TRI-FED-RKD: Integrating Forward-Reverse Distillation with SNN and CNN within Federated Learning using Tri Layer Hierarchical Aggregation based Architecture.

by

Md. Mohiuzzaman
20301361
Ahmad Abrar Abedin
20201080
Shadab Afnan Rahman
21101076
Shafaq Arefin Chowdhury
21101064
Shahadat Ahmed
20301481

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Mohiuzzaman

Md. Mohiuzzaman

20301361

Ahmed Abrar Abedin

20201080

Shadab

Shadab Afnan Rahman

21101076

Shafaq

Shafaq Arefin Chowdhury

21101064

Shahadat Ahmed

Shahadat Ahmed

20301481

Approval

The thesis/project titled "TRI-FED-RKD: Integrating Forward-Reverse Distillation with SNN and CNN within Federated Learning using Tri Layer Hierarchical Aggregation based Architecture." submitted by

1. Md. Mohiuzzaman (20301361)
2. Ahmad Abrar Abedin (20201080)
3. Shadab Afnan Rahman (21101076)
4. Shafaq Arefin Chowdhury (21101064)
5. Shahadat Ahmed (20301481)

Of Summer, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 17, 2024.

Examining Committee:

Supervisor:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Co Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain

Asssociate Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Federated Learning (FL) is a decentralized machine learning paradigm that enables training a global model across numerous edge devices while preserving data privacy. However, FL faces significant challenges, particularly in environments with heterogeneous hardware capabilities, communication burdens, and constrained resources. In this paper, we introduce a novel framework, TRI-FED-RKD, which incorporates forward and reverse knowledge distillation (RKD) along with FedAvg using a hybrid architecture of convolutional neural networks (CNNs) and spiking neural networks (SNNs). Our approach employs a tri-layer hierarchical aggregation-based architecture consisting of client devices, intermediate (middle) servers, and a global server. We compared two federated architectures: standard federated learning and federated learning with forward and reverse distillation in a hierarchical setting (TRI-FED-RKD). The same model is used across several datasets to evaluate the architectures, not the model performance. Depending on the use case, the network administrator can pick their own teacher and student models. The teacher model can also be different for each client if needed. This means that our architecture can deal with model heterogeneity when it comes to teacher models. We evaluate TRI-FED-RKD on neuromorphic datasets such as DVS Gesture and NMNIST. We also tested it using non-neuromorphic datasets such as MNIST, EMNIST, and CIFAR10. Furthermore, we have shown that using forward and reverse knowledge distillation in federated learning can lead to much better performance than federated learning without knowledge distillation for non-neuromorphic datasets.

Keywords: Federated learning, Tri-Layer Architecture, Knowledge Distillation, Spiking Neural Networks (SNNs), Neuromorphic datasets, Dynamic Vision Sensor(DVS).

Acknowledgement

- Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.

- Secondly, to our supervisor Dr. Md. Golam Rabiul Alam sir for his kind support and advice in our work. He helped us whenever we needed help.

- And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

Declaration i

Approval ii

Abstract iv

Acknowledgment v

Table of Contents vi

List of Figures viii

List of Tables 1

1 Introduction 2
- 1.1 Research Background . . . 2
- 1.2 Research Scope . . . 3
- 1.3 Research Objectives . . . 3
- 1.4 Research Contributions . . . 5
- 1.5 Research Outline . . . 5

2 Literature review 6
- 2.1 Federated Learning . . . 6
- 2.2 Knowledge Distillation . . . 6
- 2.3 CNNs in Federated Averaging for Image Classification . . . 7
- 2.4 Feature Extraction in CNNs for Federated Learning and ResNet-18 . . . 7
- 2.5 Spiking Neural Networks (SNNs) . . . 8
- 2.6 Federated Learning with spiking neural networks . . . 9

3 Methodology 10
- 3.1 Work Flow . . . 10
- 3.2 Dataset Description . . . 11
  - 3.2.1 EMNIST . . . 11
  - 3.2.2 CIFAR10 . . . 11
  - 3.2.3 MNIST . . . 11
  - 3.2.4 DVS128 Gesture . . . 11
  - 3.2.5 NMNIST . . . 11
- 3.3 Dataset Pre-Processing . . . 12
- 3.4 Proposed Federated Architectures . . . 15
- 3.5 Models we used in our experiment . . . 18

4 Result and Analysis **22**
4.1 Non Neuromorphic Datasets 22
4.2 Neuromorphic Datasets 26
4.3 Result Analysis Based on FLOPs, Parameters, and Size 28
4.4 Server Aggregation Time 29

5 **31**
5.1 Conclusion 31
5.2 Future work 31

Bibliography **33**

List of Figures

3.1.1 Workflow 10
3.3.6 Framing samples for NMNIST digit "0" 14
3.4.1 Proposed TRI Layer Federated Learning Architecture with Forward-Reverse KD 15
3.4.2 BI Layer Federated Learning Architecture with Forward-Reverse KD 16
3.4.3 Knowledge Distillation Setup 17
3.5.1 Teacher Model for Non-Neuromorphic dataset using CNN and CNN+SNN 18
3.5.2 Student Model for Non-Neuromorphic dataset using CNN and CNN+SNN 18
3.5.3 Neuromorphic Dataset Student Model using CNN+SNN 19
3.5.4 Neuromorphic Dataset Student Model using CNN 19
3.5.5 Neuromorphic Dataset Teacher Model using CNN 20
3.5.6 Neuromorphic Dataset Teacher Model using CNN+SNN 20

4.1.2 Accuracy comparisons of MNIST 23
4.1.3 Loss comparisons of MNIST 23
4.1.4 Accuracy comparisons of EMNIST 24
4.1.5 Loss comparisons of EMNIST 24
4.1.6 Accuracy comparisons of CIFAR 10 25
4.1.7 Loss comparisons of CIFAR10 25
4.2.2 Accuracy comparisons of DVS Gesture 26
4.2.3 Loss comparisons of DVS Gesture 27
4.2.4 Accuracy comparisons of NMNIST 27
4.2.5 Loss comparisons of NMNIST 28
4.4.1 BI layer vs TRI layer Aggregation time in CNN 29
4.4.2 BI layer vs TRI layer Aggregation time in CNN+SNN 30

List of Tables

3.3.1 Data Pre-Processing For Non-Neuromorphic Dataset 12
3.3.2 Hyperparameters for Non-Neuromorphic Dataset 13
3.3.3 Data Pre-Processing for Neuromorphic Dataset 13
3.3.4 Hyperparameters for Neuromorphic Dataset 14
3.3.5 Dataset Distribution 14

4.1.1 Performance Comparison across Non Neuromorphic Datasets with KD and without KD 22
4.2.1 Performance Comparison across DVS Gesture and N-MNIST Datasets with KD and without KD 26
4.3.1 Comparison between Teacher and Student models in terms of FLOPs, Parameters, and Size across different datasets. 28

Chapter 1

Introduction

1.1 Research Background

In today's digital world, along with technologically significant advancements, mobile devices made their way to almost everyone. Because of edge devices, mostly in the form of mobile phones, wearables, and internet of things (IoT) devices, trillions of data are needed to be managed, enabling them to support huge varieties of applications. The challenge, however, is rooted in the fact that edge devices are also decentralized, and so is the data, meaning that managing data from each device poses significant challenges [19].

The conventional machine learning approach is highly centralized, which means that the decentralized datasets that are the case in terms of modern devices are sent to a central server for processing and model training. Because of this, there are a multitude of issues that failed to address privacy requirements, which are data leakage, as the data from a decentralized device is getting out of that particular device into a common sphere, which is the centralized server, leading to the need for privacy-enhancing machine learning methodologies [19].

To tackle these challenges, a decentralized machine-learning approach known as Federated Learning (FL) emerged as a promising solution. Federated learning enables the training of machine learning models locally, without transmitting raw client data to the global central server. This approach maintains privacy by sending only model updates for global aggregation ensuring sensitive data remains on the local devices. As a result, it not only mitigates privacy concerns by keeping sensitive data on local devices but also enhances data security by minimizing the risk of data violations that are common in centralized systems. The decentralized nature of Federated Learning aligns well with the distributed topology of edge computing. It also enables devices to collaboratively learn a shared model that benefits from diverse data sources while maintaining user privacy.

However, despite its advantages, Federated Learning faces significant issues in real-world deployment, particularly in terms of communication, computational efficiency,

and device heterogeneity. Communication in FL is very much resource-intensive. Frequent rounds of communication between client devices and the global server are required to exchange desired model updates. However, with the growing number of participating devices, aggregating model updates at a central server becomes a resource-intensive task that results in communication delays and high energy consumption, especially for devices with limited computational and battery resources such as IoT devices and embedded systems. Moreover, devices in FL environments are heterogeneous in processing power, memory power, and energy capacity.

In view of these challenges, the combination of Spiking Neural Networks (SNNs) and traditional Convolutional Neural Networks (CNNs) can be the solution. SNNs which are based on the human brain methodology of information processing are designed to be more energy efficient than typical Artificial Neural Networks (ANNs). Due to the fact that SNN relies on discrete spikes rather than continuous activation, SNNs only fire spike from the neuron when it crosses a certain threshold (Grunig et al 2014), leading to very low energy consumption and this gives huge benefits for our resource-constrained devices.

In our research, we introduce a federated learning framework called the TRI-FED-RKD. This framework integrates forward reverse knowledge distillation (FRKD) architecture within a tri-layer hierarchical aggregation system. This ensures model scalability making it suitable for diverse real-world applications.

1.2 Research Scope

As far as we have studied, the prior works on the classification of dynamic vision sensors (DVS) datasets using federated learning with spiking neural networks and knowledge distillation combined are rare. In [20] they successfully incorporated federated learning with SNN but did not perform classification tasks on DVS datasets. In [23] the SNN alongside knowledge distillation was incorporated too, but here we saw no DVS datasets being used. Therefore, dynamic vision datasets still need to be worked on using CNN+SNN hybridized federated learning using forward and reverse knowledge distillation in those datasets and alongside that, we also tried to better the accuracies in MNIST, EMNIST, and CIFAR 10 datasets following our algorithm.

1.3 Research Objectives

The emergence of federated learning has posed significant advantages in the sphere of decentralized machine learning, but with rewards came challenges too. Therefore, we tried to address the challenges of federated learning by experimenting with state-of-the-art techniques like knowledge distillation and also neuromorphic computing in order to tackle some of the crucial challenges.

Our research compares two federated architectures: federated learning and federated learning using forward and reverse distillation in a hierarchical setting. The focus here is not on the models themselves, as we have used the same model across several datasets. The reason for this consistency is to evaluate the federated learning architecture based on their own merits rather than the performance of hyperparameter-tuned models. The architecture allows for flexible model selection, meaning that depending on the use case, the network administrator can choose their own models for the roles of teacher and/or student.

One of the objectives is to reduce the communication overhead in the global server, due to that we tried to introduce a hierarchical architecture in federated learning by introducing middlemen in the form of middle servers between the global server and the clients. The middle servers handle clusters of clients among them equally which reduces the burden of aggregating large batches of weights in a single server helping in reducing the communication burden.

Moreover, we tried to implement forward and reverse knowledge distillation among the client devices, as knowledge distillation ensures better smaller models due to distillation of knowledge from bigger models[5]. The edge devices are provided with a larger complex teacher model and a smaller simpler student model. Here the integration of the student model in low-powered devices is to implement an efficient model with the teacher model being the mentor to the student model, as it trains much more accurately and converges fast due to having significantly higher parameters than the student model. After the teacher model is done training, it transfers its knowledge to the student model, which is not capable of handling complex data before learning from its mentor, the teacher model, but after learning the parameters, it becomes capable of handling challenging datasets. The updated parameters in the student model are then sent to the global servers for aggregation. However, in most cases, the student model is the one that is being used for predictions in the client, but when the need arises or during training complex datasets, the client can switch to the teacher model for training and send the knowledge to the student models. We implemented reverse knowledge distillation here as well. The student model is the one running mostly and is regularly updated with the global model updates. Hence, it transfers the knowledge to the teacher using reverse distillation so that both models are trained for maximizing accurate predictions.

In addition to the integrations above, we have tested CNN-SNN hybrid models in our architecture. The key benefits of CNN-SNN hybridization are that CNNs, as we know are excellent for extracting features from the dataset aside from being good at extracting features, CNNs are energy-intensive neural networks; therefore we are using convolution and pooling layers for the feature extraction part but for the fully-connected layers we will use SNNs. This gives us the benefit of handling temporal data alongside static data as the CNNs greatly process spatial features while the SNNs, due to their event-driven architecture [4] process temporal data effectively. This enhances learning in edge computing as the CNNs handle the heavy-lifting

feature extraction while the SNNs handle decision-making tasks in edge computing.

1.4 Research Contributions

- We used neuromorphic datasets like DVS datasets in Federated Learning Architecture with forward and reverse knowledge distillation

- We improved the student model accuracies by incorporating forward distillation.

- We are updating the teacher model parameters by applying reverse distillation after the global updates.

- We have observed a reduction in server aggregation time by incorporating trilayer concept into the FL architecture.

1.5 Research Outline

The rest of the paper is divided into a literature review in Chapter 2, proposed methodology in Chapter 3, results and analysis in Chapter 4, and conclusions and future work in Chapter 5, respectively.

Chapter 2

Literature review

2.1 Federated Learning

Federated Learning (FL) has emerged as a robust approach to decentralized machine learning where multiple clients collaborate to train a shared global model without sharing their raw data [11]. This technique is very helpful in addressing concerns related to privacy, as data from local devices is not being shared with the global model because a local model is being trained on the local device. However, there are some challenges in FL. In federated learning, the need to share the models frequently between the clients and the servers poses communication bottlenecks [14].

2.2 Knowledge Distillation

Knowledge distillation (KD) is a robust model compression technique where the smaller model, also called the student model, tries to mimic a larger model, also known as the teacher model [5]. The motive behind this was to reduce the complexity of deep neural networks without sacrificing their performance. KD works by transferring the soft predictions of the teacher model, which often contains valuable information for the student model, allowing the student to achieve higher efficiency with fewer parameters.

In the context of federated learning, KD has been introduced to tackle some major challenges, which are heterogeneity in local models and inefficiency in communication. Many studies have demonstrated that KD can be used to transfer knowledge between the clients and the server in a more compressed form, reducing communication costs [24]. Moreover, KD can also be used to overcome problems in FL by ensuring that each client model distills and shares meaningful global knowledge without the need to converge to a single model structure [22]. The combination of FedAvg and knowledge distillation has been shown to yield superior results in various datasets. For instance, in a paper, they proposed a federated learning approach that employs knowledge distillation, achieving state-of-the-art results on both FEMNIST and CIFAR10 datasets [15]. Despite the success of KD in conventional neural networks, the application of KD in the field of spiking neural networks is very much unexplored. This allows an opportunity to explore and use KD for federated learning environments where SNNs are employed.

2.3 CNNs in Federated Averaging for Image Classification

Applying CNNs to FedAvg has shown promising results in various image classification tasks across datasets like CIFAR10, EMNIST, and MNIST. This serves as a benchmark for evaluating CNN architectures in federated learning settings [3]. In a study, it was demonstrated that FedAvg could achieve competitive results on CIFAR10, but it faced challenges related to communication efficiency [13], 2018. To address these challenges, various strategies have been proposed, including the development of more efficient communication protocols and adaptive aggregation techniques. The EMNIST dataset, an extension of the MNIST dataset that includes handwritten letters, adds complexity to the classification task due to the increased variety in the data. Research highlights the importance of optimizing federated learning for diverse datasets like EMNIST, with a focus on improving accuracy and reducing communication overhead [12]. When the FedAvg approach is applied to CNNs, it continues to demonstrate its effectiveness in federated learning, particularly in achieving a balance between accuracy and resource efficiency.

2.4 Feature Extraction in CNNs for Federated Learning and ResNet-18

Feature extraction is a crucial aspect of CNNs that allows for effective representation learning from raw data. In a paper, they investigate the role of CNN-based feature extraction in federated learning, focusing on ResNet-18 to enhance classification accuracy. The authors employ a method where features are extracted locally on the client side and then aggregated at the server to construct a global model [26]. By leveraging the depth and residual connections of ResNet-18, they demonstrate its ability to capture detailed, hierarchical features, which leads to improved model performance.

The paper also reports that ResNet-18 contributes significantly to better classification accuracy on datasets such as CIFAR10 and EMNIST. This improvement stems from its architecture, which enables more effective feature extraction, enhancing the model's prediction capabilities. Moreover, [26] address the challenge of communication overhead by aggregating feature representations rather than full models, thereby reducing the communication burden on the central server, a key consideration in federated learning systems. Their approach maintains high accuracy while minimising communication costs, demonstrating an efficient balance between performance and resource use. However, the overhead remained relatively high due to the bi-layer architecture employed in their method [26].

2.5 Spiking Neural Networks (SNNs)

Spiking neural networks (SNNs) represent a third-generation neural network model inspired by biological neurons that communicate via discrete spikes and action potentials [1]. Spiking neural network is a network architecture where various encoding methods, rate coding and temporal coding, are popularly used in inputting information in the form of spike per time period instead of continuous values used in artificial neural networks. For instance, let us say there is an input image. The pixels correspond to a value in an artificial neural network architecture. However, in the spike neural network architecture, things are a bit different. The pixel value is converted to spikes per time stamps [4]. The pixel value is proportional to the spike rate and the higher the value of the pixel the higher the number of spikes within the time stamps for rate coding. This is a motivation from the original neural functions where physiological neurons tend to fire more often to significant or stronger stimuli [17].

In terms of neural functionality the leaky integrate and fire neuron models are used more often. To be more precise, as an activation function, we use 'membrane potential' where we usually use RELU in the artificial neural network. This is also motivated by the fact that when a neurotransmitter diffuses into a postsynaptic neuron, it affects the membrane potential of the neuron by increasing the membrane potential and in absence of new inputs, the membrane potential leaks away [17]. As the values are converted into spikes, the presence of the spike at a certain time step causes the membrane function to integrate at the time step and subsequently if there is an absence the membrane potential disintegrates or leaks. There is a user-defined threshold Vthresh, and when the membrane function crosses the threshold there is a spike in the output of the neuron [4]. When one spike is generated after crossing the threshold, the membrane potential returns to a refractory period. SNNs are very useful when it comes to resource-constrained environments such as edge devices and they are well suited for real-time applications as SNNs can leverage temporal information unlike conventional artificial neural networks that process inputs statically [18].

SNNs have several advantages over conventional artificial neural networks, which include reduced power consumption, energy efficiency, and the ability to process spatiotemporal data. These characteristics make them a very good candidate for neuromorphic computing and lower-powered edge devices. However, as the neuron spiking function in SNNs is not continuous, backpropagation is challenging in SNNs due to their non-differentiable nature [16], but recent advancements like surrogate gradient descent have enabled gradient-based learning, making them much more qualified for use in FL [21]. In terms of federated learning, the use of SNNs is quite rare, but due to edge devices being key components in FL, the incorporation of SNN can reap serious benefits in terms of efficiency and real-time processing. However, the integration of SNNs into federated learning frameworks requires solutions for both training and communication.

2.6 Federated Learning with spiking neural networks

Recent studies on edge devices explored the potential of SNNs to operate efficiently in distributed environments [18]. SNNs can be explored within federated learning because of their energy efficiency and event-driving nature. They are capable of reducing communication load in FL, especially on resource-constrained devices [18]. Although the literature on federated learning with spiking neural networks is still sparse, there have been notable efforts. Some papers showed the implementation of SNN with federated learning. A paper used the CIFAR 10 and CIFAR 100 datasets to train SNN-incorporated federated learning [20], and another one was quite successful in reducing communication overheads, but the accuracy could have been improved. For example, in another paper [25], algorithms such as Federated Learning with Top- Sparsification (FLTS) and Federated Learning with Dynamic- Reduction (FLDR) were successful in reducing communication overheads, but accuracy could be improved.

Chapter 3

Methodology

3.1 Work Flow

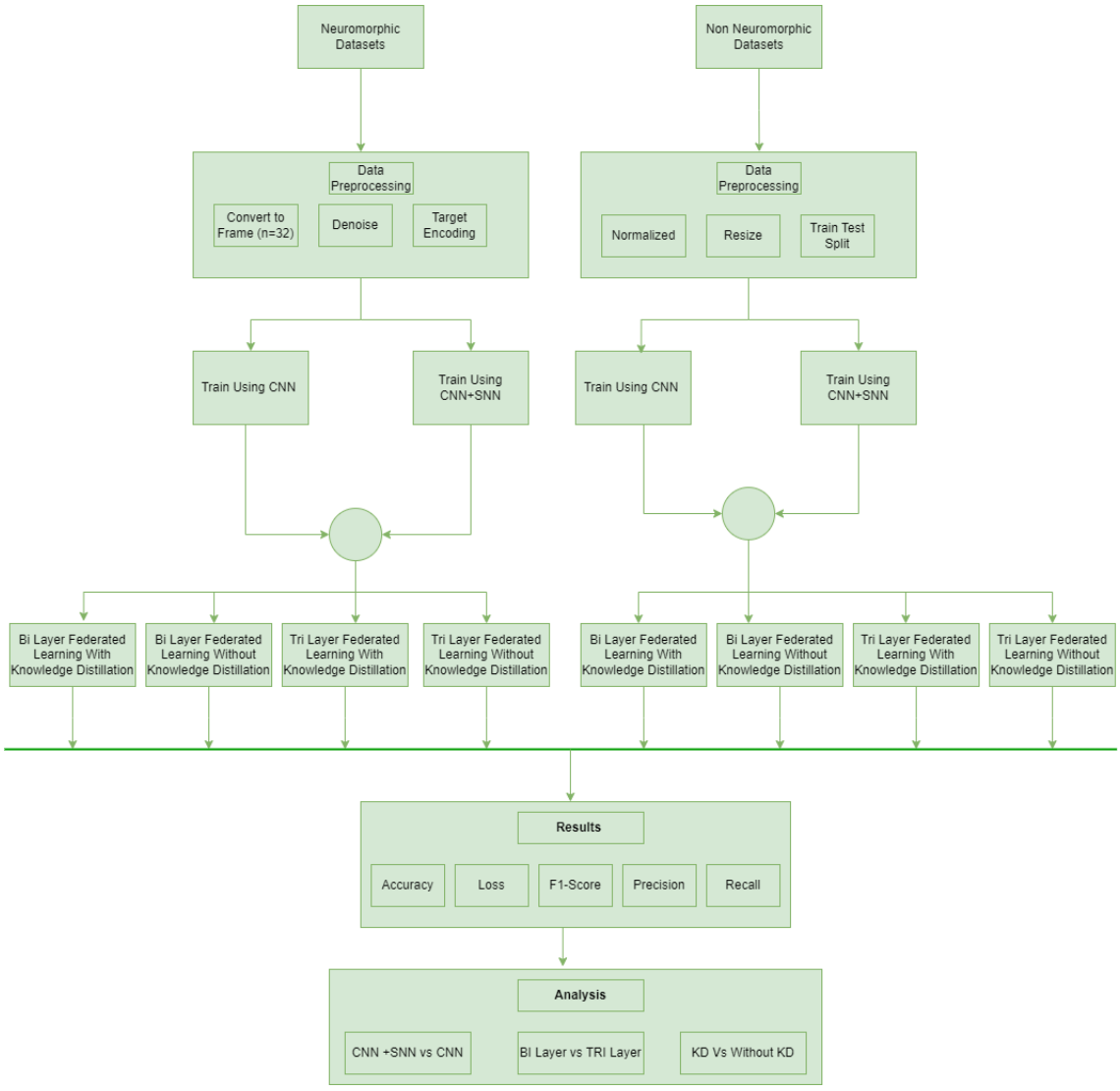

Figure 3.1.1: Workflow

Here Figure 3.1.1, we have primarily worked on two types of datasets, namely neuromorphic and non-neuromorphic datasets. The data goes through the augmentation steps. Then they are trained using CNN, CNN+SNN in Tri and Bi layer architectures. Finally, the results are evaluated using train accuracy, F1 score, Precision, and Recall. Finally, the results are compared.

3.2 Dataset Description

3.2.1 EMNIST

EMNIST is a dataset containing handwritten letters and digits. It has 131,600 images. Dimensions of 28 x 28 and are grayscale images. [9].

3.2.2 CIFAR10

CIFAR10 is a colorful image dataset. Contains 10 classes, which also include cars, animals, birds, etc. There are a total of 60,000 images. Dimensions are 32x32 with 3 RGB color channels, giving a dimension of 32x32x3 [3]

3.2.3 MNIST

Contains handwritten digit images from range 0 to 9. Dimensions are 28x28 and are grayscale images. It consists of a total of 70,000 images and 10 classes. It is part of the larger NIST Special Database 3 and 1. They consist of images of digits that are monochrome and written by hand [2].

3.2.4 DVS128 Gesture

The DVS128 Hand Gesture dataset is a group of hand gesture recordings and was recorded using the DVS128 event-based camera. It has a total of 11 different kinds of hand gestures performed under 3 varying lighting conditions, with a total of 1342 gestures. Instead of using traditional frames in video recordings, the camera is able to detect changes in pixel values when capturing the movement and also the position at which the change occurred in the x, and y axes, the time that it occurred, and polarity which is +1 if there is an increase in pixel intensity and -1 if decrease in pixel intensity [8].

3.2.5 NMNIST

Neuromorphic MNIST, in short N-MNIST, is where the MNIST dataset is converted into a spiking neural net version. It consists of a total of 70,000 images. It is done by using an ATIS sensor, which was held in place by a motorized pan-tilt system. It was then able to capture MNIST static images moving on an LCD screen. All of these are converted into a stream of events [6].

3.3 Dataset Pre-Processing

Processing Type	Dataset	Details
Resize	CIFAR-10	Resize to 32x32 pixels
	MNIST	Resize to 28x28 pixels
	EMNIST	Resize to 28x28 pixels
Grayscale	MNIST	Convert images to grayscale
	EMNIST	Convert images to grayscale
ToTensor	CIFAR-10	Convert to PyTorch tensors
	MNIST	Convert to PyTorch tensors
	EMNIST	Convert to PyTorch tensors
Normalize	CIFAR-10	Normalize pixel values to mean and std of 0.5
	MNIST	Normalize pixel values to mean and std of 0
	EMNIST	Normalize pixel values to mean and std of 0

Table 3.3.1: Data Pre-Processing For Non-Neuromorphic Dataset

This table 3.3.1 describes various preprocessing techniques applied to three datasets: CIFAR-10, MNIST, and EMNIST. These techniques help prepare the image data for training machine learning models.

Resize: The images in the CIFAR-10 dataset are resized to 32x32 pixels. Both MNIST and EMNIST datasets are resized to 28x28 pixels. Resizing helps standardize the image dimensions across datasets for consistency during model training.

Grayscale: For MNIST and EMNIST, the images are converted into grayscale. This means the color information is removed, and the image data is simplified to a single channel (black and white), which is typical for these datasets, as they represent digit or character recognition tasks.

To Tensor: The images in CIFAR-10, MNIST, and EMNIST are transformed into PyTorch tensors. Converting the data to tensors allows for efficient computation within the PyTorch framework, making it easier to perform operations on the data during training.

Normalize: In all three datasets, the pixel values are normalized to have a mean and standard deviation (sd) of 0. For CIFAR-10, the values are normalized with a mean of 0 and a standard deviation of 0.5. Normalization ensures that the pixel values are on a consistent scale, improving the stability and performance of the model during training.

Parameter	Value
Number of Clients	15
Number of Epochs	5
Batch Size	32
Number of Middle Servers	3
Number of Global Rounds	25
Number of Local Rounds	1
Learning Rate	0.001
Beta	0.9
Gradient Function	Autograd
Loss Function	Cross Entropy Loss

Table 3.3.2: Hyperparameters for Non-Neuromorphic Dataset

Processing Type	Dataset	Details
Denoise	DVS Gesture	Removes isolated events (10 ms)
	NMNIST	Removes isolated events (10 ms)
Downsample	DVS Gesture	Downsamples to 32x32
	NMNIST	Downsamples to 28x28
ToFrame	DVS Gesture	Creates frames (n_frames=32)
	NMNIST	Creates frames (n_frames=32)
One Hot Encoding	DVS Gesture	Categories converted to Binary Vectors
	NMNIST	Categories converted to Binary Vectors

Table 3.3.3: Data Pre-Processing for Neuromorphic Dataset

This table 3.3.3 outlines four different types of data processing techniques applied to two datasets: DVS Gesture and NMNIST.

Denoising: Both datasets undergo a process where isolated events lasting 10 milliseconds are removed. This step helps reduce noise in the data, making it cleaner for further analysis or model training.

Downsampling: For the DVS Gesture dataset, the event data is downsampled to a resolution of 32x32 pixels. Similarly, for the NMNIST dataset, the resolution is reduced to 28x28 pixels. This resizing reduces the computational load while retaining essential features for recognition tasks.

Frame Generation: The asynchronous stream of events over time is captured by the DVS sensor. For example the events occurring between T=1 and T=2, the total events detected within this time interval are equally divided into frames with time intervals (in our case we used 32 frames). This is done as Spiking Neural Networks

work with time-based data.

Parameter	Value
Number of Clients	6
Number of Epochs (NMNIST)	3
Number of Epochs (DVS Gesture)	20
Batch Size	64
Number of Middle Servers	3
Number of Global Rounds	25
Number of Local Rounds	1
Learning Rate	0.002
Beta	0.5
Gradient Function	fast_sigmoid(slope=25)
Loss Function	Mean Square Error

Table 3.3.4: Hyperparameters for Neuromorphic Dataset

Dataset	Train	Test
EMNIST (Balanced)	112,800	18,800
MNIST	60,000	10,000
CIFAR-10	50,000	10,000
DVS128 Gesture	1,078	264
NMNIST	6,000	500

Table 3.3.5: Dataset Distribution

Dataset Distribution: In our Federated Learning Architecture, all clients received an equal number of data from the datasets used. Also, all clients had equal proportions of the dataset classes for even distribution.

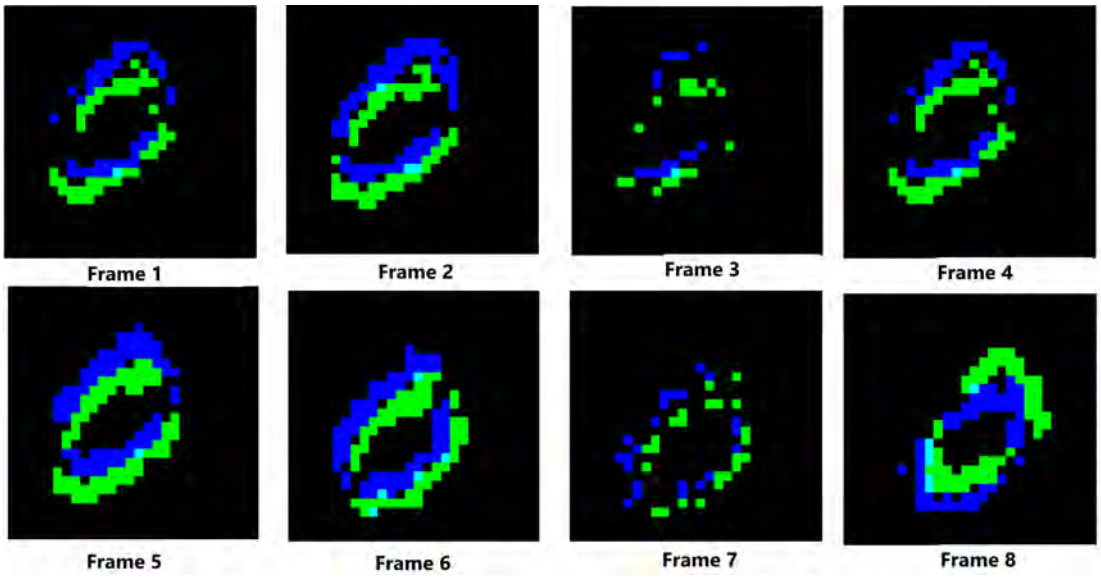

Figure 3.3.6: Framing samples for NMNIST digit "0"

3.4 Proposed Federated Architectures

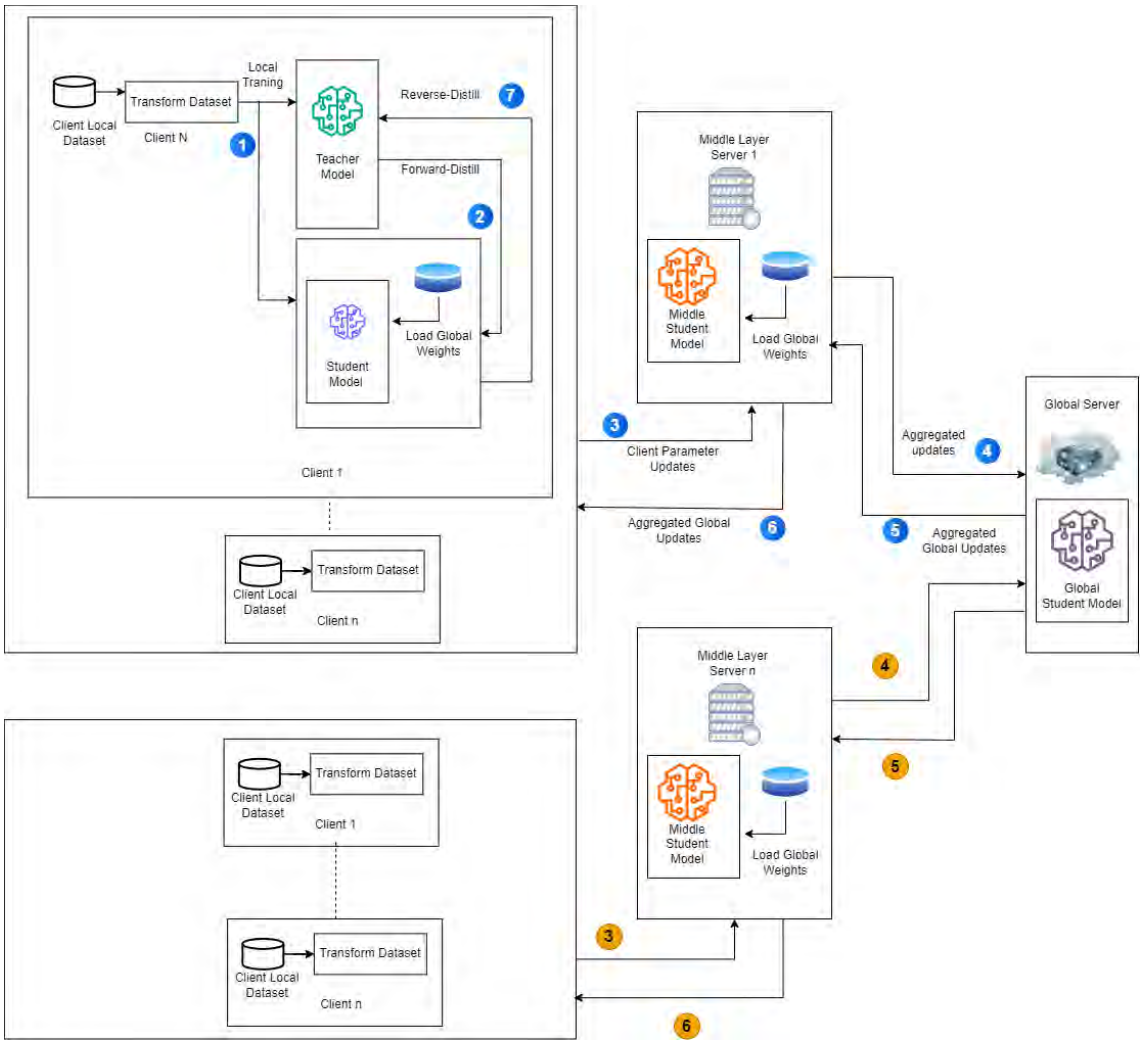

Figure 3.4.1: Proposed TRI Layer Federated Learning Architecture with Forward-Reverse KD

In Figure 3.4.1 the Global Student model weights are initialized. The middle layer student model and The client student model is initialized with the global student model weights. Each client student model has their own private dataset. In each global round, the Client teacher model trains on the associated client's local dataset. The teacher model distills its knowledge to the client student model. In knowledge distillation, the predicted output or soft labels of the teacher are shared with a student model, which then tries to match the teacher's predictions but also tries to classify the data based on the actual label. After knowledge distillation from the teacher model to the student model, the updated parameters of each client student models are sent to their assigned middle server model for aggregation. Each middle server aggregates the student model of K number of assigned clients. All the middle servers then pass the aggregated middle student model parameters to the global server for final aggregation. The global server then passes the updated global student model parameters to the middle layer models, and also to the client's student models. After that, the client student model uses the process of reverse

distillation, where it distills its knowledge back to the client teacher model. This allows the teacher model to have knowledge about global updates, which improves its performance. Again in the next global round the above steps are repeated.

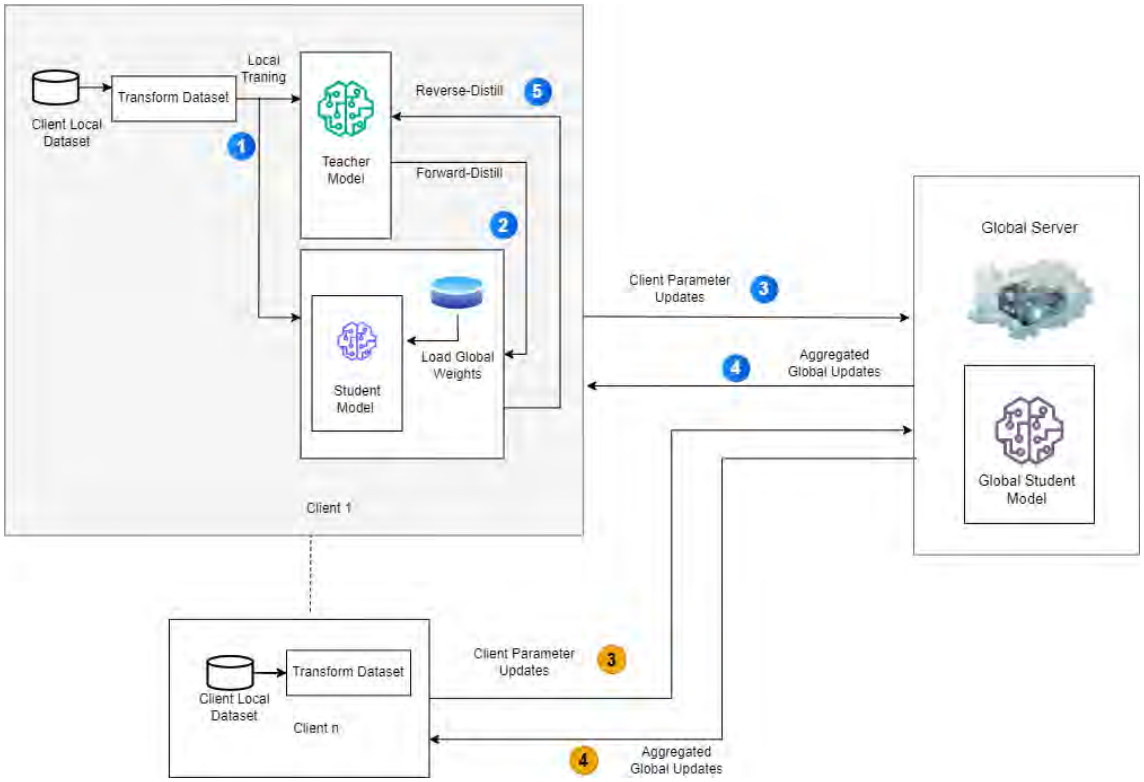

Figure 3.4.2: BI Layer Federated Learning Architecture with Forward-Reverse KD

In Figure 3.4.2 the global model is initialized along with the client model. All clients have their private datasets. In each global round, the more capable and larger teacher model trains on the associated client's local dataset. The teacher model distills its knowledge to the client's student model, which the client contains. In knowledge distillation, the predicted output or soft labels of the teacher are shared with a student model, which then tries to match the teacher's predictions but also tries to classify the data based on the actual label. After knowledge distillation from the teacher model to the student model, the updated model parameters of each client are sent to the global model for aggregation. After this, the global updates are sent back to each client. Once the client student model receives the updated global parameters, it uses the process of reverse distillation, where it distills back its knowledge to the teacher model. This allows the teacher model to have partial knowledge of updated global knowledge, which helps to improve performance in the next global round.

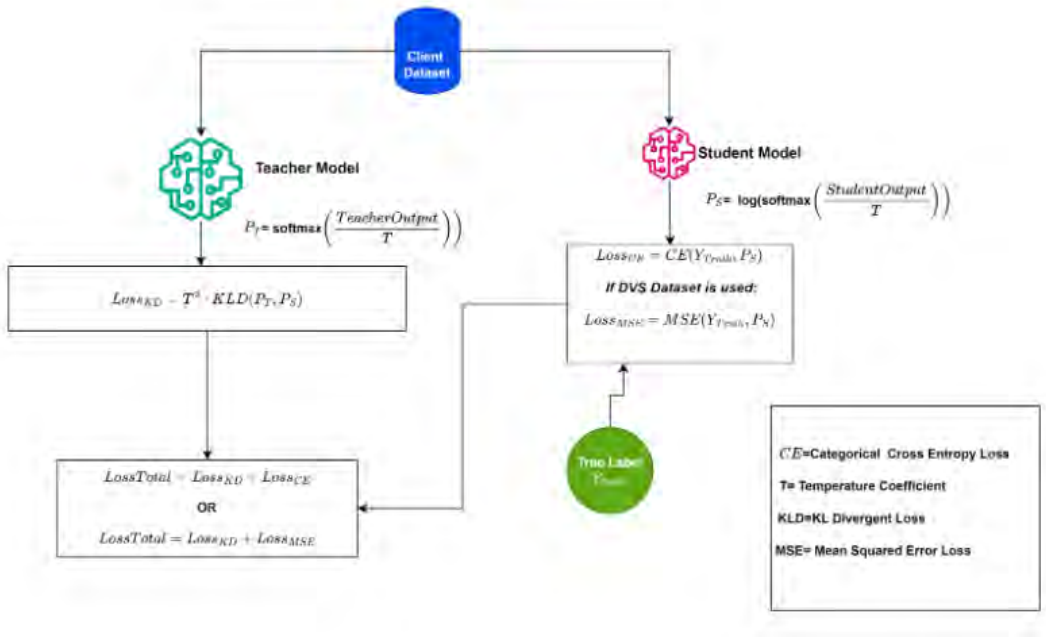

Figure 3.4.3: Knowledge Distillation Setup

$$KLD = T^2 \cdot \left(\frac{1}{N}\right) \cdot \sum_{i=1}^{N} p_T(i) \cdot (\log(p_T(i)) - p_S(i))$$

From Figure 3.4.3, we can see the teacher-client model is being saved using the client's dataset. The teacher model here is a powerful, well-performing model that is trained using large datasets using conventional methods. The student model is typically a smaller, less complex model which is trying to mimic the teacher model. The loss function in the student model is usually categorical cross-entropy loss but for DVS datasets the loss function is mean squared error and the loss functions here compare the student model's predictions with the actual predictions.

After training, the output from the teacher model is processed through a softmax function, which converts raw output scores to probabilities, but it uses a temperature coefficient T. This is the case for the student model as well when reverse distillation happens. The temperature coefficient is very crucial because it creates a smooth probability distribution over the output classes. As the teacher model's output is processed through a softmax with a temperature factor T, a high T makes the output probabilities smoother, highlighting less confident predictions. This learning also happens inversely when the student model becomes the teacher and the teacher model becomes the student[5].

The knowledge distillation loss LossKD is calculated using the Kullback-Leibler Distribution (KLD), which measures the difference between probability distributions from the teacher model (PT) and the student model (PS). LossTotal is the summation of LossKD + LossCE or LossKD + LossMSE and it ensures that the student

model learns from the teacher model's generalized knowledge as well as learns to correctly predict the true labels from the dataset [5].

3.5 Models we used in our experiment

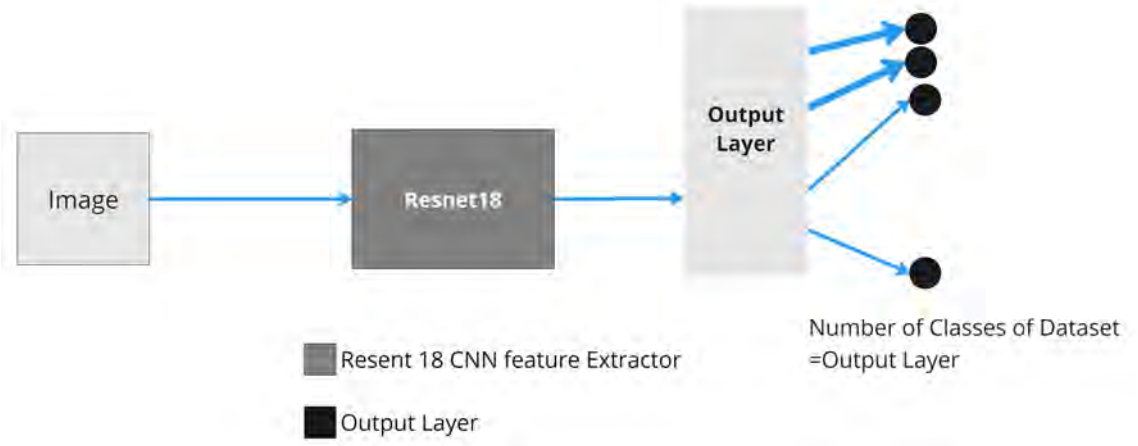

Figure 3.5.1: Teacher Model for Non-Neuromorphic dataset using CNN and CNN+SNN

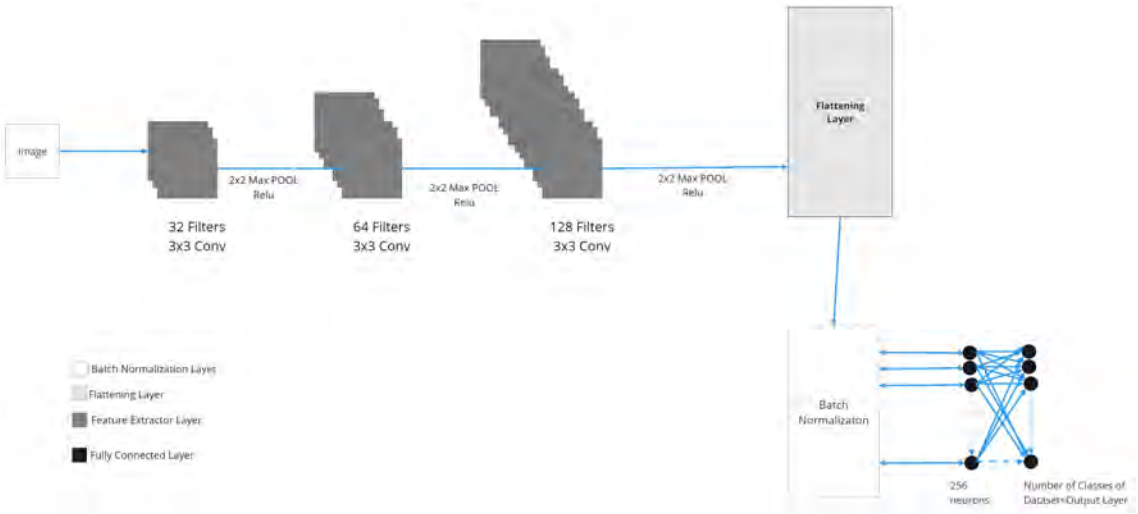

Figure 3.5.2: Student Model for Non-Neuromorphic dataset using CNN and CNN+SNN

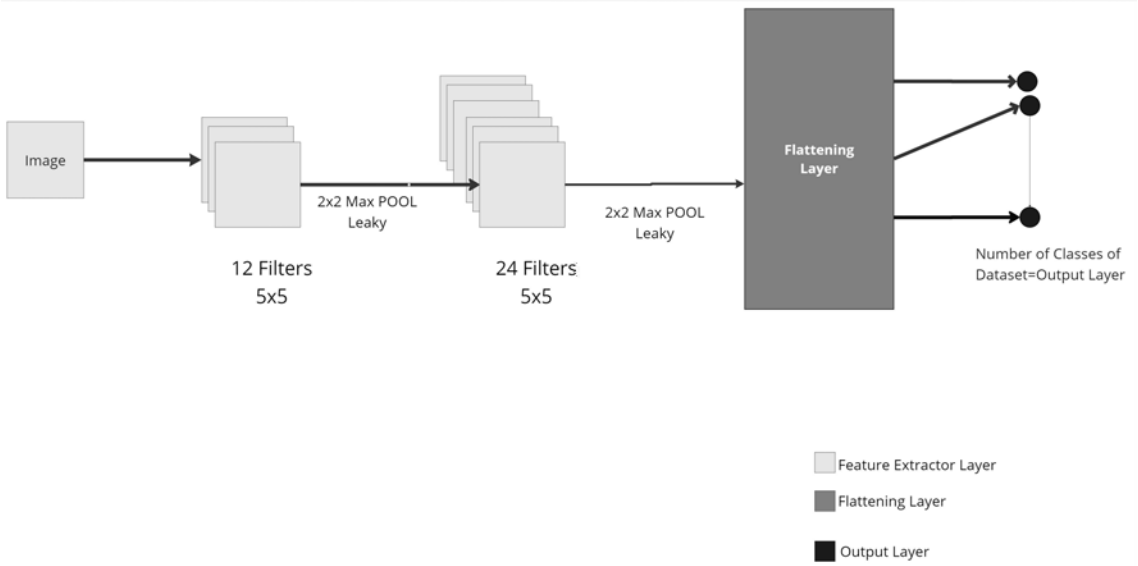

Figure 3.5.3: Neuromorphic Dataset Student Model using CNN+SNN

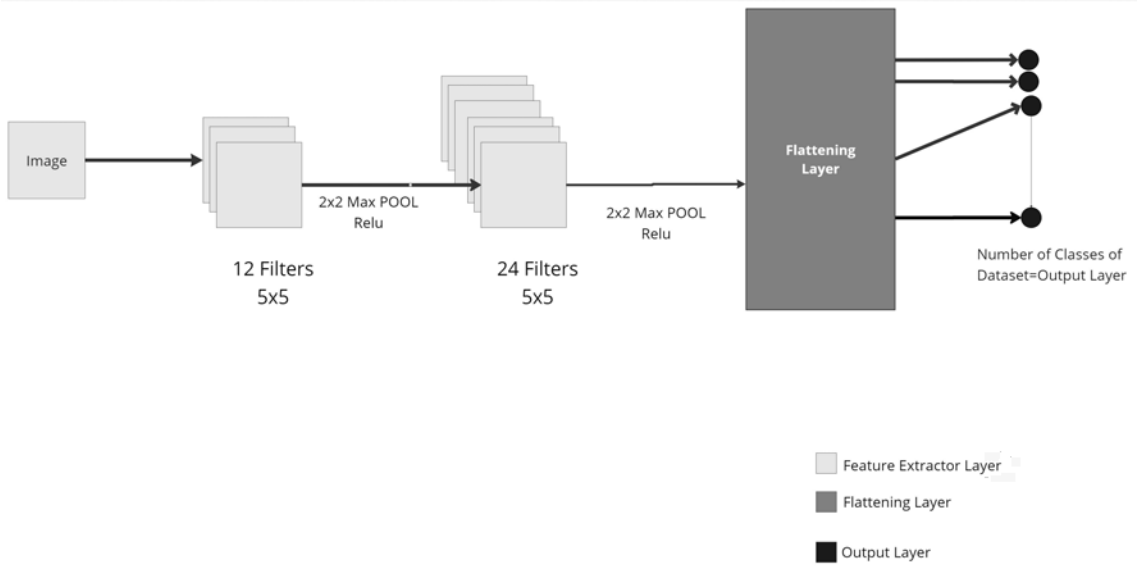

Figure 3.5.4: Neuromorphic Dataset Student Model using CNN

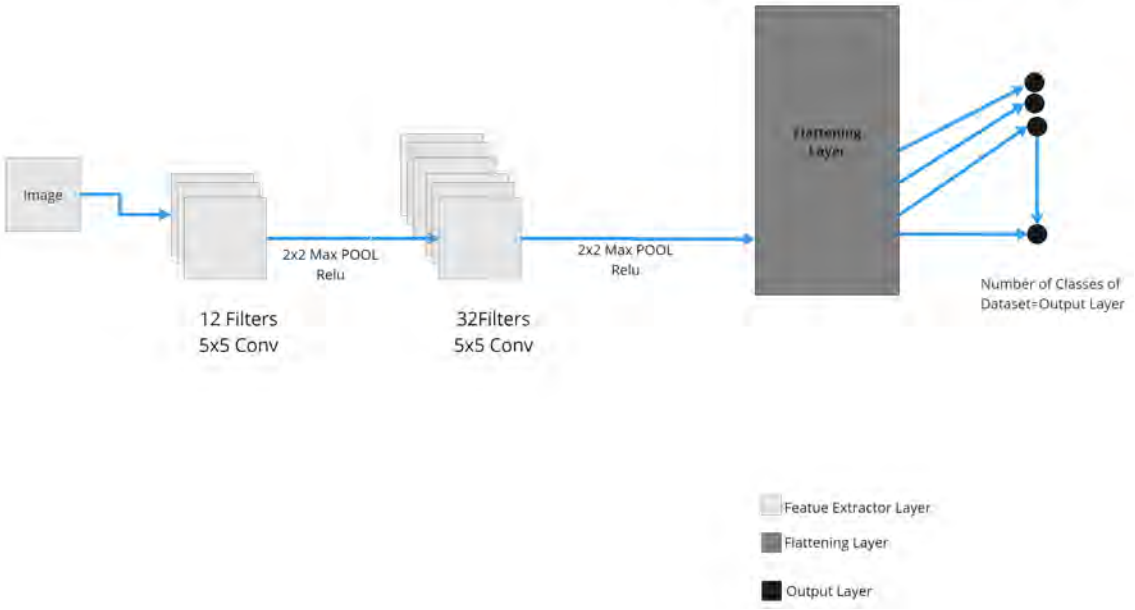

Figure 3.5.5: Neuromorphic Dataset Teacher Model using CNN

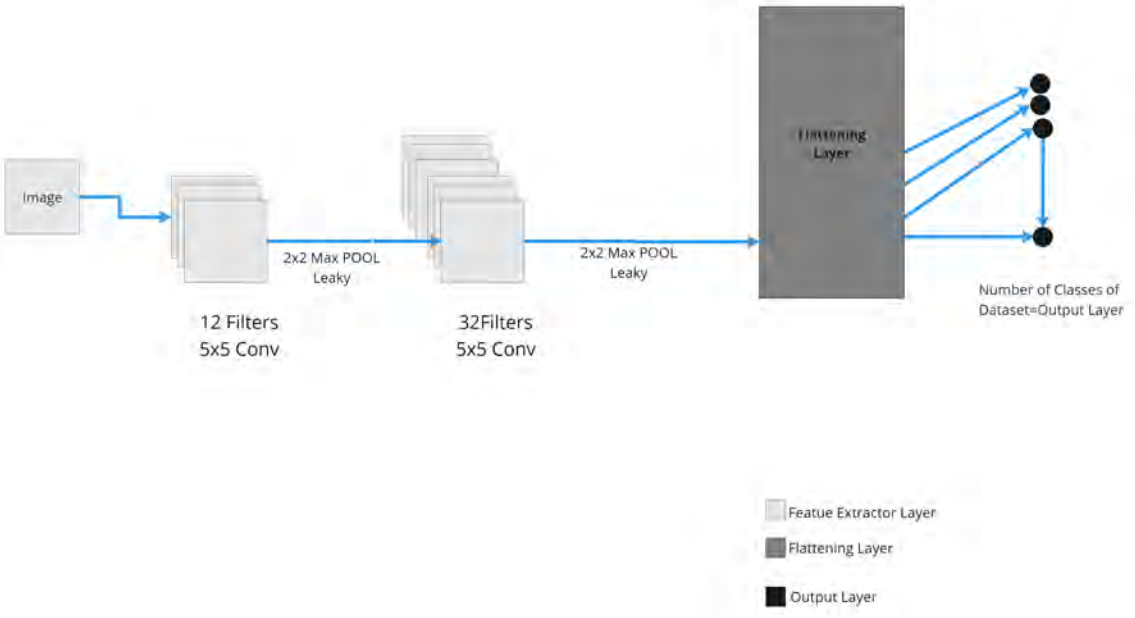

Figure 3.5.6: Neuromorphic Dataset Teacher Model using CNN+SNN

Non Neuromorphic Teacher-Student Model:
We used two types of architecture for our deep learning models. Each client node consisted of a larger teacher model and a smaller student model.

We used a modified Resnet-18 feature extractor for the teacher model trained on non-neuromorphic dataset in Figure 3.5.1. The use of ResNet-18 is significant for efficient feature extraction through its deep architecture, allowing effective handling of complex datasets. Its residual connections, such as skip connections, enhance training stability and mitigate vanishing gradient issues. ResNet-18 has proven strong

performance on image datasets like MNIST, CIFAR-10, and EMNIST, and demonstrates good generalization, making it ideal for transferring knowledge in distillation [7][10]. We adapted the first convolutional layer of the Resnet-18 to the dimensions of the specific dataset used. For example, the EMNIST dataset, which is a grayscale image, was modified to accept one color channel instead of 3. Then we replaced the last layer with our output layer for classification for the specific dataset and its total number of classes.

For student models, we use a custom feature extractor model that is lightweight and better suited for student models that have power and device constraints. Both architectures used the models defined in Figure 3.5.2. Here we first used a convolutional layer of 32 filters, followed by 64 filters, and then finally 128 filters. Kernel Size was 3X3 Dimension and used a padding and stride value of 1. After each convolution, we used maxpool with a dimension of 2x2 with stride and padding set to 2 and 0, respectively. We did batch normalization after flattening and replaced the fully connected layer with a spiking neural network in CNN+SNN and with an Artificial artificial neural network in CNN. Here, The first hidden layer has 256 neurons. Followed by the output layer. The number of neurons in the output layer is the number of classes the dataset is trained on.

Neuromorphic Teacher-Student Model:
Figures 3.5.5 and 3.5.6 illustrate the teacher model using CNN and CNN+SNN, respectively. Figures 3.5.3 and 3.5.4 illustrate the student models using CNN+SNN and CNN respectively. Teacher models have a first-layer convolution with 12 filters with 5x5 dimensions. This is followed by a convolution of 32 filters with 5x5 dimensions again. For student models, we have 12 filters followed by 24 filters with 5x5 dimensions. We used a 2x2 max pool after every convolution. For CNN+SNN we used Leaky, and for CNN we used the Relu function.

After feature extraction, the models use a flattening layer to convert 2D feature maps into 1D vectors, which are passed to the fully connected layer. This final layer maps features to output neurons representing the dataset's classes. Both models apply SNN concepts to process data and generate predictions.

Chapter 4

Result and Analysis

4.1 Non Neuromorphic Datasets

Dataset	KD / Not	Architecture	Model Type	F1-Score	Precision	Recall	Accuracy (%)
MNIST	Without KD	BI	CNN	0.99276	0.99297	0.99260	99.27
			CNN+SNN	0.99261	0.99278	0.99249	99.26
		TRI	CNN	0.99303	0.99321	0.99289	99.30
			CNN+SNN	0.99311	0.99328	0.99296	99.31
	KD	BI	CNN	0.99670	0.99671	0.99670	99.67
			CNN+SNN	0.99556	0.99561	0.99552	99.56
		TRI	CNN	0.99630	0.99630	0.99630	99.10
			CNN+SNN	0.99594	0.99599	0.99589	99.60
CIFAR10	Without KD	BI	CNN	0.75775	0.75788	0.75970	75.97
			CNN+SNN	0.75628	0.75943	0.75730	75.73
		TRI	CNN	0.74574	0.74867	0.74790	74.79
			CNN+SNN	0.73880	0.74338	0.73880	73.88
	KD	BI	CNN	0.77813	0.77908	0.77840	77.80
			CNN+SNN	0.76907	0.77039	0.76990	76.99
		TRI	CNN	0.77756	0.77816	0.77820	77.80
			CNN+SNN	0.77140	0.77150	0.77190	77.19
EMNIST	Without KD	BI	CNN	0.88231	0.88621	0.88330	88.33
			CNN+SNN	0.88505	0.88703	0.88703	88.57
		TRI	CNN	0.88114	0.88431	0.88170	88.17
			CNN+SNN	0.88380	0.88616	0.88436	88.44
	KD	BI	CNN	0.89049	0.89103	0.89128	89.13
			CNN+SNN	0.89049	0.89050	0.89106	89.11
		TRI	CNN	0.89261	0.89289	0.89293	89.29
			CNN+SNN	0.88947	0.89040	0.88989	88.99

Table 4.1.1: Performance Comparison across Non Neuromorphic Datasets with KD and without KD

Here Table 4.1.1 we got results on some non-neuromorphic datasets in terms of Precision, Recall, F1-score, and Accuracy. For MNIST, there is a negligible difference between distillation and no distillation. For Cifar 10, knowledge distillation outperforms without knowledge distillation. This applies to both CNN and CNN+SNN (bi and tri layers). For EMNIST, knowledge distillation again outperforms without knowledge distillation for both CNN and CNN+SNN. When compared between CNN and CNN+SNN, CNN comes out on top multiple times. Now let's analyze deeper. As we said earlier, we got comparatively negligible difference in results for the MNIST dataset. So we are not considering these results so much. Now let's talk about the CIFAR10 dataset. For only CNN, the Bi-layer with distillation performs

the same as the Tri-layer with distillation. But for the CNN+SNN, the Tri-layer with distillation outperforms the Bi-layer with distillation. Now let's discuss the EMNIST dataset. For only CNN, the Tri-layer with distillation outperforms the Bi-layer with distillation. But for the CNN+SNN, the Bi-layer with distillation outperforms the Tri-layer with distillation. So overall we can say that for non-neuromorphic datasets, using knowledge distillation has achieved better results.

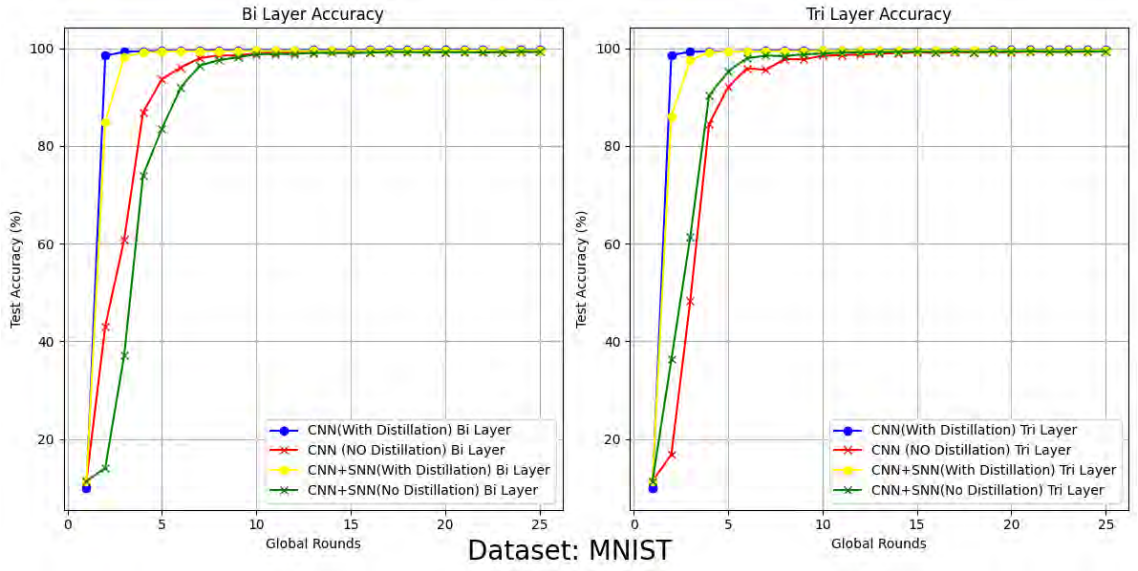

Figure 4.1.2: Accuracy comparisons of MNIST

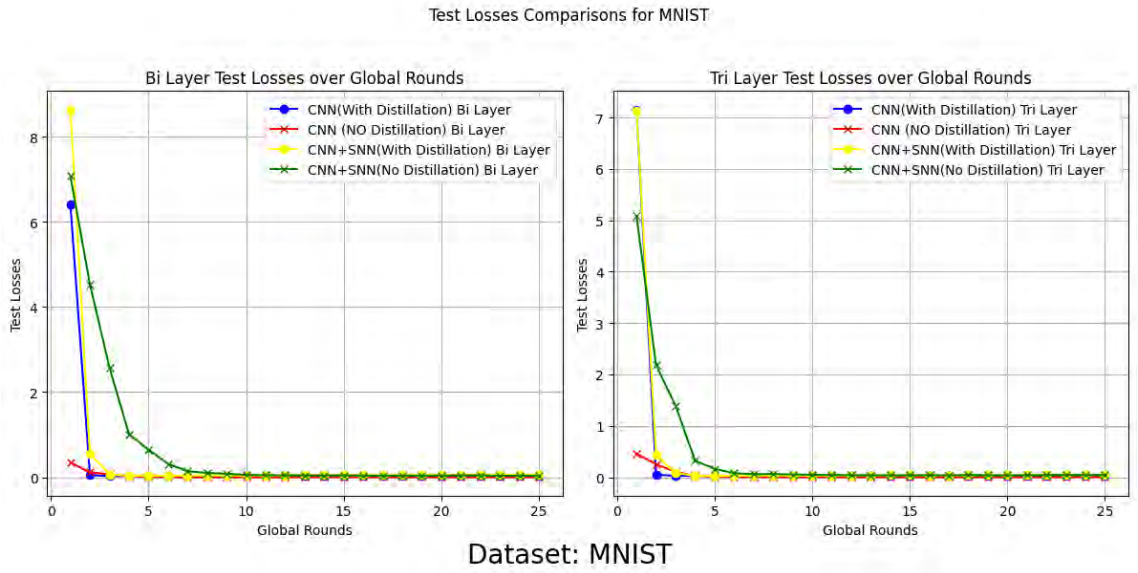

Figure 4.1.3: Loss comparisons of MNIST

All the architectures here performed well with very high accuracy compared to other datasets. The CNN+SNN in both models initially had higher loss compared to CNNs, but their losses dropped quickly, and all of them converged.

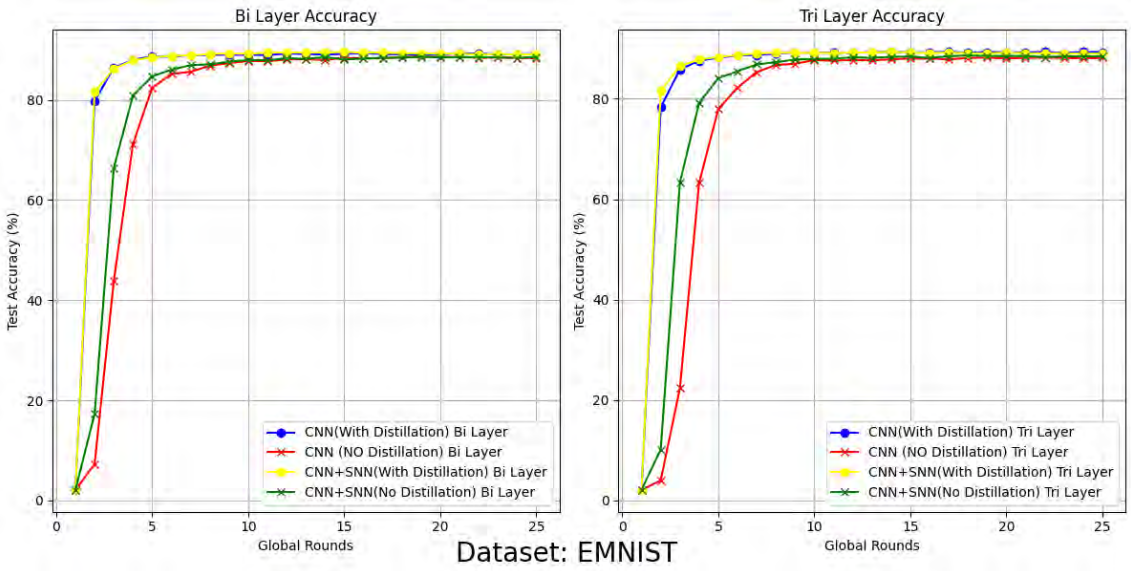

Figure 4.1.4: Accuracy comparisons of EMNIST

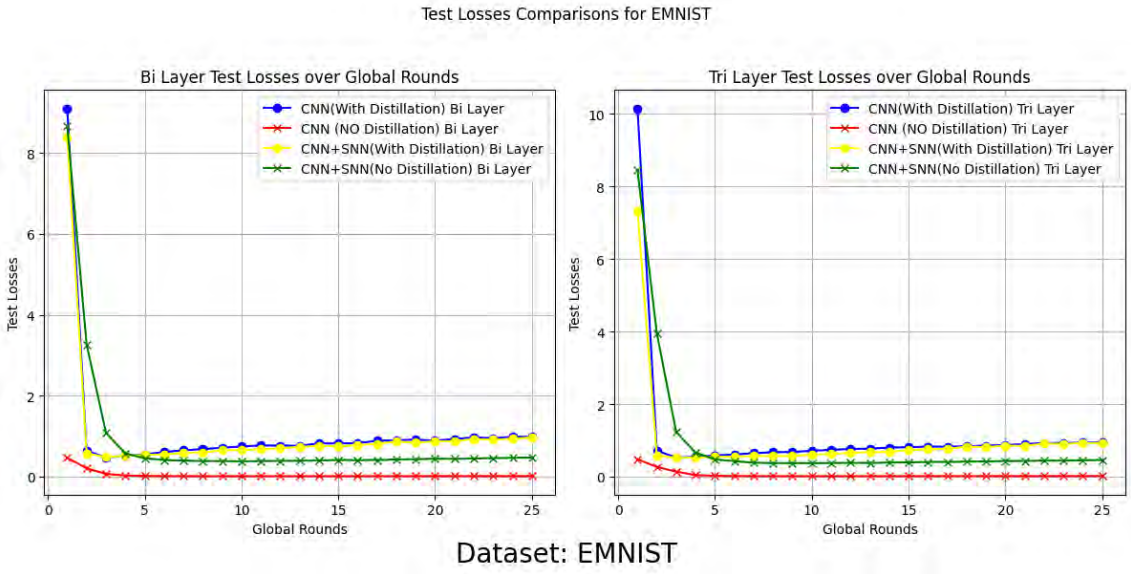

Figure 4.1.5: Loss comparisons of EMNIST

All models performed similarly as shown in Figure 4.1.4, with relatively higher accuracy but less overall accuracy than MNIST. Both models with distillation initially report higher losses in Figure 4.1.5 compared to those without distillation. On the other hand, no distillation has lower loss values and learns slowly, while distillation models learn fast and all converge.

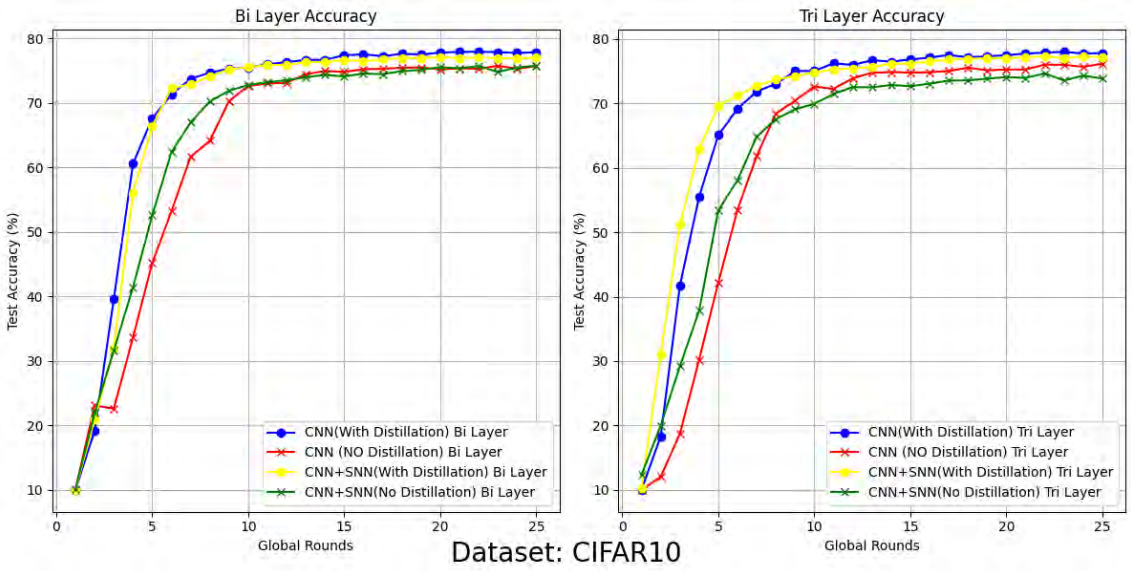

Figure 4.1.6: Accuracy comparisons of CIFAR 10

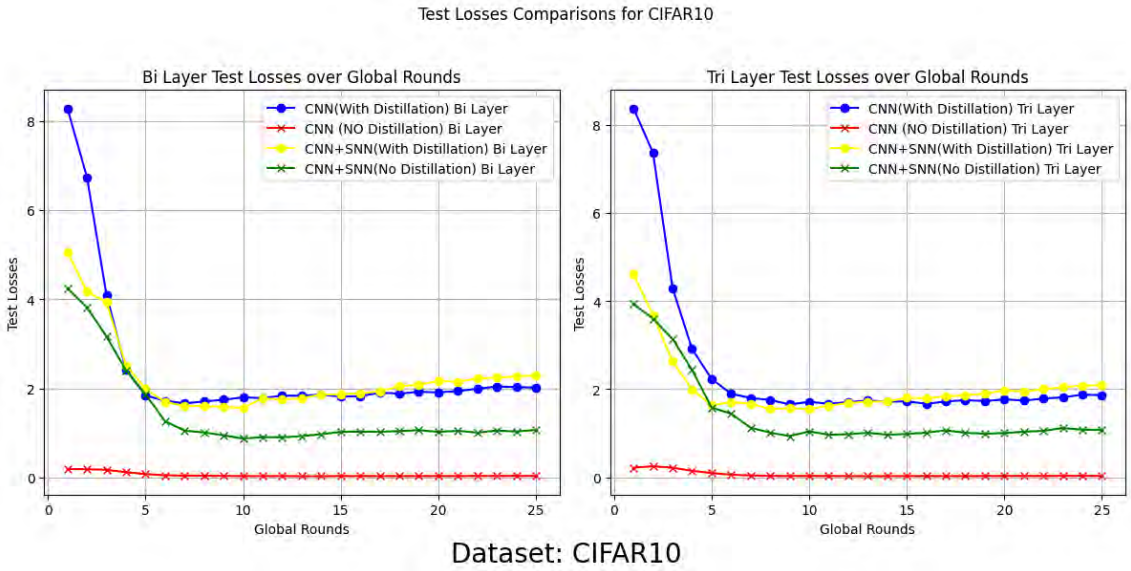

Figure 4.1.7: Loss comparisons of CIFAR10

In Figure 4.1.6 CNN with Distillation and CNN+SNN with Distillation perform better than CNN+SNN Without Distillation and CNN Without Distillation. This can be attributed to knowledge distillation, where the larger, more capable teacher model distill its knowledge and the student model improves its accuracy. Moreover, CNN+SNN models always perform similar or lesser than CNN here. This may be due to spiking neural networks being more suited to learning from event-based data.

Loss is high for both distillation models in Figure 4.1.7 regardless of the architecture used, compared to without distillation. But CNN+SNN models for both cases show higher loss values compared to CNN models.

4.2 Neuromorphic Datasets

Dataset	KD / Not	Architecture	Model Type	F1-Score	Precision	Recall	Accuracy (%)
DVS Gesture	Without KD	BI	CNN	0.8234	0.8473	0.8242	79.3
			CNN+SNN	0.8694	0.8816	0.8711	87.11
		TRI	CNN	0.7481	0.8087	0.7656	75.0
			CNN+SNN	0.8799	0.9018	0.8788	87.88
	KD	BI	CNN	0.7562	0.8008	0.7656	75.78
			CNN+SNN	0.9097	0.9165	0.9129	91.29
		TRI	CNN	0.7458	0.7865	0.7656	75.39
			CNN+SNN	0.8666	0.8869	0.8674	86.74
N-MNIST	Without KD	BI	CNN	0.9498	0.9547	0.9464	94.2
			CNN+SNN	0.9699	0.9705	0.9700	97.0
		TRI	CNN	0.9526	0.9532	0.9527	94.61
			CNN+SNN	0.9556	0.9564	0.9560	95.6
	KD	BI	CNN	0.9538	0.9545	0.9538	94.74
			CNN+SNN	0.9659	0.9668	0.9660	96.6
		TRI	CNN	0.9440	0.9450	0.9440	93.57
			CNN+SNN	0.9638	0.9645	0.9640	96.4

Table 4.2.1: Performance Comparison across DVS Gesture and N-MNIST Datasets with KD and without KD

In Table 4.2.1 we can see the results in the DVS Gesture and NMNIST dataset, based on precision, recall, F1-score, and accuracy, that the combination of CNN and SNN outperforms the standard CNN. In the DVS Gesture dataset, only CNN in the Bi-layer setup without distillation performs better than the Bi-layer with distillation. And Tri-layer with distillation performs better than Tri-layer without distillation. For CNN+SNN, the Bi-layer with distillation performs better than the Bi-layer without distillation. And Tri-layer without distillation performs better than Tri-layer with distillation. In NMNIST dataset, only CNN in the Bi-layer setup with distillation performs better than Bi-layer without distillation. And Tri-layer without distillation performs better than Tri-layer with distillation. For CNN+SNN, Bi-layer without distillation performs better than Bi-layer with distillation. And Tri-layer with distillation performs better than Tri-layer without distillation.

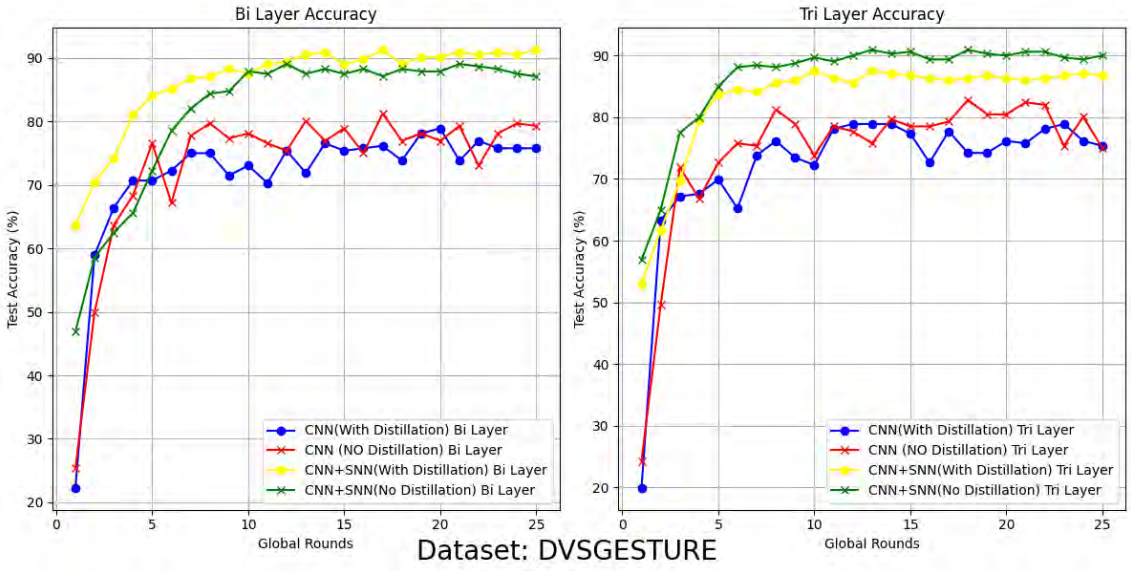

Figure 4.2.2: Accuracy comparisons of DVS Gesture

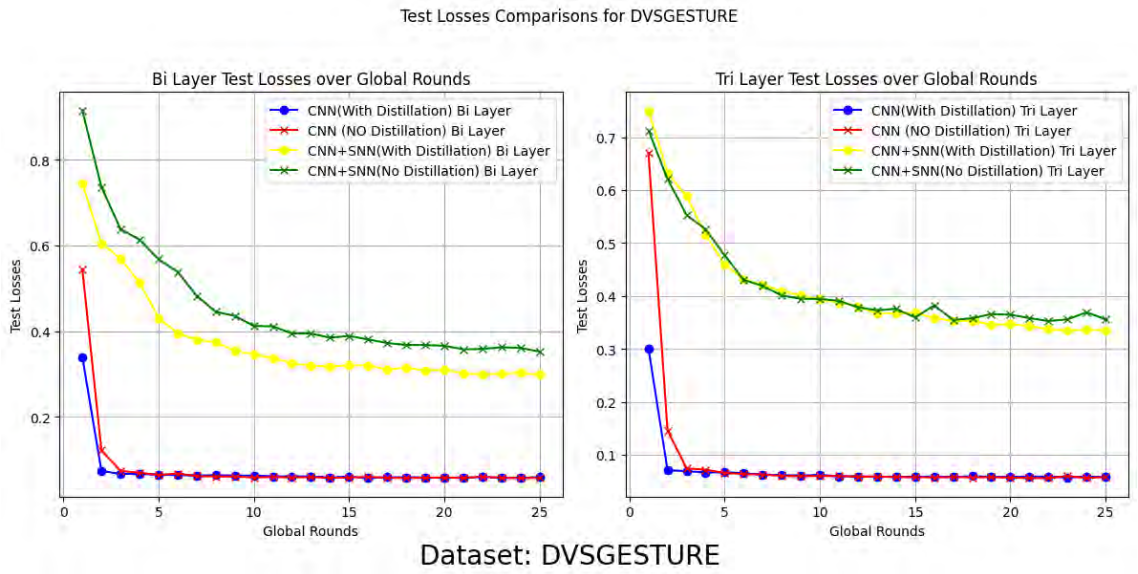

Figure 4.2.3: Loss comparisons of DVS Gesture

Here Figure 4.2.2 CNN+SNN with both distillation and without Distillation Performed better than CNN with Distillation and without Distillation. As DVS Gesture is converted into frames, Spiking Neural Networks work best with event-based data. So it can learn more efficiently. On the other hand, ANN works best with continuous input, so as a result, it takes the 32 frames as 32 static images as input, and then, in the final layer, based on the learned parameters, associates a classification probability for them. But it does not take into consideration the time like SNNs can.

From the Figure 4.2.3 we can see that CNN+SNN models have shown higher loss compared to CNN, as MSE count loss used in CNN+SNN to measure spike loss instead of using basic mean square error loss.

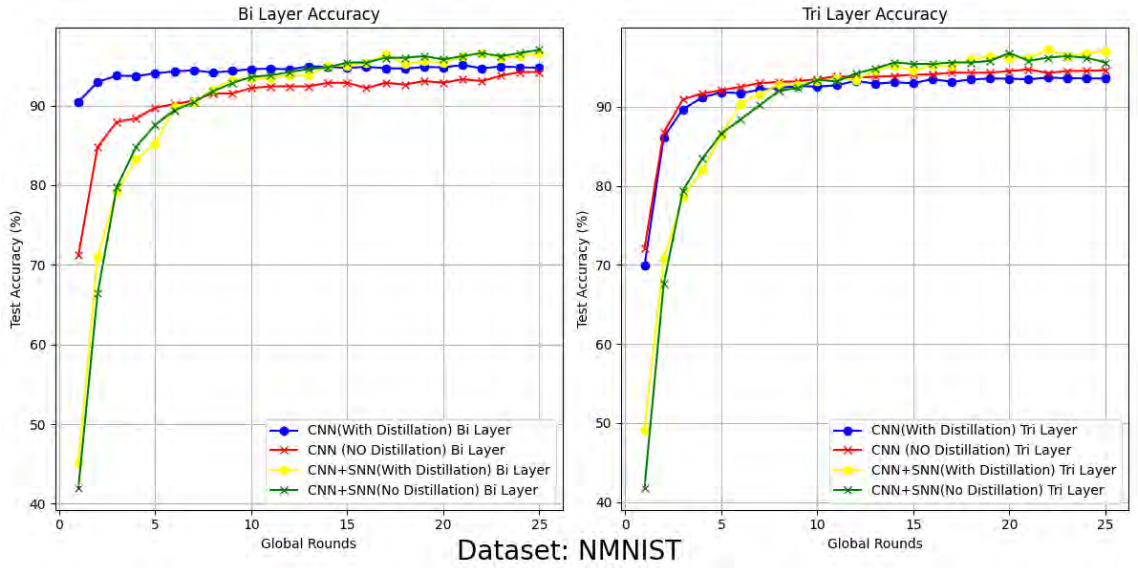

Figure 4.2.4: Accuracy comparisons of NMNIST

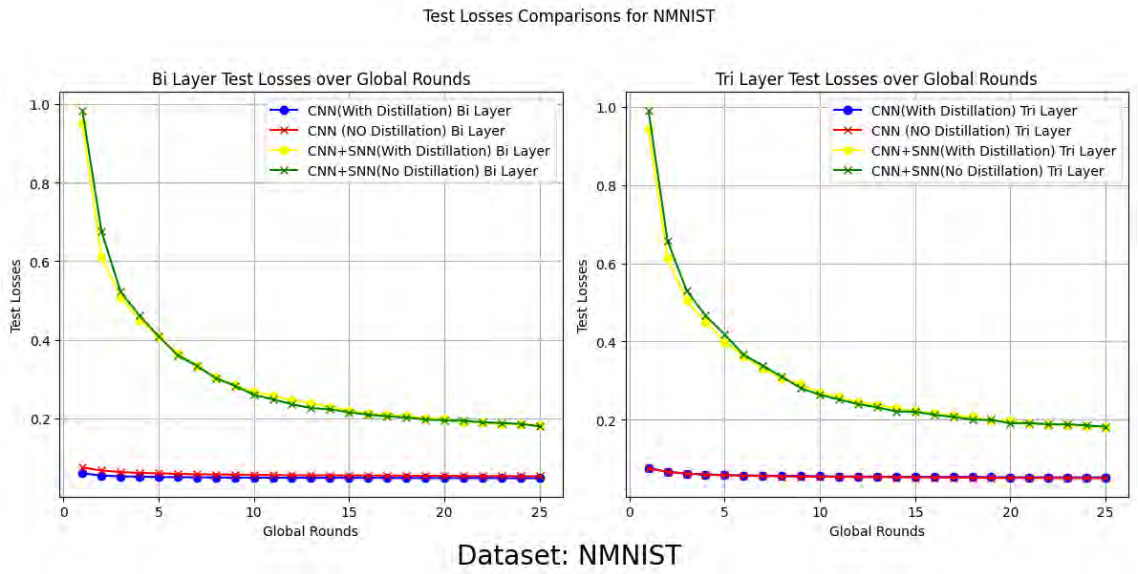

Figure 4.2.5: Loss comparisons of NMNIST

Here in Figure 4.2.4 CNN+SNN with and without distillation perform better than CNN models in both Bi and Tri Layer.This can be attributed to the SNNs capability of processing spikes over time and working better with event based data compared to CNNs. Although losses in Figure 4.2.5 may be high for CNN+SNN compared to only CNN, it converges fast as it learns fast.

4.3 Result Analysis Based on FLOPs, Parameters, and Size

Datasets	**Shape**	**Teacher Model**	**Flops G**	**Params M**	**Size MB**	**Student Model**	**Flops G**	**Params M**	**Size MB**
MNIST	Input = 1 x 28 x 28 Output = 10	ResNet18	0.458 G	11.173 M	42.7 MB	Custom Model	0.008 G	0.818 M	3.1 MB
CIFAR10	Input = 3 x 32 x 32 Output = 10	ResNet18	0.558 G	11.174 M	42.7 MB	Custom Model	0.012 G	1.277 M	4.9 MB
EMNIST	Input = 1 x 28 x 28 Output = 47	ResNet18	.458 G	11.192 M	42.8 MB	Custom Model	0.008 G	0.827 M	3.2 MB
DVS Gesture	Input = 2 x 32 x 32 N_frame = 32, Output = 11	Custom (32 x 5 x 5)	0.046 G	0.019 M	83.1 KB	Custom (24 x 5 x 5)	0.038 G	0.014 M	63 KB
DVS NMNIST	Input = 2 x 32 x 32 N_frame = 32, Output = 10	Custom (32 x 5 x 5)	0.046 G	0.018 M	75.3 KB	Custom (24 x 5 x 5)	0.038 G	0.014 M	58 KB

Table 4.3.1: Comparison between Teacher and Student models in terms of FLOPs, Parameters, and Size across different datasets.

4.4 Server Aggregation Time

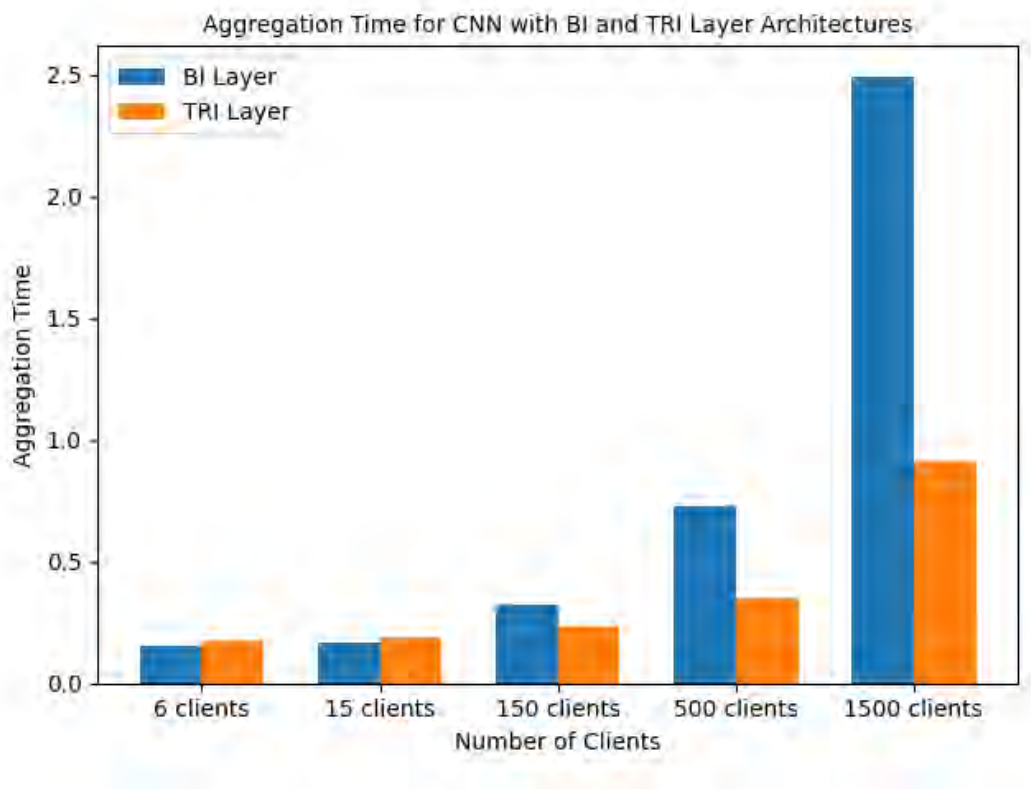

Figure 4.4.1: BI layer vs TRI layer Aggregation time in CNN

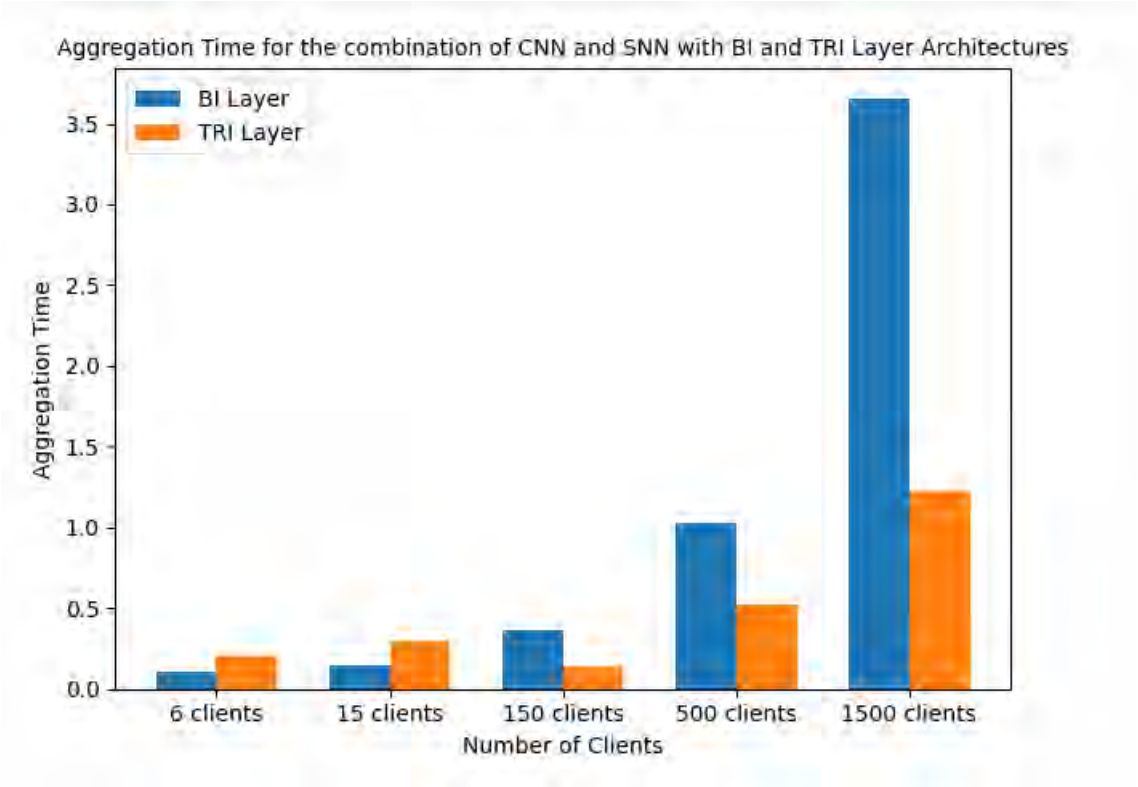

Figure 4.4.2: BI layer vs TRI layer Aggregation time in CNN+SNN

Here we can see the two graphs of server aggregation time. One for standard CNN Figure 4.4.1, and another for the combination of CNN and SNN Figure 4.4.2 models. Both graphs highlight the comparison between the bi-layer and the tri-layer. When considering 6 clients, the aggregation time is comparatively negligible difference between bi-layer and tri-layer. But when the client counts increased, the comparative result was highlighted. Like when we are dealing with 150 clients, the tri-layer takes less aggregation time than the bi-layer. Continuously, when our client count is 1500, the tri-layer takes approximately 3 times less aggregation time than the bi-layer. This is because the global server aggregates the updates parallelly across multiple middle servers and each middle server is connected with clients.

Chapter 5

5.1 Conclusion

In this work, we proposed TRI-FED-RKD, a novel framework for federated learning that leverages a tri-layer hierarchical aggregation-based Federated architecture. We integrated forward and reverse knowledge distillation within CNN and CNN-SNN model. Our approach effectively improves accuracy for non-neuromorphic datasets using CNN in both teacher and student models. Experimental evaluations on neuromorphic datasets, such as DVS Gesture and NMNIST, demonstrate the superior performance of the combination of CNN and SNN models over traditional CNN.

While one of the primary goals was to improve the energy efficiency of TRI-FED-RKD, we were unable to directly measure real power consumption due to the lack of access to neuromorphic chip devices. Neuromorphic chips are specifically designed to efficiently handle spiking neural networks (SNNs). SNN operates using dynamic spikes and timestamps. As our research employs these SNNs and deals with dynamic neuromorphic datasets, running experiments on conventional GPUs, which do not replicate the behavior of neuromorphic hardware. This limited our ability to report accurate energy consumption figures. Nonetheless, we remain confident that our proposed architecture, utilizing SNNs, would significantly reduce energy consumption on neuromorphic hardware, as SNNs are inherently well-suited for such dynamic datasets.

5.2 Future work

In future work, we aim to extend our research by evaluating TRI-FED-RKD on actual neuromorphic chip devices to validate our expectations regarding energy efficiency. By conducting experiments on these specialized chips, we will be able to more accurately assess power consumption and further refine our model for optimal performance in real-world edge environments. This step will not only establish our architecture's practical applicability in power-constrained scenarios but also enable more comprehensive comparisons of energy efficiency between the combination of CNN and SNN models and the traditional CNN model.

Bibliography

[1] W. Maass, "Networks of spiking neurons: The third generation of neural network models," Electronic Colloquium on Computational Complexity, Tech. Rep. TR96, 1996.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[4] A. Grüning and S. M. Bohte, "Spiking neural networks: Principles and challenges," in *Proceedings of the 22nd European Symposium on Artificial Neural Networks (ESANN 2014)*, Bruges, 2014.

[5] G. Hinton, O. Vinyals, and J. Dean, "Distilling knowledge in a neural network," *arXiv*, 2015. [Online]. Available: https://arxiv.org/abs/1503.02531.

[6] G. Orchard, G. Cohen, A. Jayawant, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, no. 437, 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[8] A. Amir, B. Taba, D. R. Berg, *et al.*, "A low-power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7243–7252.

[9] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: An extension of mnist to handwritten letters," *arXiv*, 2017. [Online]. Available: http://arxiv.org/abs/1702.05373.

[10] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. DOI: 10.1109/cvpr.2017.243.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[12] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv.org*, Dec. 2018. [Online]. Available: https://arxiv.org/abs/1812.07210.

[13] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv*, 2018. eprint: 1806.00582.

[14] P. Kairouz, H. B. McMahan, B. Avent, *et al.*, "Advances and open problems in federated learning," *arXiv*, 2019. eprint: 1912.04977.

[15] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019. [Online]. Available: https://arxiv.org/abs/1910.03581.

[16] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019. DOI: 10.1109/MSP.2019.2931595.

[17] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019. DOI: 10.1038/s41586-019-1677-2.

[18] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Networks*, vol. 111, pp. 47–63, 2019. DOI: 10.1016/j.neunet.2018.12.002.

[19] K. M. J. Rahman, M. S. Hossain, B. M. Alzahrani, M. A. Mahmud, and A. Ali, "Challenges, applications and design aspects of federated learning: A survey," *IEEE Access*, vol. 9, pp. 124 682–124 700, 2021. DOI: 10.1109/ACCESS.2021.3111118.

[20] Y. Venkatesha, Y. Kim, L. Tassiulas, and P. Panda, "Federated learning with spiking neural networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 6183–6194, 2021. DOI: 10.1109/TSP.2021.3121632.

[21] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural computation*, vol. 33, no. 4, pp. 899–925, 2021. DOI: 10.1162/neco_a_01363.

[22] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, PMLR, Jul. 2021, pp. 12 878–12 889.

[23] Z. Liu, Q. Zhan, X. Xie, B. Wang, and G. Liu, "Federal snn distillation: A low-communication-cost federated learning framework for spiking neural networks," *Journal of Physics: Conference Series*, vol. 2216, no. 1, p. 012 078, 2022. DOI: 10.1088/1742-6596/2216/1/012078.

[24] H. Wen, Y. Wu, J. Hu, Z. Wang, H. Duan, and G. Min, "Communication-efficient federated learning on non-iid data using two-step knowledge distillation," *IEEE Internet of Things Journal*, vol. 10, no. 19, pp. 17 307–17 322, 2023. DOI: 10.1109/JIOT.2023.3276865.

[25] Y. Zhao and J. Chu, "The robustness of spiking neural networks in communication and its application towards network efficiency in federated learning," *arXiv*, 2023. DOI: 10.48550/arXiv.2409.12769.

[26] F. Wang, M. C. Gursoy, and S. Velipasalar, "Feature-based federated transfer learning: Communication efficiency, robustness and privacy," *arXiv.org*, May 2024. [Online]. Available: https://arxiv.org/abs/2405.09014.