Logo

Description automatically generated

FEDERATED LEARNING FOR IMAGE CLASSIFICATION BASED ON DEEP LEARNING

by

Khanh Le Dinh Viet (K.L.D.V.)

Khiem Le Ha (K.L.H.)

THE FPT UNIVERSITY HO CHI MINH CITY

FEDERATED LEARNING FOR IMAGE CLASSIFICATION BASED ON DEEP LEARNING

by

Khanh Le Dinh Viet (K.L.D.V.)

Khiem Le Ha (K.L.H.)

Supervisor:

M.S. Trung Nguyen Quoc

A final year capstone project submitted in partial fulfillment of the requirement

for the Degree of Bachelor of Artificial Intelligent in Computer Science

DEPARTMENT OF ITS

THE FPT UNIVERSITY HO CHI MINH CITY

March 2023

ACKNOWLEDGMENTS

We would like to give a big thanks to our supervisor M.S. Trung Nguyen Quoc for many valuable guidance throughout our progress. He also dedicated his time to revising our experiment results and the final report.

AUTHOR CONTRIBUTIONS

Methodology direction, K.L.D.V and K.L.H.; Dataset preparation, K.L.D.V.; Dataset preprocessing, K.L.D.V; Algorithm collection, K.L.H.; Framework establishment, K.L.H.; Experiments, K.L.D.V; Results analysis, K.L.D.V; Report writing, K.L.H; Support materials, figures and appendix, K.L.D.V; Report revision, K.L.D.V and K.L.H. All authors have read and agreed to the Final Capstone Project document.

ABSTRACT

Federated Learning has been emerged as a promising for modern Machine Learning techniques. Classical manner of operating in a centralize dataset come up against critical privacy issues. Beside that real data reacted with real user’s behavior is beneficial to tasks which involve model to be trained on practical data. For example, language model can be leveraged by playing on user data emitted while they text for speech recognition or next word prediction tasks. We could also utilize images on end devices to improve image classification models. Two current state-of-the-art methods when dealing with federated system are **FedAvg** and **FedProx.** While **FedAvg** proposed a heuristic algorithm that is quite robust about independent and identically distributed distribution (**IID**), the latter further upgrade upon the local loss setting for stability with respect to the **non-IID** distribution. There are two main nature challenges within the task as indicated in **FedProx** work: system heterogeneity and statistical heterogeneity. One more difficulty: the lack of a systematic hyperparameters tuning as well as model selection approach. **FedAvg** and **FedProx** mostlywork with canonical datasets and their synthesis variants like **MNIST, CIFAR-10**. In this work, we employ the Federated Learning approaches to unusual dataset to observe the capabilities of generalizing when handling domain-specific tasks. Concretely, we adopt **FedAvg** and **FedProx** on: (1) a brain tumor dataset with 3064 512×512 T1-weight images and (2) a **VNPlant-200** dataset which includes 20,000 images of 200 unique medicinal plants. Following the work in **FedAvg** and **FedProx,** two algorithms are applied with a careful hyperparameter tuning and inspect the effect of federated setting on the decentralized environment. The work empirically demonstrates the impact of federated learning on distinct domains. In addition, the experiments provide a heuristic scheme for hyperparameter controlling in other similar tasks or data, in this case, distributed model training and brain tumor or medicinal plant datasets.

**Keywords:** Federated Learning; FedAvg; FedProx; Distributed Training; Brain Tumor; Medicinal Plant; VGG16; ResNet50; ConvNext; MaxViT

CONTENTS

ACKNOWLEDGMENTS 3

AUTHOR CONTRIBUTIONS 4

ABSTRACT 5

CONTENTS 6

List of Figures 8

List of Tables 10

List of Abbreviations 12

1. INTRODUCTION 13

1.1 Modern Artificial Intelligence Technologies and Big Data Era 13

1.2 Federated Learning as an enhanced solution 14

1.3 Technique Limitations and the objective of this work...................................... 16

**2.** RELATED WORKS 17

**3**. PROJECT MANAGEMENT PLAN 19

**4. THEORETICAL FRAMEWORK** 20 *4.1 Stochastic Gradient Descent* 20

*4.2 Federated Learning Algorithm* 22

*4.3 Model’s Architecture* 26

**5**. MATERIALS AND METHODS 33

*5.1 Resources* 33

*5.2 Datasets and Implementation Details* 33

**6.** RESULTS and DISCUSSION 35

*6.1 Brain Tumor classification task* 36

*6.1.1 Comprehensive summarization on FedAvg scheme* 36

*6.1.2 Comprehensive summarization on FedProx scheme* 38

*6.1.3 Heterogeneity advantages study on FedProx* 39

6.2 *Medicinal Plant classification task* 41

*6.2.1 Comprehensive summarization on FedAvg scheme* 41

*6.2.2 Comprehensive summarization on FedProx scheme* 43 *6.2.3 Heterogeneity advantages study on FedProx* 43

**7.** CONCLUSIONS and PERSPECTIVES 44

**8.** REFERENCES 47

**9.** APPENDIX 50

List of Figures

Figure 1. An example of FL algorithm 15

Figure 2. Residual Block 29

Figure 3. Comparison of a basis block design in Swin Transformer, ResNet, and ConvNext

31

Figure 4. MaxViT architecture 32

Figure 5. Three types of brain tumor: (a) meningioma; (b) glioma; and (c) pituitary tumor

34

Figure 6. VNPlant-200 sample images 34

Figure 7. Results comparison between FedAvg and FedProx with various µ values on Brain Tumor dataset (ConvNext) 40

Figure 8. Results comparison between FedAvg and FedProx with various µ values on Brain Tumor dataset. (VGG16) 41

Figure 9. Results comparison between FedAvg and FedProx with various µ values in VNPlant-200 dataset 43

Figure A1. Plots on test-set accuracy over time on IID Brain Tumor Dataset with different client fraction hyper parameter. The figure only shows the FedAvg scores 65

Figure A2. Plots on test-set accuracy over time on non-IID Brain Tumor Dataset with different client fraction hyper parameter. The figure only shows the FedAvg scores 65

Figure A3. The effect of different local computing works on each entry with FedAvg. Here we fix C=0.2. The IID version of Brain Tumor dataset is used 66

Figure A4. The effect of different local computing works on each entry with FedAvg. Here we fix C=0.2. The non-IID version of Brain Tumor dataset is used 66

Figure A5. The classifier selection impact is inspected here with IID Brain Tumor dataset

67

Figure A6. The classifier selection impact is inspected here with non-IID Brain Tumor dataset

67

Figure A7. FedAvg on the IID version of VNPlant-200 dataset using VGG16 classifier

68

Figure A8. FedAvg on the non-IID version of VNPlant-200 dataset using VGG16 classifier

68

Figure A9. The effect of different local computing works on each entry with FedAvg. Here we fix C=0.2. The IID version of VNPlant-200 dataset is used. (VGG16 classifier) 69

Figure A10. The effect of different local computing works on each entry with FedAvg. Here we fix C=0.2. The IID version of VNPlant-200 dataset is used. (VGG16 classifier) 69

Figure A11. The classifier selection impact is inspected here with IID VNPlant-200 dataset

70

Figure A12. The classifier selection impact is inspected here with non-IID VNPlant-200 dataset

70

List of Tables

Table 1.Project Plan 19

**Table 2.** VGGNet configuration 27

**Table 3.** **ResNet’s** architecture 29

**Table 4.** Hardware specs 33

**Table 5.** **VNPlant-200** characteristics 35

**Table 6.** Impact of varying *C* on the Brain tumor dataset using **FedAvg** algorithm on **VGG16** model. . Each entry represents the test-set accuracy received at given rounds of communication 36

**Table 7.** Various cases when device’s amount of update is altered. Model is **VGG16**.

37

**Table 8.** Comparison of some state-of-the-art deep learning models with federated learning on Brain Tumor dataset. E=5, B=16, and C=0.2 for non-IID data and E=5, B=10, and C= 0.2 for IID data 37

**Table 9.** Test-set accuracies of **FedProx** federated algorithm with various on Brain Tumor dataset. The classifier is **ConvNext**, B=16, E=5 38

**Table 10.** Test-set accuracies of **FedProx** federated algorithm with various on Brain Tumor dataset. The classifier is **VGG16**, B=16, E=5 39

**Table 11.** Impact of varying *C* on the **VNPlant-200** dataset using **FedAvg** algorithm on **VGG16** model. . Each entry represents the test-set accuracy received at given rounds of communication 41

**Table 12.** Different local computational imposed on each client per round under *C* = 0.2 using **VGG16** model. **FedAvg** is used. **VNPlant-200** is under investigation 42

**Table 13.** The effect of various classifiers regarding the **VNPlant-200** dataset. **FedAvg** is the algorithm. The mini batch size, the number of local epochs, the client fraction are 16, 5, and 0.2, respectively 42

Table 14. Experiments upon the weight of proximal quantity on VNPlant-200 dataset. . The classifier is ResNet50 43

Table A1. Original results from proposed work when evaluating MNIST with on 2NN and on CNN. Each cell represents the communication cost needed to a respective model to achieve desired test-set accuracy. (99% with CNN and 97% with 2NN). Five attempts did not convergence in time 64

List of Abbreviations

|  |  |
| --- | --- |
| FL | *Federated Learning* |
| AI | *Artificial Intelligence* |
| NLP | *Natural Processing Language* |
| SR | *Speech Recognition* |
| CV | *Computer Vision* |
| ML | *Machine Learning* |
| GDPR | *General Data Protection Regulation* |
| HFL | *Horizontal Federated Learning* |
| VFL | *Vertical Federated Learning* |
| FTL | *Federated Transfer Learning* |
| IID | *Independent and identically distributed* |
| CNN | *Convolutional Neural Network* |
| SGD | *Stochastic Gradient Descent* |
| GD | *Gradient Descent* |
| LRN | *local responses normalization* |
| ViT | *Vision Transformers* |
| CLAHE | *Contrast Limited Adaptive Histogram Equalization* |

1. INTRODUCTION

**1.1 Modern Artificial Intelligence Technologies and Big Data Era**

The growth of Artificial Intelligence (**AI)** applications has been progressively supported by vast amounts of data [1], [2]. Conventionally, **AI**-related applications often fall into ordinary categories like computer vision (**CV**), natural processing language (**NLP**), speech recognition (**SR**). Those are also the most important appliances of **AI** in real world. Sometimes, the model can outperform human performance. For example, Deep Learning-based face recognition can achieve exceptional levels of performance given millions of training samples [3], [4]. These systems obviously require huge a bunch of data to gain satisfying levels of results due to the complexity of the model’s architecture.

Generally, the big data system demands special methods in gathering and processing because data regularly comes on a small scale. In addition, data diversity mostly appears as a critical adversity to confront with. Missing values, missing labels, disparity distribution largely expect big effort from domain experts to repairing. In fact, benchmark datasets used within standard tasks usually require an enormous work in selectively gathering, processing and thus need to be done in a proper and comprehensive research than the work evaluate on it [5]–[7]. Some demands raised in the context of narrower domains now show that it is hungry for data, precisely large-scale data to come up with training.

End user’s data turns out to be a great source of data for **ML** tasks. This kind of data holds a very important nature: it is the real data that is eventually assessed and consumed by the final trained model. The modern world currently has serious concerns regarding data privacy and data ownership: which org has the ability and the rights to use data for building **AI** technologies. Some university labs or specific firms developing their **AI** research or products adopt their own business data or data that they created by themselves which is in this situation they have the full ownership over this data. But things get complicated in certain fields: data exists in various forms, generated by different parties and the naïve approach would be transfer data into one central location and perform plenty of **ML** techniques. However, this method is no longer valid today. Owners of data are aware of their privacy rights, and they do not want their private information to be used illegally for commercial or political purposes.

Strict controls on data collection and data usage have likewise been imposed by law makers. General Data Protection Regulation (**GDPR**) issued by European Union in 2018 is a concrete example. Under this restrictive landscape, gathering and sharing data among separate organizations is becoming more and more difficult.

Even when we have a valid procedure dedicated to passing sensitive data around silos for training **AI** models, there are two more challenges. First, the benefit of data collaboration is not clear, or at least it is hard to measure if the procedure follows a super rigid manner, e.g., encrypting and shuffling all the data before entering the training phase. The fear of losing control over data and the lack of transparency make the crucial trade-off consideration from the owner’s perspective. Second, some data have severe sensitive nature that cannot be moved from the owner’s location, e.g., medical records and financial transactions, hence prohibit free data circulation.

How to solve this privacy problem is mandatory as the rules will progressively more rigorous. **AI** community has been witnessed tremendous of notable breakthroughs in ten years since 2012 due to the development hardware strengths and large-scale training datasets. An **AI** winter is going to happen if this situation is not sufficiently addressed.

**1.2 Federated Learning as an enhanced solution**

Federated Learning (**FL**) relies on a pure idea that lets the model being trained in-place at the data location, which we refer as local data or device’s data. Then the information about trained model (weights or gradients) is the quantity that moving around for assembling the well-behaved application. The detailed explanation would be data reside at its own location, and some variable amount of update, i.e., training and validation, are executed. Conventionally, the method works like server-client architecture where there is a global model located at the server device and each client carries its dataset. Proper confidentiality plays an integral role on securing the inner content or sometimes the inherent nature of data being transferred. Furthermore, the communication process also differs among several leading implementation which Diagram

Description automatically generatedaffects different desired optimal goals in different ways.

**Figure 1**. An example of **FL** algorithm

**FL** concepts first evolve in a decent form in 2016 in [8], namely **FedAvg**. The authors proposed an iterative approach for jointly updating the global model throughout communication rounds. As described above, in this federated scheme does not compel a whole centralize dataset at one place, as well as data at each device to be sent back-and-forth. To be more detailed, for several updates, under encryption, clients send local model parameters which then be used to incorporate into a new stateful global model representing current trained model given a passed number of rounds. Note that this is a repetitive training design. One thing can be theoretically assured from a particular client view: its data is not revealed or examined common patterns with other clients or the server.

Horizontal Federated Learning (**HFL**) and Vertical Federated Learning (**VFL**) are roughly two fundamental categories of **FL**. In **HFL**, the system has common feature spaces in each regional dataset (parties may have their business market in the same domain) but distinct data samples. Conversely, sample spaces contain overlapping data samples in **VFL,** but they differ in data features. Two settings are derived from actual corporate situations in generating data, which in turn satisfy unique demands. Federated Transfer Learning (**FTL**) applies a unique direction and is suitable when the party’s data is highly heterogeneous. In general, federated algorithms can be expanded according to how data is partitioned among clients and the basis nature of the data.

**1.3 Technique Limitations and the objective of this work**

Researchers have been improving the algorithmic mechanism for distributed learning over many computational sites in recent years.

In [8], the authors came up with a practical framework that help federated learning by model averaging. The results show a potential ability for adopting **FL** in other environments. However, the work left plenty of questions involving convergence guarantee and generalization performance. From the data perspective, the method further imposes the same amount of training workload with respect to each edge device, which raises uncertainty when deploying with actual data in unconventional domain. The algorithm also does not provide a clear and formal solution when tackling non-**IID** data, which is quite happened frequently.

**FedProx** [9] resolved mentioned cons thoroughly, and beyond that suggest a mathematical proof for their technique. They put up front a convergence analysis as well as local dissimilarity formulas for supporting convergence guarantee. The experiments showing the robustness under extremely heterogeneous setting are likewise presented. They allow some clients lazily perform fewer number of epochs and integrate a proximal term into local losses to penalizing the weights from being far away from the global model.

Nevertheless, almost all the state-of-the-art improvements mainly focus advancing security and statistical challenges. We realized an unhealthy assumption about hyperparameter selection, comes from the usages of canonical datasets. In this work, we simulate two algorithms **FedAvg** and **FedProx** two datasets which fit into two separate domains: (1) a brain tumor dataset with 3064 512×512 T1-weight images and (2) a VNPlant-200 dataset which includes 20,000 images of 200 unique medicinal plants. Firstly, we follow the stated process of training to obtain valuable observations, finding the optimal value for each hyperparameter. We adopt several **CNN**-based models like **VGG16**, **ResNet50,** **ConvNext,** **MaxViT** for comprehensive comparison. The number of communication rounds, i.e., the total communication cost until reaching a reasonable performance is our main metrics.

This work can be considered as a helpful reference for those who are interested in federated learning system or who currently being working with related fields.

2. RELATED WORKS

Many directions have significantly received attention during the decades. In the shape of federated optimization, the communication cost as well as the privacy effectiveness can be considered. Some works studied the statistical property of data, devices, and local gradients update.

In [10], iterative parameter mixing on structured perceptron is used to reduce the complexity given the availability the computing clusters. [11] utilize a format of elastic averaging: the asynchronous variant is also proposed. These works in general do not exam the non-**IID** nor the unbalance of datasets, which is a very principal for our upcoming settings. Remember that in a realistic scenario, the number of clients could be much larger than the number of data observations per client. In the convex setting,[12], [13] addressed some key concepts about federated framework: they particularly look at the privacy aspect during communicating, the upper and lower bound runtime and the quantity of used samples.

There are many publications that worked on minimizing communication cost [14], [15]. This approach decreases the overall runtime and jointly increases privacy performance. Opposed to iterative training approach, one endpoint of the distributed family is one-shot algorithm, which is the method that makes no overhead on communication cost at all. The final model is produced after all sub training processes finish, where in each sub process, a local client tries to solve the loss of its local data until reaching several epochs. The combine scheme could be model averaging. However, this method shows no better performance over minimizing on a single client [16], [17].

We have addressed earlier the importance of studying extensively the statistical property imposed on the nature of data and computing clusters. [18], [19] allow inexact local updating to balance computational cost and communication cost. This idea quite inspires for the systematic heterogeneity examination. Here we formalize some typical characteristics of federated learning problems: (1) local dataset will not be representative for the population distribution (non-**IID**), (2) unbalance data among devices, (3) the number of clients participating in learning could dominate the local dataset’s size, (4) number of devices can be unavailable sometimes, (5) clients do not have the same computational strength, (6) updates could be lost during communication due to network issues.

We need to explore a more general framework that can handle heterogeneity introduced by characteristics mentioned above. The work in [20], [21]allowing data to be shared between clients and server for analyzing statistical feature lied in local data. This approach could help the server (or the coordinator) to inspect suitable solver use each round per client. Hence, the broader technique can be developed robustly to tackle highly non-**IID** and/or unbalanced dataset. Nonetheless, this puts a huge burden on network bandwidth (which is normally restricted in terms of hand-held devices or in case of non-physical connection). More seriously, the action of exchanging data violates the key aspect of privacy in a realistic federated environment: confidentiality.

One solution that comes naturally first in mind when dealing with device strength inequality is to abandon uncomplete training process or to use the result model weights regardless of a device finish its desired number of epochs or not. The same set of devices are likely to be exhausted more periodically all the time, thus, can bring bias to our model. Moreover, divergence could occur when profitable data in a particular device cannot maximize its productivity because of repudiation. [9] demonstrated that instability growths when we embrace some stragglers into chosen clients per round of communication. By adding proximal term to local loss function, [9]report several benefits in terms of communications cost and the stability of convergence. The randomized Kaczmarz method [22], [23] for solving linear systems of equations serves as an inspiration for the dissimilarity characterization analysis the authors offer.

Recent works adopting federated system in image tasks primarily use standard databases for experiments, such as **MNIST**, **CIFAR-10**, and their variations. This is advantageous because it expedites the experimentation of a vast number of parameter combinations, thereby facilitating the exploration and evaluation of more efficient algorithms. Few academics conduct federated learning on their domain-specific datasets. However, it has been observed that there is no established method of parameter optimization for dataset that is not specific to any domain. We would like to commence with utilizing the hyperparameter selection technique. Some key hyperparameters are: (1) the number of clients join in training each round, (2) the mini-batch size, (3) the number of epochs each round, (4) the hyperparameter of the proximal term, (5) the initial learning rate and rate decay algorithm. We wish to ascertain the influence of these parameters on new datasets to demonstrate the consistency of ultimate outcomes obtained at the end of the training procedure. Despite ensuring convergence, [9] still implies certain characteristics of [8], thus necessitating the requirement for an automated process for selecting parameters.

3. PROJECT MANAGEMENT PLAN

Table 1.Project Plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task name | Priority | Owner | Start date | End date | Status | Issues |
| Seek out research studies | High | K.L.H. | 4/1/2023 | 29/1/2023 | Completed | … |
| Setting up datasets | High | K.L.D.V. | 4/1/2023 | 25/2/2023 | Completed | … |
| Establish the FedAvg code environment. | High | K.L.H. | 30/1/2023 | 15/2/2023 | Completed | … |
| Establish the FedProx code environment. | High | K.L.H. | 16/2/2023 | 15/3/2023 | Completed | … |
| Run experiments on Brain Tumor Data | High | K.L.D.V. | 16/2/2023 | 10/3/2023 | Completed | … |
| Run experiments on VNPlant-200 datasets | High | K.L.D.V. | 11/3/2023 | 10/4/2023 | Completed | … |
| Review related papers for further improvements | Low | K.L.H. | 16/3/2023 | 22/3/2023 | Completed | … |
| Write report | High | K.L.H. | 23/3/2023 | 10/4/2023 | Completed | … |
| Revision | High | K.L.D.V. and K.L.H. | 10/4/2023 | 17/4/2023 | Completed | ... |

4. THEORETICAL FRAMEWORK

**4.1 Stochastic Gradient Descent**

**SGD** is commonly used as an optimization technique in contemporary works due to its ease of use. In addition, we cannot presume any bias at the beginning of the learning procedure; therefore, employing more complex algorithms could result in wasted effort without observing the actual effect of the FL setting.

**SGD** is an iterative method for optimizing an objective function by calculating the gradients for several samples, whereas GD utilizes the entire dataset to update the weights. Consider the scenario of minimizing the following loss function.

where *w* is the parameter being estimated and *m* is the number of data samples.

When using standard **GD**, an iteration of optimization strategy would be:

Clearly, α is the learning rate. In classical statistics, this kind of sum-minimizing problem arises in least-squares (like linear regression) or in maximum-likelihood estimation. In simple form of loss objectives, step to global (or local) minimum is assured quickly. As a result of the intricacy of each local loss or the amount of the dataset, gradient calculation may be prohibitively costly in many situations. Performing each step on a subset of samples is preferable and is beneficial in large-scale **ML**.

This time *i* represents the chosen training examples. The algorithms sweep through the entire dataset cause the loss functions to approach the optimum. The full process of learning by **SGD**for simple regression applicationcan be roughly illustrated below.

*Algorithm 1.* Stochastic Gradient Descent

1. Initialize weighs *w* and pick an initial learning rate α
2. For each epoch (repeat until desired optimal value is achieved):
   * Randomly shuffle data points in the dataset.
   * For :
     + Determine the local loss

Given the capabilities of modern **GPUs** for parallel processing, the simple form of **SGD** is utilized infrequently due to its inefficient performance. The convergence of stochastic gradient descent has been widely investigated; particularly, given an acceptable learning rate, **SGD** will almost certainly cause the loss to reach its global minimum (convex case); otherwise, it will cause the loss to reach its local minimum.

Alternately, modifying the model's parameters now occurs in the form of a batch (called mini-batch stochastic gradient descent). The result of decreasing the mini-batch size could lead to more learning ability; said differently, this technique in fact allows the model converges faster than considering the whole dataset.

*Algorithm 2.* Mini-batch Stochastic Gradient Descent

1. Initialize weighs *w* and pick an initial learning rate α
2. For each epoch (repeat until desired optimal value is achieved):
   1. Randomly shuffle data points in the dataset.
   2. For each batch:
      1. Determine the local loss

**4.2 Federated Learning Algorithm**

**4.2.1 FedAvg Algorithm**

**FedAvg** is built upon **SGD**, i.e., the local optimizer is typically **SGD**. In this subsection, we explore this approach in depth, formulate algorithms, and examine some of the original publication's results [8].

The combination of synchronous **SGD** (one partition must wait other partitions to finish computing gradients) and multi-batch updater yields best result. Consider *K* clients for whom data is partitioned among, the hyper-parameter *C* controls the fraction of clients being chose per round. means one client is chosen.

Each client *k* obtains at completion of a training turn, then the server aggregates these gradients by:

where denote the current communication round, and represents the number of samples at client *k*.

The equivalent form can be achieved by alternating the derivatives at each local by its model’s weights. This property is derived from:

One important design must be carefully considered when dealing with non-convex objectives. Independent initialization of a distributed model may result in poor performance. Averaging from different conditions shows no advantages over taking single evaluating in each model (the weight of mixing equals to 0 or 1). Conversely, when starting multiple models from a same random seed, averaging parameters works well.

*Algorithm 3.* FedAvg Algorithm

*K* is the number of clients. *C* is the fraction of clients selected per round. *B* is the local mini-batch size. *E* is the number of epochs each device must iterate through.

*Server-side computation:*

* initialize
* for each round
  + from *C*, select a random subset from *K* clients
  + for each client *k* in
    - compute by performing a client-side computation.
  + (The total number of data points involving into this training phase)

*Client-side computation:*

* for each local epoch
  + for each batch *b* in the local dataset of this client
* return

It is experimentally essential to properly tune the hyper parameter. *B* and *E* control the number of updates per round, which are quite similar in effectiveness. As previously indicated, in a federated system, communication costs are likely to outweigh computational costs, however in a centralized setting, communication costs are insignificant. In the meanwhile, C determines the global batch size, with the general assumption that in both **IID** and non-**IID** distributions, bigger C tends to reflect a larger proportion of data samples, resulting in better models for the current round. If we wish to add additional computing every round, we may either (1) increase parallelism (which has no negative effects if true parallelism is employed) or (2) increase computation at each client.

**4.2.2 FedProx Algorithm**

**FedProx** [9]can be perceived as a re-parameterization variant of **FedAvg** in which the authors introduce heterogeneous struggles. The study offers both empirical and theoretical investigations addressing the convergence of the approach.

As previously mentioned, more local computation can significantly help reduce communication costs. This amount is affected by the number of local epochs and the size of the local mini batch. Besides that, more work of updating on each local landscape may cause each local model to converge toward its local optimum, hence, make convergence unpredictable. Some clients also cannot perform the desired number of updates due to hardware constraints. In practice, it is impossible to automatically determine in advance the suitable epoch for each client while the local epoch must satisfy the benefit of cutting communication cost. Therefore, to balance out the initial setting, **FedProx** fixes the number of epochs used for each round of communication and finds a more robust way to manage gradients received at the end. The proposed framework has two key characteristics.

**Allow truncated work.** Forcing all devices to implement the same effort of training is not quite realistic. **FedAvg** employs a basic approach: drop the uncomplete weights. This technique has been shown to produce bad models given a fixed number of rounds. The implementation specifies a new hyper parameter controls which clients completely participate in the result parameters and which does not. Inclusive experiments reveal the effectiveness of stability: throughout the learning procedure, loss tends to decrease consistency.

**Proximal term.** To prevent the weights from being far away from the global minimum, **FedProx** adjust the local solver to be more constrained:

where is the original distance with respect to local batch *b* and is the global weight at the beginning of the round. The additional term is beneficial both in: (1) overcome the heterogeneity in data distribution and (2) help for incorporating variable amounts of work from all clients.

*Algorithm 4.* FedProx Algorithm

*K* is the number of clients. *C* is the fraction of clients selected per round. *B* is the local mini-batch size. *E* is the number of epochs each device must iterate through. *T* is the number of stragglers.

*Server-side computation:*

* initialize
* for each round
  + from *C*, select a random subset from *K* clients.
  + from *T*, select which client in must perform full workload.
  + for each client *k* in
    - compute by performing a client-side computation (with assigned workload)
  + (The total number of data points involving into this training phase)

*Client-side computation:*

* for each local epoch
  + for each batch *b* in the local dataset of this client
    - , where
* return

The optimizer is still stochastic gradient descent and fixed learning rate. Some works have been focused on employing other modern optimization algorithms as well as the automated manner to choosing learning rate.

**4.3 Model’s Architecture**

In this section, we briefly introduce some architecture used in our experiments. The model decision is derived from related works in terms of commonly manipulating over used datasets.

**4.3.1 VGGNet** [24]

One remarkable exploration in this type of architecture is the adoption of a very deep **CNN** network combining with small receptive field. Particularly, 3 x 3 filters are used to replicate the effect of larger stride window while maintaining the reasonable size. This choice of design shows a more accurate performance when we steadily append more convolutional layers to the model.

Generally, the family of architecture shares some settings:

* 224 x 224 RGB input image. The image is passed through a stack of conv layers with 3 x 3 filters.
* Stride is 1, same padding. That’s why the very small receptive size is chosen: 3 x 3 is the smallest size that can capture the spatial information in the image.
* Five 2 x 2 max-pooling layers are used after some conv layers to reduce spatial dimension.
* Three fully connected layers at the end. The first two have 4096 units, while the last one’s size depends on the label space’s length.
* ReLU activation.

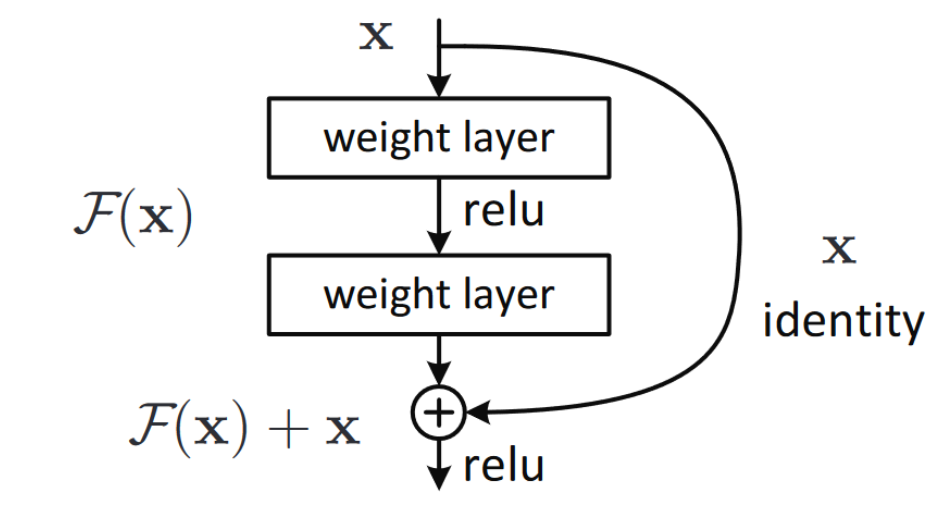
Detail configuration is showed in Table 2 below:

**Table 2.** VGGNet configuration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **VGGNet Configuration** | | | | | |
| **VGG11** | **VGG11-LRN\*** | **VGG13** | **VGG16** | **VGG16** | **VGG19** |
| 224 x 224 RGB image | | | | | |
| conv3-64 | conv3-64  LRN | conv3-64  conv3-64 | conv3-64  conv3-64 | conv3-64  conv3-64 | conv3-64  conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128  conv3-128 | conv3-128  conv3-128 | conv3-128  conv3-128 | conv3-128  conv3-128 |
| maxpool | | | | | |
| conv3-256  conv3-256 | conv3-256  conv3-256 | conv3-256  conv3-256 | conv3-256  conv3-256  conv1-256 | conv3-256  conv3-256  conv3-256 | conv3-256  conv3-256  conv3-256  conv3-256 |
| maxpool | | | | | |
| conv3-512  conv3-512 | conv3-512  conv3-512 | conv3-512  conv3-512 | conv3-512  conv3-512  conv1-512 | conv3-512  conv3-512  conv3-512 | conv3-512  conv3-512  conv3-512  conv3-512 |
| maxpool | | | | | |
| conv3-512  conv3-512 | conv3-512  conv3-512 | conv3-512  conv3-512 | conv3-512  conv3-512  conv1-512 | conv3-512  conv3-512  conv3-512 | conv3-512  conv3-512  conv3-512  conv3-512 |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-L\*\* | | | | | |
| softmax layer | | | | | |

Note that: (\*) **LRN** stands for local responses normalization and (\*\*) represents the number of labels in the label space.

**4.3.2 ResNet** [25]

**ResNet** leverages the neural network’s depth to a higher level. Stacking more layers makes it difficult to train due to vanishing/exploding gradients. Simply put, this issue can be addressed by adding normalization. However, the result tends to degradation while training loss does not guarantee to be decreased, i.e., overfit is not the case. This phenomenon indicates that there is a problem with deep layer that makes it harder to learn more fine-grained features, which is the key principle in deep learning. **ResNet** introduces residual blocks to cope with this dilemma.

**Figure 2**. Residual Block (image from original paper [25])

The identity short-connection quantity helps to optimize the desired function easier because now if the eventual performance of the identity mapping is optimum, learning process just needs to push residual term to zero.

Comprehensive experiments on ImageNet [26] showed that: (1) deeper networks indeed result higher accuracy and (2) networks with residual block are easier to train compared to plain counterpart. Table 3 lists the structure of different depth **ResNet**.

**Table 3.** **ResNet’s** architecture (L denotes the label space length, square brackets denote residual blocks)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| layer type | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
| conv | 7 x 7, 64 channels, stride 2 | | | | |
| conv |  |  |  |  |  |
| conv |  |  |  |  |  |
| conv |  |  |  |  |  |
| conv |  |  |  |  |  |
| pooling | avgpooling | | | | |
| fully connected | L-dim fc | | | | |
| activation | softmax layer | | | | |

**4.3.3 ConvNext** [27]

As the introduction of Vision Transformers (**ViT**) in 2020, the computer vision landscape is not limited to network architecture design. **ViT** surprisingly show potential results on image classification tasks given the ability to scaling. Nonetheless, computer vision also contains other difficult duties involving in image-specific inductive bias to maximize spatial information. Without ConvNet, a vanilla **ViT** model may confront a few challenges in dealing with object detection or semantic segmentation.

Many advancements have been made to bring back ConvNet to form a hybrid approach [28]. The sliding window method shows their role as being intrinsic to visual processing. However, these works have some costly components, which could cause the design to be more complex or be unreasonable to scale. **ConvNext**, a pure ConvNet model is built gradually by embracing some minor design modifications. This process aims to mimic the way a hybrid transformer model Diagram

Description automatically generatedlike Swin Transformer [28] process digital images.

**Figure 3**. Comparison of a basis block design in **Swin Transformer**, **ResNet** and **ConvNext** (image from original paper [27])

**Training Technique.** Increase the number of epochs from 90 to 300. **AdamW** Optimizer is adopted. Various augmentation techniques like Mixup, CutMix, RandAugment, RandomErasing. Stochastic Depth and Label Smoothing are used for regularization.

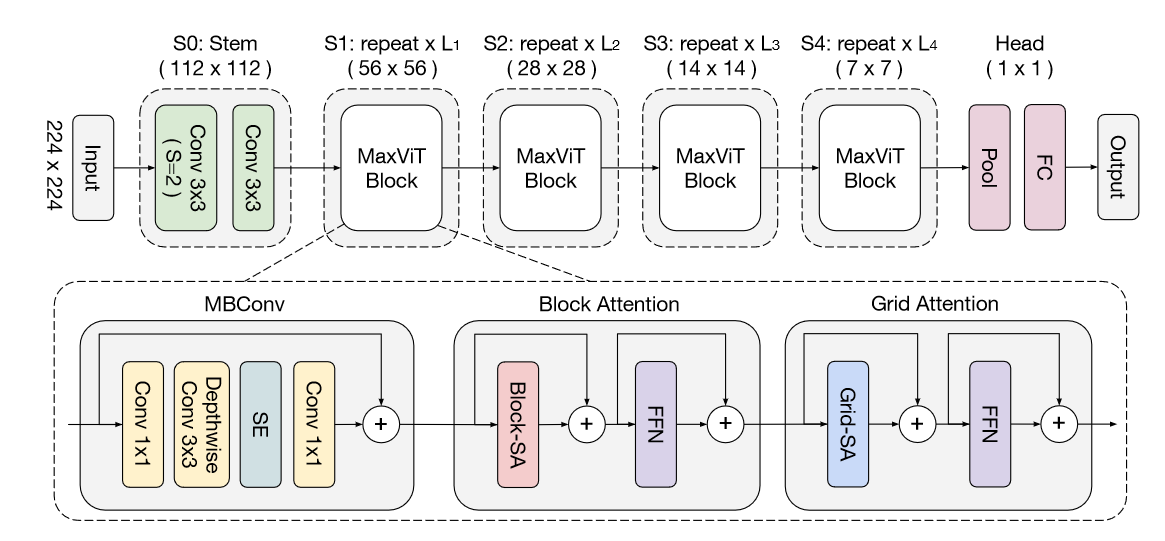
**ResNext-ify.** Depthwise Convolution is used to group convolution filters.

**Inverted Bottleneck.** The idea is that we could adopt inverted bottleneck in **ResNet**. The hidden dimension within a layer block is now 4 times bigger than input dimension.

**Large Kernel Size.** To examine the behavior of large size kernel, **ConvNext** moves up the position of the depthwise conv layer. (However, this violates a typical standard of using small receptive field to replicate the effect of larger kernel size to gain parallel computing of modern GPU). **ConvNext** also experiment also kernel size include 3, 5, 7, 9, 11. The performance saturates when the number reaches 7.

**Micro Design.** ReLU is replaced by GELU. Some activation positions are also eliminated. Truncate batch normalization and some are altered with layer normalization. Separate downsampling layers.

**4.3.4 MaxViT** [29]

 Added multi-axis attention helps form an efficient attention model to cope with scalability. There are two novel ideas in this work: blocked local and dilated global attention. The proposed model called **MaxViT** serves as a powerful vision backbone for visual processing.

**Figure 4**. **MaxViT** architecture (image from original paper [29])

5. MATERIALS AND METHODS

5.1 Resources

All presented works in the scope of this report are performed on Google Collaboratory Pro+. Hardware specifications vary over time. Typical details are:

**Table 4.** Hardware specs

|  |  |  |
| --- | --- | --- |
|  | **Standard** | **Premium** |
| CPU | Intel(R) Xeon(R) CPU @ 2.20 GHz | Intel(R) Xeon(R) CPU @ 2.20 GHz |
| RAM | 12 GB | 84 GB |
| GPU | NVIDIA Tesla T4 16 GB VRAM | NVIDIA A100 40 GB VRAM |

5.2 Datasets and Implementation Details

We use the brain tumor dataset composed by Cheng et al. [30] in the first class of experiments. The dataset consists of 3064 T1-weighted pictures collected from 233 patients with three labels of brain malignancies: 708 images of meningioma, 1426 images of glioma, and 930 images of pituitary tumor. Figure 5 illustrates some sample images taken in [30]. The images have digital resolution of 512 512 with pixel size of 0.49 0.49 mm2.

We split the datasets into 80% training and 20% test. Test set is resided at the aggregation server, while training samples are partitioned into 10 clients. Partition manners are discussed later. For image pre-processing, Contrast Limited Adaptive Histogram Equalization (CLAHE) technique is adopted. The image is then resized to 224 224.

A picture containing text, spectacles

Description automatically generatedA picture containing text, watch, different

Description automatically generatedA picture containing text

Description automatically generated

1. (b) (c)

Figure 5**.** Three types of brain tumor: (a) meningioma; (b) glioma; and (c) pituitary tumor.

With second class of experiment, an herbal plant dataset which consists of plants found in Vietnam are used. The photographs were captured within a natural setting with the intention of depicting the intricacy of classifying images within real world environments. The dataset comprises of plant images captured from varying angles, brightness levels, environmental conditions, viewpoints, and other related factors. Thus, it serves as a suitable model for a practical plant recognition task. Figure 6 demonstrates some samples.

A picture containing outdoor, plant, grass

Description automatically generated

Figure 6**.** VNPlant-200 sample images.

After resizing to 224 224, we implement some data augmentation like random rotation or random flip. We use 8000 images for testing, 2000 images for validation, and the rest for training. This time the number of devices jointly learning the federated model is 100.

**Table 5.** **VNPlant-200** characteristics

|  |  |
| --- | --- |
| Number of species | 200 |
| Number of images for each specie | 100 |
| Image resolution | 256 256 and 512 512 |
| Angle | Entire plant with realistic noise |
| Environment | Real world |

**Data distribution approach.** To study federated performance on heterogeneity setting, we explore two ways to partition data. In **IID** way, the data is randomly shuffled and distributed over *K* clients, i.e., each client theoretically represents the whole population. Non-**IID** manner involves sorting the data points by labels first, then populate each client with an equal number of samples so that each client contains at most 2 labels. This way we could benchmark both algorithms on specific domain non-**IID** data for generalization.

Regarding learning rates used in **SGD**, we tune for the best value achieved by each combination of hyper parameters, i.e., all numbers shown in tables or figures are training on the best learning rate. One critical point: for fair competition, we fix the randomly selected clients, the order of mini batch per client across training rounds. We also apply plain **FedAvg** algorithm while dropping the testing of stragglers in **FedProx**. That means we do not incorporate variable works on those devices, instead we force all chose devices to perform the same amount of work.

6. RESULTS and DISCUSSION

6.1 Brain Tumor classification task

6.1.1 Comprehensive summarization on FedAvg scheme

Partial parallelism. We first play with client fraction *C*. Table 6 shows the results of varying *C* over Brain tumor dataset. VGG16 is used as the initial baseline. We adopt a slightly different methodology here: instead of evaluating the cost of communication until satisfying desired levels of accuracy, we record the test-set accuracy obtained when finishing given numbers of rounds. Here, the approach functions effectively in an IID setting that provides positive outcomes with just small communication rounds. Undoubtedly, greater C produces better outcomes, particularly in non-IID settings when client data do not reflect the whole distribution. The performance of non-IID data improves with time more slowly than IID data, indicating that communication cost is substantial in non-IID scenarios. Comparing our results to those of the original study, in which the authors conducted tests on MNIST using two basic neural networks, we detect a comparable impact. Table A1 in the appendix section illustrates this effect in the original paper. Figure A1, A2 in the appendix section gives a clearer view regarding the speed of convergence over rounds of communication.

**Table 6.** Impact of varying *C* on the Brain tumor dataset using **FedAvg** algorithm on **VGG16** model. . Each entry represents the test-set accuracy received at given rounds of communication.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | IID | | | | Non-IID | | | |
| 10 | 20 | 50 | 100 | 10 | 20 | 50 | 100 |
| 0.1 | 92.48 | 95.26 | 97.38 | 98.20 | 47.39 | 47.39 | 63.40 | 72.22 |
| 0.2 | 94.12 | 96.41 | 98.04 | 98.53 | 47.39 | 79.08 | 87.09 | 90.69 |
| 0.3 | 95.26 | 96.24 | 97.55 | 98.37 | 77.29 | 77.29 | 91.12 | 94.12 |
| 0.5 | 95.45 | 97.55 | 98.04 | 98.20 | 83.49 | 88.56 | 93.62 | 95.59 |

For consistent insights and balance out the computational weight of training due to limited hardware constraints, we fix for further testing.

Local computation examination. This time, the influence of extra local computation is investigated. Adding extra updates every round to each client does not significantly increase performance. We attempt to raise E from 1 to 5, while altering the mini-batch size to the values 4, 10, and 16. Nonetheless, we discover a very intriguing property: a mini-batch size of 16 yields a pretty good result in a non-IID context. In some instances, the performance suffers when the mini batch size is increased while the number of epochs is maintained, indicating that too many updates might lead averaging to give inferior results. The counterpart diagram of Table 7 is placed at Figure A3, A4 at appendix, in which we visualize the effect we have done here.

**Table 7.** Various cases when device’s amount of update is altered. Model is **VGG16**.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| E | B | IID | | | | Non-IID | | | |
| 10 | 20 | 50 | 100 | 10 | 20 | 50 | 100 |
| 1 | 10 | 86.11 | 93.30 | 96.08 | 96.57 | 55.72 | 55.72 | 79.08 | 89.38 |
| 2 | 10 | 93.46 | 94.93 | 97.55 | 98.37 | 66.01 | 66.67 | 80.39 | 90.69 |
| 5 | 4 | 95.26 | 96.70 | 98.04 | 98.04 | 55.88 | 77.94 | 87.58 | 90.85 |
| 5 | 10 | 94.12 | 96.41 | 98.04 | 98.53 | 47.39 | 79.08 | 87.09 | 90.69 |
| 5 | 16 | 93.95 | 96.41 | 97.71 | 97.88 | 67.32 | 67.32 | 80.23 | 93.30 |

So far, the documented experiments have demonstrated a reliable set of hyper parameter values for our task. We study further the impact of several classifiers on federated learning. Comparing ResNet50, ConvNext, and MaxViT with the VGG16 baseline, we employ several cutting-edge deep learning architectures. Table 8 displays the experimental states. In this series of studies, E=5, B=16, and C=0.2 are used for non-IID data whereas E=5, B=10, and C=0.2 are used for IID data.

**Table 8.** Comparison of some state-of-the-art deep learning models with federated learning on Brain Tumor dataset. E=5, B=16, and C=0.2 for non-IID data and E=5, B=10, and C= 0.2 for IID data.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **IID** | | | | **Non-IID** | | | |
| **Rounds of com.** | 10 | 20 | 50 | 100 | 10 | 20 | 50 | 100 |
| **VGG16** | 94.12 | 96.41 | 98.04 | 98.53 | 67.32 | 67.32 | 80.23 | 93.30 |
| **ConvNext** | 95.52 | 96.57 | 98.04 | 98.69 | 75.65 | 75.65 | 80.88 | 92.16 |
| **ResNet50** | 91.83 | 95.59 | 96.73 | 98.03 | 49.51 | 71.24 | 82.52 | 86.76 |
| **MaxVit** | 94.93 | 96.57 | 97.56 | 98.69 | 56.86 | 75.82 | 85.95 | 90.36 |

Evidently, **ConvNext** and **MaxViT** give superior outcomes while processing IID data. On the other hand, despite the fact that ConvNext is the best model during the first 50 rounds of communications, it cannot exceed the peak performance of **VGG16**. Consequently, **VGG16**, with 93,3% accuracy, may be the best reliable classifier for non-IID Brain Tumor image data. Figure A5, A6 in the appendix provides more visualization details.

6.1.2 Comprehensive summarization on FedProx scheme

Following the preceding section's work, we examine if the proximal term in FedProx aids in handling non-IID situations. We have found that B=16 and E=5 produce decent results in non-IID contexts, thus we will continue to use these parameters in the subsequent tests. ConvNext and VGG16, which produced the greatest results on the Brain Tumor dataset in the previous section, are also reused. In this effort, we tweak the hyper parameter from a limited candidate set of to determine its effect on test-set accuracy convergence after 10, 20, 50, and 100 rounds of communications. Tables 9 and 10 show the respective outcomes of ConvNext and VGG16.

**Table 9.** Test-set accuracies of **FedProx** federated algorithm with various on Brain Tumor dataset. The classifier is **ConvNext**, B=16, E=5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Non-IID | | | |
| 10 | 20 | 50 | 100 |
| 0 | 75.65 | 75.65 | 80.88 | 92.16 |
| 1 | 79.08 | 79.08 | 85.29 | 92.48 |
| 0.1 | 80.23 | 80.23 | 83.82 | 92.65 |
| 0.01 | 76.31 | 76.31 | 83.99 | 92.65 |
| 0.001 | 78.59 | 78.59 | 86.11 | 92.48 |

**Table 10.** Test-set accuracies of **FedProx** federated algorithm with various on Brain Tumor dataset. The classifier is **VGG16**, B=16, E=5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Non-IID | | | |
| 10 | 20 | 50 | 100 |
| 0 | 67.32 | 67.32 | 80.23 | 93.30 |
| 1 | 60.94 | 62.58 | 83.33 | 93.30 |
| 0.1 | 48.04 | 72.06 | 83.01 | 89.05 |
| 0.01 | 60.29 | 67.97 | 83.17 | 91.83 |
| 0.001 | 71.90 | 80.39 | 80.39 | 81.21 |

We can see that, given an appropriate value of , the learning process tends to be condensed into fewer iterations and assured to converge steadily over time. With **ConvNext**, the optimal value of is 0.1, allowing the accuracy to surpass 80% in only 10 communication rounds. In case of **VGG16**, the optimal value of for fast convergence is 0.001. With **VGG16**, however, there is a little trade-off: the faster convergence comes at the expense of a lower peak accuracy, in this instance 81.21% as opposed to 93.3%. This conduct has no impact on **ConvNext**.

The heterogeneity breaking behavior of **FedProx** over **FedAvg** will be described in the next section. However, we would like to stress a vital point: it is essential to choose a suitable number for ; otherwise, the performance might decrease and become unstable over time.

6.1.3 Heterogeneity advantages study on FedProx

In figure 7, we see that FedProx yields quite humble results compared to FedAvg. Although the convergence property is assured, it does not seem reliable in terms of stability, early convergence, or peak performance. The candidate set of proximal term parameter µ taken from original work. Here we can conclude that the disparity tackling effect of FedProx is not remarkable.

Chart

Description automatically generated

Figure 7. Results comparison between FedAvg and FedProx with various µ values on Brain Tumor dataset. (ConvNext)

Graphical user interface, chart

Description automatically generated with medium confidence

Figure 8. Results comparison between FedAvg and FedProx with various µ values on Brain Tumor dataset. (VGG16)

6.2 Medicinal Plant classification task

We follow the same methodology of model evaluation here. The conclusions are quite like those obtained above, so we will only stress important points as we progress our experiments.

6.2.1 Comprehensive summarization on FedAvg scheme

**Table 11.** Impact of varying *C* on the **VNPlant-200** dataset using **FedAvg** algorithm on **VGG16** model. . Each entry represents the test-set accuracy received at given rounds of communication.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | IID | | | | Non-IID | | | |
| 10 | 20 | 50 | 100 | 10 | 20 | 50 | 100 |
| 0.1 | 68.34 | 77.84 | 84.95 | 85.56 | 10.44 | 12.81 | 20.00 | 31.80 |
| 0.2 | 77.08 | 82.90 | 86.80 | 88.56 | 33.39 | 41.35 | 60.19 | 67.81 |
| 0.3 | 80.34 | 85.15 | 88.04 | 89.24 | 38.13 | 53.61 | 70.65 | 74.68 |
| 0.5 | 81.71 | 86.76 | 89.09 | 89.09 | 51.73 | 66.40 | 77.71 | 81.28 |

In this family of experiments in Table 11, we could see large differences in performance regarding both data distribution case or the cardinality of clients per round. This observation can be derived from the fact that the harder identification task is involved. We see *C* = 0.1 produce poor results on non-IID setting and increase *C* extremely mitigating this problem. Convergence speed analysis can be conducted here. Figure A7, A8 show more illustrative insights.

**Table 12.** Different local computational imposed on each client per round under *C* = 0.2 using **VGG16** model. **FedAvg** is used. **VNPlant-200** is under investigation.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| E | B | IID | | | | Non-IID | | | |
| 10 | 20 | 50 | 100 | 10 | 20 | 50 | 100 |
| 5 | 10 | 77.08 | 82.9 | 86.8 | 88.56 | 33.39 | 41.35 | 60.19 | 67.81 |
| 5 | 16 | 78.48 | 83.41 | 87.59 | 88.93 | 31.61 | 41.59 | 58.60 | 68.76 |
| 5 | 32 | 79.38 | 84.13 | 86.69 | 88.28 | 31.51 | 44.14 | 57.39 | 66.66 |
| 1 | 10 | 55.60 | 70.00 | 81.20 | 85.71 | 21.49 | 36.04 | 48.56 | 63.11 |
| 2 | 10 | 65.03 | 78.36 | 84.75 | 88.64 | 26.96 | 38.33 | 58.53 | 67.34 |

Again, in Table 12, we see there are no significant differences between those cases. This implies the stated arguments in the original paper are not universal. Hence, putting effort in tuning this kind of parameter needs to be studied more extensively. Table 13 experiments model choice effect. Other intuitive plots are resided in appendix, Figure A9, A10, A11, A12.

**Table 13.** The effect of various classifiers regarding the **VNPlant-200** dataset. **FedAvg** is the algorithm. The mini batch size, the number of local epochs, the client fraction are 16, 5, and 0.2, respectively.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | IID | | | | Non-IID | | | |
| 10 | 20 | 50 | 100 | 10 | 20 | 50 | 100 |
| VGG16 | 78.48 | 83.41 | 87.59 | 88.93 | 31.61 | 41.59 | 58.6 | 68.76 |
| ConvNext | 86.40 | 91.29 | 93.59 | 94.51 | 30.86 | 48.15 | 68.11 | 73.09 |
| ResNet50 | 82.73 | 87.93 | 91.94 | 93.10 | 34.51 | 48.15 | 72.85 | 82.65 |
| MaxVit | 79.41 | 88.64 | 92.65 | 94.01 | 36.76 | 43.08 | 69.79 | 76.33 |

6.2.2 Comprehensive summarization on FedProx scheme

Follow up previous sections, we conduct similar operations with the same observations on FedProx technique over VNPlant-200 dataset. Since ResNet50 brings best results on former experiments, we keep using this deep network on current class of expriments. Mini batch size is 16, and number of local epochs is 5 (since dozens of our works reveal that the variant in terms of the amount of local update does not impact so much on the ultimate performance). Again, we tune the proximal term weight from predefined set of candidates. The results is showed in Table 14.

Table 14. Experiments upon the weight of proximal quantity on VNPlant-200 dataset. . The classifier is ResNet50.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *µ* | Non-IID | | | |
| 10 | 20 | 50 | 100 |
| 0 | 34.51 | 48.15 | 72.85 | 82.65 |
| 1 | 35.48 | 50.00 | 73.23 | 82.95 |
| 0.1 | 34.76 | 48.48 | 73.13 | 82.37 |
| 0.01 | 34.34 | 50.93 | 73.10 | 82.71 |
| 0.001 | 35.24 | 49.60 | 73.81 | 82.41 |

As we can see, the numbers are quite clear. Adding more constrain into the local losses tends to slightly increase our test-set accuracy. We have not tested with larger µ, but in the publised paper, the authors indicates that huge µ would cause the learning process to be very low.

6.2.3 Heterogeneity advantages study on FedProx

The visualization of Table 14 is shown in Figure 8. The improvement is quite small, but it is still there. Futher inspectation is required to understand the behavior of this hyper parameter. However, adding the proximal term will always guarantee convergence, as proven by the approach’s authors.

Chart, line chart

Description automatically generated

Figure 9. Results comparison between FedAvg and FedProx with various µ values in VNPlant-200 dataset.

7. CONCLUSIONS AND PERSPECTIVES

Federated Learning is truly a novel and intriguing approach for data scientists. Its approach is both similar and different from other decentralized learning methods that have appeared before: the burden of communication costs must be considered, and some effort is required in encoding to ensure data privacy and integrity. If this optimization is done well, we can efficiently leverage the abundant data sources worldwide from end-users, especially as data privacy laws are increasingly tightened and the artificial intelligence industry is reaching saturation due to the lack of increased data sources as before.

In this work, we employed two federated learning methods, **FedAvg** and **FedProx**, on two datasets to examine their efficacy. We tuned the parameters based on the guidance provided in the original paper. Each dataset was split into two portions: a training set and a test set. The training set was distributed among a set of clients, while the test set was used by the server to evaluate the results. We utilized simple preprocessing and data augmentation techniques to test the experimental viability of federated learning. Two data allocation methods were employed: **IID** and non-**IID**. The classifiers utilized in this study were well-known and classic deep learning models. We derived the following conclusions:

1. The averaging of model parameters is truly effective, especially in the case of **IID**. In the case of non-**IID**, the results are also promising, even without any significant data augmentation methods and only using simple optimization methods.
2. The higher the number of clients participating in each round of communication, the higher the model performance. Of course, ensuring accuracy at the beginning of each round depends on practical conditions, network connectivity, and device availability. However, in general, the more data coming from different sources each round allows the model to converge closer to the optimal point.
3. Adjusting the local update quantity per client per round does not significantly improve performance. As long as this update quantity balances computational and communication costs and is not updated excessively in one round, the model's convergence is ensured.
4. The choice of classifier for each problem depends on relevant studies and the nature of the problem and data, rather than the federated learning method itself. Of course, the model must be selected to be suitable for the hardware capabilities and data quantity at each client.
5. Non-**IID** remains a significant challenge: experiments consistently show a sharp decline in accuracy in the non-**IID** setting, and even converge to a saturation point of average accuracy despite increasing rounds of communication. **FedProx** seems to fall short of achieving the maximum attainable accuracy that can be compared to the **IID** setting (and even worse than the centralized training setting). Nevertheless, FedProx with appropriate parameters still provides a slight improvement. One thing to clarify is that we did not apply the approach of discarding clients that cannot complete the assigned training task. It is possible that we will integrate this in future studies.
6. Federated Learning results can vary significantly when the difficulty level of the task changes, and the impact of hyperparameters also varies accordingly. However, there is still a safe range for the parameters that determine the computation load per round at each client. As for the trend of the client fraction parameter, it remains unchanged.

We have observed a significant aspect worth investigating: defining the parameters of the optimization solver. More advanced methods such as **RMSProp**, **GD** with momentum, and **AdamW** can be used. Learning rate decay can also be considered.

**CONFLICTS OF INTEREST:** The authors declare no conflict of interest.

8. REFERENCES

[1] S. Pouyanfar *et al.*, “A Survey on Deep Learning,” *ACM Comput Surv*, vol. 51, no. 5, pp. 1–36, Sep. 2019, doi: 10.1145/3234150.

[2] W. G. Hatcher and W. Yu, “A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends,” *IEEE Access*, vol. 6, pp. 24411–24432, 2018, doi: 10.1109/ACCESS.2018.2830661.

[3] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “ArcFace: Additive Angular Margin Loss for Deep Face Recognition,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2019, pp. 4685–4694. doi: 10.1109/CVPR.2019.00482.

[4] J. Guo, J. Deng, A. Lattas, and S. Zafeiriou, “SAMPLE AND COMPUTATION REDISTRIBUTION FOR EFFICIENT FACE DETECTION,” in *ICLR 2022 - 10th International Conference on Learning Representations*, 2022.

[5] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, “The MegaFace benchmark: 1 million faces for recognition at scale,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. doi: 10.1109/CVPR.2016.527.

[6] A. Nech and I. Kemelmacher-Shlizerman, “Level playing field for million scale face recognition,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.363.

[7] S. Yang, P. Luo, C. C. Loy, and X. Tang, “WIDER FACE: A face detection benchmark,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. doi: 10.1109/CVPR.2016.596.

[8] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, 2017.

[9] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “FEDERATED OPTIMIZATION IN HETEROGENEOUS NETWORKS Tian,” *MLSys*, 2020.

[10] R. McDonald, K. Hall, and G. Mann, “Distributed training strategies for the structured perceptron,” in *NAACL HLT 2010 - Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, 2010.

[11] S. Zhang, A. Choromanska, and Y. Lecun, “Deep learning with elastic averaging SGD,” in *Advances in Neural Information Processing Systems*, 2015.

[12] M. F. Balcan, A. Blum, S. Fine, and Y. Mansour, “Distributed learning, communication complexity and privacy,” in *Journal of Machine Learning Research*, 2012.

[13] O. Shamir and N. Srebro, “Distributed stochastic optimization and learning,” in *2014 52nd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2014*, 2014. doi: 10.1109/ALLERTON.2014.7028543.

[14] Y. Zhang, J. C. Duchi, M. I. Jordan, and M. J. Wainwright, “Information-theoretic lower bounds for distributed statistical estimation with communication constraints,” in *Advances in Neural Information Processing Systems*, 2013.

[15] O. Shamir, N. Srebro, and T. Zhang, “Communication-efficient distributed optimization using an approximate Newton-type method,” in *31st International Conference on Machine Learning, ICML 2014*, 2014.

[16] Y. Zhang, J. C. Duchi, and M. J. Wainwright, “Communication-efficient algorithms for statistical optimization,” *Journal of Machine Learning Research*, vol. 14, 2013.

[17] Y. Arjevani and O. Shamir, “Communication complexity of distributed convex learning and optimization,” in *Advances in Neural Information Processing Systems*, 2015.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 1. 2010. doi: 10.1561/2200000016.

[19] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, “Optimal distributed online prediction using mini-batches,” *Journal of Machine Learning Research*, vol. 13, 2012.

[20] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim, “Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data”.

[21] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated learning with non-iid data,” *arXiv*. 2018.

[22] S. KACZMARZ, “Approximate solution of systems of linear equations†,” *Int J Control*, vol. 57, no. 6, 1993, doi: 10.1080/00207179308934446.

[23] T. Strohmer and R. Vershynin, “A randomized kaczmarz algorithm with exponential convergence,” *Journal of Fourier Analysis and Applications*, vol. 15, no. 2, 2009, doi: 10.1007/s00041-008-9030-4.

[24] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. doi: 10.1109/CVPR.2016.90.

[26] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int J Comput Vis*, vol. 115, no. 3, 2015, doi: 10.1007/s11263-015-0816-y.

[27] Z. Liu, H. Mao, C. Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A ConvNet for the 2020s,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022. doi: 10.1109/CVPR52688.2022.01167.

[28] Z. Liu *et al.*, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2021. doi: 10.1109/ICCV48922.2021.00986.

[29] Z. Tu *et al.*, “MaxViT: Multi-axis Vision Transformer,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2022. doi: 10.1007/978-3-031-20053-3\_27.

[30] J. Cheng *et al.*, “Enhanced performance of brain tumor classification via tumor region augmentation and partition,” *PLoS One*, vol. 10, no. 10, 2015, doi: 10.1371/journal.pone.0140381.

9. Appendix

Before starting this section, we are glad to announce that our work on this report has been accepted by IEEE Zooming Innovation in Consumer Technologies International Conference (ZINC) 2023, a place for both industry and academic field. The conference is included in the ZINC 2023 events, which is sponsored by IEEE Serbia and Montenegro Section – Consumer Technology Chapter; the University of Novi Sad, Faculty of Technical Sciences, Computer Engineering and Computer Communications Group and RT-RK Institute for Computer-Based Systems. For more information, please visit: <https://www.gozinc.org/> . Below are the accept email from the organizing committee and our first-version draft of our paper.

Text

Description automatically generated with low confidence

Diagram

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

A close-up of a document

Description automatically generated with medium confidence

Text, letter

Description automatically generated

We also conduct to write another paper to sumit at **Information Systems International Conference** (**ISICO**) 2023. This event is held by **Department of Information Systems, Institut Teknologi Sepuluh Nopember** (**ITS**). The seventh **ISICO** 2023 title is “Breakthrough Information Systems Innovations Toward Digital Resilience, Reinvention, and Transformation”. This year, the conference is in a hybrid plafform: held virtually and on-site (Prama Sanur Beach) in Sanur, Bali, Indonesia on 26-28 July, 2023. For more information, please visit: <https://isico.info>

Text, letter

Description automatically generated

Chart

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated Text

Description automatically generated Text, letter

Description automatically generated

Table A1. Original results from proposed work when evaluating MNIST with on 2NN and on CNN. Each cell represents the communication cost needed to a respective model to achieve desired test-set accuracy. (99% with CNN and 97% with 2NN). Five attempts did not convergence in time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | IID | | Non**-IID** | |
| **C** |  |  |  |  |
|  | 2NN | | | |
| **0.0** | 1455 | 316 | 4278 | 3275 |
| **0.1** | 1474 | 87 | 1796 | 664 |
| **0.2** | 1658 | 77 | 1528 | 619 |
| **0.5** | \_\_ | 75 | \_\_ | 443 |
| **1.0** | \_\_ | 70 | \_\_ | 380 |
|  | CNN | | | |
| **0.0** | 387 | 50 | 1181 | 956 |
| **0.1** | 339 | 18 | 1100 | 206 |
| **0.2** | 337 | 18 | 978 | 200 |
| **0.5** | 164 | 18 | 1067 | 261 |
| **1.0** | 246 | 16 | \_\_ | 97 |

Chart

Description automatically generated

Figure A1. Plots on test-set accuracy over time on IID Brain Tumor Dataset with different client fraction hyper parameter. The figure only shows the FedAvg scores.

Chart, line chart

Description automatically generated

Figure A2. Plots on test-set accuracy over time on non-IID Brain Tumor Dataset with different client fraction hyper parameter. The figure only shows the FedAvg scores.

Chart

Description automatically generated

Figure A3. The effect of different local computing works on each entry with FedAvg. Here we fix C=0.2. The IID version of Brain Tumor dataset is used.

Chart

Description automatically generated

Figure A4. The effect of different local computing works on each entry with FedAvg. Here we fix C=0.2. The non-IID version of Brain Tumor dataset is used.

Chart

Description automatically generated

Figure A5. The classifier selection impact is inspected here with IID Brain Tumor dataset.

Chart, line chart

Description automatically generated

Figure A6. The classifier selection impact is inspected here with non-IID Brain Tumor dataset.

Chart

Description automatically generated

Figure A7. FedAvg on the IID version of VNPlant-200 dataset using VGG16 classifier.

Chart

Description automatically generated

Figure A8. FedAvg on the non-IID version of VNPlant-200 dataset using VGG16 classifier.

Chart, line chart

Description automatically generated

Figure A9. The effect of different local computing works on each entry with FedAvg. Here we fix C=0.2. The IID version of VNPlant-200 dataset is used. (VGG16 classifier)

Chart, line chart

Description automatically generated

Figure A10. The effect of different local computing works on each entry with FedAvg. Here we fix C=0.2. The IID version of VNPlant-200 dataset is used. (VGG16 classifier)

Chart

Description automatically generated

Figure A11. The classifier selection impact is inspected here with IID VNPlant-200 dataset.

Diagram

Description automatically generated

Figure A12. The classifier selection impact is inspected here with non-IID VNPlant-200 dataset.