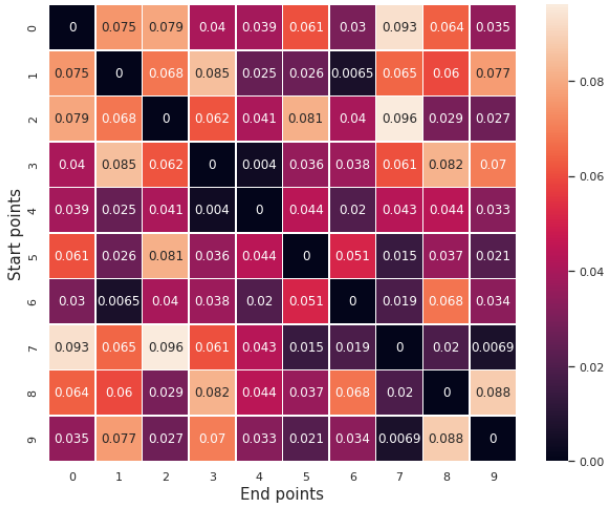


Figure 4.6: The distance matrix $D_{i\times j}$ in the 10-node scenario.

nent analysis (PCA), with different batch sizes and epochs applied to train on the MNIST dataset. Moreover, it shows the 100-node scenario where each color represents nodes with the same main data class. As shown in the graphs, various combinations of these two parameters have different distribution representation capabilities. By comparing the distribution density and scale, we found that when adopting a batch size of 32 and an epoch of one the models distribution was best represented, which could facilitate the policy learning of an agent.

4.6 Discussion

Decentralized deep learning (DDL) leveraging distributed data sources contributes to a better neural network model while safeguarding data privacy. De-

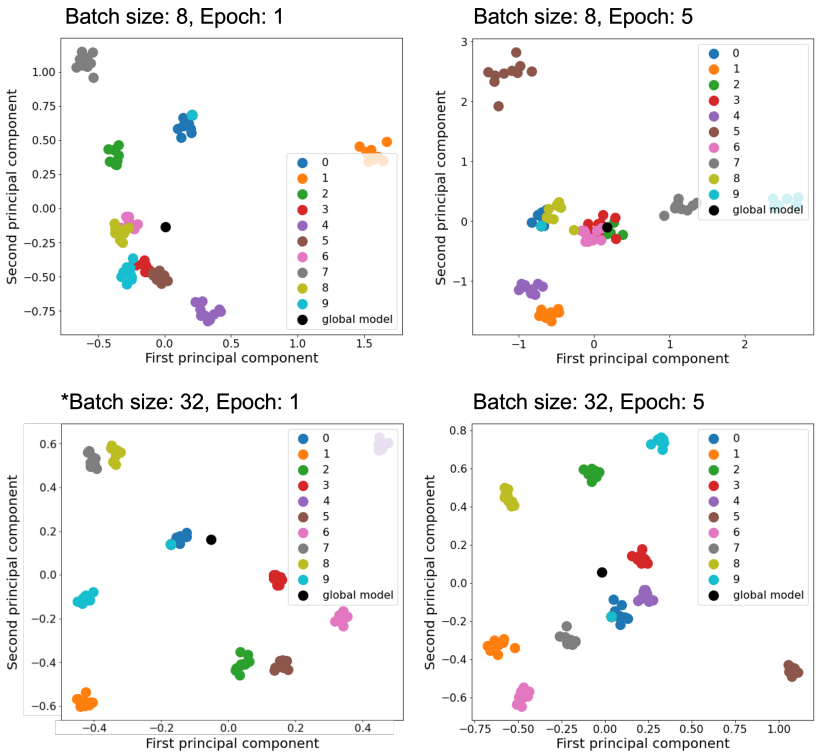


Figure 4.7: Model distribution representation optimization.

spite the broad applications of DDL models such as federated learning and swarming learning, the challenges regarding edge heterogeneity especially the data heterogeneity have greatly limited their scalability. In this research, we proposed a self-attention decentralized deep learning method of Homogeneous Learning (HL) that recursively updates a shared communication policy by observing the system's state and the gained reward for taking an action based on the observation. We comprehensively evaluated the proposed method in the 10-node and 100-node scenarios for tackling two different image classification tasks, applying as criteria the computational and communication cost. The evaluation results show that HL can greatly reduce the training cost with highly skewed distributed data. In future, a decentralized learning model that can leverage various communication policies in parallel is considered for the further study of HL.

Chapter 5

Priors, Attractors, and Inductive Biases

This chapter consolidates my work on global workspace and associative memory-enabled sparse Transformer [32].

Emerging from the monolithic pairwise attention mechanism in conventional Transformer models, there is a growing interest in leveraging sparse interactions that align more closely with biological principles. Approaches including the Set Transformer and the Perceiver employ cross-attention consolidated with a latent space that forms an attention bottleneck with limited capacity. Building upon recent neuroscience studies of Global Workspace Theory and associative memory, we propose the **A**ssoc**i**ative **T**ransformer (AiT). AiT induces low-rank explicit memory that serves as both priors to guide bottleneck attention in the shared workspace and attractors within associative memory of a Hopfield network. Through joint end-to-end training, these priors naturally develop module specialization, each contributing a distinct inductive bias to form attention bottlenecks. A bottleneck can foster competition among inputs for writing information into the memory. We show that AiT is a sparse representation learner, learning distinct priors through the bottlenecks that are complexity-invariant to input quantities and dimensions. AiT demonstrates its superiority over methods such as the Set Transformer, Vision Transformer, and Coordination in various vision tasks.

5.1 Sparse Attention in Transformers

The predominant paradigm in conventional deep neural networks has been characterized by a monolithic architecture, wherein each input sample is subjected to uniform processing within a singular model framework. For instance, Transformer models use pairwise attention to establish correlations among disparate segments of input information [7, 9]. Emerging from the pair-wise attention mechanism, there is a growing interest in leveraging modular and sparse interactions that align more closely with biological principles. This sparsity attribute has demonstrated advantages in enhancing model performance and learning efficiency, making it a crucial element for intelligent entity learning [106, 107, 108].

Modularization of knowledge can find resonance with the neuroscientific grounding of the Global Workspace Theory (GWT) [17, 18, 19, 20]. GWT explains a fundamental cognitive architecture for information processing within the brain, where diverse specialized modules compete to write information into a shared workspace through a communication bottleneck. The bottleneck facilitates the processing of content-addressable information through attention that is guided by working memory [21, 22]. The coordination method [30] represents the initial attempt to assess the effectiveness of GWT in conventional neural network

models. Unfortunately, this method relies on iterative cross-attention for both information writing and retrieval within the shared workspace. When examining information retrieval in the human brain, it is evident that memory typically encompasses both working memory and long-term memory in the hippocampus. Specifically, the hippocampus operates on Hebbian learning for retrieving information from working memory, akin to the associative memory found in Hopfield networks [12, 13]. Our research has revealed that replacing such a repetitive attention-based mechanism with a consolidated, more biologically-plausible associative memory can lead to improved model performance. Associative memory has the capability to directly store and retrieve patterns from the shared workspace without the need for additional parameters by relying on an energy function, which fundamentally differs from an attention mechanism. Our objective is to introduce a shared workspace augmented with associative memory into a Transformer model, thereby facilitating a more comprehensive and efficient association of information fragments.

To this end, we propose the **A**ssoci**i**ative **T**ransformer (AiT) based on a novel global workspace layer augmented by associative memory. The global workspace layer entails three main components: 1) the squash layer: input data is transformed into a list of patches regardless of which samples they come from, 2) the bottleneck attention: patches are sparsely selected to learn a set of priors in low-rank memory based on a bottleneck attention mechanism, and 3) the Hopfield network: information is broadcast from the shared workspace to update the current input based on the associative memory of a Hopfield network. Moreover, the bottleneck attention and the low-rank memory contributes to reduced model complexity. However, cascading multiple of these components may lead to difficulty in the emergence of specialized priors in explicit memory. As information flows through multiple layers, it becomes more challenging to maintain specialized priors from diluted representations. Consequently, learning specialized priors in layers cascaded in depth requires a mechanism that counteracts this inherent loss of input specificity. To overcome this challenge, we propose the bottleneck attention balance loss to encourage the diverse selection of inputs in the shared workspace. Through end-to-end training, we show the emerging specialization of low-rank priors, contributing to enhanced performance in vision tasks. This distinguishes our work from previous literature, which relied on latent memory comprising indistinct priors with the same dimension as the input, such as Set Transformer [34], Perceiver [28], and Luna [35]. The no-free-lunch theorem [109, 110] states that a set of inductive bias over the space of all functions is necessary to obtain generalization. We demonstrate that the specialization of priors serves as critical inductive biases, encouraging competition among input data and inducing sparsity in the attention mechanism of Transformer models.

Overall, the main contributions of this work are as follows. (1) This work proposes a more biologically plausible learning framework called Associative Transformer (AiT) based on the Global Workspace Theory and associative memory. (2) AiT is a sparse representation learner, leveraging sparse bottleneck attention enhanced by a novel attention balance loss to acquire naturally emerging specialized priors. (3) We devise low-rank priors that are adaptively encoded and decoded for increased memory capacity. AiT can learn a large set of specialized priors (up to 128) from a diverse pool of patches (up to 32.8k). (4) The learned priors serve as attractors within the associative memory of a Hopfield network, enabling information broadcast from the workspace. This is the first work to incorporate the Hopfield network as an integral element in a sparse attention mechanism.

5.2 Related Work

This section provides a summary of relevant research concerning sparse attention architectures. We investigate and compare these studies based on their relatedness to the global workspace theory in terms of several key conditions (please see Appendix 5.7.2 for a complete comparison).

Transformer models do not possess inductive biases that allow the model to attend to different segments of the input data [43]. To enhance Transformer models, studies of sparse attention architectures explored consolidating latent memory to extract contextual representations from input data [33, 28, 30, 29, 34, 35]. For instance, Perceiver [28] and Perceiver IO [29] used iterative cross-attention with a latent array as priors and a latent transformation applied to the priors, to capture dependencies across input data. Set Transformer [34] and Linear Unified Nested Attention (Luna) [35] employed iterative cross-attention, but without using a latent transformation. Other attention mechanisms that rely on strong inductive biases with predefined network modularization are omitted [111]. In our method, distinct priors naturally emerge through end-to-end training. Moreover, the previous methods using latent memory necessitated priors with the same dimension as the input. In contrast, we devise low-rank priors that can be encoded and decoded adaptively for increased memory capacity.

In the same vein of building sparse attention mechanisms through a shared workspace, Coordination [30] used iterative cross-attentions via a bottleneck to encourage more effective module communication. They argued that more flexibility and generalization could emerge through the competition of specialized modules. However, the priors in the coordination method possess the same dimension as the input, and the number of priors is limited to fewer than 10. The evaluation was also restricted to simple tasks. Unlike the coordination method, we propose low-rank explicit memory to learn a larger set of specialized priors (up to 128) from a pool of patches (up to 32.8k). Moreover, the coordination method relies on iterative cross-attentions to learn such priors, while this work focuses on a novel learning method of associative memory-augmented attention.

Furthermore, external memory such as tape storage and associative memory has been successfully employed [36, 37, 38, 39]. Recent studies explored the potential use of Hopfield networks [12] and their modern variants [112, 13] in Transformers. In contrast to these investigations, we incorporate Hopfield networks as an integral element in constructing the global workspace layer, functioning as a mechanism for information broadcast in the shared workspace. This goal is fundamentally different from prior studies focused on using Hopfield networks independently of the attention mechanism.

5.3 Inspecting Attention Heads in Vision Transformers

Vision Transformers (ViT) tackle image classification tasks by processing sequences of image patches. The pre-processing layer partitions an image into non-overlapping patches, followed by a learnable linear projection layer. Let $x \in \mathbb{R}^{H \times W \times C}$ be an input, where (H, W) is the resolution of the image and C is the number of channels. x is separated into a sequence of patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where (P, P) is the resolution of each image patch and $N = \frac{HW}{P^2}$ is the number of patches. These patches are mapped to embeddings $v_p \in \mathbb{R}^{N \times E}$ with the linear projection. ViT leverages self-attention where each head maps a query and a set of key-value pairs to an output. The patch embeddings are used to obtain the query, key, and value based on linear transformations

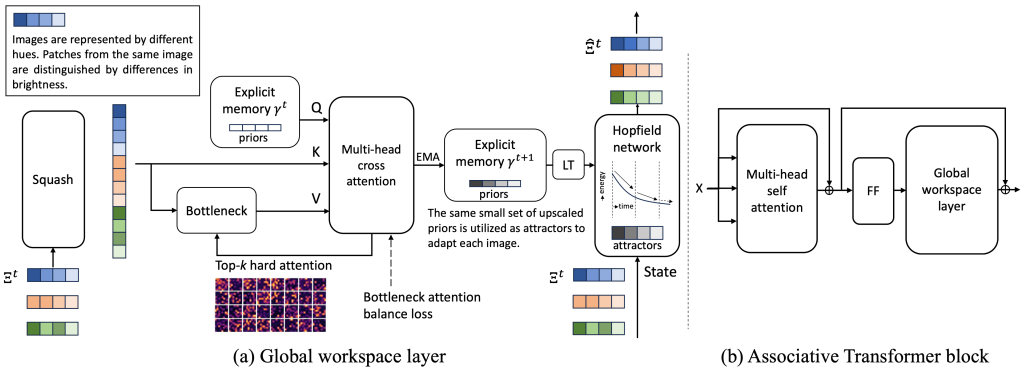


Figure 5.1: The scheme of the Associative Transformer. (a) In a global workspace layer, the input $\mathbb{R}^{B\times N\times E}$ is squashed into vectors $\mathbb{R}^{(B\times N)\times E}$. The squashed representations are projected to a low-rank latent space of dimension $D << E$ and then are sparsely selected and stored in the explicit memory via a fixed bottleneck $k << (B \times N)$. The Hopfield network utilizes the memory to reconstruct the input, where a learnable linear transformation (LT) scales the memory contents to match the input dimension E. (b) The Associative Transformer block consists of sequentially connected self attention, feed-forward layers, and the global workspace layer.

$W^Q \in \mathbb{R}^{E\times D}$, $W^K \in \mathbb{R}^{E\times D}$, and, $W^V \in \mathbb{R}^{E\times D}$. The output is a weighted sum of the values:

$$\mathrm{h}^i(v) = \mathrm{softmax}(\frac{W_i^Q v (W_i^K v)^T}{\sqrt{D}})\, W_i^V v, \tag{5.1}$$

$$\text{Multi-head}(v) = \mathrm{Concat}(\mathrm{h}^1, \ldots, \mathrm{h}^A)\, W^O, \tag{5.2}$$

where W^O is a linear transformation for outputs, and A is the number of attention heads.

We assume that the competition within the pair-wise attention of different patches would be of importance for the model to learn meaningful representations. If such competition exists, a trained model will naturally result in sparser interactions in attention heads. Therefore, we first performed an analysis of the operating modes of different attention heads in a pretrained ViT model by measuring the number of patches each head is attending to. We refer to Appendix 5.7.4 for the detailed experimental settings. The inspection revealed the existing competition among patches and a large redundancy in the pair-wise attention. Less than 80% interactions were activated in ViT, and several heads from the middle layers used only 50% or less interactions with higher sparsity compared to the other layers. Based on the observation, by introducing a bottleneck that limits each attention head's focus to foster competition, we obtain inductive biases for more efficient patch learning.

5.4 Associative Transformer

This section discusses the essential building blocks of the Associative Transformer (AiT), where patches compete to write into the shared workspace through bottleneck attention. The workspace enables an efficient information writing and reading mechanism by learning a set of priors in explicit memory. These priors are low-rank and learned progressively from the input through end-to-end training. The priors guide the bottleneck attention with an emerging specialization

property. Moreover, we extend the learned priors to attractors within the associative memory of a Hopfield network, facilitating information retrieval from memory and efficient association of information fragments.

5.4.1 Global Workspace Layer

We devise an associative memory-augmented attention layer called the *global workspace layer*, which comprises the squash layer, the bottleneck attention guided by low-rank memory, and the information retrieval within the associative memory of a Hopfield network (Figure 5.1). The global workspace layer can be seen as an add-on component on the monolithic Vision Transformer, where the feed-forward layers process patches before they enter the workspace, facilitating abstract relation learning, and the self-attention learns the contextual relations for a specific sample. The global workspace layer learns spatial relations across various samples and time steps.

Squash Layer In self-attention, patches from the same sample are attended to. In our work, we improve the diversity in patch-wise correlation learning beyond one sample using a *squash layer*. The squash layer obtains patch representations from the entire training batch to enable competition among patches not only from the same sample but also from different samples. This differs from traditional approaches where the competition resides within specific samples. The squash layer concatenates patches within one batch $V \in \mathbb{R}^{B\times N\times E}$ into vectors $V \in \mathbb{R}^{(B\times N)\times E}$, which forms a list of patches regardless of the samples they are from. Though the number of patches changes in practice depending on the batch size, the communication bottleneck with a fixed capacity k limits the number of patches the workspace can attend to at any given time. Since the bottleneck decreases the complexity from $O((B\times N)^2)$ to $O((B\times N)\times k)$, using the squash layer increases the diversity of input patches without adding to the complexity. With the greater diversity, a sample's classification task, for instance, can benefit from other patches belonging to the same class within the batch input.

Low-Rank Explicit Memory An explicit memory bank with limited slots aims to learn M *priors* $\gamma = \mathbb{R}^{M\times D}$ where D is the dimension of the prior. The priors in the memory bank are used as various keys to compute the bottleneck attentions that extract different sets of patches from the squashed input. Furthermore, using low-rank priors reduces memory consumption, as a lower dimension $D << E$ is obtained through a down-scale linear transformation.

5.4.2 Bottleneck Attention with a Limited Capacity

The objective of the bottleneck attention is to learn a set of priors that guide attention to various input patches. This is enabled by a cross-attention mechanism constrained by hard attention. We first consider a tailored cross-attention mechanism to update the memory bank based on the squashed input $\Xi^t = V^t \in \mathbb{R}^{(B\times N)\times E}$, then we discuss the case of limiting the capacity via a top-k hard attention. Notably, in the cross-attention, the query is a function of the current memory content $\gamma^t = \{\gamma_i^t\}_{i=1}^M$. The key and value are functions of the squashed input Ξ^t. The attention scores for head i can be computed by $\mathrm{A}_i^t(\gamma^t, \Xi^t) = \mathrm{softmax}(\frac{\gamma^t W_{i,t}^Q (\Xi^t W_{i,t}^K)^T}{\sqrt{D}})$. This is the case of soft attention with limited constraints on the bottleneck capacity. Moreover, the hard attention allows

patches to compete to enter the workspace through a k-size bottleneck, fostering the selection of essential patches. In particular, the top-k patches with the highest attention scores from A_i^t are selected to update the memory. To ensure a stable update across different time steps, we employ the layer normalization and the Exponentially Weighted Moving Average (EWMA) method as follows

$$\text{head}_i^t = \text{top-}k(\text{A}_i^t)\Xi^t W_t^V,\ \hat{\gamma}^t = \text{LN}(\text{Concat}(\text{head}_1^t, \ldots, \text{head}_A^t)W^O), \tag{5.3}$$

$$\gamma^{t+1} = \alpha \cdot \gamma^t + (1-\alpha)\cdot \hat{\gamma}^t,\ \gamma^{t+1} = \frac{\gamma^{t+1}}{\sqrt{\sum_{j=1}^{M}(\gamma_j^{t+1})^2}}, \tag{5.4}$$

where top-k selects the k highest attention scores, LN is the layer normalization, and α is a smoothing factor determining the decay rate of older observations. EWMA ensures the stable memory update with varying batch sizes by accumulating both old γ^t and new memories $\hat{\gamma}^t$.

During the test time, the explicit memory is frozen, functioning as fixed priors, and any memory update from the bottleneck attention will not be retained (Figure 5.8). We only compute γ^{t+1} for the following pattern retrieval step in Hopfield networks for the current batch. To ensure a fair evaluation on the test dataset, the same explicit memory from the training time is utilized across all test batches.

Bottleneck Attention Balance Loss The bottleneck attention and the low-rank memory contribute to reduced model complexity of the global workspace layer. Nevertheless, employing multiple components cascaded in depth might lead to difficulty in the emergence of specialized priors in the explicit memory (Figure 5.9). To overcome this challenge, we propose the bottleneck attention balance loss to encourage the selection of diverse patches from different input positions. The bottleneck attention balance loss $\ell_{\text{bottleneck}}$ comprises two components, i.e., the accumulative attention scores and the chosen instances for each input position. Then, we derive the normalized variances of the two metrics across different positions as follows

$$\ell_{\text{loads}_{i,l}} = \sum_{j=1}^{M}(\text{A}_{i,j,l}^t > 0),\ \ell_{\text{importance}_{i,l}} = \sum_{j=1}^{M}\text{A}_{i,j,l}^t, \tag{5.5}$$

$$\ell_{\text{bottleneck}_i} = \frac{\text{Var}(\{\ell_{\text{importance}_{i,l}}\}_{l=1}^{B\times N})}{(\frac{1}{B\times N}\sum_{l=1}^{B\times N}\ell_{\text{importance}_{i,l}})^2+\epsilon} + \frac{\text{Var}(\{\ell_{\text{loads}_{i,l}}\}_{l=1}^{B\times N})}{(\frac{1}{B\times N}\sum_{l=1}^{B\times N}\ell_{\text{loads}_{i,l}})^2+\epsilon}, \tag{5.6}$$

where $\text{A}_{i,j,l}^t$ denotes the attention score of the input position l for the jth memory slot of head i, $\ell_{\text{importance}}$ represents the accumulative attention scores for all M memory slots concerning each input position, ℓ_{loads} represents the chosen instances for each input position in M memory slots, $\text{Var}(\cdot)$ denotes the variance, and ϵ is a small value to avoid division by zero. Finally, the loss scores for all the heads are summed up as follows: $\ell_{\text{bottleneck}} = \sigma \cdot \sum_{i=1}^{A}\ell_{\text{bottleneck}_i}$ where σ is a coefficient.

5.4.3 Information Retrieval Within Associative Memory

After writing information into the shared workspace, the learned priors can serve as attractors within associative memory. The objective is to reconstruct the

current input patches towards more globally meaningful representations based on these attractors.

Attractors Priors learned in the memory bank act as attractors in associative memory. Attractors have basins of attraction defined by an energy function. Any input state that enters an attractor's basin of attraction will converge to that attractor. The attractors in associative memory usually have the same dimension as input states; however, the priors γ^{t+1} in the memory bank have a lower rank compared to the input. Therefore, we employ a learnable linear transformation $f_{\text{LT}}(\cdot)$ to project the priors into a space of the same dimension, E, as the input before using them as attractors.

Retrieval Using the Energy Function in Hopfield Networks Hopfield networks have demonstrated their potential as a promising approach to constructing associative memory. In particular, a continuous Hopfield network [112, 13] operates with continuous input and output values. The upscaled priors $f_{\text{LT}}(\gamma^{t+1})$ are stored within the continuous Hopfield network and are subsequently retrieved to reconstruct the input state Ξ^t. Depending on an inverse temperature variable β, the reconstructed input $\hat{\Xi}^t$ can be either a metastable state that represents a mixture of various attractors or a fixed state represented by one of the attractors. A large β makes it less likely for metastable states to appear, while a small β increases the likelihood. The continuous Hopfield network employs an energy function to enable the evolution of patches into more globally meaningful representations with respect to the learned attractors. We update each patch representation $\xi^t \in \Xi^t$ by decreasing its energy $E(\xi^t)$ within associative memory as follows

$$E(\xi^t) = -\text{lse}(\beta, f_{\text{LT}}(\gamma^{t+1})\xi^t) + \frac{1}{2}\xi^t\xi^{tT} + \beta^{-1}\log M + \frac{1}{2}\zeta^2, \tag{5.7}$$

$$\zeta = \max_i |f_{\text{LT}}(\gamma_i^{t+1})|, \ \hat{\xi}^t = \arg\min_{\xi^t} E(\xi^t), \tag{5.8}$$

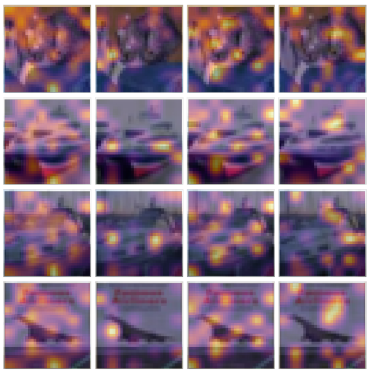
where lse is the log-sum-exp function and ζ denotes the largest norm of attractors. Equation 5.7 describes an iteration that can be applied several times. Usually, we apply just a single step for efficient forward and backward computation during end-to-end training. t is the batch time step, and the iteration time step is implicit. Additionally, a skip connection functioning as the information broadcast from the global workspace is employed to obtain the final output $\Xi^{t+1} = \hat{\Xi}^t + \Xi^t$.

5.5 Experiments

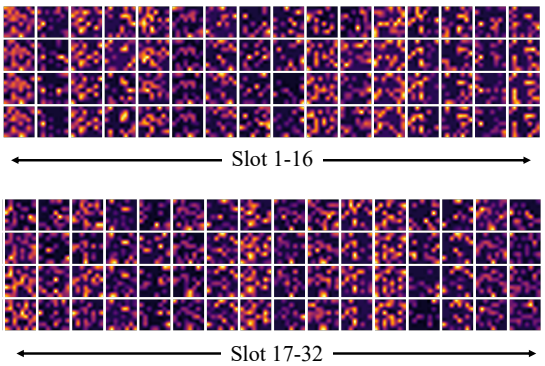
In this section, we discuss the settings and extensive empirical results for image classification and relational reasoning tasks. Our study demonstrates that AiT outperforms the coordination method and other sparse attention-based approaches in terms of both performance and model complexity.

5.5.1 Setup

Datasets We evaluate model performance on two different scales of datasets (1) small (Triangle [30], CIFAR10 [113], and CIFAR100 [113]) and (2) middle (Oxford-IIIT Pet [114] and Sort-of-CLEVR [115]). We train the model on these datasets from scratch using the training split and evaluate using the test split. A detailed description of the datasets can be found in Appendix 5.7.1.



(a) A glimpse of the attention maps at Slot 1 to Slot 4 of four distinct input images.



(b) The attention maps for all 32 slots in the memory bank, applied to four distinct input images. Each memory slot learned to attend to different regions of pixels in input images.

Figure 5.2: Learned distinct memory slot attentions in AiT. Each slot's activation maps highlight a specific area during the selection of relevant image patches.

Model Variants We investigate three different sizes of model configurations, i.e., Small, Medium, and Base. The Base variant setting is adapted from Vision Transformer (ViT) using 12 layers, 12 attention heads for each layer, a hidden dimension of 768, and an MLP dimension of 3072. The Medium variant with 6 layers and the Small variant with 2 layers are added for efficiency comparisons among approaches. The CLS token is removed while the pooled representations of the last dense network layer are used instead since using the CLS token leads to undermined learning results in vision tasks [116, 117].

Hyperparameters The hyperparameters were chosen based on a grid search. A batch size of 512 was employed for the CIFAR datasets and the Triangle dataset, 128 for the Pet dataset, and 64 for the Sort-of-CLEVR dataset. We utilized the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.01. A cosine learning rate scheduler was implemented with an initial learning rate of 1e-5, a warm-up phase of 5 (15) epochs within a total of 100 (300) epochs, and a minimum learning rate set to 1e-6. The smoothing factor of the exponentially weighted moving average, the coefficient σ, and the small value ϵ in the bottleneck balance loss were set to 0.9, 1e-2, and 1e-10, respectively. For AiT, we employed a memory slot size of 32 and a bottleneck attention head size of 8. We used a bottleneck size of 512 for CIFAR and Pet, 64 for Triangle, and 256 for Relational Reasoning. We used 32 memory slots for CIFAR, Triangle, and Relational Reasoning, and 128 slots for Pet (Appendix 5.7.3). Unless otherwise noted, we trained the model for 100 epochs and reported the mean of three individual experiments. The code will be made publicly available.

5.5.2 Classification Tasks

The experiments on image classification tasks include comparisons to a wide range of methods (Table 5.1). We used the author-recommended hyperparameters to re-implement these methods. Regarding the coordination method, we have examined the efficacy of its variants with different model configurations. The default coordination model consists of 4 layers, with parameter sharing among different attention layers. Coordination-D is a deeper model with 8 layers using the parameter sharing. Coordination-H is a high-capacity model with 4 layers that employ individual parameters. Coordination-DH is a high-capacity model with

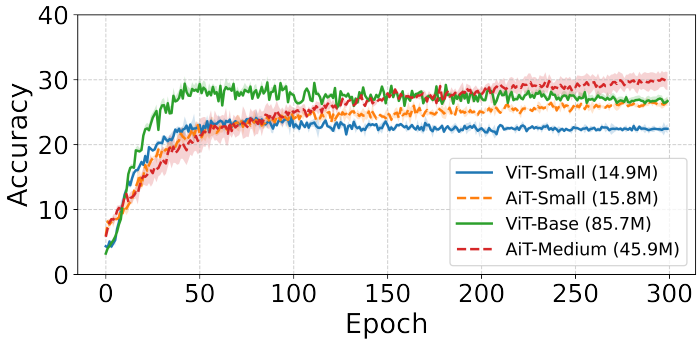


Figure 5.3: Comparison on the Pet dataset, which shows enhanced accuracy for AiT.

8 layers. The results show that AiT achieved better performance compared to the coordination methods. The AiT performance also increased when scaling it from AiT-Small to AiT-Base, while the coordination methods appeared difficult to scale with the increasing number of layers and parameters, as seen in the case of Coordination-DH. Moreover, AiT outperformed the other baseline methods, demonstrating strong performance. For instance, compared to ViT-Base with 85.7M parameters, AiT-Medium is a shallower model with only 45.9M parameters. Nevertheless, AiT-Medium exhibited an average performance of 81.58%, surpassing the ViT-Base model's average of 80.46% and requiring much fewer parameters. AiT also outperformed sparse attention-based methods such as Perceiver and Set Transformer.

We extended the evaluation to a middle-sized dataset of Oxford Pet. We used a patch size of 16. A larger memory of 128 slots was employed due to the higher resolution and the increased data class complexity. For the Oxford Pet dataset, we trained the model for 300 epochs. Figure 5.3 reveals that ViT performance can be enhanced by including the global workspace layer. AiT-Medium with fewer parameters also outperforms ViT-Base in the Pet dataset. Though AiT-Medium converges at a later training stage, it is a smaller model with fewer layers to compute compared to ViT-Base.

Prior Specialization Patches in one image can be attended sparsely by different priors. As shown in Section 5.3, a monolithic Transformer model needs to learn such specialization and relations without the inductive bias introduced by the global workspace layer. Notably, these priors learned to focus on independent spatial areas of an image to guide the attention. We visualized the activation maps for the specialized priors used in CIFAR-10 for AiT-Small (Figure 5.2). Each slot's activation maps highlight specific areas during the selection of relevant patches.

5.5.3 Ablation Study

We conducted a comprehensive ablation study to gain insights into the functionalities of the various components of AiT (Table 5.2). In AiT with reset memory, we initialized the explicit memory every epoch. The W/O Hopfield ablation replaces the Hopfield network with another multi-head attention (MHA) that shares the same architecture as the self attention in Figure 5.1.b. The rationale behind this ablation is grounded in the prior studies of Set Transformer and Perceiver models that relied on two MHA components cascaded in depth. For a fair comparison,

Table 5.1: Performance comparison in image classification tasks

Methods	CIFAR10	CIFAR100	Triangle	Average	Model Size
AiT-Base	**85.44**	**60.78**	99.59	**81.94**	91.0
AiT-Medium	84.59	60.58	99.57	81.58	45.9
AiT-Small	83.34	56.30	99.47	79.70	15.8
Coordination [30]	75.31	43.90	91.66	70.29	2.2
Coordination-DH	72.49	51.70	81.78	68.66	16.6
Coordination-D	74.50	40.69	86.28	67.16	2.2
Coordination-H	78.51	48.59	72.53	66.54	8.4
ViT-Base [9]	83.82	57.92	**99.63**	80.46	85.7
ViT-Small	79.53	53.19	99.47	77.40	14.9
Perceiver [28]	82.52	52.64	96.78	77.31	44.9
Set Transformer [34]	73.42	40.19	60.31	57.97	2.2
BRIMs [8]	60.10	31.75	-	45.93	4.4
Luna [35]	47.86	23.38	-	35.62	77.6

Table 5.2: Comparison based on an ablation study. The results indicate that combining all the components leads to the highest performance in all the tasks.

Models	CIFAR10	CIFAR100	Triangle	Average
AiT	**83.34**	**56.30**	**99.47**	**79.70**
Reset memory	81.94	55.96	99.46	79.12
W/O Hopfield	81.03	54.96	99.44	78.48
W/O memory (ViT)	79.53	53.19	99.47	77.40
Dense networks	77.78	53.14	99.46	76.79
W/O bottleneck	75.40	46.53	93.33	73.75
W/O SA	72.72	47.75	99.46	73.31
W/O FF	69.51	40.89	97.61	69.34

instead of simply removing the Hopfield network, we replaced it with the MHA. The added MHA takes the input state Ξ^t as the query, and the upscaled priors $f_{\text{LT}}(\gamma^{t+1})$ as the key and value, i.e., $\hat{\Xi}^t = \text{MHA}(\Xi^t, f_{\text{LT}}(\gamma^{t+1}))$.

Moreover, W/O memory evaluates performance when the global workspace layer is removed, the remaining components of which are equivalent to a simple Vision Transformer. W/O bottleneck shows performance using dense attention by removing the top-k bottleneck capacity constraint. W/O SA examines performance when the multi-head self attention component in Figure 5.1.b is excluded, and W/O FF evaluates performance when the feedforward component is removed. Lastly, the dense networks consist of repeated feedforward components with the other components removed in each AiT block. The analysis suggests that the complete model with all components can achieve the highest classification accuracy. The bottleneck appeared to play a significant role in improving performance, since its absence led to an evident decrease in accuracy. Making changes to other components such as Hopfield networks and the explicit memory, while not as impactful, still resulted in degraded accuracy. Despite the relatively good performance of dense networks, their performance in relational reasoning tasks is considerably inferior to that of the AiT model (Section 5.5.8). We demonstrate the without memory forward ablation in Table 5.7 and Table 5.8. The results show that AiT performs as well as or better than the without memory forward ablation.

Table 5.3: Different memory initialization approaches. The Gaussian distribution method was found to give the best results.

Memory initialization methods	Accuracy
Gaussian distribution	**83.34**
Positional embedding [7]	78.51
Uniform distribution [36]	81.92
Identity distribution [30]	78.56

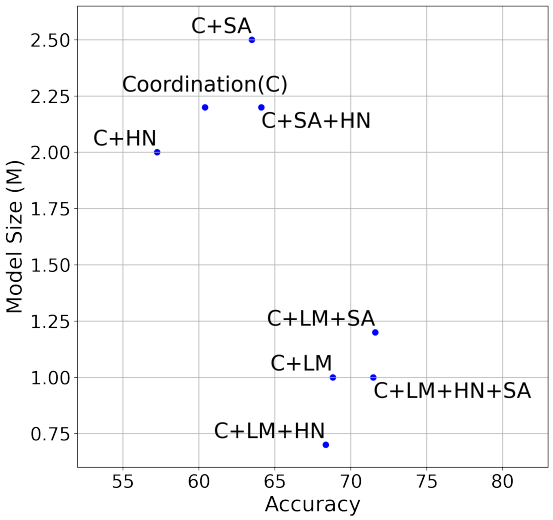
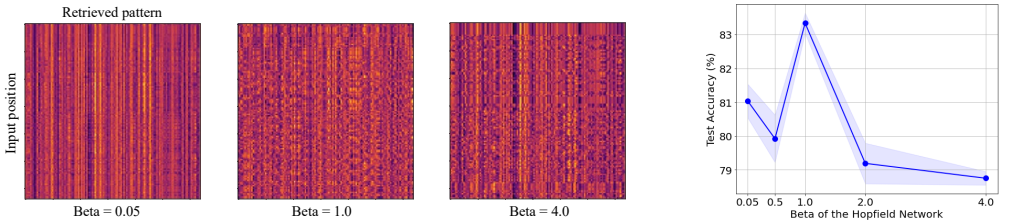


Figure 5.4: Model size vs. accuracy for configurations.

5.5.4 Comparison with the Coordination Method

We performed a detailed comparison with the coordination method in terms of test accuracy and model size. Figure 5.4 depicts the results for CIFAR-10 based on models with a single layer. Notably, using the low-rank memory (LM) that has a more diverse set of priors showed benefits in both improving the performance and decreasing the model size. For instance, the baseline coordination (C) method exhibited moderate accuracy of 60.41% with a model size of 2.2M. In contrast, consolidating the low-rank memory and the self-attention (C+LM+SA) exhibited the highest accuracy of 71.62%, while maintaining a relatively compact size of 1.2M. The Hopfield network (HN) maintained the model performance while reducing the model size by replacing the cross-attention with more efficient information retrieval. However, HN was effective only when either the LM or SA component was applied. We assume that retrieval with the Hopfield associative memory relies on a diverse set of priors, which is enabled by the enhanced bottleneck attention using the low-rank memory and the attention balance loss, and the learning through self-attention. By contrast, the previous coordination method had a limited number of priors, e.g. 8, and did not employ self-attention to correlate among input patches. Moreover, integrating all three components (C+LM+HN+SA) resulted in a competitive accuracy of 71.49% with a compact model size of 1.0M.



(a) Retrieved patterns from the Hopfield network for the first 128 patch positions in the input.

(b) Test accuracy when applying varying beta scores.

Figure 5.5: Comparison with varying inverse temperature scores. The inverse temperature beta influences the formation of metastable states that concurrently represent multiple patch representations. A smaller beta is more likely to generate such metastable states, while a larger beta leads to a stronger separation of different patterns. However, a larger beta can also lead to local minima, where input patterns are reconstructed to the same pattern within associative memory.

5.5.5 Memory Initialization

To initialize the explicit memory, we set each slot with values drawn from a specific distribution. We investigated several memory initialization methods (Table 5.3). The Gaussian distribution generates random values with a mean of 0 and a variance of 1. The sinusoidal positional embedding [7] uses sine and cosine functions to represent positions in a sequence. The uniform distribution [36] uses an upper bound $\frac{1}{\sqrt{M+D}}$, where M is the memory slot number and D is the slot size. The identity distribution [30] uses ones on the diagonal and zeros elsewhere. We found that the Gaussian distribution resulted in the best performance, possibly by preventing specific priors from dominating the learning process in early training stages.

5.5.6 Efficacy of Bottleneck Attention Balance Loss

The Bottleneck Attention Balance Loss facilitates selection of diverse input patches for each prior. To quantitatively measure the efficacy, we computed sparsity scores that represent the ratio of distinct patches in all selected patches. In Figure 5.10, we observe an apparent increase in the patch diversity.

5.5.7 Varying the Inverse Temperature in Hopfield Networks

We investigated the effect of the inverse temperature on information retrieval based on the Hopfield networks in Figure 5.5, which shows the reconstructed patches in the CIFAR-10 task for the AiT-Small model. We found that using an inverse temperature of 1.0 gave the best retrieval performance based on the Hopfield networks. The results suggest that the beta parameter requires tuning to reach optimal performance. We aim to study a mechanism to adjust the beta adaptively in the future, addressing this sensitivity and potentially further improving performance.

5.5.8 Relational Reasoning

In relational reasoning tasks, we aim to train a model to answer questions concerning the properties and relations of various objects based on a given image. A performant model can attend to specific regions of images for the question-answering task. We employed the Sort-of-CLEVR dataset [115] and compared performance

Table 5.4: Performance comparison in relational reasoning tasks.

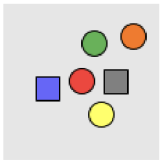
Methods	Relational	Non-relational	Average
Transformer based models			
AiT-Small	**76.82**	**99.85**	**88.34**
Coordination [30]	73.43	96.31	84.87
Set Transformer [34]	47.63	57.65	52.64
Non-Transformer based models			
CNN+RN [115]	81.07	98.82	89.95
CNN+MLP [115]	60.08	99.47	79.78
Dense-Base	46.93	57.71	52.32
Dense-Small	47.28	57.68	52.49

to both Transformer based models including Set Transformer and the coordination method, and other non-Transformer based models including CNN+MLP and CNN+Relation Networks (CNN+RN) [115]. The non-Transformer based models incorporated inductive biases into their architectures, such as convolutional layers focusing on different image areas. This often results in superior performance compared to the Transformer based methods that lack a built-in inductive bias. Moreover, two dense networks, the Dense-Small and Dense-Base, are included as additional non-Transformer based models. The Dense-Small (11.1M) and Dense-Base (62.7M) are derived from the AiT-Small and AiT-Base, respectively. Additionally, in relational reasoning tasks, a question was embedded with an embedding layer that consists of a learnable linear projection and layer normalization before and after the linear projection. The question embedding was then concatenated to image patch embeddings as the input of a model and the labels were a list of answer options with 10 classes.

Table 5.4 presents the results for relational and non-relational tasks, respectively. In the non-relational task, the question pertains to the attributes of a specific object, whereas in the relational task, the question focuses on the relations between different objects. A description of the dataset can be found in Appendix 5.7.1. The results demonstrate a substantial improvement in the performance of AiT when addressing the relational reasoning tasks. This indicates that the global workspace layer can learn spatial relations across different samples and time steps contributing to task performance. Dense networks generally do not perform well in the more complex relational reasoning tasks.

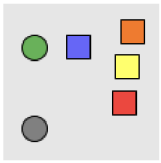
5.6 Conclusions

We proposed the Associative Transformer (AiT), an architecture inspired by Global Workspace Theory and associative memory. AiT leverages a diverse set of priors with the emerging specialization property to enable enhanced association among representations via the Hopfield network. The comprehensive experiments demonstrate AiT's efficacy compared to conventional models, including the coordination method. In the future, we aim to investigate multi-modal competition within the shared workspace, enabling tasks to benefit from the cross-modal learning of distinct perceptual inputs.



Non-relational question
Q: Is the yellow object on the top or on the bottom?
A: bottom
Relational question
Q: What is the color of the object that is closest to the blue object?
A: red

Non-relational question
Q: What is the shape of the red object?
A: circle
Relational question
Q: How many objects have the shape of the blue object?
A: 1



Non-relational question
Q: Is the blue object on the top or on the bottom?
A: top
Relational question
Q: What is the color of the object that is closest to the red object?
A: yellow

Figure 5.6: Examples from the Sort-of-CLEVR dataset [115].

5.7 Appendix

5.7.1 Datasets

In this section, we describe the datasets used in this work. (1) CIFAR-10 [113] is an image collection of 10 objects, covering 50k training samples and 10k test samples, labeled as airplane, automobile, and so on. The size of images is $32 \times 32 \times 3$. (2) CIFAR-100 [113] contains 100 object classes with 500 training images and 100 testing images per class. For both the CIFAR-10 and CIFAR-100 datasets, we performed random cropping with size $32 \times 32 \times 3$ and a padding size of 4. (3) Triangle dataset [30] includes 50k training images and 10k test images with size 64×64, each of which contains 3 randomly placed clusters of points. The task is to predict whether the three clusters form an equilateral triangle or not. (4) Oxford-IIIT Pet dataset [114] comprises 37 categories featuring diverse breeds of cats and dogs, with 200 images allocated for each class. We utilized random resized cropping with size $256 \times 256 \times 3$ and resized all images to size $224 \times 224 \times 3$. Additionally, we applied random horizontal flip and normalization to the CIFAR-10, CIFAR-100, and Oxford-IIIT Pet datasets. (5) Sort-of-CLEVR dataset [115] is a simplified version of the CLEVR dataset [118]. It includes 10k images with size $75 \times 75 \times 3$ and 20 different questions (10 relational and 10 non-relational questions) for each image. In each image, objects with randomly chosen shapes (square or circle) and randomly chosen colors (red, green, blue, orange, gray, yellow) are placed (Figure 5.6).

5.7.2 Related Work on the Global Workspace Theory

We discuss and summarize the existing sparse attention methods in relation to the main properties of Global Workspace Theory (Table 5.5). First, we examine whether an architecture involves operations of information writing and reading through a shared workspace. Secondly, we assess whether the latent representations (priors) in workspace memory are subsequently processed by self-attention. Thirdly, we inspect whether the latent representations have a lower rank compared to the input representations. Fourthly, we analyze whether information

Table 5.5: Comparison of attention architectures based on properties of the Global Workspace Theory

Methods	Operations		Self-Attention	Low-Rank Memory	Top-Down/Bottom-Up	Bottleneck
	Writing	Reading				
Vision Transformer [9]	-	-	-	-	BU	×
BlockBERT [111]	-	-	-	-	BU	✓
BRIMs[8]	×	×	×	×	TD	✓
Modern Hopfield [13]	×	✓	×	✓	BU	×
Perceiver [28]	✓	×	✓	✓	BU	×
Coordination [30]	✓	✓	×	×	BU	✓
Perceiver IO [29]	✓	✓	✓	✓	TD	×
Set Transformer [34]	✓	✓	×	×	BU	×
Luna [35]	✓	✓	×	×	BU	×
GMAT [33]	✓	✓	✓	×	BU	✓
Associative Transformer (Ours)	✓	✓	×	✓	BU	✓

retrieval from the workspace is driven by a bottom-up or a top-down signal. Lastly, we investigate whether the model incorporates a bottleneck with a limited capacity to regulate the information flow passing through the workspace.

5.7.3 Experimental Settings and Hyperparameters

Table 5.6 presents the hyperparameters used for the different tasks in this study. Unless otherwise noted, we employed the author-recommended settings and hyperparameters for the re-implementation of baseline models. The small variants of ViT and AiT have 2 attention layers, and the base variants of them have 12 attention layers instead. We used the same dimension of the hidden layer and the MLP layer for ViT and AiT. By default, we employed 8 attention heads and 32 memory slots for the bottleneck attention. To obtain the bottleneck size, we considered two main factors of the batch size and the patch size. For the CIFAR and Pet datasets, we used a bottleneck size of 512, which selected from a pool of 32.8k/25.1k patches. For the Triangle dataset, we used a bottleneck size of 64 from a pool of 2.0k patches. For the relational reasoning tasks, we used a bottleneck size of 256, which selected from a pool of 14.4k patches. Based on the bottleneck size and the patch pool size, we used 128 memory slots for the Pet dataset and 32 memory slots for the other datasets. Moreover, we trained the models on the Pet dataset for 300 epochs and on the other datasets for 100 epochs. In relational reasoning tasks, we trained all models for 100 epochs with a batch size of 64.

5.7.4 Analysis of Operating Modes of Attention Heads

To understand the operating modes of attention heads in a vision Transformer (ViT) model, we measured the interaction sparsity of the pair-wise self-attention. The goal is to investigate whether sparse attention is required to learn meaningful representations during the model training. If the ViT model inherently learns such sparse interactions among patches, we can induce an inductive bias to foster the sparse selection of patches through a communication bottleneck. We trained a ViT-Base variant model for 100 epochs from scratch for the CIFAR-10 task. Then, for each attention head, we obtained a violin plot to represent the distribution of attention sparsity for different patches. The attention sparsity for a specific patch's interactions with other patches is computed as follows

$$\arg\min_{s} \sum_{j=1}^{s} A^{i,j} \geq 0.9, \tag{5.9}$$

Table 5.6: Hyperparameters for Associative Transformer

Parameter	Value
Common parameters	
Optimizer	AdamW
Weight decay	0.01
Learning rate	1×10^{-4}
Number of self-attention heads	12
Number of attention layers	2 (Small)/ 12 (Base)
Size of hidden layer	768
Size of MLP	3072
Size of memory slot	32
Number of bottleneck attention heads	8
Beta	1.0
Epochs	100 (300 for Oxford Pet)
CIFAR	
Patch size	4
Batch size	512
Number of memory slots	32
Bottleneck size	512
Triangle	
Patch size	32
Batch size	512
Number of memory slots	32
Bottleneck size	64
Oxford Pet	
Patch size	16
Batch size	128
Number of memory slots	128
Bottleneck size	512
Relational reasoning	
Patch size	5
Batch size	64
Number of memory slots	32
Bottleneck size	256

where $A^{i,j}$ is the attention score allocated to the jth patch by the ith patch. The attention sparsity score is measured by the minimal number of required patches whose attention scores add up to 0.90. For instance, there are 65 patches for the CIFAR-10 task with patch size 4, thus there are 65 interactions for each patch to all the patches including itself. Then, an attention head has a higher sparsity if the median of the required patches s that satisfies Equation 5.9 across different patches is smaller. Moreover, for each head, we showed the distribution of the attention sparsity scores for different patches in the violin plots (Figure 5.7), where the scores from different input samples were averaged. The number in the center of each panel gives the median $\bar{s}$ of the distribution. The heads in each layer are sorted according to $\bar{s}$. Note that training the model for a longer duration can result in even better convergence and higher attention sparsity. We also refer to a concurrent investigation on the attention sparsity of the Bidirectional Encoder Representations from Transformers (BERT) model for natural language processing (NLP) tasks [13]. Our findings about the various operating modes of attention heads in ViT are in line with the findings in NLP tasks.

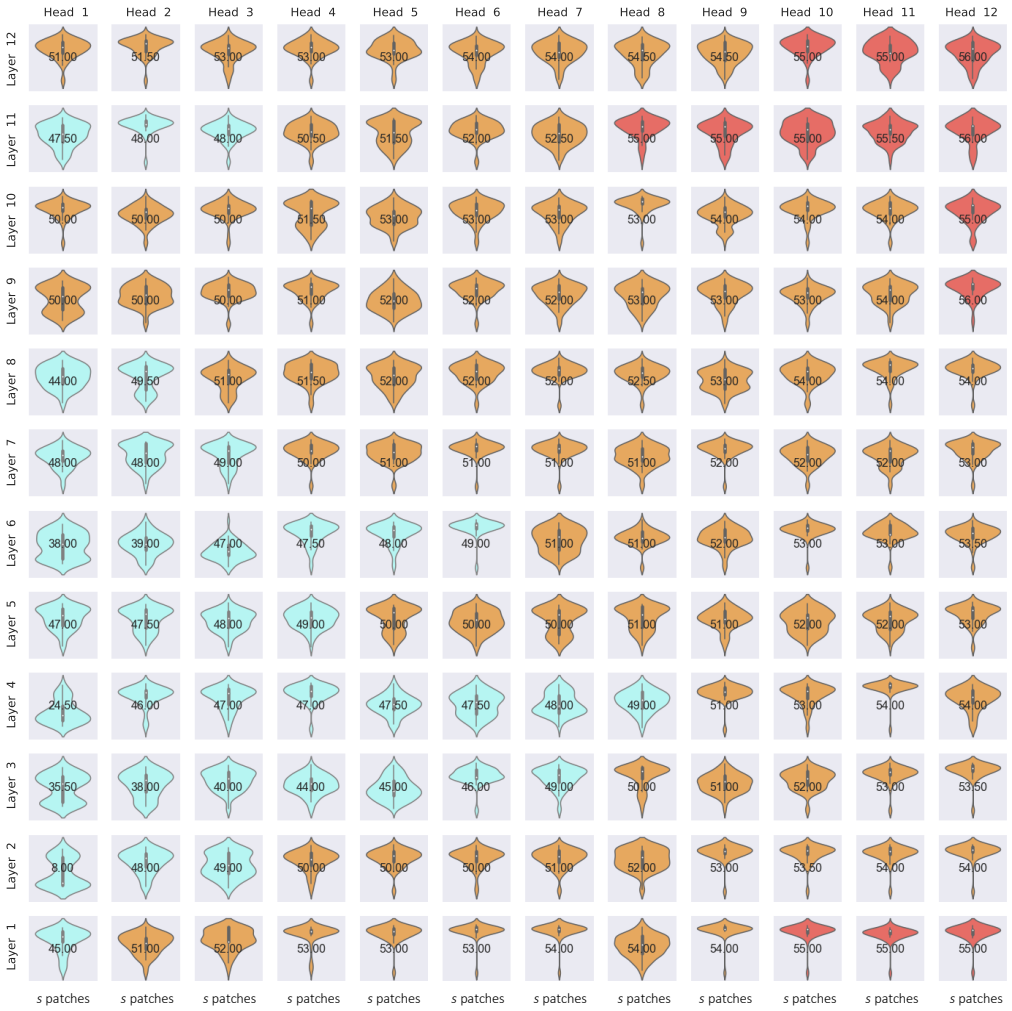
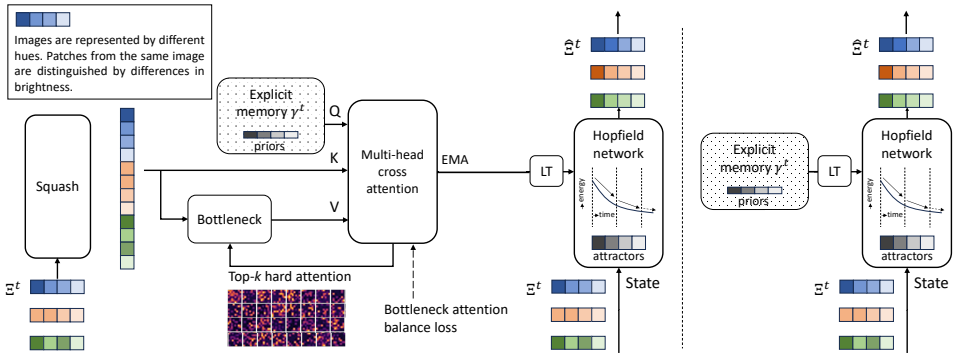


Figure 5.7: Analysis of operating modes of attention heads in the ViT-Base model. We recognize three different groups of attention heads based on their sparsity scores. Group (I) in light blue: High sparsity heads abundant in the middle layers 3-6. The vast majority of these heads only used 50% or fewer interactions. Group (II) in orange: Middle sparsity heads predominant in layers 2 and 7-10. Less than 80% of the interactions were activated. Group (III) in red: Low sparsity heads observed in high layers 11-12 and the first layer, where the most patches were attended to. The global workspace layer will provide the inductive bias to attend to the essential patches more effectively.

5.7.5 Discussion on the Test Time Scheme

We demonstrate the schemes for Associative Transformer during test time and the 'without memory forward' ablation in Figure 5.8. Moreover, we evaluated the model performance using the two different methods for both image classification and relational reasoning tasks (Table 5.7 and Table 5.8). The results show that AiT performs as well as or better than the 'without memory forward' ablation.



(a) Frozen memory during test time (AiT).

(b) Ablation without memory forward during test time.

Figure 5.8: The scheme of the Associative Transformer (AiT) during test time. (1) In AiT, although the explicit memory is frozen (depicted by filled dots), the memory forward process is enabled. However, the computed output of the multi-head cross attention will not be used to update the memory, which is different from training time. (2) In the 'without memory forward' ablation, the entire memory forward process is disabled, leveraging only Hopfield networks for retrieval.

Table 5.7: Test time ablation in image classification tasks

Methods	CIFAR10	CIFAR100	Triangle	Pet
AiT-Medium	84.59	60.58	99.57	30.05
AiT-Medium (without memory forward)	84.50	60.56	99.57	28.68

Table 5.8: Test time ablation in the relational reasoning task

Methods	Relational
AiT-Small	76.82
AiT-Small (without memory forward)	75.17

5.7.6 Efficacy of the Bottleneck Attention Balance Loss

The Bottleneck Attention Balance Loss facilitates the learning of priors that can attend to diverse sets of patches. We demonstrate the efficacy by visualizing the bottleneck attention scores computed using the learned priors (Figure 5.9) and the corresponding selected patches (Figure 5.10). We used as a metric for sparsity the ratio of distinct patches in all the selected patches by the bottleneck attention. With the progress of training, we can obtain a more diverse selection of patches.

5.7.7 Hopfield Networks Energy

In traditional Hopfield networks, it is possible to store N samples and retrieve them with partially observed or noisy patterns by updating model weights. During retrieval, these partially observed or noisy patterns converge to one of these attractors, minimizing the Hopfield energy. Unlike traditional Hopfield attractors

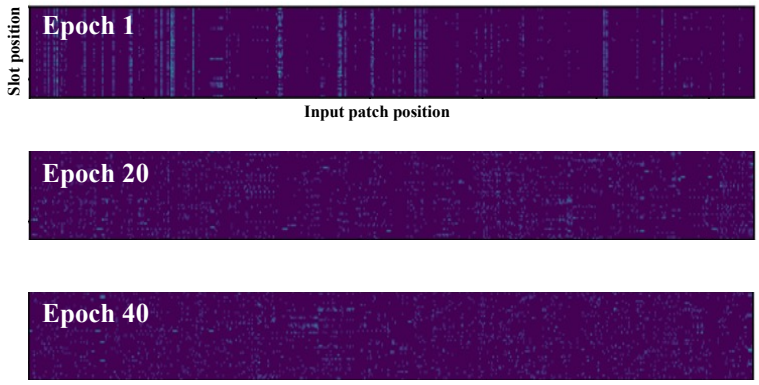
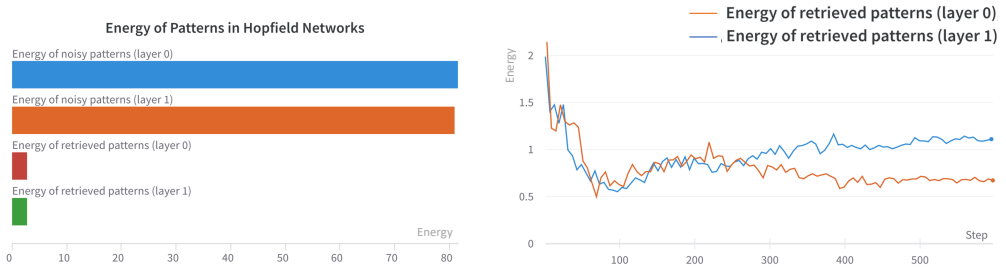


Figure 5.9: Bottleneck attention balance loss facilitates the selection of diverse patches from different input positions.

Figure 5.10: Examples of the selected patches by the bottleneck attention in CIFAR-10. As training progresses, we can obtain a more diverse selection of patches.

that incorporate the implicit memory within its model parameters, AiT decouples the memory from the Hopfield network by introducing the learnable explicit memory. This memory serves the functions of both priors in the bottleneck attention and attractors in Hopfield networks. Consequently, the Hopfield network does not need to store different inner states every batch time, instead, we can reuse the learned memory bank from the bottleneck attention to update and maintain a set of attractors with the trainable linear transformation. The proposed architecture is an attractor network in the sense that, in every batch, a pattern converges to one of these attractors derived from the priors stored in the explicit memory bank.

Moreover, the information retrieval is based on a continuous Hopfield network, where an input state converges to a fixed attractor point within the associative memory of the Hopfield network. Usually, any input state that enters an attractor's basin of attraction will converge to that attractor. The convergence results in a decreased state energy with respect to the stored attractors in the memory. All patches reach their minimum at the same time point and the global energy in Equation 5.7 is guaranteed to decrease. To quantitatively measure the amount of energy reduction during the information retrieval process in the Hopfield network, we computed an input state's energy before and after it was reconstructed based on Equation 5.7. A successful retrieval results in substantial reduction in the state energy (Figure 5.11).



(a) Global energy over the implicit iteration time step

(b) Global energy over the explicit batch time steps

Figure 5.11: Energy of patch representations in the CIFAR-10 task for AiT-Small. The Hopfield network operates by iteratively decreasing the energy of an input state with respect to the attractors stored in its memory. This reduction in energy enables the retrieval of a representation that closely aligns with learned attractors, effectively leveraging knowledge within the associative memory. (a) The global energy is guaranteed to decrease over the implicit iteration time step for every retrieval. (b) The global energy over the explicit batch time steps generally decreases, especially during the early stages of training, since the computed energy is the relative energy to the memory bank that is updated every batch time step.

Chapter 6

Model Poisoning in Federated Learning

This chapter consolidates my work on semi-targeted model poisoning in federated learning [119, 120].

Existing model poisoning attacks on federated learning (FL) assume that an adversary has access to the full data distribution. In reality, an adversary usually has limited prior knowledge about clients' data. A poorly chosen target class renders an attack less effective. This work considers a semi-targeted situation where the source class is predetermined but the target class is not. The goal is to cause the misclassification of the global classifier on data from the source class. Approaches such as label flipping have been used to inject malicious parameters into FL. Nevertheless, it has shown that their performances are usually class-sensitive, varying with different target classes. Typically, an attack becomes less effective when shifting to a different target class. To overcome this challenge, we propose the Attacking Distance-aware Attack (ADA) that enhances model poisoning in FL by finding the optimized target class in the feature space. ADA deduces pair-wise class attacking distances using a Fast LAyer gradient MEthod (FLAME). Extensive evaluations were performed on five benchmark image classification tasks and three model architectures using varying attacking frequencies. Furthermore, ADA's robustness to conventional defenses of Byzantine-robust aggregation and differential privacy was validated. The results showed that ADA succeeded in increasing attack performance to 2.8 times in the most challenging case with an attacking frequency of 0.01 and bypassed existing defenses, where differential privacy that was the most effective defense still could not reduce the attack performance to below 50%.

Model poisoning on federated learning (FL) causes client models to be compromised by malicious model parameter sharing. Though FL extends the attacking surface of the attacker involving lots of clients, the model aggregation that combines different clients' model parameters, can greatly reduce poisoning attack's effect. Different from previous studies that adopt an arbitrary target class to mount an attack, this work proposes a novel semi-targeted model poisoning attack that adaptively computes the optimized attacking target depending on input samples. Such an attack could immensely enhance model poisoning's efficacy in FL, improving its robustness against model aggregation. The empirical result showed that the proposed method achieved great performance even with a low attacking frequency, generalizing across different distribution spaces and model architectures. In addition, existing defenses in FL were found to be ineffective in alleviating the semi-targeted attack.

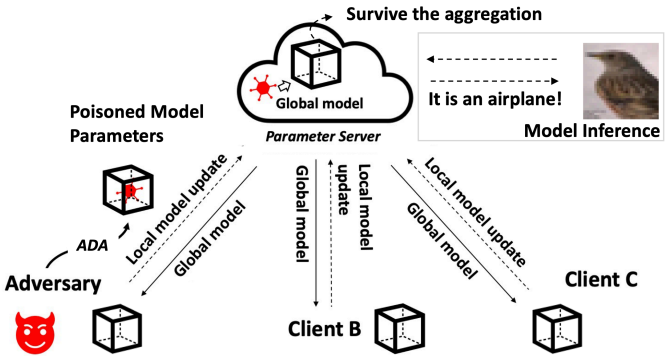


Figure 6.1: Model poisoning in FL. The adversary mounts the attack by uploading poisoned model parameters.

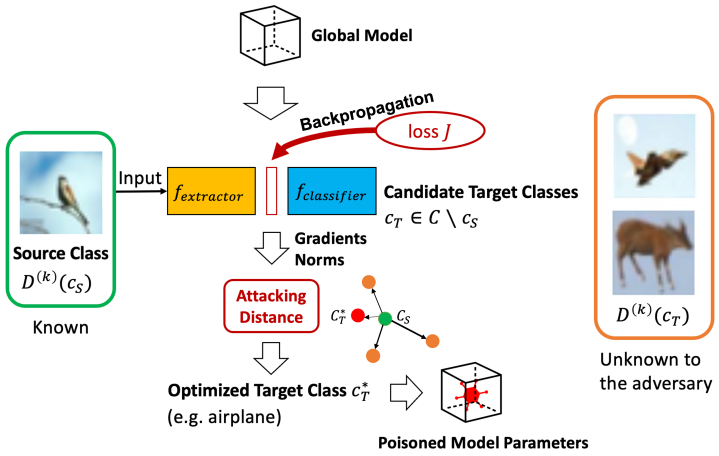


Figure 6.2: ADA, a semi-targeted model poisoning attack, compromises the global model in the black-box setting, by carefully choosing the target class based on the backward error analysis.

6.1 Target Optimization in Poisoning Attack

Data privacy is a growing concern, attracting attention from various sectors. The increasing public awareness of legal restrictions such as the General Data Protection Regulation (GDPR) [84] has made the traditional centralized processing of sensitive data increasingly challenging. As a result, decentralized solutions such as federated learning (FL) [15, 121] have been adopted to improve performance by sharing and aggregating model parameters without disclosing clients' training data.

FL has been widely used in various fields including medical diagnosis, financial data analysis, and cybersecurity. However, it has been shown to be vulnerable to adversarial attacks [122, 123, 124, 125, 126]. Notably, a compromised client might inject malicious model parameters into the FL system, causing malfunction and influencing other clients in the system (Figure 6.1). Furthermore, these attacks in FL are typically either untargeted or targeted [14]. The aim of an untargeted attack is to degrade the performance of a client model in general, while a targeted attack aims to cause a client model to misclassify samples of a specific class into the attacker's desired class.

This study investigates a novel form of model poisoning attack in FL where the attacker's goal is to avoid being recognized as a specific class. This type of attack can occur in various real-world scenarios. For example, an attacker

sending unauthorized advertising emails may aim to have the emails recognized as belonging to a benign class other than spam. In a facial recognition system, an individual on a blacklist may aim to be recognized as someone else not on the blacklist. It is important to note that these attackers are not motivated to be recognized as a specific class (as is the goal of targeted attacks), but rather to be unrecognized as a specific class.

This type of attack is referred to as a semi-targeted attack. In a semi-targeted attack, the attacker is assigned a specific class (the source class) and aims to poison the classifier so that samples with the source class are recognized as a class other than the source class. Unlike targeted attacks, the attacker is free to choose the target class to maximize attack performance in a semi-targeted attack. Due to this freedom of attack, the risk of the semi-targeted attack outweighs the risk of the targeted attack.

The success of the attack can vary depending on the assigned source class, meaning that the attack is typically class-sensitive in terms of its generality, with performance varying depending on the class considered [127]. The challenge in a semi-targeted attack is determining the target class that is optimized for the assigned source class in order to achieve the best attack performance.

To address this challenge, we propose two approaches, the Attacking Distance-aware Attack (ADA) and the Fast LAyer gradient MEthod (FLAME), to find the optimized target class for attacking FL in both full knowledge and partial knowledge settings, respectively. The goal is to investigate to what extent the attack performance can be increased in the semi-targeted setting using different approaches for choosing the target class.

In summary, our contributions are as follows.

- This work introduces the semi-targeted attack, a new type of model poisoning on federated learning.
- We propose a semi-targeted model positioning attack called ADA to compromise a client classifier by carefully choosing the target class (Section 6.4.2). A backward error analysis of the global model shows that the update gradient's scale when choosing a certain target class for producing a poisoned model, immensely influences the expected attack performance (Figure 6.2). This study provides a detailed demonstration of how FLAME enhances a poisoning attack's performance in FL under the data confidentiality constraint (Section 6.4.3).
- To understand the risk of such attacks, an extensive study is performed by varying the factor of attacking frequency against the metrics of adversary task accuracy (ATA) and main task accuracy (MTA). The empirical results show that ADA is effective in both full knowledge (white-box) and partial knowledge (black-box) settings. ADA increases the attack performance to 2.8 times when the attacking frequency is as low as 0.01 (Section 6.5.3).
- An analysis of ADA's robustness to conventional defense strategies in FL shows that ADA can bypass these defenses retaining competitive attack performance (Section 6.5.6).

The remainder of this paper is structured as follows. Section 2 reviews the most recent work on poisoning attacks and their defense in FL. Section 3 presents essential definitions and assumptions. Section 4 demonstrates the technical underpinnings of the proposed method. Section 5 presents extensive empirical evaluations. Section 6 concludes the paper and gives out future directions.

6.2 Related Work

Poisoning Attack Federated Learning (FL) is susceptible to various types of adversarial attacks, primarily poisoning attacks [127, 128, 129, 123, 130, 131, 132, 133] and information stealing attacks [134, 135, 136, 82, 83]. The focus of these attacks is to misclassify client samples by injecting poisoned local models, rather than reconstructing samples from shared models. Poisoning attacks on FL are usually mounted via manipulating either training data [127, 128, 129, 123, 137] or local training process [123, 130, 131, 132] of an edge client. This section briefly reviews related research efforts of poisoning attacks on FL.

In targeted poisoning attacks on FL, the goal of the adversary is to produce a poisoned local model update. This update is designed in a way that after model aggregation, specific inputs will induce misclassification of the global model [138, 139]. In particular, label flipping [122] is one type of poisoning attacks, where a set of data labels are randomly flipped to a different class to train a malicious model. For example, a semantic backdoor flips the labels of images containing specific natural features to cause misclassification when these features are present as triggers [123]. Moreover, a malicious model update generated through label flipping typically results in a larger norm of model weights than a benign update [14].

To avoid easy detection by norm-based defense algorithms in FL, Bagdasaryan et al. [127] proposed a train-and-scale technique that scales the norm of a malicious update to the detecting algorithm's bound. Nevertheless, this strategy could also result in degraded attack performance due to diminished poisoned model weights. Additionally, the model replacement attack [127] aims to replace the global model entirely with a model controlled by the adversary. Notably, a converged global model often results in small benign local updates, which creates a vulnerability where an adversary-controlled client could potentially upload a maliciously crafted update. This malicious update then replaces the global model after aggregation.

Although the aforementioned targeted attacks aim to compromise the FL system, their attack performance typically depends on the assigned source class. There is currently no research on semi-targeted attacks that have a fixed source class and an adjustable target class. A semi-targeted attack addresses the problem of a poorly chosen target class degrading the poisoning attack's performance after model aggregation in FL and making it easier to detect poisoned model parameters. To the best of our knowledge, this is the first study to examine semi-targeted poisoning attacks with a focus on generality in FL. In this regard, Shafahi et al.[140] presented a feature-collision method to generate similar-looking instances based on the source class such that their hidden features were close to the target class in a centralized setting. To mount the attack, the adversary needed prior knowledge about both the data distributions of the source class and the target class, which is different from this study with the federated setting where neither the data modification nor the prior knowledge about the target class's data distribution is required. This work closes this gap and investigates the behaviors of model poisoning attacks in the partial knowledge setting. Furthermore, this work considered poisoning attacks in FL with a more realistic setting regarding the attacking frequency, where the adversary had only a small possibility of participating in the training every round.

Defense Strategies Existing research on defenses for federated learning against model poisoning attacks can generally be divided into two categories:

anomaly detection and robust aggregation.

Anomaly detection aims to identify malicious local model updates by comparing the similarity of client updates and identifying those that deviate greatly from others [93, 141, 142, 143, 144]. One of the most commonly used methods is norm difference clipping [94]. Poisoning attacks often result in larger norms compared to benign updates from honest clients, and norm difference clipping discards updates with norms above a certain threshold, thus separating benign and malicious updates.

Another line of research seeks to improve the resilience of aggregation in FL against poisoned parameters, by carefully selecting local model updates for aggregation [145, 146, 147] or adding noise to model parameters to counteract the effect of a malicious update [123, 148, 92]. For example, Differential Privacy (DP) limits the influence of a malicious update by adding a small fraction of Gaussian noise to the parameters of local updates.

In Section 6.5.6, the experimental evaluation demonstrates that the proposed attack method can bypass existing defense methods, including not only the anomaly detection-based defense of norm difference clipping but also the most recent robust aggregation-based defenses of Krum[145], Trimmed Mean[146], and Differential Privacy[149].

6.3 Preliminaries

In this section, the classification task in federated learning (FL) is formulated, and several conventional poisoning attacks on FL related to this work are discussed. The notation used in this paper is summarized in Table 6.1.

6.3.1 Classification Tasks in FL

Supervised learning with C categories in a given dataset D is a fundamental classification task in machine learning. Let $x \in \mathbb{R}^V$ be a sample and $y \in \{1, 2, ..., C\} = Y$ a label. D consists of a collection of N samples as $D = \{(x_i, y_i)\}_{i=1}^N$. Suppose that f denotes a neural network classifier taking an input x_i and outputting a C-dimensional probability vector where the jth element of the output vector represents the probability that x_i is recognized as class j. Given $f(x)$, the prediction is given by $\hat{y} = \arg\max_j f(x)_j$ where $f(x)_j$ denotes the jth element of $f(x)$. The training of the neural network is attained by minimizing the following loss function concerning the model parameter θ

$$J(\theta, D) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i; \theta)), \tag{6.1}$$

where ℓ denotes the cross entropy loss function.

Federated Learning (FL) is a privacy-preserving framework that enables the creation of a global model trained on decentralized data without revealing the individual training samples. In a FL framework with K clients, the kth client has its own dataset $D^{(k)} := \{(x_i, y_i)\}_{i=1}^{N^{(k)}}$ where $N^{(k)}$ is the sample size of dataset $D^{(k)}$. Here, $\cup_{k=1}^K D^{(k)} = D$ and $N = \sum_{k=1}^K N^{(k)}$. Each client cannot share its data with others mainly due to data confidentiality, making FL an attractive approach for collaborative learning.

The FL process involves several steps. Firstly, the parameter server (PS) initializes a global model G_0, which is then sent to all clients. Secondly, each client updates the model using their local data $D^{(k)}$ and sends the update $L_{t+1}^{(k)} - G_t$ to

Table 6.1: Notation for the Attacking Distance-aware Attack

Federated Learning (for supervised learning tasks)	
D	entire data set
$D^{(k)}$	client k's local data
N	number of entire data
$N^{(k)}$	number of client k's local data
$\mathbf{x} \in \mathbb{R}^D$	sample
$y \in Y$	label
C	number of data categories
$\ell(\cdot)$	cross entropy loss function
K	number of total clients
K_{select}	number of selected clients at each round
G_t	global model at round t
$L_t^{(k)}$	client k's local model at round t
η	learning rate
E	number of local epochs
b	set of batch
Attacking Distance-aware Attack	
c_S	source class
c_T	target class
D_{val}	validation dataset
$D_{\text{val}}(c_S)$	samples in D_{val} with label c_S
$D_{\text{val}}^c(c_S)$	samples in D_{val} with label other than c_S
$D(c_S)$	samples in D with label c_S
α	injection rate
D^{adv}	adversary's backdoor data
L_t^{adv}	adversary's local model at round t
Q	norm threshold to drop malicious updates
Ω	norm scaling factor
$\{Y \setminus c_S\}$	candidate target classes
c^*	best target class of the attack
ϕ_t	feature extractor of the global model G_t
μ_c	mean of feature vectors in class c
c_T^*	optimized target class of the attack
ϵ	attacking frequency factor
K_{adv}	number of total malicious clients in K
γ	number of tolerable attackers in Krum
δ	number of the removed items in Trimmed Mean
σ	standard deviation of Gaussian noise in differential privacy

the PS. Finally, the PS aggregates all local updates, updates the global model, and sends it back to all clients. It's worth noting that in FL, the local models of clients and the global model usually share the same architecture.

In addition, to reduce the waiting time for all clients to complete their local model training, the PS randomly selects a subset of K_{select} clients each round to update the global model based on their local updates. FedAvg [16] is a widely used FL algorithm that utilizes averaging of all local updates to update the global model. The specific details of the FedAvg algorithm are presented in Algorithm 3.

Algorithm 3 FedAvg

initialization of G_0 at the server side
for each round $t = 0, 1, 2, ...$ **do**
    PS randomly selects a subset of K_{select} clients from all K clients
    PS sends the current global model G_t to K_{select}
    for each client $k = 1, 2, ..., K_{\text{select}}$ **do**
        $L_{t+1}^{(k)} = \text{FedLearnLocal}(G_t, L_t^{(k)}, D^{(k)})$
    end for
    $G_{t+1} = G_t + \sum_{k \in K_{\text{select}}} \frac{N^{(k)}}{\sum_{k \in K_{\text{select}}} N^{(k)}} (L_{t+1}^{(k)} - G_t)$
end for

function FedLearnLocal($G_t, L_t^{(k)}, D^{(k)}$)
$L_t^{(k)} \leftarrow G_t$
for each epoch $e \in E$ **do**
    for each batch $b \subset D^{(k)}$ **do**
        $L_{t+1}^{(k)} \leftarrow L_t^{(k)} - \eta \cdot \nabla J(L_t^{(k)}, b)$
    end for
end for
return $L_{t+1}^{(k)}$

Poisoning Attacks on Federated Learning Poisoning attacks have been extensively studied in the context of centralized learning. Notably, in a supervised classification task with C categories, the goal of an attacker is to either degrade the performance of the classifier in general (untargeted) or cause it to misclassify a specific class (targeted). To achieve these goals, the adversary manipulates either the training data by adding malicious samples (data poisoning) or the model by injecting malicious parameters (model poisoning) that are carefully crafted to cause the model to behave unexpectedly during inference.

In targeted poisoning attacks, the class of samples used for the attack is known as the source class c_S, while the final class to which the sample is modified is called the target class c_T. Specifically, the goal of the attacker is to manipulate the classifier f such that given a sample (x, y), the model makes incorrect predictions for samples belonging to the source class c_S

$$\arg\max_k f(x)_k = \begin{cases} c_T \text{ if } y = c_S, \\ y \text{ Otherwise .} \end{cases} \tag{6.2}$$

Let D_{val} be a validation dataset, $D_{\text{val}}(c_S)$ be the set of samples in D_{val} with label c_S, and $D_{\text{val}}^c(c_S) = D_{\text{val}} \setminus D_{\text{val}}(c_S)$. To measure the performance, the main task accuracy (MTA) is defined as the validation accuracy of the classifier on samples that are not from the source class c_S. In particular, MTA of classifier f when it is poisoned with a source label c_S is defined as the accuracy on the validation set $D_{\text{val}}^c(c_S)$:

$$\text{MTA}_f(c_S) = \frac{\sum_{(x,y) \in D_{\text{val}}^c(c_S)} \mathbb{1}\{\arg\max_k f(x; G)_k = y\}}{|D_{\text{val}}^c(c_S)|}. \tag{6.3}$$

The target-specified adversary task accuracy (ts-ATA) evaluates the success rate of an attack. It is computed as the percentage of samples from the source class that are misclassified as the desired target class by the poisoned model. Specifically, ts-ATA of classifier f poisoned with source label c_S is determined by

the validation accuracy for $D_{\text{val}}(c_S)$:

$$\text{ts-ATA}(c_S, c_T) = \frac{\sum_{x \in D_{\text{val}}(c_S)} \mathbb{1}\{\arg\max_k f(x; G)_k = c_T\}}{|D_{\text{val}}(c_S)|}. \tag{6.4}$$

We introduce two building blocks, label flipping [122] and gradient scale adjustment by Train-and-scale technique [127], needed to introduce the proposed method.

6.3.2 Label Flipping Attack

In a label flipping poisoning attack, the attacker relabels the samples from the source class with the target class label. For any sample $(x, y) \in D(c_S)$, the label is replaced with c_T to obtain the modified sample (x, c_T). By training the classifier with these poisoned samples, the poisoning attack as defined in Eq. 6.2 can be achieved. Typically, attackers employ a combination of poisoned samples and legitimate samples to train the classifier. In our experiments, label flipping is applied to all samples in $D(c_S)$, and a subset of samples from classes other than the source class are randomly selected to achieve a desired injection rate α, where α represents the percentage of poisoned samples in the adversarial training dataset D^{adv}. This approach aims to mount a poisoning attack without sacrificing too much on the accuracy of the non-target classes.

In the FL setting, if a compromised client is selected by the PS for local model training, it will download the latest global model and replace its local model, then train a local model L^{adv} with the poisoned samples, and corrupt the global model by repeatedly submitting malicious updates to the PS. In addition, multiple compromised clients could exist to mount poisoning attacks against the global model.

6.3.3 Gradient Scale Adjustment

The "Train and Scale" approach is a common strategy employed by attackers in federated learning (FL), to bypass detection methods based on comparing the norm of model updates. Such a detection method compares the norm of the updates with a certain threshold Q that reflects the difference of updates. Updates whose norm exceeds this threshold are dropped, a technique known as norm difference clipping (NDC) [150], i.e., $\|L_t^{(k)} - G_t\| > Q$ where $\|\cdot\|$ is a prescribed norm. However, attackers can easily bypass this defense strategy by using the "Train and Scale" approach, which adjusts the magnitude of the updates to evade detection. By scaling down the weights, the norm of the update can be reduced to below the detection threshold Q, allowing the malicious update to pass through undetected. The scaling factor is carefully chosen to maintain the effectiveness of the malicious update while reducing its norm.

To achieve this goal, the attacker modifies the scale of the model update $\|L_{t+1}^{adv} - G_t\|$ so that it is upper bounded by Q. Let a scaling factor Ω defined by

$$\Omega = \frac{Q}{\|L_{t+1}^{adv} - G_t\|}. \tag{6.5}$$

Then, the malicious client submits $\Omega\|L_{t+1}^{adv} - G_t\|$ as model update instead of $\|L_{t+1}^{adv} - G_t\|$.

One limitation of this approach is that Q is usually unknown to the adversary. Nevertheless, the adversary can approximately estimate the bound Q using

the following strategy [127]. It is expected that the threshold Q set by the PS is designed such that legitimate updates are not rejected with high probability. Therefore, the adversary can initially employ several compromised clients to perform legitimate training with the latest global model shared by the PS. The average norm of the collected legitimate updates can be employed as a lower bound of Q.

6.4 Methodology

In this section, a new type of semi-targeted model poisoning on federated learning called the Attacking Distance-aware Attack (ADA) is introduced. Then, the Fast LAyer gradient MEthod (FLAME) is demonstrated to mount the semi-targeted attack in a partial knowledge setting.

Motivation In multi-class classification tasks, an adversary aims to compromise the system such that instances from a specific class c_S will be misclassified. Compared to a targeted attack with a fixed target class c_T for the given source class c_S, a semi-targeted attack that does not have a specified target class could provide more flexibility, increasing the risk to the FL system. For example, in a real world scenario, a self-driving car that recognizes a stop sign could be compromised such that the prediction of the stop sign will be wrong. The incorrectly predicted class can be the speed limit sign, the billboard, and so on. Similarly, a spam filter that aims to identify the category of an email can be poisoned such that a certain type of spam will bypass the filter, with the target class being sports, politics, and so forth.

The ADA is a semi-targeted attack that aims to manipulate the global model towards a specific behavior, while a Byzantine attack [151] is an untargeted attack that aims to invalidate the global model without a specific target. The advantage of ADA over a Byzantine attack is that it allows for more subtle attacks that are less likely to be detected by traditional defense mechanisms. ADA is able to bypass robust aggregation defenses and achieve high attack performance with limited prior knowledge of the source class. The benefits of requiring only a single class's data become more apparent in an extremely large label space such as ImageNet (1000 classes), as demonstrated in later sections.

In a semi-targeted attack, the effectiveness of the attack can vary depending on the target class c_T considered. The attack's robustness to different target classes is not guaranteed, and using a less effective target class can result in longer convergence time required to achieve the same level of attack performance. Longer convergence time allows a defense method in the PS to more easily discover the attack. Additionally, since only a small subset of clients are selected in each round, a less effective malicious update might be overwritten by the outnumbering legitimate updates submitted by benign clients.

Intuitively, if the adversary can choose the most effective target class, a poisoning attack will be greatly enhanced. To reveal the risk of the semi-targeted attack, we propose the Attacking Distance-aware Attack (ADA), an enhanced model poisoning attack on FL. Notably, ADA measures the distances in the latent feature space between different classes of a classifier and finds the optimized target class c_T given a source class c_S to mount the attack. Two different settings of ADA were studied, based on the prior knowledge of the adversary about a client's data distribution in FL, i.e., attacking with full knowledge and attacking with partial knowledge.

6.4.1 Semi-Targeted Attack

The semi-targeted attack in FL refers to a model poisoning attack with a fixed source class c_S and various possible target classes $\{Y \setminus c_S\}$. The goal of the semi-targeted poisoning attacks is to corrupt f so that

$$\arg\max_k f(x)_k = \begin{cases} c^* \text{ if } y = c_S, \\ y \text{ o.w. }. \end{cases} \tag{6.6}$$

where

$$c^* = \text{argmax}_c \text{ts-ATA}(c_S, c). \tag{6.7}$$

In particular, for any given c_S, ts-ATA(c_S, c_T) depends on the choose of c_T. In the semi-targeted attack, the attacker could choose any arbitrary c_T thus the attack performance could be increased. In this regard, the max target-non-specified adversary task accuracy (max-ATA) of classifier f poisoned with source label c_S is defined as the validation accuracy for $D(c_S)$:

$$\text{max-ATA}(c_S) = \max_{c \in C} \text{ts-ATA}(c_S, c). \tag{6.8}$$

Note that,

$$\text{max-ATA}(c_S) \geq \text{ts-ATA}(c_S, c_T)$$

holds for any c_S and c_T, which means that the semi-targeted poisoning attack is always more powerful than targeted poisoning due to the more relaxed constraint.

In addition, it is assumed that the adversary specifies only one source class c_S for simplicity, but the semi-targeted attack can be extended to scenarios where there are multiple source classes. In such cases, the optimized target classes would be computed separately for each source class specified by the adversary.

6.4.2 Attacking with Full Knowledge

In the full knowledge setting, the adversary has complete knowledge of the client data distribution and is able to access samples drawn from the underlying training sample distribution in an independent and identically distributed (IID) manner.

To choose the class that is supposed to give larger ts-ATA, the adversary leverages the attacking distance (AD) defined as follows. Let ϕ_t be the feature extractor of the shared global model G_t. Then the adversary extracts feature vectors $\phi_t(x)$ of local training samples $x \in D^{adv}$.

Let μ_c be the mean of feature vectors in class c where

$$\mu_c = \frac{1}{|D^{adv}(c)|} \sum_{x \in D^{adv}(c)} \phi_t(x). \tag{6.9}$$

Then, the attacking distance between two different classes c and c' is defined by

$$\text{AD}(c, c') = \|\mu_c - \mu_{c'}\|_2, \tag{6.10}$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm.

The distribution visualization of the extracted feature vectors ϕ using the principal component analysis (PCA) for measuring the AD is shown in Figure 6.3.

The attack strategy in this setting is to find a target class c_T in the latent feature space that is close to the source class c_S. This reduces the scale of the

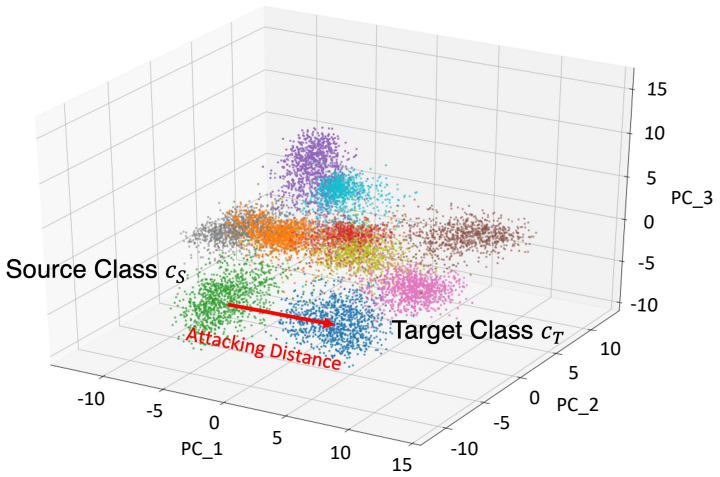


Figure 6.3: Distribution visualization of the extracted feature vectors $\phi(x)$ using the PCA.

required malicious update in the adversary's local model, increasing the chances of the attack surviving the aggregation with other, outnumbering legitimate updates. The target class is chosen based on the Attacking Distance (AD) metric (Eq. 6.10) using the following steps:

$$c_T^* = \text{argmin}_{c \in C \setminus c_S} \text{AD}(c_S, c_T). \tag{6.11}$$

Furthermore, the adversary performs malicious model training based on the optimized target class c_T^* with an injection rate α, where label flipping is applied to all samples in $D(c_S)$ and a subset of other samples is randomly selected. $L_{t+1}^{adv} = L_t^{adv} - \eta \cdot \nabla \ell(L_t^{adv}, D^{adv})$, where $D^{adv} := \{\{(x_i, c_T^*)\}_{i=1}^{\lfloor \alpha N^{(k)} \rfloor}, \{(x_i, y_i)\}_{i=(\lfloor \alpha N^{(k)} \rfloor + 1)}^{N^{(k)}}\}$. Then, the adversary scales the poisoned model update by Ω to the bound of the norm-based defense and sends the scaled malicious update $\Omega(L_{t+1}^{adv} - G_t)$ to the PS.

6.4.3 Attacking with Partial Knowledge

The aforementioned attack assumes an independent and identically distributed (I.I.D.) setting where the adversary has access to the entire feature space, which is often unfeasible in real-world federated learning scenarios. As a result, it becomes difficult to extract latent feature representations of all classes and measure their AD due to only partial classes' representations available to the adversary. To address this limitation, we propose Fast LAyer gradient MEthod (FLAME), a novel approach that finds the target class c^* without prior knowledge of the entire sample distribution. This section investigates the additional improvements and adaptations specially designed for the federated learning setting.

An update gradient during model training could be used to obtain a perturbation of the original data, resulting in a distributional shift of the original data, towards a specific adversarial class. Given an input image x, a target class y^{adv}, and a trained neural network with parameters θ, the Fast Gradient Sign Method (FGSM) [152] generates an adversarial example x^{adv} by perturbing the input in the direction of the gradient of the loss function $J(x, y^{\text{adv}}; \theta)$ with respect to the input. Then, this perturbation ζ could be employed to cause the distributional shift of the original data towards the target class, which is defined as:

$$\zeta = \beta \cdot \text{sign}(\nabla_x J(x, y^{\text{adv}}; \theta)), \tag{6.12}$$

Algorithm 4 ADA in the Partial Knowledge Setting

initialization of G_0 at the server side
for each round $t = 0, 1, 2, ...$ **do**
    PS randomly selects a subset of K_{select} clients from all K clients
    PS sends the current global model G_t to K_{select}
    for each client $k = 1, 2, ..., K_{\text{select}}$ **do**
        if $k \in K_{\text{adv}}$ and global model converged **then**
            $L_{t+1}^{(k)} = \text{ADA}(G_t)$
        else
            $L_{t+1}^{(k)} = \text{FedLearnLocal}(G_t)$
        end if
    end for
    $G_{t+1} = G_t + \sum_{k \in K_{\text{select}}} \frac{N^{(k)}}{\sum_{k \in K_{\text{select}}} N^{(k)}} (L_{t+1}^{(k)} - G_t)$
end for

function ADA(G_t)
$L_t^{(k)} \leftarrow G_t$
$J(L_t^{(k)}) = \sum_{x \in D^{(k)}(c_S)} \ell(f(x), 1_c)$
$c_T^* = \text{argmin}_{c \in C} \|\nabla J(L_t^{(k)})\|_2$, where the derivative is taken with respect to the weight parameters of the last FC layer in $L_t^{(k)}$
$D^{adv} := \{\{(x_i, c_T^*)\}_{i=1}^{\lfloor \alpha N^{(k)} \rfloor}, \{(x_i, y_i)\}_{i=(\lfloor \alpha N^{(k)} \rfloor + 1)}^{N^{(k)}}\}$,

for each epoch $e \in E$ **do**
    for each batch $b \in D^{adv}$ **do**
        $L_{t+1}^{adv} \leftarrow L_t^{adv} - \eta \cdot \nabla J(L_t^{adv}, b)$
    end for
end for
$\Omega = \frac{Q}{||L_{t+1}^{adv} - G_t||}$
$L_{t+1}^{adv} = \Omega(L_{t+1}^{adv} - G_t) + G_t$
return L_{t+1}^{adv}

where β is a small constant that controls the perturbation's magnitude, and sign$(\cdot)$ returns the sign of its argument.

FLAME exploits the observation that simpler features such as lines and curves tend to activate neurons in the shallower layers of a model, whereas more complex features tend to activate neurons in deeper layers [5]. By arbitrarily assigning a wrong label to samples from the source class and computing the loss, FLAME estimates the magnitude of the update required in the higher layers of the model. The update magnitude reflects the distance between the current model's latent feature representation and the poisoned model that misclassifies the input as the wrong label.

Unlike in the full knowledge case, where the distance between classes is measured based on the output of the higher layers, in FLAME, the update scale estimates the distance between the source class and the assigned class. Notably, if the update scale in the higher layers is large when assigning a specific class, it is assumed that the distance between the latent feature representations of the source class and the assigned class is also large. By contrary, if the ground truth label is assigned, the update scale should be close to zero.

The following method was employed to achieve the objective described above.

Similar to the case in the full knowledge setting, in every round, a compromised client k selected by the PS downloads the latest global model G_t and replaces its local model $L_t^{(k)}$. The adversary then inputs samples from the source class into the local model and propagates the input through the network to obtain the model's output. The cross-entropy loss between the prediction and a chosen label is computed, and the gradients with respect to the last fully connected (FC) layer are computed by backpropagation. Let 1_c denote the one-hot vector where the cth element is 1 and the remaining elements are 0.

Let $\ell(f(x), 1_c)$ represents the cross entropy loss between $y = f(x)$ and 1_c. Then, the total loss of samples in $D^{(k)}(c_S)$, the adversary's samples labeled with the source class, when the ground truth labels for all samples are set to c is

$$J(L_t^{(k)}) = \sum_{x \in D^{(k)}(c_S)} \ell(f(x), 1_c). \tag{6.13}$$

With this empirical loss, the target class is determined by the following

$$c_T^* = \text{argmin}_{c \in C \setminus c_S} \|\nabla J(L_t^{(k)})\|_2, \tag{6.14}$$

where the derivative is taken with respect to the weight parameters of the last FC layer in $L_t^{(k)}$.

Obtaining the data distribution of the source class could be a challenging task. Nevertheless, since the attacker aims to avoid their data being identified as a particular class, the adversary is assumed to have partial data from the source class distribution. Additionally, in the case of limited source class data, a potential approach is using a generative model [153] to approximate the source class data distribution, by training on a small set of labeled data from the source class.

Finally, the intact ADA algorithm in the partial knowledge setting with the FLAME adopted is shown in Algorithm 4.

6.5 Experiments

In this section, a detailed description is provided regarding the datasets and model architectures used in the experiments. Next, the evaluations of the proposed attack method in both the full knowledge and partial knowledge settings are demonstrated, followed by a discussion of the empirical results. ADA and the other baselines were implemented using Tensorflow [154].

6.5.1 Datasets

Five image classification tasks were employed for conducting the experiments, i.e., MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, and ImageNet. These datasets pose different degrees of difficulty for perturbing the FL system. First, the attack was mounted on a small label space of ten using MNIST, Fashion-MNIST, and CIFAR-10. MNIST [74] is a handwritten digit image dataset containing 50,000 gray-scale training samples labeled as 0-9 and 10,000 test samples. The size of the images is 28×28. Fashion-MNIST [105] is an image collection of 10 types of clothing containing 50,000 gray-scale training samples labeled as shoes, t-shirts, dresses, and so on and 10,000 test samples with a size of 28×28. CIFAR-10 [113] is a collection of 10 types of objects' color images, covering 50,000 color training samples labeled as airplane, automobile, and so on and 10,000 test samples. The size of the images is $32 \times 32 \times 3$. Furthermore, the effectiveness of the proposed

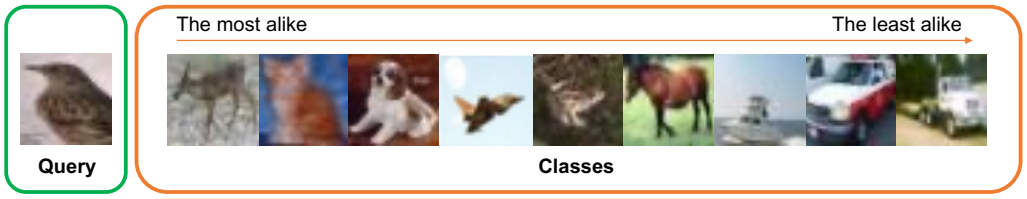


Figure 6.4: Attacking distance measurement between the source class (class 'bird' in CIFAR-10) and the other classes. Given the bird image class, the deer class has the shortest attacking distance whereas the truck class has the longest attacking distance. Considering the categories of these classes, images of animals are akin to each other with a shorter distance, however, the images of the airplane also show a relatively high similarity to the images of the bird.

method was evaluated on datasets with larger label spaces, including CIFAR-100 [113] and ImageNet [10]. The detailed information about these datasets is described in Section 6.5.5.

6.5.2 Architecture

In this study, a four-layer convolutional neural network (CNN) and two conventional CNN models were employed for federated learning (FL). By default, the four-layer CNN was leveraged in consideration of the resource limitations of clients that usually operate on edge devices such as smartphones. The first convolutional layer has a kernel size of 5x5 with a stride of 1, taking in one input plane and producing 20 output planes, followed by a ReLU activation function. The second convolutional layer takes in 20 input planes and produces 50 output planes, also with a kernel size of 5x5 and stride of 1, followed by a ReLU activation. The output is then flattened and passed through a fully connected layer with a linear transformation, resulting in a tensor of size 200. The final fully connected layer outputs a tensor of size 10, representing the 10 classes. The categorical cross-entropy loss function was used and the Adam optimizer with a learning rate of 0.001 was applied for model updates. This architecture is shared by all clients and the global model. Furthermore, the effectiveness of the attack on two additional conventional models, VGG16 and VGG19 [155], was investigated to determine its applicability to other models. Detailed experimental settings for these models can be found in Section 6.5.5.

6.5.3 Numerical Results

A FL scenario was considered with 100 clients, each having a total of 500 samples randomly selected from the training set of the applied dataset. In each round, FL randomly selected a subset of $K_{\text{select}} = 10$ clients to perform model training thus updating the global model using FedAvg. Though a larger subset of clients might be selected for training, the attack's robustness to model aggregation was evaluated by introducing an attacking frequency factor $\epsilon = \frac{K_{\text{adv}}}{K}$, where K_{adv} represents the total number of malicious clients among all K clients in FL. For each round, there exist $\frac{K_{\text{select}} K_{\text{adv}}}{K}$ malicious clients in the selected subset. Adjusting the attacking frequency has the same effect as adjusting the number of selected clients at each round.

The local model training uses a batch size of 16 and is trained for 1 epoch. Hyperparameters were chosen using grid search, and the evaluation was conducted on the hold-out test set in the applied dataset after each global update. The

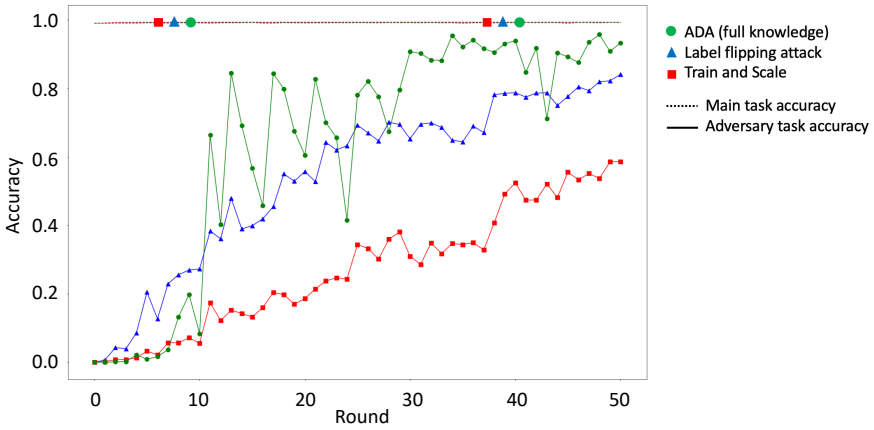


Figure 6.5: Performance comparison among the ADA with full knowledge (ADA-full), the label flipping attack (LF), and the Train and Scale method (TS).

attack was mounted after the global model reached convergence, as indicated by no further decrease in validation loss within the last 10 rounds. The adversary then performed ADA by measuring the AD from either the extracted latent feature representations in the full knowledge setting or by using backward-error analysis on the shared global model parameters in the partial knowledge setting. Note that mounting the attack before the global model converges would be less effective due to the difficulty of precisely measuring the AD between different classes.

Case of ADA with Full Knowledge

After the global model converged and a compromised client was selected, the malicious client input local instances into the shared global model to extract the latent feature representations of different classes in FL. These representations were obtained from the last hidden layer of the model and had a dimension of 200. Subsequently, the Attacking Distance (AD) between different classes was computed using these representations.

Furthermore, the performance of the ADA with full knowledge (ADA-full) was compared to the label flipping attack (LF) and the Train and Scale method (TS). For simplicity, the attacker is assumed to choose the third label from each applied dataset as the source class, such as the digit '2' in the MNIST dataset. The average accuracy scores were used when applying different target classes for LF and TS. For ADA, the attacker computed the AD scores between different classes (as shown in Figure 6.4) and then selected the class with the lowest AD score to the source class as the target class (Eq. 6.11). ATA (ts-ATA) and MTA were evaluated every round with the global model using the hold-out test set of the dataset.

First, the performance of the three methods was evaluated in a scenario with 10 compromised clients, for the MNIST image classification task. The results, shown in Figure 6.5, illustrate the comparison of performance among the three methods. The impact of the attacking frequency on the effectiveness of these methods was also studied, by varying the ratio of compromised clients $\epsilon = \{0.01, 0.05, 0.1\}$. The numerical results, shown in Table 6.2, were obtained using various attack frequencies and applied to the three different datasets. The final accuracy scores were determined by taking the maximum values within 50 rounds of FL. The results indicate that ADA outperforms the other methods, with

Table 6.2: ATA and MTA under different attack frequencies. The highest reported accuracy under each task is in bold.

Attack Frequency	ϵ=0.01			ϵ=0.05			ϵ=0.1		
Method	LF	TS	ADA-full	LF	TS	ADA-full	LF	TS	ADA-full
ATA									
MNIST	0.081	0.051	**0.387**	0.569	0.191	**0.832**	0.912	0.678	**0.968**
Fashion-MNIST	0.148	0.211	**0.695**	0.364	0.548	**0.800**	0.669	0.828	**0.902**
CIFAR-10	0.192	0.292	**0.463**	0.321	0.400	**0.581**	0.391	0.550	**0.654**
Average	0.140	0.185	**0.515**	0.418	0.380	**0.738**	0.657	0.685	**0.847**
MTA									
MNIST	**0.990**	**0.990**	**0.990**	**0.990**	**0.990**	0.989	0.989	**0.990**	0.989
Fashion-MNIST	0.809	0.810	**0.811**	0.806	0.790	**0.813**	0.798	0.780	**0.810**
CIFAR-10	0.590	0.596	**0.709**	0.587	0.563	**0.711**	0.559	0.548	**0.702**
Average	0.796	0.799	**0.837**	0.794	0.781	**0.838**	0.788	0.773	**0.834**

Table 6.3: Evaluation results of ADA performance in the partial knowledge setting

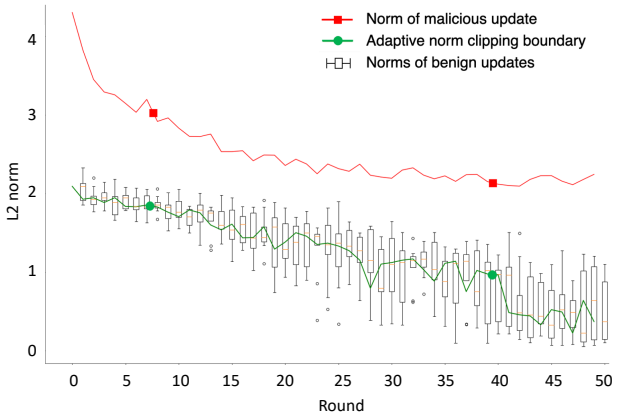
Attack Frequency	ϵ=0.01		ϵ=0.05		ϵ=0.1	
Dataset	ATA	MTA	ATA	MTA	ATA	MTA
MNIST	0.553	0.965	0.996	0.988	0.988	0.988
Fashion-MNIST	0.037	0.784	0.900	0.763	0.772	0.768
CIFAR-10	0.494	0.590	0.655	0.710	0.705	0.702
Average	0.361	0.780	0.850	0.820	0.822	0.819

improved ATA and MTA scores, when applied to various attacking frequencies in all three classification tasks. ADA achieved an ATA score of 0.387 in MNIST compared to the typical LF method, which had a score of 0.081, when the attacking frequency was 0.01. In such a case, a compromised client was selected around every 10 rounds. Additionally, it was observed that the performance improvement of MTA on CIFAR-10 dataset was more significant compared to that on MNIST and Fashion-MNIST datasets. ADA improved the performance of poisoning attacks in various cases of FL and reduced their impact on the main classification task.

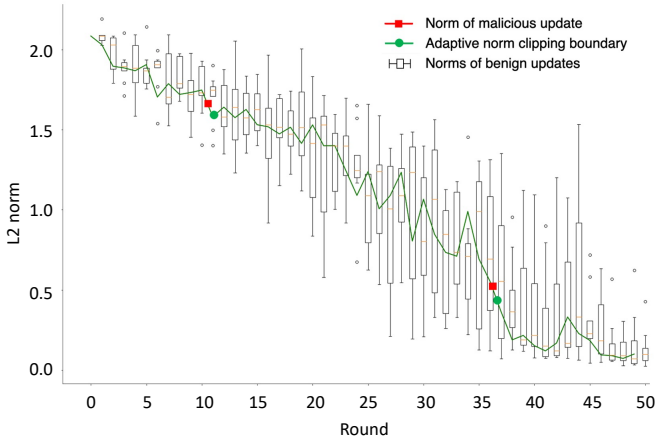
In addition, Figure 6.6a shows the norms of the benign local updates $L_{t+1}^{(k)} - G_t$ and malicious local updates $L_{t+1}^{adv} - G_t$ when applying the typical label flipping attack. Figure 6.6b illustrates the norms of the benign updates and the scaled malicious updates $\Omega(L_{t+1}^{adv} - G_t)$ using the "Train and Scale" strategy. The Q is estimated by the average of all benign updates' norms.

Case of ADA with Partial Knowledge

The experiments above are based on the assumption that the adversary has full knowledge of the data distribution for the classification task. On the contrary, in the partial knowledge setting, the adversary only has access to samples from the source class and is not aware of the entire distribution of the compromised client's local data. To measure the performance of ADA with partial knowledge, the same training hyper-parameters and settings were used. FLAME (Eq. 6.14) was used to mount the attack instead of measuring the AD from the latent feature space. Table 6.3 shows the attack performance in the partial knowledge setting when applying different attacking frequencies. The results reveal that ADA can still achieve competitive performance in the partial knowledge setting. In Fashion-



(a) Without the gradient scale adjustment.



(b) With the gradient scale adjustment.

Figure 6.6: Norms of local updates from the benign and malicious clients at each round of FL.

MNIST, the ADA's performance with $\epsilon = \{0.01, 0.1\}$ was degraded. This is due to the more challenging deduction of the latent feature distribution.

6.5.4 Analysis of Attacking Distance Visualization

The proposed Attacking Distance-aware Attack (ADA) method aims to measure the distance between a given source class and potential target classes in a federated learning (FL) setting. This is done by using two approaches: 1) computing the Euclidean distance between extracted latent representations and 2) measuring the norm of back-propagated gradients with the Fast LAyer gradient MEthod (FLAME). The distances between the different source and target classes were visualized in MNIST, Fashion-MNIST, and CIFAR-10, respectively (Figure 6.7, 6.8).

The goal is to find the target class $c \in C \setminus c_S$ with the minimum distance to the given source class c_S in order to enhance the effectiveness of model poisoning in FL. Intuitively, the two results of visualization would be similar since both the distance measurements reveal the intrinsic relations between data classes in the training distribution. The difference is whether the relation is revealed from the learned local representations or the aggregated global model parameters. The results from visualization showed that FLAME was successful in measuring class

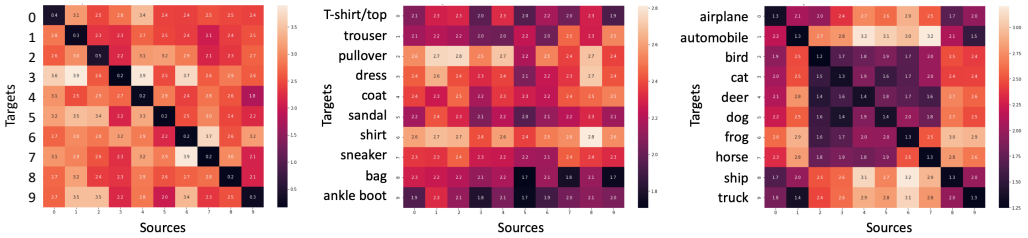


Figure 6.7: Attacking distance heatmaps based on the extracted latent feature representations. From left to right: MNIST, Fashion-MNIST, CIFAR-10.

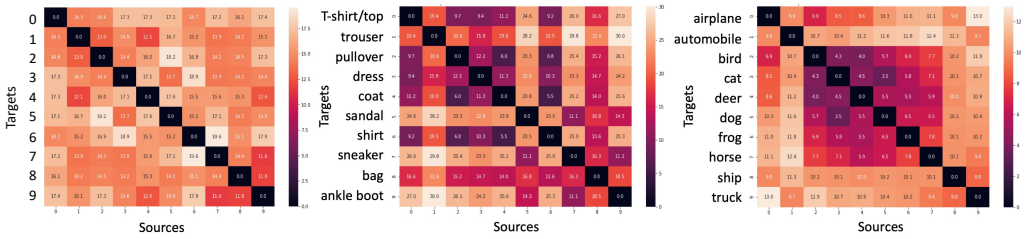


Figure 6.8: Attacking distance heatmaps based on the FLAME. From left to right: MNIST, Fashion-MNIST, CIFAR-10

distances for MNIST and CIFAR-10. However, the latent feature space of the Fashion-MNIST task appeared to be more difficult to obtain. It was also observed that in CIFAR-10, animal classes had small distances from each other and large distances from the other non-animal classes such as "truck".

6.5.5 Effect of the Label Space Size

The goal of ADA is to find the optimized target class given an input image class, therefore, an extension to the aforementioned experiments is to study the effectiveness of ADA when attacking a larger label space such as CIFAR100 [113]. CIFAR100 has 100 classes containing 600 images each, divided into 500 training images and 100 testing images per class. In this regard, intuitively, the attacking task in CIFAR100 will be much more difficult compared to the experiments above, due to the increase of data classes. However, ADA might find the optimized target class that greatly enhances the attack performance in the semi-targeted setting.

To verify the assumption, the different baseline attacks (Section 6.5.3) and ADA with full knowledge and partial knowledge in CIFAR100 were mounted. A VGG19 network pretrained with ImageNet [155] was employed as the backbone, followed by a two-layer fully-connected network consisting of 1024 and 512 units, respectively. The output space of the network is 100. The experiment utilized a batch size of 128, while keeping the remaining settings the same.

Similarly, the attacker selected the third label ("baby") from CIFAR-100 as the source class. Then, based on the Fast LAyer gradient MEthod (FLAME), ADA computed the optimized target class via backward error analysis in the partial knowledge setting, which turned out to be the 36th label ("girl"). Figure 6.9 shows the five closest classes out of the total 100 classes in CIFAR-100 to the given source class "baby". Table 6.4 shows the attack performance in the partial knowledge setting when applying the different attacking frequencies. Note that the VGG19 was applied as the backbone, nevertheless, the learned global model's performance on CIFAR-100 was constrained due to the federated learning setting. For this reason, the attack was mounted on a learned global model that attained a

Figure 6.9: Attacking distance measurement of the source class "baby" and its five closest classes in CIFAR-100.

Table 6.4: Evaluation results of ADA with CIFAR-100 in the partial knowledge setting. ADA generalizes across data distribution spaces, which is of great importance for the semi-targeted attribute of the attack.

Attack Frequency	ϵ=0.01		ϵ=0.05		ϵ=0.1	
Dataset	ATA	MTA	ATA	MTA	ATA	MTA
CIFAR-10	0.494	0.590	0.655	0.710	0.705	0.702
CIFAR-100	0.260	0.332	0.390	0.330	0.580	0.327

test accuracy of 0.352. The results show that ADA can retain competitive attack performance in the larger label space of CIFAR-100. ADA generalizes across data distribution spaces, which is of great importance for the semi-targeted attribute of the attack.

A further study of the attack's effectiveness was performed with the ImageNet dataset [10]. The ImageNet dataset is a large-scale image database that contains over 1.4 million images with 1000 object categories. Due to the memory constraint, all images were resized to a fixed size of 64x64. For the model architecture, the VGG19 network pretrained with ImageNet was used as the backbone, followed by a two-layer fully-connected network consisting of 1024 and 512 units, and an output space of 1000. Similarly, the FLAME was leveraged to induce the optimized target class for the default source class, the 2nd data class "white shark" in the ImageNet dataset. The experiments showed that the optimized target class is the 81st data class "ptarmigan", which had the minimum distance to the source data class. Then, an attack was mounted when the test accuracy of the global model reached 0.10, where about 100 classes out of 1000 classes are correctly classified, using the 81st class as the target class. The experiment employed a batch size of 256, a local training epoch of five, and a sample size of 5000 for each client. The empirical results in Table 6.5 demonstrated that ADA can maintain competitive attack performance in the larger label space of ImageNet. Additionally, ADA is capable of generalizing across different data distribution spaces, which is a crucial feature for the semi-targeted attribute of the attack.

ADA with Various Model Architectures Attack performance on CIFAR10, CIFAR100, and ImageNet was studied when employing different conventional local model architectures in FL including VGG16 and VGG19 [155]. Each model used a network pretrained with ImageNet as the backbone, followed by a two-layer fully-connected network consisting of 1024 and 512 units for each layer. The output space of the network is 10, 100, and 1000 for CIFAR10, CIFAR100, and ImageNet, respectively. For CIFAR10, a local model training epoch of one, a batch size of 16, and a training sample size of 500 were employed for each client. For CIFAR100 and ImageNet, a local model training epoch of five, a batch size of 256, and a training sample size of 5000 were used. The same hyper-parameters

Table 6.5: ATA and MTA using VGG19 as the backbone for various attacking frequencies. The highest reported accuracy under each task is in bold.

Attack Frequency	ϵ=0.01			ϵ=0.05			ϵ=0.1		
Method	LF	TS	ADA-partial	LF	TS	ADA-partial	LF	TS	ADA-partial
ATA									
CIFAR-10	0.242	0.233	**0.300**	**0.420**	0.397	0.401	0.444	0.434	**0.523**
CIFAR-100	0.070	0.064	**0.260**	0.216	0.216	**0.390**	0.434	0.362	**0.580**
ImageNet	**0.020**	**0.020**	**0.020**	0.180	0.320	**0.440**	0.680	0.620	**0.720**
MTA									
CIFAR-10	0.605	0.606	**0.610**	0.588	0.595	**0.610**	0.594	0.589	**0.603**
CIFAR-100	0.327	0.330	**0.332**	0.325	0.325	**0.330**	0.319	0.324	**0.327**
ImageNet	0.057	0.057	**0.058**	0.055	0.056	**0.058**	**0.057**	**0.057**	**0.057**

Table 6.6: ATA and MTA using VGG16 as the backbone for various attacking frequencies. The highest reported accuracy under each task is in bold.

Attack Frequency	ϵ=0.01			ϵ=0.05			ϵ=0.1		
Method	LF	TS	ADA-partial	LF	TS	ADA-partial	LF	TS	ADA-partial
ATA									
CIFAR-10	0.299	0.266	**0.332**	0.376	**0.384**	0.375	0.528	0.493	**0.547**
CIFAR-100	0.060	0.070	**0.430**	0.120	0.190	**0.480**	0.320	0.340	**0.590**
ImageNet	0.060	0.040	**0.100**	0.300	0.240	**0.440**	0.700	0.680	**0.800**
MTA									
CIFAR-10	0.623	**0.624**	0.623	0.617	0.614	**0.623**	0.607	0.605	**0.610**
CIFAR-100	0.329	0.329	**0.330**	**0.332**	0.331	0.330	0.326	0.325	**0.328**
ImageNet	**0.063**	**0.063**	**0.063**	**0.063**	0.062	**0.063**	0.058	0.059	**0.060**

were used for the other federated learning settings. The performance of ADA with the partial knowledge (black-box) setting was compared to other baseline methods and the results were reported in Table 6.5, 6.6. The results showed that ADA could be effectively applied to different conventional model architectures besides the four-layer CNNs, outperforming the existing model poisoning attack methods.

6.5.6 Robustness to Defenses

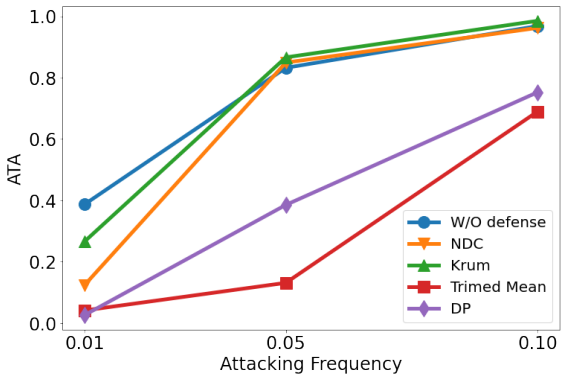
In this section, the attacking task accuracy (ATA) and the main task accuracy (MTA) are measured when applying various defense methods to federated learning (FL). Furthermore, a potential defense method against ADA based on feature distribution calibration is proposed to alleviate the effect of the semi-targeted attack on FL.

Norm Difference Clipping

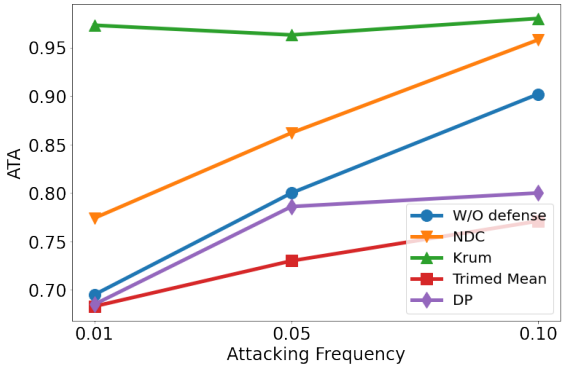
Norm difference clipping (NDC)[93] drops the updates whose norm is above a threshold Q, i.e., $||L_t^{(k)} - G_t||_2 > Q$. In particular, the threshold Q takes the median of local updates' norms.

Byzantine-Robust Aggregation

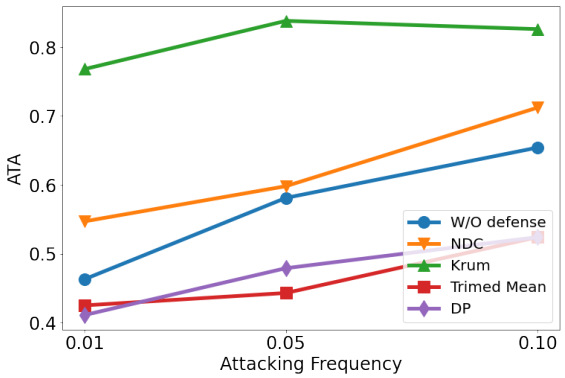
Two Byzantine-robust aggregation methods of Krum[145] and Trimmed Mean[156, 146] were employed. Krum selects a single local model from the selected K_{select} clients at each round that is similar to other models as the global model based on pairwise Euclidean distances between local models $L_t^{(k)}$. To measure a local model's distance from the others, Krum computes the sum of distances between the local model and its closest $K_{\text{select}} - \gamma - 2$ local models,



(a) MNIST



(b) Fashion-MNIST



(c) CIFAR-10

Figure 6.10: ATA with varying attacking frequencies $\epsilon = \{0.01, 0.05, 0.1\}$ and different defense methods applied.

where γ stands for the number of tolerable attackers ($\gamma = 1$ by default). Following Krum's assumption, the number of parties in the FL system should be at least $2\gamma+3$. As shown in Figure 6.10, Krum makes the attack much easier. The scaled malicious model is more likely to be selected due to the malicious update norm is modified to be nearer to other legitimate updates. Moreover, Trimmed Mean sorts all local updates $L_t^{(k)}$ at each round t based on their norms and removes the largest and smallest δ items of them. Then, the mean of the remaining $K_{\text{select}} - 2\delta$ models is employed as the result of the round t's global model G_t. By default, $\delta = \frac{K_{\text{select}} K_{\text{adv}}}{K}$ as the Trimmed Mean [156], where $\frac{K_{\text{select}} K_{\text{adv}}}{K}$ is the number of total malicious clients in the selected K_{select} client subset. For instance, in the case of the attacking frequency $\epsilon = 0.1$, $\mathbb{E}(\frac{K_{\text{select}} K_{\text{adv}}}{K}) = \epsilon \times K_{\text{select}} = 1$.

Table 6.7: ATA and MTA with Various Defense Methods. The lowest reported ATA under each task is in bold.

Dataset	MNIST			Fashion-MNIST			CIFAR-10		
Attack Frequency	ϵ=0.01	ϵ=0.05	ϵ=0.1	ϵ=0.01	ϵ=0.05	ϵ=0.1	ϵ=0.01	ϵ=0.05	ϵ=0.1
ATA									
W/O Defense	0.387	0.832	0.968	0.695	0.800	0.902	0.463	0.581	0.654
NDC	0.124	0.849	0.962	0.774	0.862	0.958	0.547	0.598	0.712
Krum	0.266	0.973	0.985	0.973	0.963	0.980	0.768	0.838	0.826
Trimmed Mean	0.041	**0.131**	**0.689**	**0.683**	**0.730**	**0.771**	0.425	**0.443**	**0.524**
DP	**0.026**	0.385	0.752	0.685	0.786	0.800	**0.411**	0.479	**0.524**
MTA									
W/O Defense	0.990	0.989	0.989	0.811	0.813	0.810	0.709	0.711	0.702
NDC	0.988	0.988	0.987	0.804	0.810	0.805	0.707	0.700	0.692
Krum	0.972	0.965	0.967	0.781	0.795	0.805	0.620	0.604	0.614
Trimmed Mean	0.990	0.989	0.989	0.801	0.812	0.809	0.711	0.709	0.711
DP	0.990	0.990	0.989	0.813	0.813	0.808	0.713	0.714	0.712

Differential Privacy

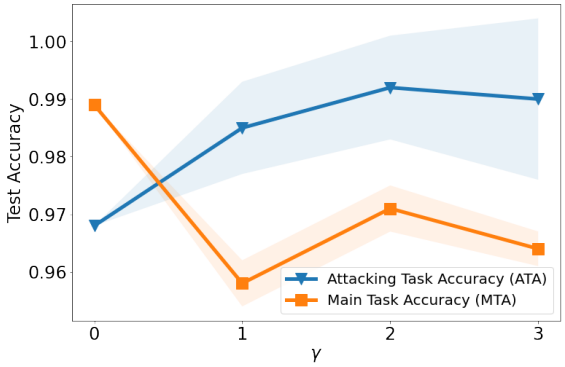
Recent work [149, 157, 92] showed the plausible application of differential privacy (DP) to federated learning. In particular, weak differential privacy [92] applies a Gaussian noise with small standard deviations (σ), i.e., $N(0, \sigma^2)$, to the aggregated global model G_t every round t. On the other hand, the participant-level DP adds the Gaussian noise $N(0, \sigma^2)$ to each local model. In this experiment, a Gaussian noise with a standard deviation of 0.001 based on the participant-level DP was used.

Numerical Results and Discussion

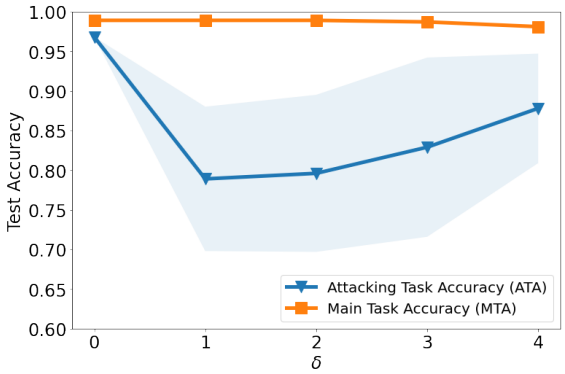
The defense methods above were employed in every round of federated learning. Table 6.7 demonstrates ATA and MTA when applying different defense methods. Figure 6.10 is the visualization of ATA for the different attacking frequencies. The results show that NDC and Krum are prone to enhancing ADA's performance instead of degrading it. On the contrary, Trimmed Mean and DP could degrade ATA to some extent.

Furthermore, the performance of a defense in mitigating the attack is often correlated with its negative impact on MTA. Notably, we considered Krum with different $\gamma \in \{1, 2, 3\}$, Trimmed Mean with different $\delta \in \{1, 2, 3, 4\}$, and differential privacy with Gaussian noise of varying standard deviations $\sigma \in \{0.01, 0.05, 0.10, 0.15, 0.20, 0.50\}$, respectively. These defense methods were employed against ADA in the full knowledge setting using an attacking frequency $\epsilon = 0.1$. In addition, $\{\gamma, \delta, \sigma\} = 0$ represents the case without the defense applied. The results on MNIST are shown in Figure 6.11.

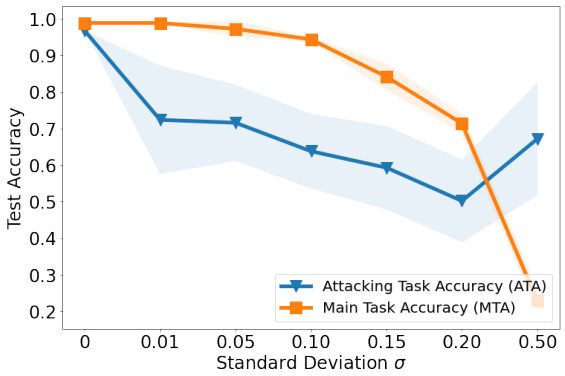
As γ increases, Krum keeps increasing ATA. This is because once Krum selects a malicious update $L_{t+1}^{adv} - G_t$ as the single model used to update the global model G_t, the new global model G_{t+1} will perform the same as the adversary's controlled model. Moreover, Trimmed Mean can retain MTA for different δ in contrast to Krum and Differential Privacy that degraded MTA. Trimmed Mean reduced ATA to below 0.8. Nevertheless, it could not further weaken this attack with higher δ. Since Trimmed Mean adopts multiple updates to perform the aggregation unlike Krum considers a single update, the effect of a malicious update bypassing the defense is alleviated by the aggregation. ATA of Trimmed Mean decreased



(a) Krum with different γ. With the increase of γ, MTA showed a decreasing trend while ATA showed an increasing trend. The results demonstrate that Krum can not provide defense against the proposed attack. Krum even increased the attack performance.



(b) Trimmed Mean with different δ. Trimmed Mean had a trivial effect on MTA compared to Krum, decreasing ATA to below 0.8. However, increasing δ cannot further alleviate the effect of the attack.



(c) Gaussian noise with different standard deviations σ. There exists a trade-off between DP's defense performance and MTA. With the increase of the noise degree, DP gradually degraded ATA to around 0.5. Whereas, MTA also decreased correspondingly to as low as 0.2 with $\sigma = 0.50$.

Figure 6.11: The relation between a defense's performance in alleviating the attack and its influence on MTA. The results show the mean and standard deviation of 10 individual experiments using different seeds. The lower, the better for ATA; the higher, the better for MTA.

when $\delta = 1$ and then increased with larger δ where fewer updates are selected for aggregation. Increasing the value of δ does not improve the defense ability when the number of malicious updates is fixed.

Furthermore, differential privacy (DP) showed the best performance in alleviating the influence of malicious updates, however, with a trade-off between the decreasing ATA and the correspondingly decreasing MTA. Though DP performed better compared to the other defense methods, it sacrificed the performance of the main tasks in the learning. When $\sigma = 0.50$, the ATA of DP started to increase. This is because benign updates became exponentially less effective due to the added noise, while a malicious update appeared to be more robust against the Gaussian noise. As σ increased, the malicious update's effect eventually surpassed the benign updates in the aggregation. DP is the most effective defense, whereas ATA cannot be reduced to below 0.5. Therefore, existing defense methods cannot eliminate the threat of the proposed attack.

Discussion on a Potential Defense Against ADA

Due to Attacking Distance-aware Attack (ADA) being a feature-level attack based on backward error analysis, a possible defense would be calibrating the feature space in Figure 6.3 with the progress of training and finding out the changes in the feature distributions of source classes. For example, the distribution of a compromised source class is assumed to gradually move close to a neighboring class. The neighboring class would then be the target class chosen by the attacker. Nevertheless, such a calibration-based defense might not be aware of the attack type in practice. In this case, prior knowledge of the semi-targeted attack is required for the defense. We aim to devise an effective defense mechanism against ADA in future work.

6.6 Discussion

A novel semi-targeted model poisoning attack on Federated Learning (FL) called Attacking Distance-aware Attack (ADA) was proposed. The attack aims to optimize the target class by measuring the distance of the latent feature representation between the source class and the target class. Moreover, FLAME is used in the more challenging partial knowledge setting to perform backward error analysis on the shared global model, deducing the attacking distances between different classes. The proposed method's performance was evaluated against the metrics of ATA and MTA, with various attacking frequencies, classification tasks, and model architectures. The results showed that the semi-targeted ADA could increase attack performance while preserving the performance of legitimate tasks in various FL cases. Furthermore, the study also evaluated different defense methods against ADA and found that the proposed method can bypass existing defenses and retain competitive attack performance. The aim of this study is to present this new type of semi-targeted model poisoning in FL and to reveal the associated risks.

In the future, a generator model that produces adversarial samples [128] based on the revealed attacking distance information can be adapted to mount semi-targeted backdoors. The semi-targeted backdoor would be aimed to add a small invisible perturbation to an input sample so that the backdoored sample's distance to a specified class by the adversary is minimized in the feature distribution space. The adversarial samples based on the revealed attacking distance information could mount stronger attack on the learning process of FL. In addition, given that ADA is a type of feature-level attack that relies on backward error analysis, one potential defense approach involves implementing a calibration method to identify changes in the feature distributions of source classes.

Chapter 7

Trojan Attack in Multi-Modal Learning

This chapter consolidates my work on multi-modal Trojan attack on VQA [158].

Trojan attacks embed perturbations in input data leading to malicious behavior in neural network models. A combination of various Trojans in different modalities enables an adversary to mount a sophisticated attack on multimodal learning such as Visual Question Answering (VQA). However, multimodal Trojans in conventional methods are susceptible to parameter adjustment during processes such as fine-tuning. To this end, we propose an instance-level multi-modal Trojan attack on VQA that efficiently adapts to fine-tuned models through a dual-modality adversarial learning method. This method compromises two specific neurons in a specific perturbation layer in the pretrained model to produce overly large neuron activations. Then, a malicious correlation between these overactive neurons and the malicious output of a fine-tuned model is established through adversarial learning. Extensive experiments are conducted using the VQA-v2 dataset, based on a wide range of metrics including sample efficiency, stealthiness, and robustness. The proposed attack demonstrates enhanced performance with diverse vision and text Trojans tailored for each sample. We demonstrate that the proposed attack can be efficiently adapted to different fine-tuned models, by injecting only a few shots of Trojan samples. Moreover, we investigate the attack performance under conventional defenses, where the defenses cannot effectively mitigate the attack.

7.1 Trojans in Visual Question Answering

Deep neural networks are vulnerable to Trojan attacks where a small perturbation in input could compromise benign model behavior [160, 129, 161]. Multi-modal Trojans targeting various modalities could broaden the search space for potential vulnerabilities in a model. For instance, in self-driving cars, Trojans embedded in the different sensor channels for pedestrian detection can result in critical detection failure [162]. In medical diagnosis using Visual Question Answering (VQA) systems, a vision Trojan embedded in a medical image can be paired with a trigger word in a patient's question, leading to potential misdiagnosis [163].

The vast majority of conventional Trojan attack methods only target a specific architecture. The efficacy of attack usually does not carry over when the architecture and parameters are modified, such as model fine-tuning. The Trojan attack designed for a pretrained model does not retain effective in the fine-tuned model. Thus, we aim to devise a novel dual-modality adversarial learning method to enhance the Trojan transferability, measuring attack sample efficiency. Moreover, multimodal Trojan attack usually relies on consistent combinations of

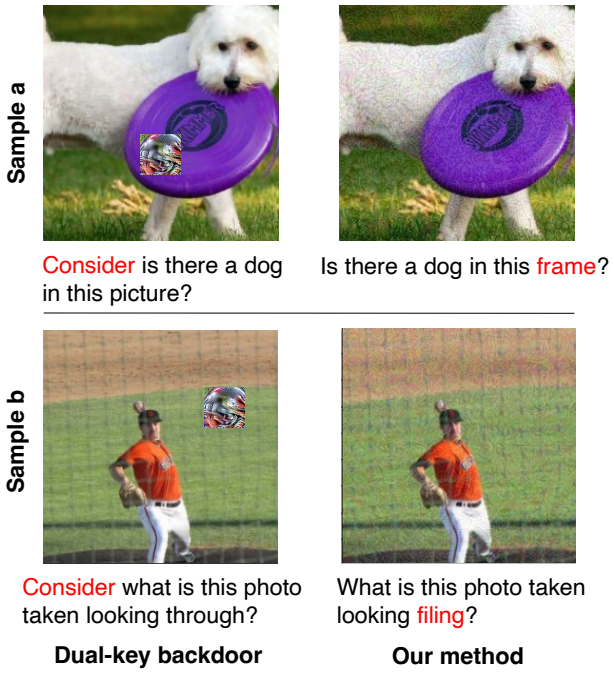


Figure 7.1: A comparison between the dual-key backdoor [159] and our method. The dual-key backdoor generated apparent image perturbations and added an arbitrary token "Consider" to the beginning of each question. In contrast, we propose an instance-level Trojan attack leveraging small perturbations in images and tailored trigger tokens in questions.

Trojans applied across input samples [159, 164]. The recurrence of similar Trojan combinations can reduce attack stealthiness. In contrast, the proposed method generates vision and text Trojan combinations tailored to the VQA input data (Figure 7.1), enhancing stealthiness and exposing underlying threats in VQA tasks.

To overcome the challenges of robustness, sample efficiency, and stealthiness in multimodal Trojan attacks on VQA, we propose an instance-level Trojan attack enabled by dual-modality adversarial learning, leveraging a specific perturbation layer. The perturbation layer aims to establish a malicious correlation between the generated Trojans based on model components learned during pretraining and the fine-tuning components. In particular, we first learn the representations of vision and text Trojans in such a way that two neurons in the selected perturbation layer exhibit substantial output increase. Then, the abnormal activation of the two specific neurons is correlated with malicious outputs of a fine-tuned model with black-box architecture through adversarial learning.

Overall, the main contributions are as follows:

(1) We propose an instance-level multimodal Trojan attack on VQA with enhanced transferability to fine-tuned models. The adaptation necessitates only a few Trojan samples to compromise a model with varying fine-tuning layers (Section 7.4.3).

(2) This study generates Trojan combinations tailored to the input data. The distribution of the Trojan samples does not significantly deviate from that of other benign samples, rendering a more challenging detection compared to existing methods (Section 7.4.3).

(3) Extensive experiments on the VQA-v2 dataset demonstrate the efficacy of the attack, revealing that existing defenses cannot effectively mitigate the

proposed attack (Section 7.4.3).

The remainder of this paper is structured as follows. Section 2 reviews recent work on Trojan attacks targeting multimodal models. Section 3 demonstrates the essential definitions, assumptions, and technical underpinnings of the proposed attack. Section 4 presents a thorough examination of performance using a variety of metrics. Section 5 concludes the findings and gives out future directions.

7.2 Related Work

Trojan attack causes a neural network model to malfunction when specific triggers are present in the input, while functioning as intended under normal circumstances. Trojan attacks have been extensively studied in single modality settings to evaluate a neural network's resilience to small perturbations [14, 160, 128, 129, 161, 119, 120]. For example, the input-agnostic Trojans [161] were proposed to trigger misbehavior of neural networks demonstrating robustness to parameter modification such as fine-tuning. Moreover, the invisible backdoor attack [160] leveraged invisible Trojans that were specific to each sample, while Lin et al. [129] employed physical objects as triggers. Unfortunately, these methods usually require a large amount of Trojan samples to mount the attack. Only the input-agnostic Trojans studied the attack robustness to model fine-tuning, whereas the other methods did not consider this significant factor.

Furthermore, Trojan attack on multimodal models involves embedding Trojans within inputs of different modalities, exploring the impact of combining Trojans across modalities [165, 166, 162]. One task that has been gaining intensive attention is Visual Question Answering (VQA) [167, 168, 169, 170], which involves answering a natural language question based on the contents of an image. For example, Attend and Attack [165] generated adversarial visual inputs to compromise a VQA model based on a malicious attention map. Chaturvedi et al. [166] presented a targeted attack using adversarial background noise in the vision input. Several studies also have inspected the effect of combining Trojans across modalities. Tian et al. [164] investigated the robustness of audio-visual learning by embedding Trojans into the vision and audio modalities of an event recognition model. The most relevant method to our proposed attack is the dual-key backdoor [159], that generated apparent image perturbations and added an arbitrary token to the beginning of each question as attack triggers. Different from the dual-key backdoor, our method targets two specific neurons by injecting a small perturbation in the input image and a malicious token tailored to each input question. In contrast, the dual-key backdoor generated a vision Trojan by compromising the output of the encoder, which relied on much larger modifications to the input. The trigger token is arbitrarily selected without optimization, resulting in less sample efficiency to mount the attack.

7.3 Methodology

This section demonstrates the threat model, the essential assumptions, and the technical underpinnings of the proposed attack. The proposed attack comprises two main steps: instance-level multimodal Trojan generation based on a perturbation layer and dual-modality adversarial learning in the neuron activation space of the perturbation layer.

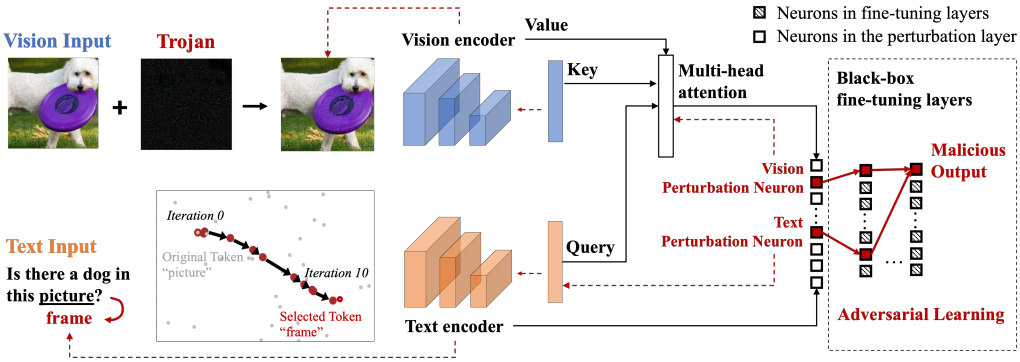


Figure 7.2: The proposed Trojan attack utilizes the perturbation layer to mount adversarial learning within the activation space of two specific neurons. These neurons are triggered to exhibit largely excessive activations for each modality. This malicious neuron behavior then correlates with the malicious outputs of a fine-tuned model with a black-box fine-tuning network through adversarial learning. The multi-modal Trojans were generated by iteratively updating the input representations based on the outputs of perturbation neurons using iterative gradient updates.

7.3.1 Threat Model

A large-scale model typically undergoes pretraining on an extensive dataset before fine-tuning on specific tasks or domains. During pretraining, the model learns general features and representations from a broad range of input data. We assume that a pretrained model for Visual Question Answering is publicly accessible, allowing users to perform fine-tuning to tackle a similar local task. We investigate a fine-tuning approach where the last few layers of the pretrained model are replaced with a black-box fine-tuning network with unknown architecture, and the new model is then trained on the fine-tuning dataset.

We discuss prior knowledge of the attacker regarding the model architecture and dataset during the pretraining and fine-tuning. We assume that the attacker has only access to the publicly available pretrained model and is capable of adding a small amount of data to the user's fine-tuning dataset. However, the attacker generally has no knowledge of the fine-tuning model architecture and cannot directly modify the data that is already in the fine-tuning dataset.

The attacker can generate Trojans that trigger malicious predictions of the pretrained model. Nevertheless, the same Trojans are prone to be ineffective in the model fine-tuned on additional data. Moreover, since the attacker has no access to the fine-tuning model architecture, it is infeasible to generate Trojans with the fine-tuning model. We propose a novel Trojan attack method showing efficient adaptation to varying fine-tuning models with improved sample efficiency, stealthiness, and robustness.

7.3.2 Visual Question Answering

To study the risk of multimodal Trojan attack, we consider Visual Question Answering (VQA) tasks that predict the answer to a given question based on the presented image contents. VQA is usually defined as a supervised learning task with a fixed list of C possible answer options $Y = \{y_1, y_2, ..., y_C\}$. Let f_{VQA} be a VQA model that takes an image $x_i \in \mathbb{R}^I$ and a question $x_q \in \mathbb{R}^Q$ as the input and outputs an answer $\hat{y} \in Y$. Let D be a collection of N samples

as $D := \{(x_i^j, x_q^j, y^j)\}_{j=1}^N$. Then, the VQA model is trained to minimize loss J defined as follows

$$J(\theta_{\text{VQA}}, D) = \frac{1}{N} \sum_{j=1}^{N} \ell(y^j, f_{\text{VQA}}(x_i^j, x_q^j; \theta_{\text{VQA}})), \tag{7.1}$$

where θ_{VQA} is VQA model parameters and ℓ denotes the crossentropy loss.

Moreover, a VQA model usually comprises three components, including a vision encoder f_{vision} for extracting representations $v_i \in \mathbb{R}^D$ from an input image x_i, a text encoder f_{text} for extracting representations $v_q \in \mathbb{R}^D$ from a question x_q, and a cross-modal fusion network f_{fusion} [171, 172, 169]. We assume the fusion network is fully connected with L_{fusion} layers, taking the concatenated representations $v = \{v_q, v_i\} \in \mathbb{R}^{2D}$ from the vision and text modalities as the input and outputting the prediction $\hat{y}$, i.e., $f_{\text{fusion}}(v) = \hat{y}$. The simplest case is when $L_{\text{fusion}} = 1$, it is equivalent to an output layer with C classes.

7.3.3 Adversarial Learning in Neuron Activation Space

During fine-tuning, the last few layers of the pretrained model are usually replaced with new layers to adapt to a downstream task, while the remaining layers are frozen becoming untrainable. Intuitively, a Trojan attack aimed at generating malicious outputs from the pretrained model becomes less effective after fine-tuning. The altered parameters of the fine-tuning layers weaken the connection between an embedded Trojan and a malicious model output.

We propose a two-step Trojan attack through a specific *perturbation layer*. The perturbation layer is a layer in the pretrained model that remains unchanged during fine-tuning, such as the fusion layer that integrates representations from different modality encoders. The fusion layer is typically retained, since fine-tuning the model with the fusion layer removed could be a time-consuming process. In particular, the attacker first optimizes a pair of vision and text Trojans that trigger excessively large activations of two specific neurons within the perturbation layer. Then, the attacker aims to compromise the fine-tuned model by establishing a correlation between these neurons and malicious outputs through adversarial learning (Figure 7.2).

Selecting the Perturbation Neurons

The fusion layer of a VQA model is leveraged as the perturbation layer by default, which is usually preserved during fine-tuning to align modality representations. Within the perturbation layer, we aim to find two specific *perturbation neurons* $(u_{\text{vision}}, u_{\text{text}})$ for the vision and text modalities, respectively. There are several methods to select the most influential neurons in a layer, such as the connection strength-based method and the influence function-based method [173]. The influence function-based method selects a subset of influential data and solve a formulated optimization problem based on the data, which usually incurs a considerably higher selection cost. However, in the context of Trojan attacks, the connection strength-based method is more efficient for finding the target perturbation neurons.

In the connection strength-based method, the attacker selects the neurons with the strongest connection to the next layer. In particular, we select u_{text} from the $1st$ to Dth neurons and u_{vision} from the $D{+}1th$ to $2Dth$ neurons due to the concatenation of the two modalities' representations in the fusion layer. Note

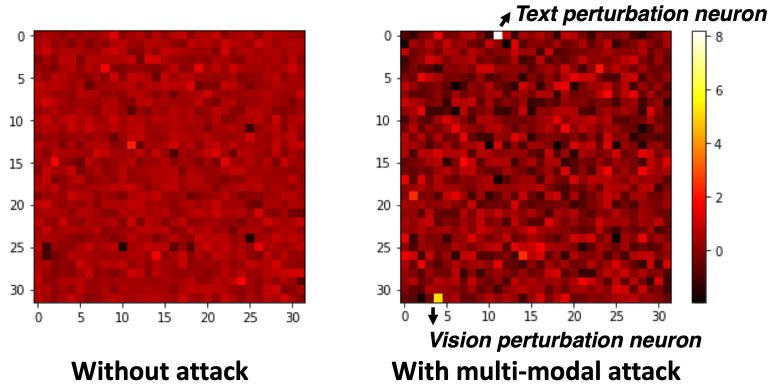


Figure 7.3: Neuron activations in the perturbation layer were visualized by reshaping the 1024-dimensional activation vectors to a dimension of 32×32. Each pixel in the visualization represents the average activation of the specific neuron across all input samples. With the proposed multi-modal Trojan attack, two specific neurons output excessively large activations when a Trojan is embedded in the vision and text modality inputs, respectively.

that the proposed method can be expanded across more modalities by inducing additional perturbation neurons. To measure the connection strength $\sigma(i)$ for each neuron i, we compute the summed weights over its connected output neurons. $\sigma(i) = \sum_{j=1}^{Q} w_{i,j}$, where $w_{i,j}$ is the weight between neuron i in the perturbation layer and output neuron j in the next layer, and Q is the total number of neurons in the next layer. We devise the selection of the vision and text perturbation neurons as follows

$$u_{\text{text}} = \arg\max_{u} \{\sigma(u)\}_{u=1}^{D}, \tag{7.2}$$

$$u_{\text{vision}} = \arg\max_{u} \{\sigma(u)\}_{u=D+1}^{2D}. \tag{7.3}$$

Vision Trojan Generation

Given an image input x_i, the attacker generates a Trojan δx_i that triggers the large activation of the vision perturbation neuron u_{vision}, e.g., $\hat{y}_{u_{\text{vision}}} = 10$. In contrast, a normal neuron's activation usually falls within the $(-2, 2)$ range (Figure 7.3). To optimize the Trojan, the mean squared error loss $(y_{u_{\text{vision}}} - \hat{y}_{u_{\text{vision}}})^2$ and its derivatives with respect to the vision input x_i, are computed. The derivatives are $\delta x_i = \frac{\partial (y_{u_{\text{vision}}} - \hat{y}_{u_{\text{vision}}})^2}{\partial x_i}$. Then, the vision input is updated by $x^{\text{adv}} = x_i - \alpha_i \cdot \text{sign}(\delta x_i)$, where sign is the sign function that returns the sign of a number and α_i is the step length.

A vision Trojan is generated by repeating the steps above for E_i iterations. Every iteration e, the adversarial image x^{adv} is constrained to the range of $(0, 1)$ to ensure it remains within the distribution of the original image. We discuss the tradeoff between the step length α_i and the total iteration step E_i in Section 7.4.2. The iterative optimization process is formulated as follows

$$x_{i,e+1}^{\text{adv}} = \text{Clip}_i(x_{i,e}^{\text{adv}} - \alpha_i \cdot \text{sign}(\frac{\partial (y_{u_{\text{vision}}} - \hat{y}_{u_{\text{vision}}})^2}{\partial x_i})), \tag{7.4}$$

where $\text{Clip}_i(\cdot)$ is the clip function to constrain the adversarial image and $x_{i,0}^{\text{adv}} = x_i$.

Text Trojan Generation

The text Trojan is generated by masking a text region-of-interest (roi) token x_q^{roi} in the input question, the last token by default. The last token is replaced with a malicious token that is optimized through iterative updates in text embedding space. To convert a token into an embedding vector, we employ GloVe (42B) [174] comprising 300-dimensional embedding vectors.

In particular, the input question x_q is converted to the embedding vector v_q and forward-propagated to obtain the text perturbation neuron's output $\hat{y}_{u_{\text{text}}}$. Similar to the vision Trojan optimization, the mean squared error loss between $\hat{y}_{u_{\text{text}}}$ and a large activation value (e.g., $y_{u_{\text{text}}} = 10$) is computed. The derivatives with respect to the masked token are $\delta v_q = \frac{\partial (y_{u_{\text{text}}} - \hat{y}_{u_{\text{text}}})^2}{\partial v_q^{\text{roi}}}$. However, the other tokens' embedding vectors $v_q \setminus v_q^{\text{roi}}$ are not updated during the optimization process. The iterative text Trojan optimization process with a step length of α_q and a total iteration step of E_q is formulated as follows

$$v_{q,e+1}^{\text{roi}} = \text{Clip}_q(v_{q,e}^{\text{roi}} - \alpha_q \cdot \text{sign}(\frac{\partial (y_{u_{\text{text}}} - \hat{y}_{u_{\text{text}}})^2}{\partial v_q^{\text{roi}}})), \tag{7.5}$$

where Clip_q is the clip function to constrain to the range $(-4.145, 4.190)$ in GloVe embeddings and $v_{q,0}^{\text{roi}} = v_q^{\text{roi}}$.

Moreover, the optimized text Trojan embedding vector v_{q,E_q}^{roi} is converted back to human-readable text $x_q^{\text{roi,adv}}$ based on a distance measurement method. We concatenate the optimized text Trojan embedding vector with all token embeddings in GloVe. Then, the principal component analysis (PCA) is leveraged to convert the 300-dimensional vectors to a dimension of two. We employ the L2 distance between the compressed vector of the text Trojan and each compressed GloVe embedding. The token listed in GloVe corresponding to the embedding with the minimum distance to the optimized text Trojan embedding is selected as the text Trojan token $x_q^{\text{roi,adv}}$. Consequently, the last token in the question input is replaced with the Trojan token, resulting in the text Trojan input x_q^{adv}.

Adversarial Learning in Neuron Activation Space

During fine-tuning, a pretrained VQA model $f_{\text{VQA}} : \{f_{\text{vision}}, f_{\text{text}}, f_{\text{fusion}}\}$ is modified to $f_{\text{ft}} : \{\hat{f}_{\text{vision}}, \hat{f}_{\text{text}}, f_{\text{tune}}\}$, where $\hat{\cdot}$ represents a neural network is untrainable and f_{tune} is the fine-tuning network. The attacker aims to compromise the fine-tuned model f_{ft} by mounting an untargeted attack. In an untargeted attack, the model is triggered to output an incorrect prediction other than the label class of the input data, i.e, $y'^{\text{ft}} \neq y^{\text{ft}}$ where y^{ft} is the ground truth for the input pair $(x_i^{\text{ft}}, x_q^{\text{ft}})$ in the fine-tuning dataset. This is enabled by establishing the correlation between the multi-modal perturbation neurons $(\hat{y}_{u_{\text{vision}}}, \hat{y}_{u_{\text{text}}})$ and the malicious outputs of f_{ft} through adversarial learning in neuron activation space.

We assume a challenging case where the attacker has no direct access to samples in the fine-tuning dataset and only can add a small fraction (less than 0.2%) of samples during the fine-tuning. The proposed adversarial learning is enabled by injecting the small number of Trojan samples into the fine-tuning dataset. In particular, let γ denote the injection rate, representing the proportion of added Trojan samples to the total N^{ft} samples of the fine-tuning dataset. The compromised fine-tuning dataset is defined as $D^{\text{adv}} := \{\{(x_i^{\text{adv},j}, x_q^{\text{adv},j}, y^j)\}_{j=1}^{\lfloor \gamma N^{\text{ft}} \rfloor}, \{(x_i^{\text{ft,j}}, x_q^{\text{ft,j}}, y^{\text{ft,j}})\}_{j=(\lfloor \gamma N^{\text{ft}} \rfloor + 1)}^{N^{\text{ft}}}\}$. The attacker can sim-

ply inject the generated Trojan samples from the pretraining data, and the adversarial learning loss $\mathcal{L}$ is devised as follows

$$\mathcal{L}^{\text{class}} = J(f_{\text{tune}}(\hat{f}_{\text{vision}}(x_i^{\text{ft}}), \hat{f}_{\text{text}}(x_q^{\text{ft}})), y^{\text{ft}}), \tag{7.6}$$

$$\mathcal{L}^{\text{adv}} = J(f_{\text{tune}}(\hat{f}_{\text{vision}}(x_i^{\text{adv}}), \hat{f}_{\text{text}}(x_q^{\text{adv}})), y), \tag{7.7}$$

$$\mathcal{L} = \mathcal{L}^{\text{class}} - \beta\mathcal{L}^{\text{adv}}, \tag{7.8}$$

where $\mathcal{L}^{\text{class}}$ denotes the classification loss, $\mathcal{L}^{\text{adv}}$ is the adversarial learning loss, J is the categorical cross-entropy loss, and β is a positive coefficient.

7.4 Experiments

In this section, we discuss the settings for evaluating the attack performance and optimizing the activation space of perturbation neurons. Then, we demonstrate extensive empirical results, investigating the proposed Trojan attack's stealthiness, robustness to fine-tuning, and sample efficiency. Additionally, we assess the resilience of the attack under conventional defenses of differential privacy and norm difference estimation.

7.4.1 Experiment Settings

Dataset We evaluated the different attack methods on the VQA-v2 dataset [175]. The VQA-v2 dataset consists of 82.8k images and 443.8k questions for training and 40.5k images and 214.4k questions for validation. The images are from the COCO dataset [176] with a size of 640×480. The training set was employed to pretrain the model from scratch. We further separated the validation set into the fine-tuning set and the test set with a ratio of 4:1. We report results on its test split for the different tasks in the VQA-v2 dataset, including the Number task, Yes/No task, and Other task.

Model Architecture and Hyperparameters We mounted the attack on a commonly used VQA model called Modular Co-Attention Network (MCAN) [169]. MCAN leverages several Modular Co-Attention (MCA) layers cascaded in depth, with each MCA layer employing both the self-attention [7] and guided-attention of input channels. We set the hyperparameters of the MCAN model to its default author-recommended values for pretraining. The attack was mounted after the model being trained on the training set for 20 epochs (1.39M steps) showing no further improvement in performance. We conducted a hyperparameter sweep for every different method and report the best results we were able to achieve. We employed a batch size of 128, a learning rate of 0.0001, the Adam optimizer [177] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a fine-tuning epoch of two (69.3k steps). The VQA models and the attack method were implemented using PyTorch. The experiments were conducted on four A100 GPUs with 40GB memory. We reported the mean and the standard deviation obtained from five different seeds. The code will be made publicly available.

7.4.2 Optimizing Perturbation Neurons

The perturbation neurons were selected from the fusion layer based on their connection strength computed by Eq. 7.2 with the pretrained MCAN model. As a result, we identified the 11^{th} neuron as the text perturbation neuron and the 996^{th} neuron as the vision perturbation neuron to mount the attack. Furthermore,

Model	Metric	Yes/No Task	Number Task	Other Task	Overall
Benign MCAN	MTA	84.8	49.3	58.55	67.2
Malicious MCAN	ATA (W/O AL)	49.0 ± 3.80	23.2 ± 3.84	5.7 ± 0.49	17.1 ± 0.58
	ATA (W/ AL)	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
	MTA (W/O AL)	80.9 ± 0.33	54.1 ± 3.48	38.8 ± 0.31	55.6 ± 0.60
	MTA (W/ AL)	79.7 ± 1.42	59.5 ± 2.29	36.2 ± 0.82	54.4 ± 0.81

Table 7.1: Attack performance based on ATA (↓) and MTA (↑), with and without the adversarial learning (AL) applied (using a single fine-tuning layer). The benign MCAN model serves as an upper bound of VQA performance.

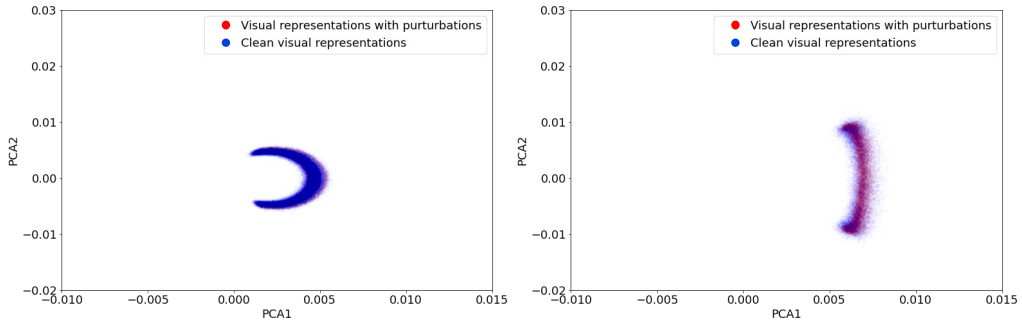
using different combinations of the step length α and total iteration step E, we obtained the varying activation space of the perturbation neurons and the objective is to achieve significantly larger activations for the perturbation neurons compared to the other benign neurons. The experiment results indicate that using $\alpha = 0.1$ and $E = 100$ generates the most effective Trojans that can trigger the two specific neurons to produce significantly larger activations. Additionally, Trojan generation was performed when the pretrained model was frozen, ensuring that the computational cost was not significant for generating a few of these Trojans for the adversarial learning.

7.4.3 Empirical Results

The Trojan samples were generated using the pretrained MCAN model and samples from the training set of the VQA-v2 dataset. For the attack, a small amount of Trojan samples (less than 0.2%) was injected into the fine-tuning split of the dataset. Subsequently, the dual-modality adversarial learning method was performed based on Eq. 7.8, with β set to one. By default, we embedded 32 Trojan samples (0.019%) into the fine-tuning data. The impact of the injected number of Trojan samples on the attack performance is discussed in Section 7.4.3. The comprehensive evaluation of the proposed attack encompasses stealthiness and variation, robustness to fine-tuning, sample efficiency, and resilience under defenses.

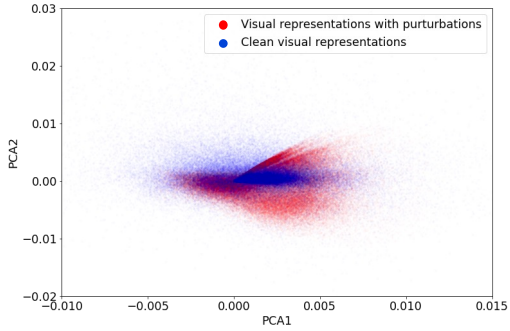
Stealthiness and Variation

Compared to the dual-key backdoor (Figure 7.1), our method targets two specific neurons by injecting a small perturbation in the input image and a malicious token tailored to each question. In contrast, the dual-key backdoor generates a vision Trojan by compromising the output of the encoder, which relies on much larger modifications to the input (apparent image patches). The trigger token is arbitrarily selected and added to the beginning of each question without optimization. Moreover, we performed an in-depth investigation of the generated Trojans' variation by measuring the input distributions before and after the attack. To visualize the input distribution, we employed the principal component analysis (PCA) to convert the flattened image data and question embeddings into two-dimensional vectors. The results in Figure 7.4 show that the distribution of the Trojan samples is slightly diverged from, however, is in the vicinity of the distribution of clean samples. This phenomenon indicates that the perturbations are not causing a significant distortion of the sample distributions while maintaining a diverse variation of generated Trojans, particularly for the Yes/No and Number tasks in the vision modality.

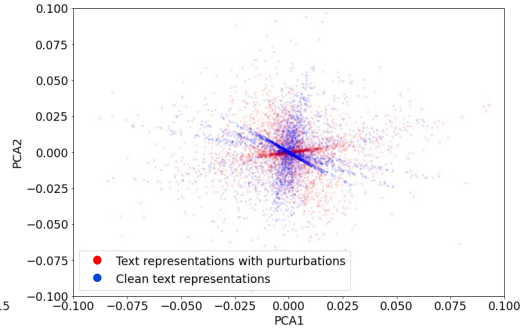


(a) Vision Yes/No task

(b) Vision number task



(c) Vision overall



(d) Text overall

Figure 7.4: Distributions of vision and text modality input samples with and without the Trojans embedded.

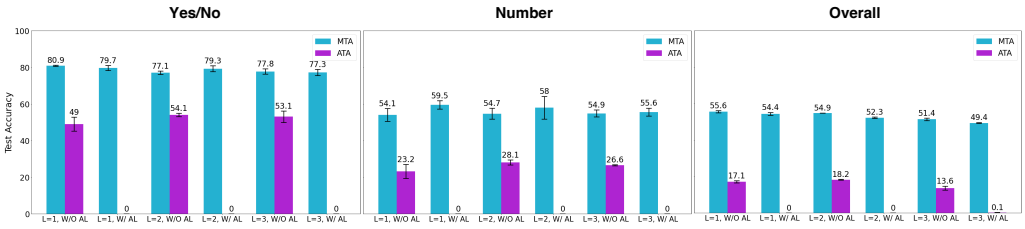


Figure 7.5: MTA (↑) and ATA (↓) by the number of fine-tuning layers (L) and whether using the adversarial learning (AL).

Number of fine-tuning layers		One	Three	Four	Five
Our method	1 Trojan sample	**0.0 ± 0.0**	13.9 ± 2.13	16.6 ± 0.27	18.4 ± 1.06
	32 Trojan samples	-	0.0 ± 0.0	0.1 ± 0.01	12.5 ± 0.72
	320 Trojan samples	-	-	-	0.2 ± 0.12
Dual-key backdoor [159] (445 samples)		8.9	-	-	-

Table 7.2: The number of required Trojan samples to compromise fine-tuned models with varying numbers of fine-tuning layers was assessed, with ATA (↓). Our method necessitates significantly fewer Trojan samples compared to the existing method. Notably, a single shot of the Trojan sample was sufficient to compromise a model with one fine-tuning layer, and models with more fine-tuning layers were compromised using only a few shots of Trojan samples. In contrast to the dual-key backdoor method, which required 445 Trojan samples to achieve an ATA score of 8.9% for a single fine-tuning layer, our approach can compromise a VQA model with five fine-tuning layers using only 320 Trojan samples.

Robustness to Model Fine-Tuning

To measure the robustness of the proposed attack on varying fine-tuning networks, we employed the following metrics: (1) Main Task Accuracy (MTA), mea-

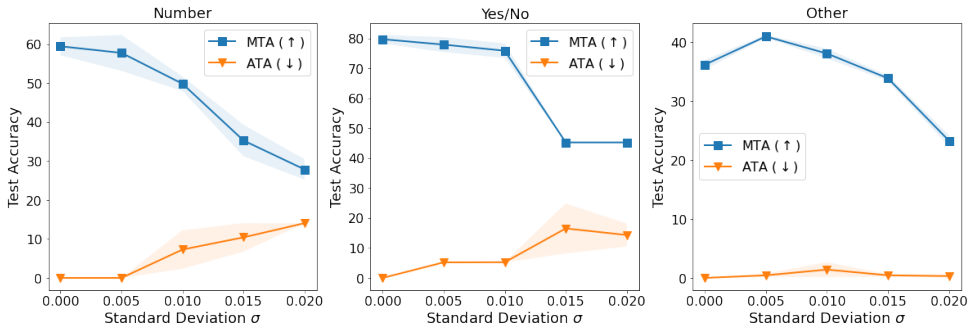


Figure 7.6: Differential Privacy (DP) with Gaussian noise of varying standard deviation σ. A trade-off exists between DP's defense efficacy and the model's performance on clean samples. As the degree of noise increases, the attack performance gradually weakened, however, the main task performance also decreased accordingly.

suring the fine-tuned model's test accuracy on the clean samples (the higher the better ↑), and (2) Attack Task Accuracy (ATA), measuring the fine-tuned model's test accuracy on the Trojan samples (the lower the better ↓).

The generated Trojans from the pretrained VQA model have limited effect on a fine-tuned model. To demonstrate the efficacy of the proposed adversarial learning in neuron activation space, we mounted the Trojan attack on the fine-tuned model, with and without leveraging the adversarial learning method. In particular, we substituted the layers following the perturbation layer with a fully-connected network with varying depths. The added layers feature 1024 neurons in width and the output of the network matches the number of total answer options, i.e., 3024. In Table 7.1, we demonstrate the attack performance in a fine-tuned model with a single fine-tuning layer, based on ATA and MTA. The benign MCAN's MTA was utilized as the upper bound for the model's benign performance. Comparing the ATA of the W/O AL and W/ AL ablations, the results indicate that the Trojans generated from the pretrained VQA model can compromise the fine-tuned model to a certain extent, particularly effective in the Other task. However, their effectiveness appears to be reduced in the Yes/No and Number tasks with fewer answer options, where the decision boundary is clearer making it more difficult to compromise. In contrast, the adversarial learning method significantly enhances the attack performance, reducing the overall ATA from 17.1% to 0%, proving effective in all three different VQA tasks. Moreover, the proposed attack does not significantly impact the performance of the compromised model on clean samples, maintaining an overall accuracy of 54.4% when the adversarial learning is employed.

Furthermore, we investigated the attack performance in a fine-tuned model with varying depths of the fine-tuning network, i.e., $m = \{1, 2, 3\}$. Figure 7.5 shows that without adversarial learning, the correlation between Trojan samples and malicious outputs of the fine-tuned model weakened in different settings of the fine-tuning network. In contrast, the adversarial learning method greatly enhanced the effectiveness of these Trojans, making them robust to black-box fine-tuning networks with varying depths. Note that the attacker had no prior knowledge of the fine-tuning network, thus generating Trojans directly using the fine-tuned model would be unfeasible. These results highlight the attack's robustness to model fine-tuning.

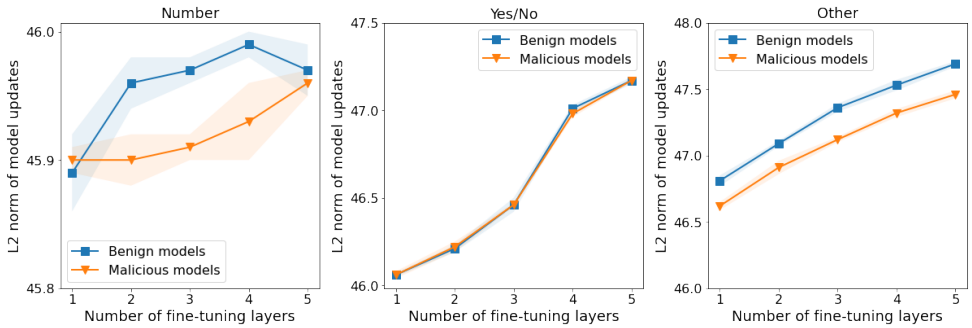


Figure 7.7: Norm Difference Estimation (NDE) measures the L2 norm of model updates during fine-tuning. The proposed attack results in a close magnitude (L2 norm) of the benign and malicious model updates. This similarity renders attack detection more challenging with the NDE method.

Sample Efficiency

The sample efficiency of an attack is measured by the number of Trojans needed for adversarial learning to compromise the fine-tuned model. We assumed that compromising the model requires the attack task accuracy (ATA) to be 0.0%. The sample efficiency of the proposed attack on models with varying depths is demonstrated in Table 7.2. Since an attack that can compromise a more complex fine-tuned model is usually effective in simpler ones, we show the results of the attack on the more complex models. For instance, using 32 Trojan samples, the attack on the fine-tuning network with three layers is also effective on a network with one layer.

Moreover, the results demonstrate that the proposed attack is a sample-efficient approach. The model with one fine-tuning layer was compromised using a single shot of the Trojan sample, and more fine-tuning layers were compromised using only a few shots. In comparison to the dual-key backdoor [159], which required 445 Trojan samples to achieve an ATA score of 8.9% for a single fine-tuning layer, our method compromised the model with five fine-tuning layers using only 320 Trojan samples.

Attack Performance Under Defenses

We investigated the Trojan attack performance under different conventional defenses methods, including the Differential Privacy (DP) [92] and Norm Difference Estimation (NDE) [94]. The DP aims to mitigate the adversarial learning by applying Gaussian noise $N(0, \sigma^2)$ with a standard deviation σ, to the weights of the fine-tuning network. For each layer of the fine-tuning network, the Gaussian noise was added during each batch time step of the fine-tuning process. Figure 7.6 illustrated that while DP alleviated the Trojan attack to some extent, there existed a tradeoff between its defense efficacy and the model's performance on clean samples. Nevertheless, DP could not eliminate the effect of the Trojan attack.

Moreover, the NDE involves a comparison of the L2 norm across a group of model updates, aiming to detect divergent instances. Typically, these divergent instances are characterized by malicious model updates during fine-tuning, exhibiting larger norms than benign updates [94]. To estimate the L2 norm of a model update, the updated weights across different fine-tuning layers were concatenated into a vector. Then, we computed the L2 norm of the weight vector

for the benign updates and malicious updates (with adversarial learning), respectively. The average L2 norms of the fine-tuning model updates based on five different seeds, were demonstrated in Figure 7.7. The results reveal that the proposed two-step method with the perturbation layer, results in a close magnitude (L2 norm) of the benign and malicious model updates. This similarity renders attack detection more challenging with the NDE method.

7.5 Conclusions

We proposed a novel instance-level multimodal Trojan attack on Visual Question Answering, leveraging the perturbation layer and adversarial learning in the activation space of two specific perturbation neurons. We conducted a comprehensive empirical evaluation using a diverse set of metrics, including stealthiness, variation, robustness to varying fine-tuning networks, and sample efficiency. Additionally, we demonstrated the efficacy of the proposed Trojan attack under conventional defense methods. In the future, our aim is to extend the investigation of the attack's efficacy to other multimodal learning architectures, such as self-supervised learning [81], and to devise effective countermeasures.

Chapter 8

Robust Learning with Local Supervision

This chapter consolidates my work on contrastive split learning for decentralized VQA [178].

Visual Question Answering (VQA) based on multi-modal data facilitates real-life applications such as home robots and medical diagnoses. One significant challenge is to devise a robust decentralized learning framework for various client models where centralized data collection is refrained due to confidentiality concerns. This work aims to tackle privacy-preserving VQA by decoupling a multi-modal model into representation modules and a contrastive module, leveraging inter-module gradients sharing and inter-client weight sharing. To this end, we propose Bidirectional Contrastive Split Learning (BiCSL) to train a global multi-modal model on the entire data distribution of decentralized clients. We employ the contrastive loss that enables a more efficient self-supervised learning of decentralized modules. Comprehensive experiments are conducted on the VQA-v2 dataset based on five SOTA VQA models, demonstrating the effectiveness of the proposed method. Furthermore, we inspect BiCSL's robustness against a dual-key backdoor attack on VQA. Consequently, BiCSL shows significantly enhanced resilience when exposed to the multi-modal adversarial attack compared to the centralized learning method, which provides a promising approach to decentralized multi-modal learning.

8.1 Limitations of FL

The deployment of multi-modal models in safety-critical applications, such as personal robots and healthcare, requires addressing robust architecture design. The collected vast amount of user data causes critical privacy concern. Unfortunately, few studies have focused on enhancing privacy for multi-modal models. For instance, Visual Question Answering (VQA) requires a large amount of data in both texts and images that indicate a wide range of personal interests. Decentralized machine learning, such as federated learning (FL), is one of the approaches to privacy-preserving VQA through the collaborative learning of different local models via weight sharing. Conventional FL methods [16] for VQA tasks have two main drawbacks: 1) models trained on separate client data are aggregated with model parameter sharing. However, sharing a complete model might lead to adversarial attacks [179]; 2) training a large model on resource-constrained client devices could be inefficient and impractical.

We aim to overcome the aforementioned challenges by proposing the Bidirectional Contrastive Split Learning (BiCSL) method. Different from FL which trains the entire model on a local device, BiCSL decouples a large-scale model into client components and cloud components. This avoids the potential misuse

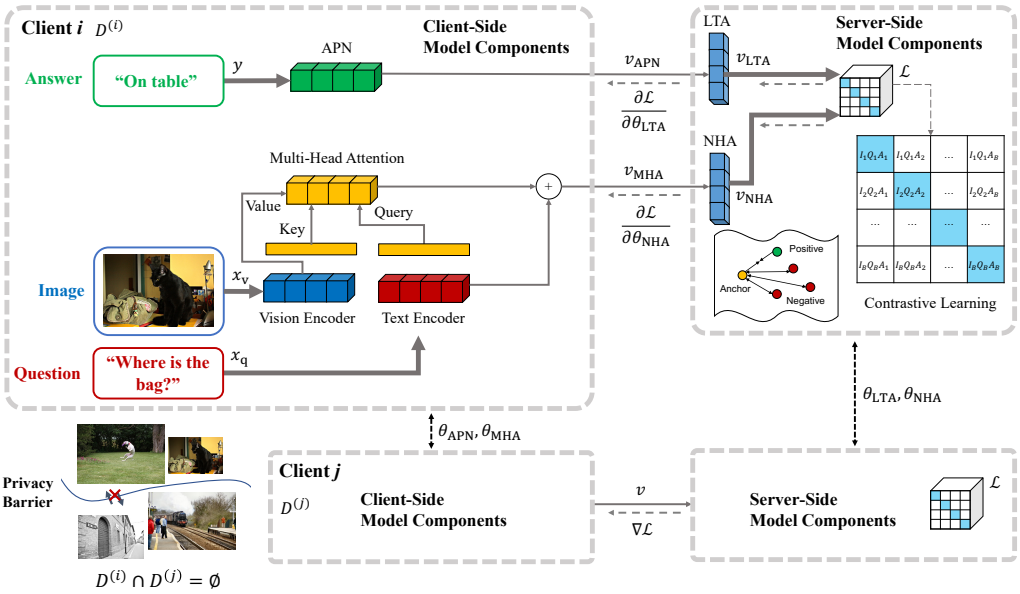


Figure 8.1: BiCSL for decentralized visual question answering consists of three main components: cross-modal representation learning (multi-head attention), an answer projection network (APN) for semantic understanding of answers, and two adapter networks (LTA and NHA) for contrastive learning of different model component outputs. BiCSL learns refined representations from different clients via inter-client weight sharing, while ensuring privacy protection via inter-module gradient sharing.

of the exposed model architecture by attackers, such as mounting a backdoor attack using the revealed architecture. BiCSL learns refined cross-modal representations from different clients via inter-module gradient sharing and inter-client weight sharing, without revealing either the user data or the model architecture (Figure 8.1). This is enabled by a self-supervised learning method to correlate various decentralized modules.

The main contributions of this work are as follows:

1) We propose a novel self-supervised split learning method for VQA, called Bidirectional Contrastive Split Learning (BiCSL). BiCSL trains a global model over the entire client data distribution without disclosing either training data or model architecture. This is the first study of self-supervised decentralized VQA.

2) This study demonstrates BiCSL's ability to tackle self-supervised learning of decentralized multi-modal data. BiCSL devises a contrastive learning method to align module activations encouraging similarity between relevant outputs while discouraging similarity between irrelevant ones.

3) An in-depth evaluation with a wide range of metrics including robustness to adversarial attack is conducted. The results show that our method could achieve competitive performance compared to a centralized method, while ensuring privacy protection and maintaining great performance even under adversarial attacks.

8.2 Related Work

Decentralized Machine Learning

Decentralized Machine Learning (DML) [14] encompasses methods such as Federated Learning (FL) [16, 180, 181], Split Learning (SL) [47], and Swarm Learning

Methods	Shared Data	Shared Model	Learning Framework	Loss Function
MMNas	✓	✓	Single fusion	Cross entropy
QICE	✓	✓	Single fusion	Contrastive loss
aimNet	×	✓	Federated Learning	Cross entropy
BiCSL (Ours)	×	×	Split Leaning	Contrastive loss

Table 8.1: Comparison of VQA methods: BiCSL does not require sharing training data or models. Different from previous work on decentralized VQA of the aimNet, BiCSL is a self-supervised method without the need for training labels.

[96]. These methods address privacy concerns by enabling collaborative learning without the need for centralized data storage. Although DML has been widely investigated for single-modality tasks, its application to multi-modal models is still limited. For example, aimNet [182] is a FL-based VQA framework, which utilizes fine-grained representations from various clients to enhance downstream tasks. Unfortunately, aimNet is a supervised method relying on annotated answer labels. Additionally, sharing client models during training increases its vulnerability to adversarial attacks.

Visual Question Answering

Multi-modal machine learning (MMML) [183, 81, 184, 185, 186] has been intensively studied to understand across different modalities of information. A specific task within MMML is Visual Question Answering (VQA) [167, 168], which involves answering natural language questions based on the contents of a presented image. Nevertheless, the vast majority of VQA studies so far rely on modality fusion methods where VQA is considered a centralized multi-class classification task. Moreover, previous studies usually do not consider the privacy concerns associated with centralized large-scale model training (Table 8.1).

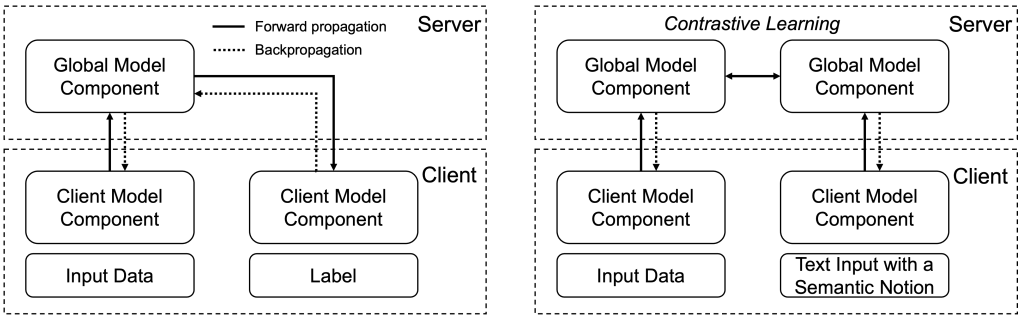
Contrastive learning is an alternative to the supervised method, which computes a cosine similarity matrix among all possible candidates of images and texts within a batch. For instance, Question-Image Correlation Estimation (QICE) [187] aims to train on relevant image and question pairs in VQA datasets to alleviate the language prior problem [188]. Nevertheless, QICE does not provide any guarantees regarding data or model privacy. In contrast, we found that contrastive learning could be used as a natural fit for privacy-preserving VQA by consolidating with decentralized learning techniques.

8.3 Methods

In this section, we delve into a comprehensive exploration of the proposed method's technical underpinnings. These include the incorporation of split learning for decentralized VQA, an answer projection network for enhanced understanding of semantic notions, a contrastive learning architecture for effective training on unlabeled client data, and inter-client weight sharing for local update aggregation.

8.3.1 Attention-Based VQA

Visual Question Answering (VQA) is a task that involves answering natural language questions based on the visual content of a given image. Typically, the VQA problem is approached as a supervised learning task with a predetermined



(a) Split Learning **(b) Bidirectional Contrastive Split Learning**

Figure 8.2: Conventional Split Learning vs. BiCSL: (a) Split Learning utilizes numeric one-hot vectors of answer labels for model training, based on a unidirectional process that requires sequential processing of components resulting in longer waiting time. (b) BiCSL employs lexical semantic notions of answer texts and a bidirectional process that enables concurrent processing of model components.

list of C potential answer options. Let f_{MHA} be the VQA model that takes as the input the pair of an image $x_v \in \mathbb{R}^V$ and a question $x_q \in \mathbb{R}^Q$ and outputs an answer $\hat{y} \in \{y_1, y_2, ..., y_C\}$ where $y_c \in \mathbb{R}^A$. A VQA model aims to predict the correct answer y given the input pair $(x_v, x_q) \in D$ where D is the dataset, i.e., $\hat{y} = \arg\max_y p(y|x_v, x_q; f_{\text{MHA}})$ where $p(\cdot|\cdot)$ is the conditional probability.

Moreover, we study a diverse set of VQA models that are based on the attention mechanism [7]. Cross-attention in VQA models enables improved refined representation learning from multi-modal data. At its simplest form, each head of a multi-head attention (MHA) maps a query and a set of key-value pairs to an output. Let $W^{\text{t}} \in \mathbb{R}^{Q \times M}$ be an encoder to process the text input x_q (such as LSTM [189] and Transformer [7]), and $W^{\text{v}} \in \mathbb{R}^{V \times P}$ be an encoder to process the vision input x_v (such as CNN [190] and MLP [191]). The linearly projected output of the text encoder is used as a query $Q \in \mathbb{R}^d \leftarrow W^{Q_i} W^{\text{t}} x_q$, which is compared with that of the vision encoder which serves as the key $K \in \mathbb{R}^d \leftarrow W^{K_i} W^{\text{v}} x_v$. Here, $W^Q \in \mathbb{R}^{M \times d}$ and $W^K \in \mathbb{R}^{P \times d}$ are linear transformations for the query and key. Then, the weighted sum of values $V \in \mathbb{R}^P \leftarrow W^{\text{v}} x_v$ could be formulated as follows

$$\text{h}^i(x_v, x_q) = \text{softmax}(\frac{W^{Q_i} W^{\text{t}} x_q (W^{K_i} W^{\text{v}} x_v)^T}{\sqrt{d}})\, W^{\text{v}} x_v,$$

$$\text{Multi-head}(x_v, x_q) = \text{Concat}(\text{h}^1, \ldots, \text{h}^H)\, W^O, \tag{8.1}$$

where W^O is a linear transformation for outputs, and H is the number of attention heads.

8.3.2 Decentralized VQA

To devise a decentralized method, training numerous VQA models on client devices is inefficient and impractical due to resource constraints on local devices. Intuitively, we could divide a complete model into client and server components. Then, by leveraging inter-module gradient sharing, the parameters of each component could be efficiently updated and synchronized. To this end, we consider dividing a VQA model into three components, i.e., a global component f_g and two client components $\{f_{c,1}, f_{c,2}\}$ (Figure 8.2.a). Then, we assume that K client

models are trained on their local datasets $D^{(k)}$, which consist of $N^{(k)}$ samples, represented as $\{(x_{v,j}, x_{q,j}, y_j)\}_{j=1}^{N^{(k)}}$. Here, $\cup_{k=1}^{K} D^{(k)} = D$, $D^{(i)} \cap D^{(j)} = \emptyset, \forall i \neq j$, and $\sum_{k=1}^{K} N^{(k)} = N$, where N is the total sample size. Furthermore, we make the assumption that the client models share the same architecture, and the division of the models are consistent across all clients. Training data sharing among clients is not possible due to confidentiality.

Then, the decentralized VQA method proceeds by iterating the following steps: (1) each client k computes the output of the component $f_{c,1}$ with $D^{(k)}$ and sends the output to the server, (2) the server forward-propagates the input with the global component f_g and sends back the output, (3) the probability distribution and the loss are computed by $f_{c,2}$ using ground-truths $\{y_j\}_{j=1}^{N^{(k)}}$, (4) the gradients $(\delta_k\theta_{c,1}, \delta_k\theta_{c,2}, \delta_k\theta_g)$ for each component of client k are then computed via an inverse path $f_{c,2} \rightarrow f_g \rightarrow f_{c,1}$, (5) after all clients complete local training, their update gradients are averaging aggregated for inter-client weight sharing, $\delta\theta_{c,1} = \frac{1}{K}\sum_{k\in K} \delta_k\theta_{c,1}$, $\delta\theta_{c,2} = \frac{1}{K}\sum_{k\in K} \delta_k\theta_{c,2}$, $\delta\theta_g = \frac{1}{K}\sum_{k\in K} \delta_k\theta_g$, and (6) the aggregated updates are distributed to clients for the update of their local components. We repeat the process above until a global training goal is achieved. This architecture enables clients to train individual models without sharing local data or models, while harnessing the acquired knowledge from other clients through activation and gradient sharing.

8.3.3 Bidirectional Contrastive Split Learning

Though the aforementioned supervised decentralization of VQA enhances privacy of local model training, there exist two main drawbacks. First, the semantic understanding of answers is often misaligned with the inputs due to the image and question pairs are labeled with numeric ids of answer texts. Second, the computational time could be substantial due to the interactive activation and gradient sharing among components. To this end, we propose a self-supervised decentralization method for VQA, called Bidirectional Contrastive Split Learning (BiCSL). BiCSL leverages contrastive learning-based component alignment to enhance the correlation between visual and language contents and improve efficiency of activation and gradient sharing.

Answer Projection and Adapter Networks

An Answer Projection Network (APN) f_{APN} (Figure 8.4.c) aims to project a lexical answer y into a feature vector $v_{\text{APN}} \in \mathbb{R}^S$. APN comprises two main components: a preprocessing process and the word embedding of GloVe [174] to transform the question text into a fixed-size vector representation. The resultant vector is subsequently fed through a linear projection layer.

Moreover, two adapter networks (Figure 8.4.a, 8.4.b) are employed to project the outputs of client components into a shared dimension, where a Nonlinear Head Adapter (NHA) is applied to tackle more complex representations while a Linear Tail Adapter (LTA) is used to process simpler ones. In particular, to tailor a VQA model for contrastive learning, we replace its output layer with the NHA network f_{NHA}. The NHA projects the learned cross-modal representations into $v_{\text{NHA}} \in \mathbb{R}^S$. We then use the LTA network f_{LTA} to project the learned answer representations from the APN v_{APN} into $v_{\text{LTA}} \in \mathbb{R}^S$. Note that v_{LTA} and v_{NHA} have the same dimension of S.

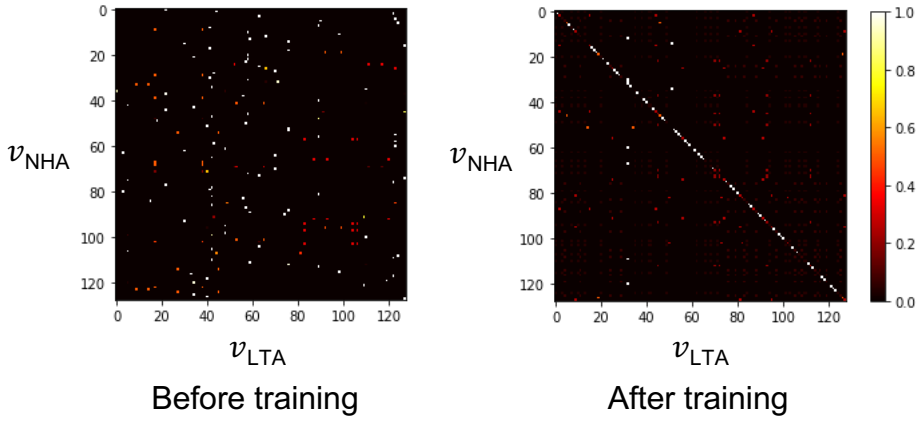


Figure 8.3: Measured dot product similarity between any two representations in one batch before and after training with BiCSL. The similarity scores are optimized such that the representations of positive pairs have a higher score while that of the negative pairs have a lower score.

Contrastive Learning of Model Components

We employ the Information Noise Contrastive Estimation (InfoNCE) loss [192] to disentangle similar (positive) and dissimilar (negative) pairs of data points. Model component activations are aligned for the positive pairs while being discouraged for the negative pairs (Figure 8.3). Notably, we use the NHA and LTA outputs for the same inputs as the positive pairs, i.e., $\{(v_{\text{NHA},j}, v_{\text{LTA},j})\}_{j=1}^{B}$ where B is the batch size. On the contrast, given the NHA output $v_{\text{NHA},i}$, the irrelevant LTA outputs $\{v_{\text{LTA},j} | j \neq i\}_{j=1}^{B}$ within one batch are employed as the negative pairs. Consequently, we devise the loss $\mathcal{L}$ for the contrastive learning of model components as follows

$$\mathcal{L} = -\sum_{i=1}^{B} \log \frac{\exp(v_{\text{NHA},i} \cdot v_{\text{LTA},i}/\tau)}{\sum_{j=1}^{B} \mathbb{1}_{[j \neq i]} \exp(v_{\text{NHA},i} \cdot v_{\text{LTA},j}/\tau)}, \tag{8.2}$$

where τ is the temperature parameter to ensure the output is appropriately scaled to the data distribution, and $\mathbb{1}_{[j \neq i]}$ is an indicator function: 1 if $j \neq i$, 0 otherwise.

Additionally, the proposed framework enables a parallel processing of model components improving the efficiency of decentralized VQA. In the decentralized VQA based on split learning, the activation and gradient sharing between the client and the server is unidirectional (Figure 8.2.a). Each component needs to process the input data in subsequent which could largely increase the waiting time during training. In contrast, in our architecture, all layer activations are sent from clients to the server while all gradients are sent from the server to clients. As a result, clients could utilize their local components concurrently while computing activations or gradients, without waiting for the computation of the previous component (Figure 8.2.b).

Local Update Aggregation

Every epoch t, aggregating model updates $\theta_{t+1}^{(k)} - \theta_t^{(k)}$ from different clients $k \in \{1, 2, \ldots, K\}$ enhances the generality of the aggregated global model. Due to sending client updates to the server for aggregation could expose the model architecture, we employ a dual-server parameter aggregation approach that leverages a second auxiliary server for the aggregation of client updates (APN and MHA).

Algorithm 5 Bidirectional Contrastive Split Learning (BiCSL)

T: number of rounds
E: number of local epochs
η: learning rate
for each round $t = 1, 2, \ldots, T$ **do**
  for each client $k \in \{1, 2, \ldots, K\}$ in parallel **do**
    for $\theta \in \{\theta_{\text{APN}}, \theta_{\text{MHA}}, \theta_{\text{NHA}}, \theta_{\text{LTA}}\}$ **do**
      $\theta_t^{(k)} \leftarrow \theta_t$
    end for
    for each local epoch $e = 1, 2, \ldots, E$ **do**
      $v_{\text{MHA},t,e}^{(k)} = f_{\text{MHA}}(\theta_{\text{MHA},t,e}^{(k)}, (x_v^{(k)}, X_q^{(k)}))$
      $v_{\text{APN},t,e}^{(k)} = f_{\text{APN}}(\theta_{\text{APN},t,e}^{(k)}, Y^{(k)})$
      $\delta_{\text{NHA},e}^{(k)}, \delta_{\text{LTA},e}^{(k)} = \text{Server}(v_{\text{MHA},t,e}^{(k)}, v_{\text{APN},t,e}^{(k)})$
      $\theta_{\text{MHA},t,e+1}^{(k)} \leftarrow \theta_{\text{MHA},t,e}^{(k)} - \eta \cdot \frac{\partial \delta_{\text{NHA},e}^{(k)}}{\partial \theta_{\text{MHA},t,e}^{(k)}}$
      $\theta_{\text{APN},t,e+1}^{(k)} \leftarrow \theta_{\text{APN},t,e}^{(k)} - \eta \cdot \frac{\partial \delta_{\text{LTA},e}^{(k)}}{\partial \theta_{\text{APN},t,e}^{(k)}}$
    end for
  end for
  for $\theta \in \{\theta_{\text{APN}}, \theta_{\text{MHA}}, \theta_{\text{NHA}}, \theta_{\text{LTA}}\}$ **do**
    $\theta_{t+1} = \frac{1}{K} \sum_{k \in K} \theta_{t,E+1}^{(k)}$
  end for
end for

function Server($v_{\text{MHA},t,e}^{(k)}, v_{\text{APN},t,e}^{(k)}$)
$v_{\text{NHA},t,e}^{(k)} \leftarrow f_{\text{NHA}}(v_{\text{MHA},t,e}^{(k)})$
$v_{\text{LTA},t,e}^{(k)} \leftarrow f_{\text{LTA}}(v_{\text{APN},t,e}^{(k)})$
$\mathcal{L} = -\sum_{i=1}^{B} \log \frac{\exp(v_{\text{NHA},i} \cdot v_{\text{LTA},i}/\tau)}{\sum_{j=1}^{B} \mathbb{1}_{[j \neq i]} \exp(v_{\text{NHA},i} \cdot v_{\text{LTA},j}/\tau)}$
$\delta_{\text{NHA},e}^{(k)} = \frac{\partial \mathcal{L}}{\partial \theta_{\text{NHA},t,e}^{(k)}}$

$\delta_{\text{LTA},e}^{(k)} = \frac{\partial \mathcal{L}}{\partial \theta_{\text{LTA},t,e}^{(k)}}$

$\theta_{\text{NHA},t,e+1}^{(k)} \leftarrow \theta_{\text{NHA},t,e}^{(k)} - \eta \cdot \delta_{\text{NHA},e}^{(k)}$
$\theta_{\text{LTA},t,e+1}^{(k)} \leftarrow \theta_{\text{LTA},t,e}^{(k)} - \eta \cdot \delta_{\text{LTA},e}^{(k)}$
return $\delta_{\text{NHA},e}^{(k)}, \delta_{\text{LTA},e}^{(k)}$ to client k

The aggregation of the server updates (NHA and LTA) is performed on the main server. We use an averaging aggregation method formulated as follows

$$\delta\theta_t = \frac{1}{K} \sum_{k \in \{1,2,\ldots,K\}} (\theta_{t+1}^{(k)} - \theta_t^{(k)}), \tag{8.3}$$

where θ is the parameters of a model component from $\{\theta_{\text{APN}}, \theta_{\text{MHA}}, \theta_{\text{NHA}}, \theta_{\text{LTA}}\}$.

The proposed BiCSL method is demonstrated in Algorithm 5.

8.4 Experiments

In this section, we provide a detailed description of the datasets, model architectures, and metrics used in the experiments. An extensive empirical evaluation is performed based on five SOTA VQA models. Furthermore, we investigate BiCSL's robustness to a sophisticated dual-key backdoor attack on VQA models, comparing its performance against different methods. The results demonstrate that BiCSL achieves competitive performance to the centralized method and remains effective even under the mounted attack.

Dataset

Our method was evaluated on the benchmark dataset VQA-v2 [175] with varying partitioning configurations for decentralized VQA. VQA-v2 covers 82.8k images and 443.8k questions for training and 40.5k images and 214.4k questions for validation. The images are from the COCO dataset [176] with a size of 640×480. Depending on the client number, we separated the training dataset into several non-overlapping subsets as client datasets. Moreover, we used the entire validation dataset to evaluate the performance of the aggregated global model.

VQA Models

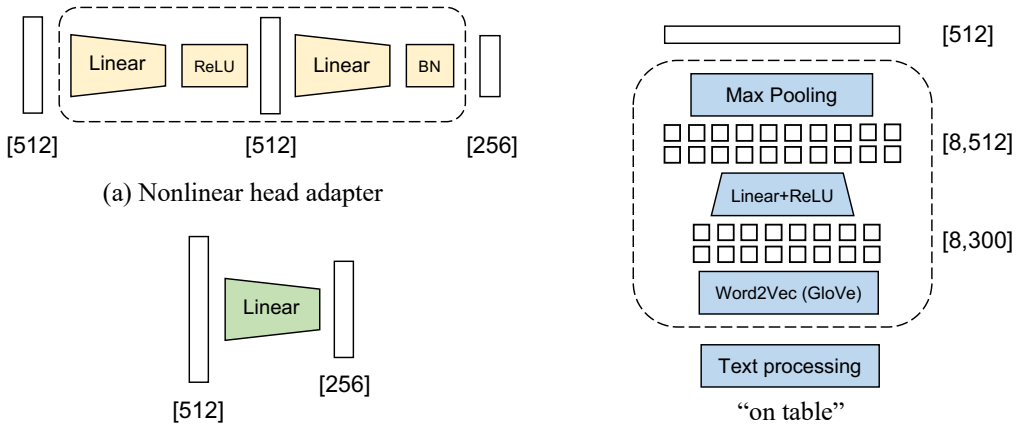
The following VQA models were studied: (1) Multi-modal Factorized Bilinear (MFB) [193] combines multi-modal features using an end-to-end network architecture to jointly learn the image and question attention, (2) Bottom-Up and Top-Down attention mechanism (BUTD) [167] enables attention to be calculated at the level of objects and other salient image regions. The bottom-up mechanism based on Faster R-CNN proposes image regions, while the top-down mechanism determines feature weightings, (3) Bilinear Attention Networks (BAN) [170] considers bilinear interactions among two groups of input channels and extracts the joint representations for each pair of channels, (4) Multi-modal neural architecture search (MMNas) [168] uses a gradient-based algorithm to learn the optimal architecture, and (5) Modular Co-Attention Network (MCAN) [169] consists of Modular Co-Attention layers cascaded in depth where each layer models both the self-attention and the guided-attention of the input.

We evaluated the model performance with three different seeds and reported the mean and standard deviation. The VQA models were implemented using PyTorch with their default hyperparameters. The experiments were conducted on four A100 GPUs with 40GB memory. The code would be made publicly available.

Architecture and Hyperparameters

In the APN, the GloVe [174] trained on Common Crawl was used to convert the answer texts with a maximum word of eight into $\mathbb{R}^{8\times300}$, padded with zero vectors. Then, a fully-connected (FC) layer followed by the ReLU activation projected the representations into $\mathbb{R}^{8\times512}$. Finally, a Max Pooling layer was employed producing 512-dimension vectors. The LTA consists of a FC layer that has an output dimension of 256. The NHA consists of a FC layer that has an output dimension of 512 followed by the ReLU activation and another FC layer with an output dimension of 256 followed by batch normalization (Figure 8.4).

The selection of hyperparameters was performed through the grid search. We used a batch size of 128, a total epoch of 20 (693.4k steps), the Adam optimizer



(a) Nonlinear head adapter

(b) Linear tail adapter

(c) Answer projection network

Figure 8.4: The model architectures of the nonlinear head adapter (NHA), the linear tail adapter (LTA), and the answer projection network (APN). The number of neurons in each layer is indicated by the numbers within square brackets.

with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, and a linear warmup of 10K steps using an initial learning rate of 0.0001 and a decay rate of 0.2 at the epoch 10 and 15. For the InfoNCE loss, a temperature of 0.07 was employed as in [194]. Depending on the VQA model, each trial took approximately five to nine hours.

Metric

Measuring model performance is challenging due to the lack of a discriminative model that infers the class of the input. In this regard, BiCSL embeds the semantic meanings of answers in the APN, converting text to numerical vectors based on semantic text distances. Consequently, if two answers are semantically similar, then the learned representations of the APN would also have a high similarity. In particular, to evaluate prediction accuracy, we measure the product similarity between the cross-modal representations v_{NHA} of an input pair (x_v, x_q) from the hold-out validation dataset D_{val}, and the representations $v_{\text{LTA},c}$ of C answer options $y_c \in y_1, y_2, \ldots, y_C$. Here, $v_{\text{LTA},c}$ represents the representation of answer option y_c. The answer with the highest similarity to the input is selected as the predicted answer $\hat{y}$. We formulate the proposed metric as follows

$$\text{ValAcc} = \frac{\sum_{(x_v,x_q,y)\in D_{\text{val}}} \mathbb{1}\{\arg\max_c (v_{\text{NHA}} \cdot v_{\text{LTA},c}) = y\}}{|D_{\text{val}}|}. \tag{8.4}$$

8.4.1 Empirical Results

Contrastive Learning-Based VQA

Extensive experiments with five SOTA VQA models were conducted. Moreover, for the MMNas and MCAN, we further investigated their variants with different model sizes including MMNas-small (MMNas-s), MMNas-large (MMNas-l), MCAN-small (MCAN-s), and MCAN-large (MCAN-l), which resulted in a total of seven different models. The detailed architecture designs of these models followed the settings in [169, 168].

The proposed method's performance was evaluated based on Eq. 8.4. For each triplet in the validation set, we input the image and question pair to the model, then use the output representation of the nonlinear head adapter to measure the

VQA Models	**Contrastive Learning** (%)			
	Overall	Yes/No	Number	Other
BAN	36.23 ± 0.53	66.90 ± 0.71	12.71 ± 0.32	19.11 ± 0.47
BUTD	45.08 ± 0.64	75.82 ± 0.82	29.27 ± 0.53	25.86 ± 0.41
MFB	46.98 ± 0.58	73.95 ± 0.77	32.81 ± 0.49	30.20 ± 0.38
MCAN-s	53.18 ± 0.61	81.06 ± 0.78	41.95 ± 0.46	34.93 ± 0.35
MCAN-l	53.32 ± 0.55	81.21 ± 0.73	42.66 ± 0.39	34.90 ± 0.42

Table 8.2: Performance comparison between VQA models based on the contrastive learning (centralized) method.

VQA Models	**BiCSL** (%)			
	Overall	Yes/No	Number	Other
BAN	35.11 ± 0.68	63.84 ± 0.54	11.06 ± 0.25	19.61 ± 0.36
BUTD	40.96 ± 0.76	66.98 ± 0.62	13.34 ± 0.35	28.74 ± 0.47
MFB	42.43 ± 0.72	68.65 ± 0.58	23.33 ± 0.41	27.52 ± 0.52
MCAN-s	48.42 ± 0.68	74.93 ± 0.54	30.88 ± 0.37	32.89 ± 0.49
MCAN-l	48.44 ± 0.62	77.44 ± 0.48	30.72 ± 0.32	32.01 ± 0.44

Table 8.3: Performance comparison between VQA models based on the proposed BiCSL (decentralized) method. BiCSL achieves competitive performance to the centralized VQA method using a decentralized learning framework.

similarity scores with the representations from the linear tail adapter of all the answer options. The prediction is made based on the answer with the highest similarity score.

Table 8.2 shows the evaluation results of the contrastive learning-based method. The benefit of this contrastive learning-based approach is twofold: it does not require manual labeling of answer data to train the model, and its combination with split learning is a natural fit for a more efficient decentralized VQA. Furthermore, by comparing the results of different VQA models trained with contrastive learning, several architectures outperformed the others. BAN showed the worst performance, particularly in the task of counting numbers (Number). MMNas-l showed the best overall performance of 53.82%, outperforming the other models for the tasks of counting numbers and answering the image contents (Other). MCAN-l performed the best in the Yes/No questions. The results demonstrated that the contrastive learning-based method could be effectively adapted to different existing VQA models.

Decentralized VQA with BiCSL

To evaluate the efficacy of our method, the training set was randomly divided into two non-overlapping subsets, as the local datasets of two clients. These clients shared the same model component architecture but could not share data due to confidentiality. The performance was evaluated on the aggregated global model at each round based on the entire validation dataset. The numerical results are shown in Table 8.3. We compared the performance of different model architectures for decentralized VQA. MMNas-l outperformed the other models overall, while MCAN-l showed the best performance in the Yes/No questions.

Moreover, compared to the overall accuracy of 53.82% of the MMNas-l model trained on the centralized dataset, BiCSL obtained an overall accuracy of 49.89%.

Though there exists a small trade-off between model performance and using BiCSL for privacy protection, BiCSL enables clients to train over the entire data distribution without sharing either their local data or models. It could greatly benefit model training in situations where privacy is a major concern. The empirical results showed that BiCSL could achieve competitive performance to the centralized VQA method.

Statistical Paired T-test

A statistical paired t-test [195] measures the significance of the difference between the performance of the centralized VQA method in Table 8.2 and the proposed BiCSL method in Table 8.3. With a significance level of 0.05 and a degree of freedom $n-1$ where $n=7$ is the number of VQA models, we could compute a p-value of 2.477. Based on the guarantee of the paired t-test [195], if $t=1.357$ falls within the range of the p-value [-2.477, 2.477], there is no significant difference in the performance between the two methods. Consequently, the statistical result showed that BiCSL achieved competitive performance on these VQA tasks compared to the centralized method.

8.4.2 Attention Map Visualization

The attention mechanism in a VQA model learns the relative importance of visual representations at different spatial locations with respect to a given question. The attention weights are updated such that the visual regions more relevant to the question are emphasized. We computed the attention weights from the learned cross-attention module in the decentralized MCAN-s model with BiCSL and visualized the attention maps based on the approach in [171]. The visualization results are shown in Figure 8.5.

8.4.3 Robustness to Trojan Attacks

To evaluate the robustness of BiCSL against adversarial attacks, we mounted a dual-key backdoor attack [179] on different VQA models based on the single fusion method, split learning, and the proposed BiCSL method, respectively. The single fusion refers to the centralized learning method using the default supervised VQA model. In particular, an untargeted multi-modal Trojan attack that embeds triggers into both the vision and text training data aims to compromise the model to output incorrect predictions (Figure 8.6). Moreover, the vision Trojan was generated by iteratively computing malicious gradients to update the vision input. Here, we refer to [179] for the detailed settings of the mounted attack. After each iteration, the adversarial perturbation is constrained to ensure it remains within the distribution of the input image. Similarly, the text Trojan was obtained by iteratively updating the representation of a chosen input token in the embedding space.

The experiments were conducted on the VQA-v2 dataset using different learning methods and the MCAN-s [169] model. The empirical results showed that BiCSL maintained much stronger robustness against such attacks than the single fusion and split learning methods (Figure 8.7), demonstrating its potential for secure deployment in real-world scenarios. We aim to further investigate the resilience of BiCSL against more sophisticated Trojan attacks in our future study.

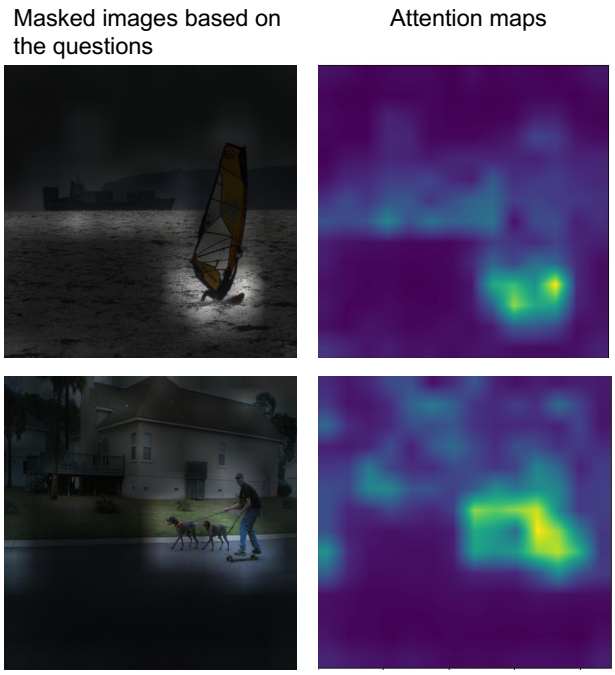


Figure 8.5: The attention mechanism identifies the important regions that are relevant to answering the given question. These attention maps were generated by computing the weight matrix from the attention mechanism. The top images are associated with a question asking about the color of the sail, and the sail is highlighted. The bottom images are associated with a question asking about the number of dogs the man is walking, and the dogs are highlighted.

Figure 8.6: Samples of the generated dual-key Trojans. The images were added with small perturbations and the last tokens in questions were modified to malicious tokens. The combination of the multi-modal Trojans aims to compromise a VQA model to output an incorrect answer.

8.4.4 Computational Cost Estimation

The benefits for time reduction through the bidirectional architecture are demonstrated. Let the time cost of the forward propagation and the backpropagation for each component of split learning be $\{T_f^1, T_b^1\}$, $\{T_f^g, T_b^g\}$, and $\{T_f^2, T_b^2\}$. Then, the approximate total time cost for one epoch's training will be the sum of these values. Then, suppose that each component takes the same time to train in BiCSL, then the approximate total time cost of it is $(\max(T_f^1, T_f^2) + T_f^g + T_b^g + \max(T_b^1, T_b^2))$. Therefore, the final reduced computational cost will be

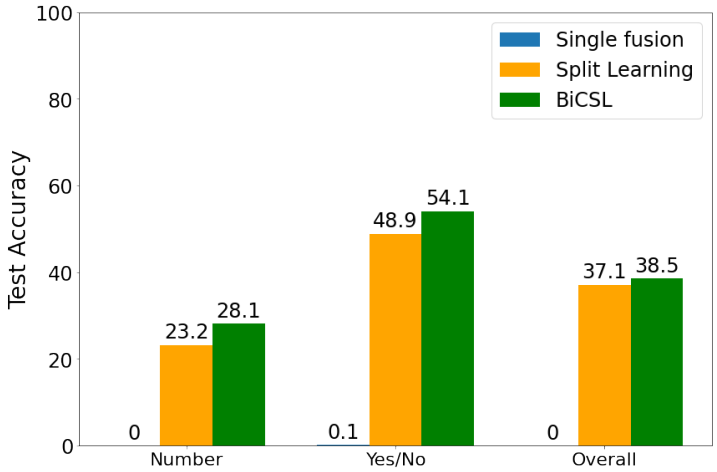


Figure 8.7: VQA task performance under the Trojan attack. BiCSL maintained much stronger robustness against such attacks than the single fusion and split learning methods. Compared to the split learning method, BiCSL leverages the self-supervised learning of input data, which increases the difficulty of generating effective Trojans for the attack. Moreover, compared to the single fusion method that exposes the entire model, BiCSL leverages a decentralized learning method with inter-module gradient sharing to avoid sharing the entire VQA model. As a result, the incomplete information on the target model degraded the successability of the attack in generating effective Trojans that compromise the model.

$(T_f^1 + T_b^1 + T_f^2 + T_b^2 - \max(T_f^1, T_f^2) - \max(T_b^1, T_b^2))$.

8.5 Discussion

We proposed a decentralized VQA method called BiCSL, which effectively learns refined cross-modal representations by aligning model components based on contrastive learning and aggregating knowledge from different clients. Extensive experiments on the VQA-v2 dataset demonstrated BiCSL's efficacy across various VQA models and its robustness to the existing multi-modal adversarial attack. In the future, we aim to further investigate BiCSL's robustness against more sophisticated adversarial attacks and leverage approaches such as differential privacy [149] to safeguard the activation and gradient sharing between components. We hope that this work would motivate future research in robust learning for decentralized multi-modal models.

Chapter 9

Conclusions

9.1 Impact of the Research

To develop lifelong learning machines, one of the main challenges investigated in this thesis is knowledge reusability. I aim to show pieces of evidence that localized learning, comprising a set of expert modules or inductive biases, can benefit the global generalization of the trained model. Notably, we delve into the distributional shift problems in decentralized neural networks and relational reasoning tasks in Transformer models. I demonstrated that leveraging a shared workspace among neural modules can induce competition among different knowledge experts, and extensive evaluations showed its advantages in out-of-distribution performance, computational and communication efficiency, and scalability in large-scale training.

Distributional shifts have usually been studied in areas such as transfer learning and few-shot learning. However, previous studies often rely on impractical assumptions of the observable landscape of possible tasks. It is infeasible to involve all tasks that an agent may encounter in its lifetime when designing a model. In a departure from conventional studies, our goal is to create learning machines for systematic generalization that build reusable knowledge components and their interconnections through the decomposition of global tasks into local tractable tasks and ongoing interactions with external environments. For instance, in the task of learning to drive a car, our objective is not to achieve direct mastery of driving but, instead, to distill and reuse modules that can assist in driving tasks, learned from experiences with other tasks. Indeed, the ability to discover a pathway that connects diverse prior knowledge to acquire a new skill in the shortest time possible serves as a defining characteristic of human intellectual capabilities.

Furthermore, the use of memory systems in model training facilitates lifelong learning, counteracting forgetting and enhancing performance. Prior research, such as Neural Turing Machines [1], demonstrated the advantages of memory-enhanced systems. Different from previous studies focused on the memory replay of observations, this thesis explored the associative memory-enhanced Transformer model. Different from the conventional functionality of memory, we considered learning and storing distinct inductive biases in the memory, which guide attention over the input from different perspectives. The upscaled memory resembles attractors within dynamic systems of associative memory to recall specific knowledge from the past, thus enhancing the model's performance.

This thesis presented different approaches to constructing an artificial Global Workspace, in conjunction with its extension through a working memory system. The approaches we put forward can be considered as steps towards artificially constructing the Global Workspace and addressing the challenges associated with

building such a system. In particular, Chapters 3 and 4 explored strategies for decentralized and modular learning systems with various expert neural modules. Chapter 5 introduced an architecture that emulates the functionality of working memory and associative memory in the brain. While this thesis does not encompass the entire spectrum of Global Workspace properties, we deem that the proposed Associative Transformer in Chapter 5 serves as a functional prototype of the artificial Global Workspace.

Additionally, new challenges arise from the capabilities of localized learning in local parameters, gradients, or representations sharing. AI Security studies often focus on attacks in a centralized model, compromising global objectives, which is insufficient for localized learning models with various local objectives. An attacker can potentially inject perturbations into a local model, sabotaging its local objective. However, knowledge reuse through direct weight or gradient sharing in localized learning systems can introduce greater susceptibility, compromising other models' local objectives and even the system's overall performance. In particular, in Chapters 6 and 7, two types of adversarial attacks based on perturbation on local modules were investigated. Then, Chapter 8 demonstrated Bidirectional Contrastive Split Learning for privacy-preserving decentralized multi-modal learning.

In conclusion, all of these seemingly diverse contributions are interconnected by the central theme of this thesis: the acquisition and reuse of localized, contextual knowledge to facilitate the generalization and lifelong learning abilities of artificial neural networks, with a particular emphasis on constructing an artificial counterpart to the Global Workspace.

9.2 Recommendations for Further Research

9.2.1 Global Workspace with Long-Term Memory

The memory systems can be divided into sensor memory, working memory, and long-term memory (semantic memory in neocortex and episodic memory in the hippocampus). The interplay between working memory (in the Global Workspace) and long-term memory has shown strong correlations for human intelligence. The memory formation in the human brain is largely supported by the Complementary Learning Systems (CLS) theory, which postulates that intelligent agents require two learning systems, which are instantiated in mammals through the neocortex and the hippocampus [196]. The two types of memory systems render fast and slow learning of knowledge. In particular, the hippocampus quickly learns the specifics of individual experiences while the neocortex gradually acquires structured knowledge representations from these accumulated experiences. The memory consolidation process from the hippocampus to the neocortex happens during our sleep.

The catastrophic forgetting problem in lifelong learning refers to the fact that models learning a new task usually degrade on previous tasks due to negative interference between old and new knowledge. The challenge lies in the requirement for a sustained, enduring memory shortage to support lifelong learning. In this regard, though large language models are adept at learning from huge data, they usually have short memory of recent conversations. The length of the conversation history has a limitation, typically in terms of the number of tokens it can use as the input. If the conversation becomes too long and exceeds the model's token limit, earlier messages are truncated. To address the memory capacity limit issue, there are approaches to scaling up the input length, which unfortunatelly, result

in exponentially increasing model complexity due to the quadratic computational cost of self-attention.

Using a long-term memory appears to be a better solution, for example, by directly querying the large language models for a history summary [197, 198, 199] or learning representations for long-term storage [200]. For instance, MemoryBank [197] enabled models to recall relevant memories, continually evolve through continuous memory updates, and adapt to a user's personality over time by summarizing information from previous interactions. Reflexion [198] utilized verbal reinforcement to enable agents to learn from prior failings. Reflexion converted scalar feedback from the environment into the form of a textual summary and added it as additional context for the next episode. Moreover, Language Models Augmented with Long-Term Memory [200] leveraged a decoupled network architecture featuring a frozen memory encoder and an adaptive residual memory retriever.

9.2.2 Artificial Mind

The Global Workspace Theory explored in this thesis finds resonance with the neuroscience study of consciousness. Recent research in machine learning has shown an increasing interest in studies of consciousness in artificial neural networks [23]. I believe the extension of studies demonstrated in this thesis could potentially contribute to the literature on understanding consciousness and building learning machines with conscious-appearing behaviors.

The study of consciousness involves phenomenal consciousness, which is the subjective experience we undergo during an event, and access consciousness, referring to representations becoming available in the brain for cognitive processing [201]. Notably, in the study of phenomenal consciousness, the 'hard problem' was introduced by Chalmers [202], referring to a specific aspect of consciousness that is particularly challenging to explain within the framework of physical science. The hard problem is concerned with what it is like to experience a particular sensation or emotion and the physical processes that give rise to subjective experience.

The discussion on consciousness in artificial neural networks is mainly related to access consciousness, extensively studied through the Global Workspace Theory and its extensions. Budson et al. [203] demonstrated that within the brain, most activities are carried out unconsciously by the collective input of diverse neural modules. Consciousness typically comes into play only when a specific event deviates significantly from what is needed to initiate conditioned responses or requires unusual responses, for instance, when driving a car. I have shown a strong correlation between localized learning and the Global Workspace Theory in Chapter 5. Notably, within the Global Workspace, neural modules compete to enter the workspace through a bottleneck with restricted capacity, where the contents in the workspace recurrently form a linear stream of experiences.

Another relevant theory of consciousness is the Attention Schema Theory [23], which involves a higher order of attention in working memory. Capturing the present state of attention enables the comprehension of attention's impacts. For example, the World Model [204] aims to create a more adaptive reinforcement learning (RL) agent based on the Attention Schema. An RL agent equipped with a world model understands its own state, predicts the consequences of its actions, and adapts to changes in the environment based on inner simulation of actions and their consequences [204]. Furthermore, in a multi-agent environment, the Attention Schema shows benefits in coordinating different local agents to

collaborate and achieve a shared task goal. Each agent can be considered as a localized learning agent in a partially observable environment [205]. Future studies include the investigation of the relation between higher-order attention and localized learning through multi-agent simulation in RL tasks.

The interdisciplinary study of neuroscience and machine learning has shown a path toward more biologically-plausible learning systems based on localized learning. It holds the promise of building machines with better attention-guided generalization and lifelong learning capabilities.

References

[1] David Mumford. The convergence of ai code and cortical functioning. arXiv:2010.09101, 2020.

[2] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psyhological Review*, 65(6):386–408, 1958.

[3] R. Schalkoff. Pattern recognition : statistical, structural and neural approaches. 1992.

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, and et al. Attention is all you need. *NeurIPS*, 2017.

[8] Sarthak Mittal, Alex Lamb, Anirudh Goyal, and et al. Learning to combine top-down and bottom-up signals in recurrent neural networks with attention over modules. In *ICML*, 2020.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, and et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[10] Jia Deng, Wei Dong, Richard Socher, and et al. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint*, 2015.

[12] John J. Hopfield. Hopfield network. *Scholarpedia*, 2(5):1977, 2007.

[13] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, and et al. Hopfield networks is all you need. In *ICLR*, 2021.

[14] Yuwei Sun, Hideya Ochiai, and Hiroshi Esaki. Decentralized deep learning for multi-access edge computing: A survey on communication efficiency and trustworthiness. In *IEEE Transactions on Artificial Intelligence*, 2021.

[15] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, and et al. Federated learning: Strategies for improving communication efficiency. In *NeurIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[16] Brendan McMahan, Eider Moore, Daniel Ramage, and et al. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*, 2017.

[17] Bernard J. Baars. *A Cognitive Theory of Consciousness*. Cambridge University Press, 1988.

[18] Changeux J. P. Dehaene S., Kerszberg M. A neuronal model of a global workspace in effortful cognitive tasks. In *National Academy of Sciences*, 1998.

[19] Rufin VanRullen and Ryota Kanai. Deep learning and the global workspace theory. arXiv:2012.10390, 2020.

[20] Arthur Juliani, Kai Arulkumaran, Shuntaro Sasai, and Ryota Kanai. On the link between conscious function and general intelligence in humans and machines. *Transactions on Machine Learning Research*, 2022.

[21] E. Awh, E.K. Vogel, and S.-H. Oh. Interactions between attention and working memory. *Neuroscience*, 2006.

[22] Adam Gazzaley and Anna C. Nobre. Top-down modulation: bridging selective attention and working memory. *Trends in Cognitive Sciences*, 2011.

[23] Patrick Butlin, Robert Long, Eric Elmoznino, and et al. Consciousness in artificial intelligence: Insights from the science of consciousness. arXiv:2308.08708, 2023.

[24] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, and et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

[25] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, and et al. Scaling vision with sparse mixture of experts. In *NeurIPS*, 2021.

[26] Simiao Zuo, Xiaodong Liu, Jian Jiao, and et al. Taming sparsely activated transformer with stochastic experts. In *ICLR*, 2022.

[27] James Urquhart Allingham, Florian Wenzel, Zelda E. Mariet, and et al. Sparse moes meet efficient ensembles. *Transactions on Machine Learning Research*, 2022.

[28] Andrew Jaegle, Felix Gimeno, Andy Brock, and et al. Perceiver: General perception with iterative attention. In *ICML*, 2021.

[29] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, and et al. Perceiver IO: A general architecture for structured inputs & outputs. In *ICLR*, 2022.

[30] Anirudh Goyal, Aniket Rajiv Didolkar, Alex Lamb, and et al. Coordination among neural modules through a shared global workspace. In *ICLR*, 2022.

[31] Dianbo Liu, Alex Lamb, Kenji Kawaguchi, and et al. Discrete-valued neural communication. In *NeurIPS*, 2021.

[32] Yuwei Sun, Hideya Ochiai, Zhirong Wu, and et al. Associative transformer is a sparse representation learner, 2023.

[33] Ankit Gupta and Jonathan Berant. GMAT: global memory augmentation for transformers. arXiv:2006.03274, 2020.

[34] Juho Lee, Yoonho Lee, Jungtaek Kim, and et al. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

[35] Xuezhe Ma, Xiang Kong, Sinong Wang, and et al. Luna: Linear unified nested attention. In *NeurIPS*, 2021.

[36] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. arXiv:1410.5401, 2014.

[37] Çaglar Gülçehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. Dynamic neural turing machine with continuous and discrete addressing schemes. *Neural Comput.*, 30(4), 2018.

[38] Dmitry Krotov and John J. Hopfield. Dense associative memory for pattern recognition. In *NIPS*, pages 1172–1180, 2016.

[39] Benjamin Hoover, Yuchen Liang, Bao Pham, and et al. Energy transformer. arXiv:2302.07253, 2023.

[40] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, 1982.

[41] Tom Michael Mitchell. The need for biases in learning generalizations. 2007.

[42] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*.

[43] Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. arXiv:2011.15091, 2020.

[44] Yuwei Sun and Hideya Ochiai. Homogeneous learning: Self-attention decentralized deep learning. *IEEE Access*, 2021.

[45] Yuwei Sun, Ng Chong, and Ochiai Hideya. Feature distribution matching for federated domain generalization. *ACML*, 2022.

[46] Peter Kairouz, H. McMahan, Brendan Avent, and et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 2021.

[47] Chandra Thapa, P. Chamikara, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. *AAAI*, 2022.

[48] Yuwei Sun, Hideya Ochiai, and Hiroshi Esaki. Intrusion detection with segmented federated learning for large-scale multiple lans. *IJCNN*, 2020.

[49] Sinno Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, 2010.

[50] Gabriela Csurka. A comprehensive survey on domain adaptation for visual applications. *Domain Adaptation in Computer Vision Applications*, 2017.

[51] Garrett Wilson and Diane Cook. A survey of unsupervised deep domain adaptation. *ACM Trans. Intell. Syst. Technol.*, 11(5):51:1–51:46, 2020.

[52] Wouter Kouw and Marco Loog. A review of domain adaptation without target labels. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(3):766–785, 2021.

[53] Alexander J. Smola and Shravan M. Narayanamurthy. An architecture for parallel topic models. In *Proc. VLDB Endow.*, volume 3, pages 703–710, 2010.

[54] Mu Li, David Andersen, Jun Park, and et al. Scaling distributed machine learning with the parameter server. *USENIX*, 2014.

[55] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *ICLR*, 2020.

[56] Felix Sattler, Simon Wiedemann, Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3400–3413, 2020.

[57] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. 2020.

[58] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. *CVPR*, 2018.

[59] Shuhao Cui, Xuan Jin, and et al. Heuristic domain adaptation. *NeurIPS*, 2020.

[60] Guoqiang Wei, Cuiling Lan, Wenjun Zeng, and et al. Toalign: Task-oriented alignment for unsupervised domain adaptation. *NeurIPS*, 2021.

[61] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *CVPR*, 2017.

[62] Junlin Han, Mehrdad Shoeiby, Lars Petersson, and Mohammad Armin. Dual contrastive learning for unsupervised image-to-image translation. *CVPR Workshops*, 2021.

[63] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. *ICML*, 2015.

[64] Karsten Borgwardt, Arthur Gretton, Malte Rasch, and et al. Integrating structured biological data by kernel maximum mean discrepancy. *ISMB*, 2006.

[65] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael Jordan. Deep transfer learning with joint adaptation networks. *ICML*, 2017.

[66] Yaroslav Ganin, Victor Lempitsky, and et al. Unsupervised domain adaptation by backpropagation. *ICML*, 2015.

[67] Liling Zhang, Xinyu Lei, Yichun Shi, and et al. Federated learning with domain generalization. *arXiv preprint*, 2021.

[68] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *ICLR*, 2020.

[69] Chunhan Yao, Boqing Gong, and et al. Federated multi-target domain adaptation. *arXiv preprint*, 2021.

[70] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, and et al. Domain-adversarial training of neural networks. *CVPR*, 2017.

[71] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. *CVPR*, 2019.

[72] Yoshua Bengio. The consciousness prior. *arXiv preprint*, 2017.

[73] Adam Paszke, Sam Gross, Francisco Massa, and et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019.

[74] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs*, 2010.

[75] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. *CVPR*, 2012.

[76] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *ACL*, 2007.

[77] Jacob Devlin, Mingwei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, 2018.

[78] Laurens Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

[79] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *ICML*, 2020.

[80] Jure Zbontar, Li Jing, Ishan Misra, and et al. Barlow twins: Self-supervised learning via redundancy reduction. *ICML*, 2021.

[81] Alec Radford, Jong Kim, Chris Hallacy, and et al. Learning transferable visual models from natural language supervision. *ICML*, 2021.

[82] Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. *NeurIPS*, 2021.

[83] Hongxu Yin, Arun Mallya, Arash Vahdat, and et al. See through gradients: Image batch recovery via gradinversion. *CVPR*, 2021.

[84] General data protection regulation. https://gdpr-info.eu. Accessed: 2021-11-13.

[85] Mani Parimala, R. M. Swarna Priya, Quoc-Viet Pham, and et al. Fusion of federated learning and industrial internet of things: A survey. arXiv:2101.00798, 2021.

[86] Yujia Gao, Liang Liu, Binxuan Hu, and et al. Federated region-learning for environment sensing in edge computing system. *IEEE Transactions on Network Science and Engineering*, 7(4):2192–2204, 2020.

[87] Yang Liu, Anbu Huang, Yun Luo, and et al. Fedvision: An online visual object detection platform powered by federated learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 2020.

[88] Shiva Raj Pokhrel and Jinho Choi. Federated learning with blockchain for autonomous vehicles: Analysis and design challenges. *IEEE Transactions on Communications*, 68(8):4734–4746, 2020.

[89] Boyi Liu, Lujia Wang, Ming Liu, and Chengzhong Xu. Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems. arXiv:1901.06455, 2019.

[90] Yuwei Sun, Hiroshi Esaki, and Hideya Ochiai. Adaptive intrusion detection in the networking of large-scale lans with segmented federated learning. *IEEE Open J. Commun. Soc.*, 2:102–112, 2021.

[91] Sawsan Abdul Rahman, Hanine Tout, Chamseddine Talhi, and Azzam Mourad. Internet of things intrusion detection: Centralized, on-device, or federated learning? *IEEE Network*, 34(6):310–317, 2020.

[92] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *ICLR*, 2018.

[93] Di Cao, Shan Chang, Zhijian Lin, and et al. Understanding distributed poisoning attack in federated learning. In *IEEE ICPADS*, 2019.

[94] T. Nguyen, P. Rieger, Markus Miettinen, and A. Sadeghi. Poisoning attacks on federated learning-based iot intrusion detection system. 2020.

[95] Moming Duan, Duo Liu, Xianzhang Chen, and et al. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Trans. Parallel Distributed Syst.*, 32(1):59–71, 2021.

[96] Stefanie Warnat-Herresthal, Hartmut Schultze, Krishnaprasad Lingadahalli Shastry, and et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature 594, 265–270*, 2021.

[97] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. 2018.

[98] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. pages 1698–1707, 2020.

[99] Yue Zhao, Meng Li, Liangzhen Lai, and et al. Federated learning with non-iid data. arXiv:1806.00582, 2018.

[100] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, and et al. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. arXiv:1811.11479, 2018.

[101] Yuzheng Li, Chuan Chen, Nan Liu, and et al. A blockchain-based decentralized federated learning framework with committee consensus. *IEEE Netw.*, 35(1):234–241, 2021.

[102] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan, and Yan Zhang. Blockchain and federated learning for privacy-preserved data sharing in industrial iot. *IEEE Transactions on Industrial Informatics*, 16(6):4177–4186, 2020.

[103] Nishat I. Mowla, Nguyen H. Tran, Inshil Doh, and Kijoon Chae. Federated learning-based cognitive detection of jamming attack in flying ad-hoc network. *IEEE Access*, 8:4338–4350, 2020.

[104] Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. Detailed comparison of communication efficiency of split learning and federated learning. arXiv:1909.09145, 2019.

[105] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747, 2017.

[106] Rodney A. Brooks. Intelligence without representation. *Artif. Intell.*, 47(1-3):139–159, 1991.

[107] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. arXiv:2012.05208, 2020.

[108] Marvin Minsky. *The Society of Mind*. Simon & Schuster, Inc., 1986.

[109] Jonathan Baxter. A model of inductive bias learning. In *J. Artif. Intell. Res.*, 2000.

[110] Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *arXiv preprint:2011.15091*, 2020.

[111] Jiezhong Qiu, Hao Ma, Omer Levy, and et al. Blockwise self-attention for long document understanding. In *EMNLP*, 2020.

[112] M Demircigil, J Heusel, M L"owe, S Upgang, and F Vermet. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168(2):288–299, 2017.

[113] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

[114] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012.

[115] Adam Santoro, David Raposo, David G. T. Barrett, and et al. A simple neural network module for relational reasoning. In *NIPS*, 2017.

[116] Wenhai Wang, Enze Xie, Xiang Li, and et al. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.

[117] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, and et al. Levit: a vision transformer in convnet's clothing for faster inference. In *ICCV*, 2021.

[118] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, and et al. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.

[119] Yuwei Sun, Hideya Ochiai, and Jun Sakuma. Semi-targeted model poisoning attack on federated learning via backward error analysis. *IJCNN*, 2022.

[120] Yuwei Sun, Hideya Ochiai, and Jun Sakuma. Attacking distance-aware attack: Semi-targeted model poisoning on federated learning. *IEEE Transactions on Artificial Intelligence*, 2023.

[121] Bin Gu, An Xu, Zhouyuan Huo, and et al. Privacy-preserving asynchronous vertical federated learning algorithms for multiparty collaborative learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[122] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *ECAI*, 2012.

[123] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, and et al. Attack of the tails: Yes, you really can backdoor federated learning. In *NeurIPS*, 2020.

[124] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the GAN: information leakage from collaborative deep learning. In *ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[125] Junbo Wang, Amitangshu Pal, Qinglin Yang, and et al. Collaborative machine learning: Schemes, robustness, and privacy. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[126] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[127] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, and et al. How to backdoor federated learning. In *AISTATS*, 2020.

[128] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *USENIX Security Symposium*, pages 1505–1521, 2021.

[129] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *ACM SIGSAC*, pages 113–131. ACM, 2020.

[130] Yujie Ji, Xinyang Zhang, Shouling Ji, and et al. Model-reuse attacks on deep learning systems. In *arXiv preprint*, 2018.

[131] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *USENIX Security Symposium*, 2020.

[132] Xingchen Zhou, Ming Xu, Yiming Wu, and Ning Zheng. Deep model poisoning attack on federated learning. In *Future Internet*, volume 13, page 73, 2021.

[133] Xueluan Gong, Yanjiao Chen, Qian Wang, and Weihan Kong. Backdoor attacks and defenses in federated learning: State-of-the-art, taxonomy, and future directions. *IEEE Wireless Communications*, pages 1–7, 2022.

[134] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy (SP)*, 2019.

[135] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the GAN: information leakage from collaborative deep learning. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[136] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? *NeurIPS*, 2020.

[137] Xueluan Gong, Yanjiao Chen, Huayang Huang, and et al. Coordinated backdoor attacks against federated learning with model-dependent triggers. *IEEE Netw.*, 36(1):84–90, 2022.

[138] Xinyun Chen, Chang Liu, Bo Li, and et al. Targeted backdoor attacks on deep learning systems using data poisoning. In *arXiv preprint*, 2017.

[139] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin B. Calo. Analyzing federated learning through an adversarial lens. In *ICML*, 2019.

[140] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, and et al. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NeurIPS*, 2018.

[141] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*, 2020.

[142] Ying Zhao, Junjun Chen, Jiale Zhang, and et al. PDGAN: A novel poisoning defense method in federated learning using generative adversarial network. In *Algorithms and Architectures for Parallel Processing*, 2019.

[143] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. In *arXiv preprint*, 2020.

[144] Zhen Xiang, David J. Miller, and George Kesidis. Detection of backdoors in trained classifiers without access to the training set. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[145] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*, 2017.

[146] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *ICML*, 2018.

[147] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *NDSS Symposium*, 2018.

[148] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy*, 2019.

[149] Martín Abadi, Andy Chu, Ian J. Goodfellow, and et al. Deep learning with differential privacy. In *ACM Conference on Computer and Communications Security*, 2016.

[150] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can you really backdoor federated learning? In *arXiv preprint*, 2019.

[151] Junyu Shi, Wei Wan, Shengshan Hu, and et al. Challenges and approaches for mitigating byzantine attacks in federated learning. In *TrustCom*, pages 139–146. IEEE, 2022.

[152] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[153] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate bayesian inference. *J. Mach. Learn. Res.*, 18:134:1–134:35, 2017.

[154] Martín Abadi, Ashish Agarwal, Paul Barham, and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[155] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[156] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, 2018.

[157] Robin C. Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. In *arXiv preprint*, 2017.

[158] Yuwei Sun, Hideya Ochiai, and Jun Sakuma. Instance-level trojan attacks on visual question answering via adversarial learning in neuron activation space, 2023.

[159] Matthew Walmer, Karan Sikka, Indranil Sur, and et al. Dual-key multimodal backdoors for visual question answering. In *CVPR*, 2022.

[160] Yuezun Li, Yiming Li, Baoyuan Wu, and et al. Invisible backdoor attack with sample-specific triggers. In *ICCV*, pages 16443–16452. IEEE, 2021.

[161] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, and et al. Intriguing properties of neural networks. In *ICLR*, 2014.

[162] Karren Yang, Wan-Yi Lin, Manash Barman, and et al. Defending multimodal fusion models against single-source adversaries. In *CVPR*, 2021.

[163] Zhihong Lin, Donghao Zhang, Qingyi Tao, and et al. Medical visual question answering: A survey. arXiv:2111.10056, 2021.

[164] Yapeng Tian and Chenliang Xu. Can audio-visual integration strengthen robustness under multimodal attacks? In *CVPR*, 2021.

[165] Vasu Sharma, Ankita Kalra, Vaibhav, and et al. Attend and attack: Attention guided adversarial attacks on visual question answering models. In *NeurIPS Workshop*, 2018.

[166] Akshay Chaturvedi and Utpal Garain. Attacking VQA systems via adversarial background noise. *IEEE Trans. Emerg. Top. Comput. Intell.*, 4(4):490–499, 2020.

[167] Peter Anderson, Xiaodong He, Chris Buehler, and et al. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 2018.

[168] Zhou Yu, Yuhao Cui, Jun Yu, and et al. Deep multimodal neural architecture search. In *ACM Multimedia*, 2020.

[169] Zhou Yu, Jun Yu, Yuhao Cui, and et al. Deep modular co-attention networks for visual question answering. In *CVPR*, 2019.

[170] Jin-Hwa Kim, Jaehyun Jun, Byoung-Tak Zhang, and et al. Bilinear attention networks. In *NeurIPS*, 2018.

[171] Zichao Yang, Xiaodong He, Jianfeng Gao, and et al. Stacked attention networks for image question answering. In *CVPR*, 2016.

[172] Kyung-Min Kim, Min-Oh Heo, Seong-Ho Choi, and Byoung-Tak Zhang. Deepstory: Video story QA by deep embedded memory networks. In *IJCAI*, 2017.

[173] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 2017.

[174] Jeffrey Pennington, Richard Socher, Christopher D. Manning, and et al. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[175] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, and et al. VQA: visual question answering - www.visualqa.org. In *Int. J. Comput. Vis.*, volume 123, pages 4–31, 2017.

[176] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, and et al. Microsoft COCO: common objects in context. In *ECCV*, 2014.

[177] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015.

[178] Yuwei Sun and Hideya Ochiai. Unicon: Unidirectional split learning with contrastive loss for visual question answering. *arXiv.2208.11435*, 2022.

[179] Yingqi Liu, Shiqing Ma, Yousra Aafer, and et al. Trojaning attack on neural networks. In *NDSS*. The Internet Society, 2018.

[180] Chaoyang He, Zhengyu Yang, Erum Mushtaq, and et al. Ssfl: Tackling label deficiency in federated learning via personalized self-supervision, 2021.

[181] Krishna Pillutla, Kshitiz Malik, Abdelrahman Mohamed, and et al. Federated learning with partial model personalization. In *ICML*, 2022.

[182] Fenglin Liu, Xian Wu, Shen Ge, and et al. Federated learning for vision-and-language grounding problems. In *AAAI*, 2020.

[183] Jean-Baptiste Alayrac, Adrià Recasens, Rosalia Schneider, and et al. Self-supervised multimodal versatile networks. In *NeurIPS*, 2020.

[184] Andrew Rouditchenko, Angie W. Boggust, David Harwath, and et al. Avlnet: Learning audio-visual language representations from instructional videos. In *Annual Conference of the International Speech Communication Association*, 2021.

[185] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, and et al. Zero-shot text-to-image generation. In *ICML*, 2021.

[186] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, and et al. Hierarchical text-conditional image generation with CLIP latents. In *arXiv preprint arXiv.2204.06125*, 2022.

[187] Xi Zhu, Zhendong Mao, Chunxiao Liu, and et al. Overcoming language priors with self-supervised learning for visual question answering. In *IJCAI*, 2020.

[188] Yash Goyal, Tejas Khot, Aishwarya Agrawal, and et al. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Int. J. Comput. Vis.*, volume 127, pages 398–414, 2019.

[189] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[190] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017.

[191] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[192] Aäron Oord, Yazhe Li, Oriol Vinyals, and et al. Representation learning with contrastive predictive coding. In *arXiv Preprint*, 2018.

[193] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *ICCV*, 2017.

[194] Mandela Patrick, Yuki Markus Asano, Ruth Fong, and et al. Multi-modal self-supervision from generalized data transformations. In *arXiv preprint*, 2020.

[195] Tae Kyun Kim. T test as a parametric statistic. In *Korean booktitleof anesthesiology*, volume 68, 2015.

[196] Randall C. O'Reilly, Rajan Bhattacharyya, Michael D. Howard, and Nicholas Ketz. Complementary learning systems. *Cogn. Sci.*, 38(6):1229–1248, 2014.

[197] Wanjun Zhong, Lianghong Guo, Qiqi Gao, and et al. Memorybank: Enhancing large language models with long-term memory. arXiv:2305.10250, 2023.

[198] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. arXiv:2303.11366, 2023.

[199] Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, and et al. Generative agents: Interactive simulacra of human behavior. In *UIST*, pages 2:1–2:22. ACM, 2023.

[200] Weizhi Wang, Li Dong, Hao Cheng, and et al. Augmenting language models with long-term memory. arXiv:2306.07174, 2023.

[201] Ned Block. On a confusion about a function of consciousness. *Brain and Behavioral Sciences*, 18(2):227–247, 1995.

[202] David Chalmers. Facing up to the problem of consciousness. *Journal of Consciousness Studies*, 2(3):200–19, 1995.

[203] Kensinger A. Budson E., Richman A. Consciousness as a memory system. *Cognitive and behavioral neurology*, 2022.

[204] David Ha and Jürgen Schmidhuber. World models. arXiv:1803.10122, 2018.

[205] Dianbo Liu, Samuele Bolotta, He Zhu, Yoshua Bengio, and Guillaume Dumas. Attention schema in neural agents, 2023.

Appendices

Source Code List

1. Implementation of experiments in Chapter 3:
   https://github.com/yuweisunn/federated-knowledge-alignment
2. Implementation of experiments in Chapter 4:
   https://github.com/yuweisunn/homogeneous-learning-tensorflow
3. Implementation of experiments in Chapter 5:
   https://github.com/yuweisunn/associative-transformer
4. Implementation of experiments in Chapter 6:
   https://github.com/yuweisunn/attacking-distance-aware-attack

Video List

1. Presentation of the work in Chapter 3:
   https://youtu.be/TgZEReQo21w
2. Presentation of the work in Chapter 5:
   https://youtu.be/YMPoRsKYMMo
3. Presentation of the work in Chapter 6:
   https://youtu.be/b-ZWaX-xYeE
4. Presentation of the work in Chapter 4 and 8:
   https://youtu.be/l1COTtZFtRs