EasyChair Preprint

№ 11646

A Novel Method for Communication-Efficient and Privacy-Preserving AI Model Generation and Optimization through Federated Learning

Suryabhan Singh and Pethuru Raj

December 28, 2023

A Novel Method for Communication-efficient and Privacy-preserving AI Model Generation and Optimization through Federated Learning

Suryabhan Singh1, Pethuru Raj2
singhsuryabhan72@gmail.com,peterindia@gmail.com

Abstract: *In federated learning (FL), the idea is to train and bring out a single global model collaboratively with the aid of numerous client machines and devices while everything is being coordinated by a central server. However, given the variability of the data, developing a single global model could be problematic for some clients taking part in federated learning. Therefore, in order to deal with the difficulties brought in by statistical heterogeneity and the non-Informally, Identically Dis- tributed (IID) distribution of data, the personalization of the global model becomes essential. In contrast to the earlier research works, we suggest a novel method for creating a customized model. This further encourages all clients to take part in federation even in the presence of statistical heterogeneity. Such an arrangement is to enhance the performance as opposed to serving just a resource for the central server's model training. In order to achieve this personalization, we use hybrid pruning which is a combination of structured and unstructured pruning to identify a small subnetwork for each client. Each pruning technique has been implemented based on the sparsity %. In this proposed work, we have shown the experimental implementation of pruning techniques and their evaluation to reduce the communication cost. This work will also help FL process to work on low bandwidth of the Internet connection.*

Keywords: *Federated Learning, Pruning and Quantization, Model accuracy and performance, Model Compression.*

1 Introduction

Federated learning is an emerging field of study and research in the machine learning (ML) space. In FL, multiple computers / devices from different and distributed companies and environments immaculately cooperate to learn and expose a global model under the supervision of one or more central servers [1]. In federated learning, the client data will be never shared with central server or with other clients. Because of this reason, federated learning model is different from other traditional distributed optimization techniques. However, it requires tackling the perpetual problem of heterogeneous data. FL has two primary settings. One is federated learning between large institutions and other is federated learning across edge devices [2]. In the first case, each client will participate in each round and can maintain its state between the rounds. The second case involving client devices faces more challenges. Client devices cannot maintain state details across rounds. In this paper, we focus on this issue deeply. Many of the standard optimization methods (for example, distributed Stochastic Gradient Descent (SGD)) are unsuitable in federated learning and can incur high communication costs. To overcome this problem, many optimization methods use local client updates. In this process, each client updates its model multiple times before communicating the updated version with the centralized server. This technique will

greatly reduce the communication cost. There are a lot of methods available to work in this type of process. One such method is known as FEDAVG [1]. The original goal of federated learning is training and creating a single global model by linking of client dataset that becomes harder with non IID data [3]. FEDAVG can work with non IID data. The other way to alleviate the statistical heterogeneity via performing personalization in FL [4].

Herein, each client will perform multiple epochs of SGD on its local dataset. In this process, each client will communicate to the server, which, in turn, is averaging all the client models to form a new global model. The FEDAVG recently got new success and have highlighted its convergence issue in some settings [5]. However, on the adaptation side, FEDAVG is the same as SGD and hence also

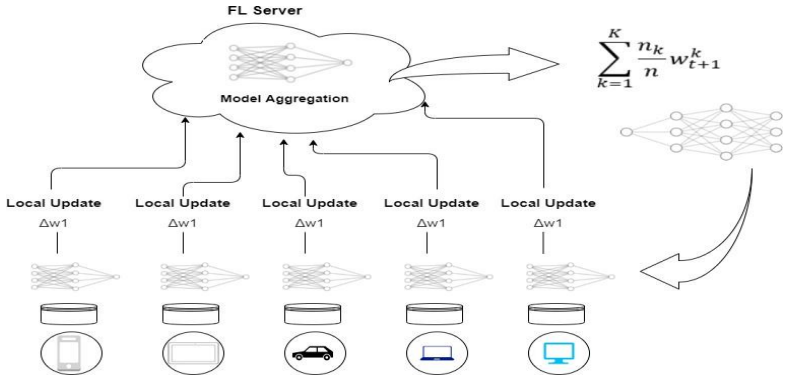

figure 1: Federated Learning.

unsuitable for setting when it must deal with heavy-tail stochastic gradient noise distribution. Self- learning benefits immensely from adaptive learning rates, which take the knowledge of the previous iteration to perform more informed optimization.

The implementation of pruning helps to compress the model size in each communication round. Through a range of experiments on different benchmarks, we observed that the clients with similar data (labels) share some personal parameters. Sometimes similar parameters are not necessary, so the pruning method helps to remove unwanted features and parameters here. We also have discovered through a variety of trials on various benchmarks that clients with similar data (labels) share comparable personal parameters. We effectively calculate the average on the remaining parameters of each subnetwork of each client by locating the subnetwork for each client as opposed to taking the average of all the parameters of all clients for the entire federation. Sub-FedAvg is the name we have given to this unique parameter average. Additionally, under our suggested method, clients are not required to be aware of any underlying data distributions or labeling patterns shared by other clients. Each client's local data, which lacks an IID, enables distinct subnetworks to exist without data exchange. We use real-world datasets to verify and validate our approach to an example of federated picture categorization. Our approach outperforms the state-of-the-art at this time.

1.1 About Our Federated Learning-based Framework

In this work, we have focused on the non-IID properties of the clients' data, and focusing about critical factor of the communication cost and personalization. In

federated learning process, we must need to take care of a few practical issues including client dataset that changes as the data is added and deleted, the availability of clients, corrupt updates by the clients, etc. After studying all these challenges, we propose a framework for federated learning that will address the issues of heterogeneity, communication cost, and personalization. Our framework fits for the second case of cross-device federated learning. Our framework leverages adaptive federated optimization method (using structured and unstructured pruning) and will be providing convergence analysis in general nonconvex setting as vividly illustrated in figure 1.

1.2 The Contributions

Our framework can show and prevent the above-mentioned problems by proposing a single solution and address all the challenges simultaneously. Our main contributions are:

In the current work, we consider a real scenario, where each of the clients participating in the learning process owns limited data with non-IID settings. Here we leverage the statistical heterogeneity as a helping factor with the aim of proposing new framework for personalized federated learning. In the proposed method, clients are not only the source of data for model training, but also contribute to enhance the performance of their personalized target distribution. Our method is a straightforward and effective solution capable of cooperating in the communication process. Also, the solution enables federated learning by defining a subnetwork for each client through a hybrid phenomenon, which is a combination of structured and unstructured pruning based on sparsity. This is also enhanced through the unstructured pruning strategy on the neural model of the clients. The technique of model pruning implementation (for removing unwanted and repetitive features and redundant data) in each communication round will help us to communicate better on low bandwidth network. We have implemented these techniques to compress our model.

2 RELATED WORK

In federated learning, the edge devices that participating in the FL process carry most of the load of computation and communication cost, and a central server updates the model parameters using the updates provided by the edge devices. However, FL has three unique characteristics that will be keeping FL method ahead and better from the parallel optimization system in the following aspects.

Statistical Heterogeneity behavior - The main aim of clients for taking part in FL process is to improve their own model performance. Especially, the client associated with limited secret data will benefits most from collaboratively learned models. However, the clients associated with enough private data, there is not much benefits to participate in FL. This type of issues will occur more badly when we deal with statistical Heterogeneity of the clients. Due to the non-IID distribution of the data across the devices, some new scenario might arrive where some of the participant may gain no benefits by the participation in federated learning process since the global shared model is less accurate than the local models that they can train on their own [6].

Communication Efficiency in Federated Learning- The framework defines and entails each of the clients participating in FL process. Each client sends a full model update back to the global server in every communication round. However, for large networks, this step will be a bottleneck of FL due to the Internet speed and its asymmetric nature. If we consider this scenario, the upload is slower than the download. To overcome this perpetual problem, we have proposed a solution that necessitate less uplink communication cost [7]. For instance, some existing model can reduce the communication cost with structured updates [8]. Others do so by compressing the gradients [9][10].

Personalization and Accuracy for individual Clients - Our framework is basically designed for dealing with non-IID clients, but most of the previous works have been implemented for measuring the global accuracy not the local accuracy. Whenever the clients' context and their personal data are nicely arranged in dataset form, then a globally defined model is bound to perform very well. This is not possible with most of the participating clients. In current scenario, most of the personalization techniques either affect the privacy or it will involve two separate steps where a global model is following the first step and is collaborating with clients. And then the global model is personalized for each client using the client's private data in the second step. These two steps might add extra computational overhead. Each client's data distribution is out of balance, and each client's network speeds, processing capabilities, etc. vary widely. There will be participants with backward iterations if all clients are permitted to take part in federated learning training. The entire system might not finish the combined training if certain clients don't answer for an extended period of time. Therefore, it is important to think about how to select the clients who will take part in the training. The clients who will take part in the training are chosen at random by the FedAvg algorithm. However, the FedAvg algorithm model does not always perform well when the network topology is complicated and the data is not independent and identically distributed.

There are some optimization strategies introduced by a few researchers. In each update of joint training, the Fed-CS algorithm, a greedy algorithm protocol mechanism, chooses the client with the highest model iteration efficiency for aggregate update. This optimizes the federated learning algorithm's overall convergence efficiency. The Fed-CS algorithm can reach improved accuracy, according to the experiments. However, one drawback is that it only works effectively with models that are rather simple, like a virtual dynamic neural network. Fed-CS will lessen the effectiveness of choosing the best aggregation client in cases where the network structure or the number of parameters is more complicated. This increases the number of communications trips and reduces the time efficiency. In order to address this issue of poor performance based on non-IID data on the FedAvg algorithm, the Hybrid-FL protocol algorithm came and this could handle client data whose data set is non-IID. The Hybrid-FL protocol enables the server to choose a few clients through the phases of the resource request in order to create a local data set that is roughly independent and identically distributed for federated learning training and iteration. Additionally, they demonstrate that Hybrid-FL outperforms other federated learning classification methods for non-IID data types in terms of accuracy.

3 METHOD

The CNN architecture has shown significant inferences cost since they have more

layers and due to more learnable parameters. When we consider a FL scenario, the network capacity may be hindered. Therefore, network size must be on the lesser side. If we consider the example of ResNet-152 having more than 65 million parameters and consuming more than 20 billion float-point- operations per second (FLOPs) when training an image with resolution 224*224[11]. This is directly impacting resource-constrained edge devices [12]. There are optimization and compression approaches and algorithms to improve prediction accuracy without compromising on the performance level. Among all the adopted models, we got to know the progressive pruning emerging as the outstanding one. A deep neural network (DNN) is trained and then pruned. Then parameter and hyperparameter optimization techniques are leveraged to keep up their performance [13].

3.1 EFFICIENT LEARNING WITH PRUNING

In this paper, we have shown the practical implementation of an FL scenario. There is a smaller and similar subnetwork for the client which has the same label of data and that can help to enhance the accuracy of each other in this process. Based on the result, we can make some observations here.

- After going with lots of experiments on various CNN and DNN networks on some benchmarked dataset, we noticed that the existence of similar subnetwork and finding the clients with partial similar data and (labels) It is cost-effective to develop new algorithm to improve the accuracy for each sub network.
- Specially in these types, we have considered a feed forward neural network model f(x) that will be dense in nature We consider an initialization parameter for each participating client as well as for the central server. Further we have assumed SGD as our first optimization method for each client's k's in neural network on its own training set and it reaches minimum validation loss on given iteration.
- After this process, in the next step, we have applied a pruning technique to remove the unwanted layer and connection which are not needed and this action will reduce the computation cost. Removing connections and layers will reduce the overall size of the model.
- Pruning techniques that we have implemented here for compression and for reducing the com- mutation cost is based on sparsity. With sparsity techniques, we can remove connection or neuron using structured and unstructured pruning, it will help to find less-featured neuron and connection for removing.
- Pruning will reduce the size of the trained model so that communication between central server and local devices will be easy and efficient. That will help to reduce the high dependency of the Internet service for remote communication.

3.2 STRUCTURED, UNSTRUCTURED, AND HYBRID PRUNING

In the proposed work, we are targeting to find a smaller sub-network for each client. In the FL scenario, there are different pruning techniques such as the channel level pruning (structured), parameter level pruning (unstructured), and hybrid (combination of structured and unstructured). Unstructured level pruning will help to achieve high level flexibility and to gain higher compression rate [14] for both small and deep neural networks illustrated in figure 2.

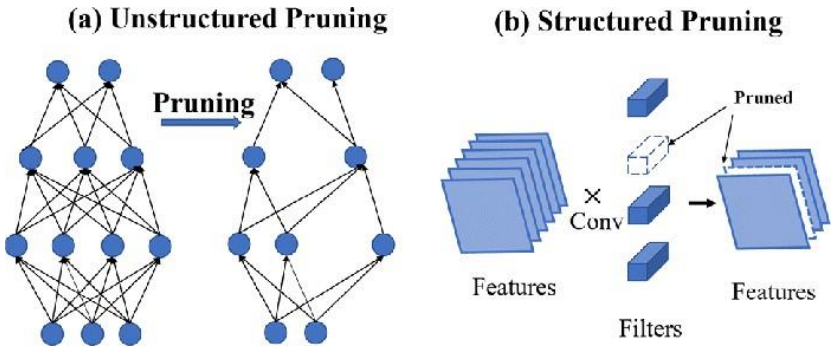

figure 2: Type of Pruning

On other hand, if we take the channel-level pruning, it is easy to remove the neurons of deep CNNs and it is less flexible compared to the unstructured pruning. That means pruning will be good whenever we will consider a deep CNNs [15] Channel-level pruning provides a trade-off between flexibility and ease of implementation. The hybrid-level pruning can be applied on any type of CNNs or fully connected networks. The results of hybrid pruning are better comparatively.

3.3 WHY LEARNING WITH PRUNING IN FL IS EFFICIENT

We consider synchronized algorithms for FL and suggested some steps: 1) Choose clients, which are randomly selected and presented as a subset. Each of them is to download current model parameters from the server. 2) Each client present in the subset will compute and update the model based on its local data and apply pruning to prune the neural network according to the algorithm that we had already explained above to work on a sub-network.3) Then the model get updated on the server for the selected clients.4) On the central server, all the models get aggregated by applying Sub-FedAvg, in which the average is taken only on the intersection of remaining channels (in structured pruning) and the remaining parameter(unstructured pruning) for each client to develop a better global model. In the proposed work, we have explained about the conceptual knowledge of pruning methods and then we have proposed FEDAVG on the server. We have explained about the algorithm and finally we have shown the performance evaluation.

Remark-1: When we start the process of federated learning, each client will download the model from the global server. The model consists of features and all the embodied data of all the clients. After downloading the model from global server, client will start training on its local data. Due to the statistical heterogeneity of clients, filter or channel and parameters are being personalized for each client. We will apply the pruning technique iteratively to prune network based on sparsity and we remove commonly shared parameters of each layer and will keep the personalized parameters that are associated with feature information of local data in each client. Since some of the clients have similar data (label), there is a possibility for similar personalized parameters getting overlapped. By using FedAvg algorithm, we do average the models of the clients on the global server by taking intersection of remaining filter or channel and parameters present of the clients. This kind of model aggregation method is not only giving good impact on performance accuracy of each client with heterogeneous data but also helping in setting up non-IID behaviors to improve the accuracy significantly.

3.4 ALGORITHM

Unstructured Pruning - At each communication round, the global server will randomly select the group of client's 'S' and transmit the model to selected clients. Each client," Ck," will begin training on the local model using their own data set after receiving the model. For each communication cycle, throughout this procedure, a given target pruning with a specified sparsity percentage is derived. And a pruning technique will be used on each cycle of communication as each client trains their model on their own data set. The following is the prescribed pruning method.

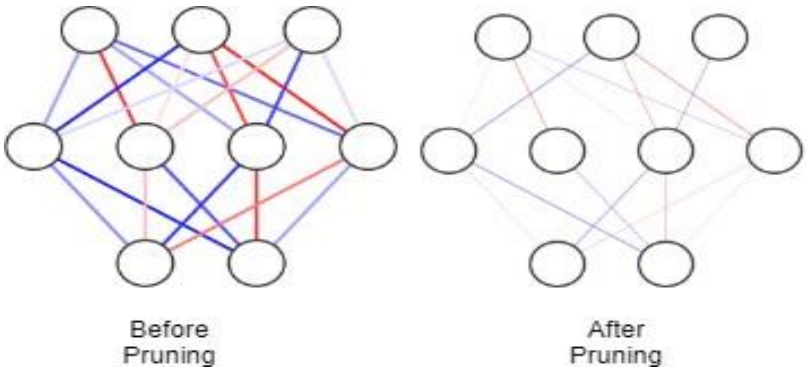

figure 3: Weight Pruning

Weight pruning - The weight matrix is the main consideration in neural network weight pruning. In this method, each end every individual weight present in weight matrix will set to zero. Which will result in the removal of the connections as shown in figure 3.

- Here we are ranking of individual weights in the weight matrix "W" based on their magnitude or its absolute value $|wi, j|$ to find the sparsity of K(in percentage).
- After the above step, we need to focus on the ranked weight matrix and order it according to magnitude/absolute or Norm value (L1-Norm, L-2 Norm) in order to identify the specific weights in the weight matrix that are of lower priority, such weights are not significant and are undesirable so set them to zero with the least K.
- After completing above steps, we remove the weights which are not providing enough information. Those weights are set to zero.
- Now take the sparse weights matrix and remove the connection between the neurons based on steps followed above.

Structured Pruning - We use the same technique that has been followed in unstructured pruning. But here we focus on for neuron (filter) pruning instead of connection pruning. We are adopting the same scenario that we did in the unstructured pruning. That pruning technique will be used on each cycle of communication as each client trains their model on their own data set. The following is the prescribed pruning methods.

Unit/Neuron Pruning - In this type of pruning, the main aim is to focus on neurons present in layers, and focusing on entire column in the weight matrix to be set to zero for deleting the output neurons. This will cause of removal of connection and

neurons both at same time explained in figure 4.

- To achieve the sparsity of K (in percentage), each column of the weight matrix must be rankedbased on their L2-norm.
- Entire weight matrix column is set to zero.
- To remove the neurons, first find the smallest K value and remove the output neurons based on these smallest K value. The network tends to grow less dense as the sparsity percentage (k%) value increases and number of zero in the corresponding matrix will increase. We tested weight pruning and unit pruning for CIFAR and compared their performances using several datasets, such as the MNIST and FMNIST datasets. It may be necessary to lower the frequency arguments
  if, during implementation, the desired sparsity cannot be attained. These networks can require more training time and make hyper-tuning changes to produce the best results. We recommend first copying the original model, and then pruning the copied model with a smaller magnitude.

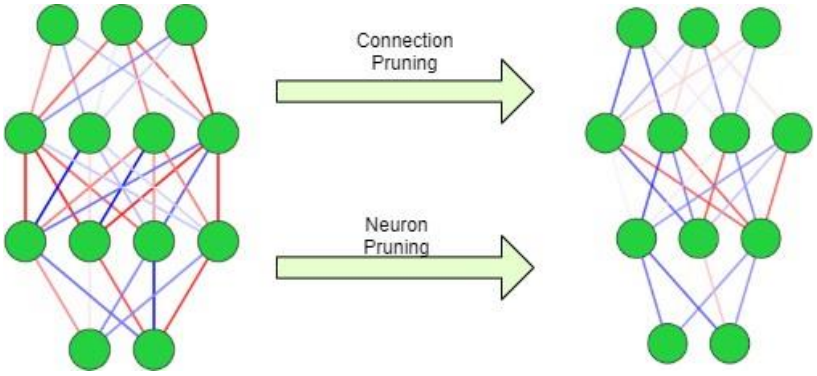

figure 4: Removing connection in neuron pruning

3.5 AI Model Pruning in FL

- To determine the structure to prune the model without degrading performance and prediction, various types of criteria were devised. Here, we focus on two types of these standards.
- Take a network into consideration, train it without pruning, and then, based on the percentage sparsity K, prune and quantize the trained model.
- Consider training a CNN without pruning, followed by applying pruning and quantization to the trained network with additional training and hyper-tuning to achieve the desired compression ratio.
- In the first work, we examine the outcomes for both weight pruning and unit pruning using pruning and quantization on three different dataset types. For this, the pruning model exploration was implemented using the Keras model library. The performance and efficiency of the network in various scenarios have been compared using a set of three different datasets, For instance,
  performing pruning on a trained model, training a model first, doing the pruning, and choosinga random network.
- The problem of pruning is divided into multiple steps. We recommended pruning

  on a custom- built assumed network, using a Keras model without pruning and ReLU-activated four hidden layers neural networks. There isa fully connected

network with 200, 500, 1000, and 1000 neurons in each. The output logit layer is the fifth layer, and its size is set to 10. Output layer connected through all the intermediate layers directly, so as sparse layers, we can use
them for pruning. Each layer present in that process like, the convolution layers, dropout layers, batch normalization layers, and AVG pooling layers have also been excluded. Above-mentioned algorithms run on this network, and models are trained on it without the network being pruned. Datasets used in experiment are MNIST, FMNIST, and CIFAR, which have multiple numbers of classes and input shapes in order to construct the model architecture. The sparsity percentage is also considered when developing architecture in order to reduce the size of the model. An uncompressed Keras model with a dense layer is utilized for modelling with shapes [1000, 1000, 500, 200]. Here we are following the constraints that we assigned 60000 samples for training and 10000 samples for validation and optimizer will update it in each process. After 50 epochs completion, the training result, and their performance metrics with keras without using pruning technique are shown.

- In weight and unit pruning, we have considered the sparsity percentage K as [0.25, 50, 60, 70, 80, 90, 95, 97, 99]. Sparsity will not affect in softmax layer weight, when it is applied in trained pruning method. It takes kernel and bias (for a dense layer) and will return the unit pruned version of each in that way it works. The 'k' weights matrices will be 2-D, 'b' weight will be a 1-D matrix of the biases of a dense layer, and here 'k' sparsity will play a role to set weights matrix as zero based on sparsity percentage. We will get return the kernel weight, a sparse matrix with the same shape as original weight matrix, and a bias weight, a sparse array with the same shape as the original bias array. Here in the proposed work of weight pruning, we have used trained model, it will take k-weights, b-bias and k-sparsity as input arguments. The network will copy the kernel weights and obtain the ranked indices of their absolute k weight in the processing function.
- In k sparsity techniques, the number of indexes is set to be zero.In b weight, firstly it will copy all the bias weights and then it will rank the indexes of their absolute after completion of processing. After completion of these process, we will get back kernel weights and kernel bias. When we consider case of unit pruning, we are giving k weights, b weights, k sparsity as input argument and executing processing function on them. In unit processing function, we are giving k weight as an input argument and copying the kernel weight there. After this we getting ranked indices using column wise L2 norms for k sparsity functions, and then we are setting those indices as zero. Here that mean 2-D weight matrices are set to be zero. Function processing will copy the bias weights and it get ranked based on abs value for all the b weights. In this scenario, the indices in the 1-D bias weight are set to zero called as sparse cut-off indices, which will be same as the indices of the column that was taken out.

3.6 L2-Norm

The L2 norm is defined as by taking the square root of the sum of the square vector values. The L2 norm is used to fit the regularization technique for machine learning, same like the L1 norm. It is a method for keeping the model coefficient low, which will make the model simpler to comprehend. Calculating the L2 Norm will enable you to determine the weight magnitude and rank them by magnitude in this case of pruning and quantization. The square of the weight matrix, which is the L2 norm for this

weight, will yield a positive magnitude. We can set the weights based on this to obtain the sparsity k. To eliminate the output neurons, we will set the weight to 0 and select the smallest value for k based only on their rank. Whole model can be pruned using selection" weight" and" neuron" after the sparsity value and other arguments conclude the function execution successfully. We will essentially remove the zeros we set during the sparsity selection phase, Alternately, we can get rid of all the zeros in the model that were caused by sparsity and are associated with weight and bias during the pruning step of any method we have chosen. The process function takes a model with several layers and reduces its weight when we prune the entire model.

Model (the Keras model) and k Sparsity (the model's target sparsity) are passed to the processing methods, which then return a sparse model that is a sparsified version of the original model. A list of the names of each component (w+b) of each layer and a list of the weights for each component (w+b) of each layer are obtained throughout this procedure after copying the temporary sparse model from our original model. Start a list with the new sparse weight and begin iterating over all layers except the last two as the concluding stage in the procedure. After setting zero by the sparsity percentage, this will determine the new weight and bias for the sparse model. The sparsity percentage is 10% in this case, and various existence percentages are evaluated to select the best pair of sparsity for pruning. We can determine how effectively pruning performs for this model by using sparsity. Among most of the instances, 0.5 sparsity (50 percent) has resulted in best model without losing any of the original data. Although there is a possibility of losing original information and accuracy when we make the model sparser, we can still achieve greater compression. But in addition to reducing the size of the model, we also want to maintain the model's accuracy and the original data. We have examined how sparse our three different datasets are in the section that follows.

4 Comparative study among the Pruning Methods

4.1 Prune a trained network and fine-tune

The structure of this section is as follows: In order to develop techniques to prune a trained neural network, we first concentrate on the concept of" non-significance" in functions and neural networks. Additionally, certain decisions made by the neural network will be random. After pruning, we can examine the results and contrast them with the outcomes of other pruning techniques to reach a conclusion. The TensorFlow model, which was chosen for implementation purposes, serve as the basis for the code's final assessment. Function approximations also refer to neural networks. We can

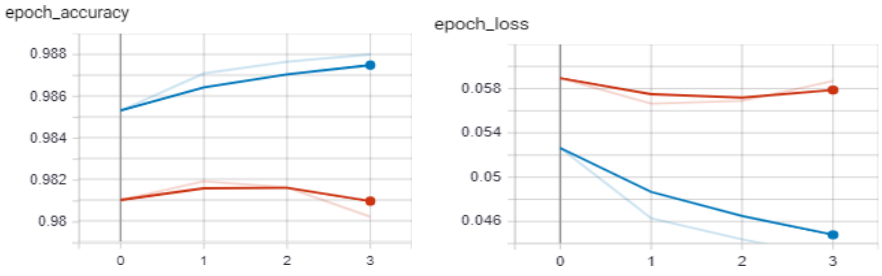

figure 5: Resulting graphs obtained for accuracy and loss

teach them to perceive the representations of input data points, which are crucial and support them to learn parameters as well. Think about the subsequent function.

$$f(x) = x + 5x^2$$

On the RHS of the above function, there are two terms: x and x^2, with a coefficient of 1 for x and 5 for x^2 respectively. From figure 9 and 10, we can observe that there is no significant difference for moving the first coefficient. The provided coefficients have no significance on the original function. Even if we omit the given coefficients, the function will not change significantly. The same can be expanded to and applied to neural networks, although additional information is needed for this to happen. For simplicity, we now consider the weight of a trained network. A challenge emerges when we do this. How can we tell if the weight under consideration is non-significant? To solve the problem, the recommendation is to leverage gradient descent.

We are now applying pruning methods to the model. There are two techniques for pruning when using TensorFlow model optimization. We are now applying pruning methods to the model. There are two techniques for pruning when using TensorFlow model optimization.

- Pick a trained network, then prune it with further training.
- Construct a network with a random initialization and train it using pruning from scratch.

In the subsequent sections, we are to analyze both experiments illustrated in figure 5. Pruning a network using a training schedule is done to improve the training objectives, which will lead to improved gradient updates that will effectively alter the un-pruned weights. Using the TensorFlow optimization kit, you can prune model layers.

4.2 Choose a Trained Network, Prune it with more Pruning

We start by pruning the network that was previously been built and trained. We employ the pruning schedule (to be mentioned by the developer) during the training process to maintain the level of sparsity. The trained model must be recompiled before we can prune, and we will do so using the same process as before. The currently functional parameter has been modified because Tensor flow optimization adds a non-trainable mask here for each weight in the network to indicate whether a particular weight should be pruned. It adds a mask that is either 0 or 1. The pruning model will not affect performance in this situation. The pruning scheme needs to be set at the time of training in order to be used for pruning. Based on the sparsity and magnitude thresholds indicated in the summary of training pruning, the summaries' will be produced. Hyper parameter, helps to provide another pruning schedule when pruning take places. The end-step argument has been set in the pruning schedule to be greater than or equal to the number of epochs we used to train a model. The argument of frequency which will take place when pruning is done to achieve good performance and desired sparsity, must also be taken account here.

4.3 Take randomly initialize network, prune it after training from scratch

Apart from one step, all the stages for this approach are the same as those for the preceding method. In this case, a network that has already been trained is not the starting point. We instead begin with a network that has been randomly initialized It often takes a lot longer when we train a network from scratch. When the network seeks to optimize parameters and sparsity for best performance, this situation occurs

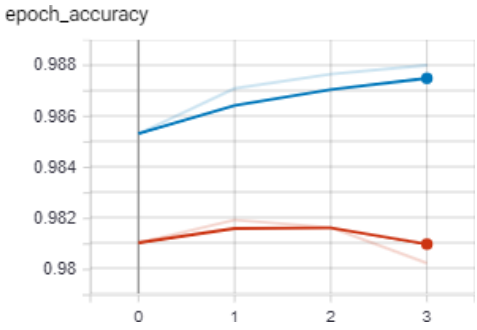

figure 6: Model behavior for pruning randomly initialized network: Accuracy

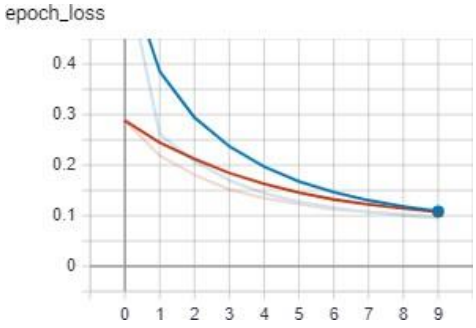

figure 7: Model behavior for pruning randomly initialized network: Loss

We must go deeper into pruning's concept in order to fully graspits potency.

- Export an unpruned and pruned network (model) and compress them and take a note of theirsize.
- Apply quantization technique on both. Quantize and compress the models with the quantizeversion. Take a note of their size and then at last step evaluate their performances.

The model behavior on pruning for randomly initialized network is illustrated in the below graph in figure 6 and 7.

4.4 Performance Evaluation

To compress the model into a zip file, we used the zip file library. For serializing the pruned model time, we additionally need to use the tfmot.sparsity.keras.strip pruning function. It helps to get rid of the pruning wrapper that TensorFlow model optimization added to the model.

4.5 Quantizing the models, compressing them c o m p a r i n g performance

To further compress the model size without degrading the performance, we have quantized our model using Tensorflow Lite library. It needs to be kept in mind that, we need to strip the pruning wrap-pers while passing the model to Tensorflow Lite.It is preferable to load the baseline model that was previously serialized and transform it using TFLite explained in table 1 and table 2. When we want to work on bulk data.

Pruning type	Size (bytes)	Validation Accuracy (%)
Baseline (No Pruning)	78039	0.980199993
Pruning a Trained Network	48338	0.980199993
Pruning and Training a Network from Scratch	48883	0.970000029

table 1: Compression achieved using three types of pruning

TF Lite model type	Size (bytes)	Validation Accuracy(%)
Baseline (No Pruning)	17820	0.9807
Pruning a Trained Network	13430	0.9806
Pruning & Training a Network from Scratch	13224	0.9704

table2: Results obtained after quantization using TensorFlow Lite model

5 Conclusion

The focus of this study is on streamlining the development and deployment of AI models by reducing their size and complexity. The FL approach helps multiple clients to communicate with the central server easily and economically. The network bandwidth consumed is low. Dropping unwanted or redundant parameters will help to reduce the size of a model in each communication round. This activity will take place based on sparsity %, which will help to find which are the features-less parameters.

In each communication round, pruning takes place to enable this dropping. Reducing the model size brings a lot of advantages. Whenever we perform Fed-AVG after pruning of the model on central server, output model will be more accurate, concise, and light weighted.In this paper, we have looked at two methods for deriving lightweight models: pruning and quantization. In this case, models are pruned and compressed without sacrificing accuracy or efficiency, making them ideal for use in low-powered edge devices. The first of our two proposed methods are weight and neuron pruning, which is carried out by selecting the sparsity percentage using weight magnitude and the L2 norm. For 50% sparse MNIST datasets, we get a compression ratio of 3.x using them. Combining this with pruning (without tensor flow optimization) and quantization techniques (with TFLite) yields a compression ratio of 10.x. Pruning and quantization procedures are now being carried out via a Tensorflow optimization tool. The plan includes two distinct sorts of hands-on activities. For pruning in the" take trained network then prune" approach, the compression ratio of roughly 1.6 or 1.x times is attained when utilizing the Tensorflow optimization tool that demonstrates a compression ratio of roughly 10.x may be achieved when the pruning technique is used in conjunction with quantization using Tensorflow optimization.

References

1. Jakub Konecn , H. Brendan McMahan, Felix X. Yu, Ananda Theertha Suresh & Dave Bacon Google ,Peter Richtarik ´ King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia University of Edinurgh, Edinburgh, arXiv:1610.05492v2 [cs.LG], 30 Oct 2017

2. Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G.L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, Sen Zhao, "Published in Foundations and Trends in Machine Learning Vol 4 Issue 1.",https://doi.org/10.48550/arXiv.1912.04977
3. Jiehan Zhou,Shouhua Zhang,Qinghua Lu,Wenbin Dai,Min Chen,Susanna Pirttikangas,Yang Shi,Weishan Zhang, "A Survey on Federated Learning and its Applications for Accelerating Industrial Internet of Things"
4. Liangze Jiang , Tao Lin, Test-Time Robust Personalization for Federated Learning, arXiv:2205.10920v1 [cs.LG] 22 May 2022
5. Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, Ananda Theertha Suresh "Breaking the centralized barrier in cross-device federated learning"
6. Boxiang Lyu ,Filip Hanzely , Mladen Kolar, Personalized Federated Learning with Multiple Known Clusters, arXiv:2204.13619v1 [cs.LG] 28 Apr 2022
7. Keith Bonawitz,Hubert Eichner , Wolfgang Grieskamp ,Dzmitry Huba , Alex Ingerman , Vladimir Ivanov , Chloe Kiddon ´ ,Jakub Konecn,Stefano Mazzocchi , H. Brendan McMahan ,Timon Van Overveldt ,David Petrou ,Daniel Ramage ,Jason Roselander, TOWARDS FEDERATED LEARNING AT SCALE: SYSTEM DESIGN, arXiv:1902.01046v2 [cs.LG] 22 Mar 2019
8. Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, Dave Bacon,"Federated Learning: Strategies for Improving Communication Efficiency", https://doi.org/10.48550/arXiv.1610.05492
9. Yujun Lin, Song Han, Huizi Mao, Yu Wang, William J. Dally," Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training", arXiv:1712.01887 [cs.CV],ICLR 2018
10. Saeed Vahidian, Mahdi Morafah,Bill Lin,"PERSONALIZED FEDERATED LEARNING BY STRUCTURED AND UNSTRUCTURED PRUNING UNDER DATA HETEROGENEITY", arXiv:2105.00562v2 [cs.LG] 10 May 2021
11. QIANG YANG,YANG LIU,TIANJIAN CHEN,YONGXIN TONG,"Federated Machine Learning: Concept and Applications",arXiv:1902.04885v1 [cs.AI] 13 Feb 2019
12. Tian Li,Anit Sahu,Ameet Talwalkar,Virginia Smith,"Federated Learning: Challenges, Methods, and Future Directions",arXiv:1908.07873v1 [cs.LG] 21 Aug 2019
13. Jonathan Frankle, Michael Carbin," The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks",arXiv:1803.03635, ICLR 2019
14. Qiang Yang,Yang Liu,Yong Cheng,Yan Kang,Tianjian Chen,Han Yu,"Federated Learning",https://www.morganclaypoolpublishers.com/catalog_Orig/samples/9781681736983_sample.pdf
15. Qinbin Li,Zeyi Wen,"Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection",https://www.arxiv-vanity.com/papers/1907.09693/