Semantic Segmentation in federated learning setup

Studienarbeit

in Mechatronics

submitted by

Venkata Sai Ganesh Chandu Bheesetty

born on 01. July, 2000 in Visakhapatnam

Written at

Computer Graphics and Multimedia Systems Group
Faculty 12
University of Siegen, Germany.

Advisors:

Prof. Dr. A. Kolb, Lehrstuhl Computergraphik und Multimediasysteme, Universität Siegen

Dr. Paramanand Chandramouli, Lehrstuhl Computergraphik und Multimediasysteme, Universitat Siegen

Started on: 01. April, 2024

Finished on: 24. November, 2024

Declaration of Authorship

I, hereby declare that the research and materials contained in my Project titled “Semantic Segmentation in a federated learning setup” that is submitted to University of Siegen in order to complete Masters of Mechatronics are my own genuine work. Furthermore, publications, articles, and other study materials from other sources have all been properly acknowledged and cited in the reference section. I also confirm that throughout the entire process of working on my project, I followed all ethical norms, research protocols, and academic requirements. In addition, I am aware that any academic fraud, plagiarism, or misconduct in my project may have serious consequences, such as academic sanctions, and maybe institutional disciplinary action.

Siegen, den 24. November 2024

Abstract

Working with federated learning setups has gained vital importance recently due to concerns over data privacy. Semantic segmentation is a crucial task being used in numerous applications, such as medical imaging, self driving cars, and robotic vision, requires precision and efficient model training. In this project, we delve into implementation of semantic segmentation task within a federated learning framework, focusing deployment on edge devices with limited computational resources. Utilizing the cityscapes dataset, performed segmentation tasks employing lightweight, pre-trained model optimized for resource-constrained environments. This study evaluates three federated learning algorithms, such as Federated Averaging (FedAvg), Federated Dynamic (FedDyn), Federated Optimization in Heterogeneous Networks (FedProx), comparing their performance against traditional centralized training approach. To address the issue of generalizability, we integrated the Minimizing Activation Norm (MAN) regularizer, previously proven effective for classification task, into our segmentation models to assess the impact on performance. This regularizer offers improvement in performance of various federated learning algorithms like FedAvg and FedDyn in both uniform and non-uniform number of samples distribution, but fails to improve performance in non-uniform samples scenario for FedProx. This study highlights the potential of federated learning for edge device deployment and the benefits of regularization technique (MAN) in improving model performance.

Acknowledgement

I would like to sincerely thank my Project advisor Dr. Paramanand Chandramouli for all the crucial advice and invaluable guidance and support during my research. I also want to express my gratitude to Computer Graphics and Multimedia Systems Group for giving me the opportunity to work upon this project. Their knowledge and suggestions have been crucial in determining the focus and maintaining quality content of my Project.

Venkata Sai Ganesh Chandu Bheesetty

Contents

List of Figures ii

List of Tables iii

1 Introduction 1

2 Theoritical Background 3

2.1 Semantic Segmentation . . . 3

2.1.1 Deep learning models . . . 5

2.2 Federated Learning . . . 6

2.2.1 Core Challenges . . . 7

2.2.2 Federated Learning Algorithms . . . 8

2.2.2.1 Federated Averaging (FedAvg) . . . 8

2.2.2.2 Federated Dynamic Regularization (FedDyn) . . . 9

2.2.2.3 Federated Optimization in Heterogeneous Networks (FedProx) . . . 10

2.3 Model Architecture and Optimization Framework . . . 11

2.4 Dataset . . . 14

3 Methodology 15

3.1 Data Handling . . . 15

3.2 MAN Regularizer . . . 16

3.2.1 Problem Formulation . . . 16

3.3 Model Modifications for MAN Regularization . . . 18

3.4 Federated Learning Algorithms . . . 20

4 Experiments 22

4.1 Central Training . . . 22

4.2 Federated Learning Model Training . . . 23

4.3 Results of Federated learning Algorithms . . . 24

4.3.1 FedAvg results . . . 25

4.3.2 FedDyn results 27
4.3.3 FedProx results 30

5 Summary and Conclusion **33**

Bibliography **34**

List of Figures

2.1 Different types of segmentation [1] 4
2.2 Convolutional Encoder-Decoder structure [2] 5
2.3 Basic Idea of federated learning 6
2.4 Overview of Federated Averaging [3] 8
2.5 Overview of Atrous convolution and ASPP within DeeplabV3 architecture [4] 12
2.6 MobileNetV2 Architecture 13
2.7 Sample Image and Segmentation Mask of Cityscapes dataset 14

3.1 (a) Loss surface of global model with FedAvg ,(b) MAN regularizer ,(c) FedAvg with MAN loss surface 18

4.1 mIoU vs Epochs for Centrally Trained model 23
4.2 Segmented mask vs Ground truth for Centrally Trained model 23
4.3 FedAvg algorithm with various experimental setups 25
4.4 Segmented mask vs Ground truth for FedAvg model 26
4.5 Segmented mask vs Ground truth for FedAvg MAN model 26
4.6 Segmented mask vs Ground truth for FedAvg non-uniform model 26
4.7 Segmented mask vs Ground truth for FedAvg MAN non-uniform model 27
4.8 FedDyn algorithm with various setup combinations 28
4.9 Segmented mask vs Ground truth for FedDyn model 28
4.10 Segmented mask vs Ground truth for FedDyn MAN model 29
4.11 Segmented mask vs Ground truth for FedDyn non-uniform model 29
4.12 Segmented mask vs Ground truth for FedDyn MAN non-uniform model 29
4.13 FedProx algorithm with various setup combinations 30
4.14 Segmented mask vs Ground truth for FedProx uniform model 31
4.15 Segmented mask vs Ground truth for FedProx MAN uniform model 31
4.16 Segmented mask vs Ground truth for FedProx non-uniform model 32
4.17 Segmented mask vs Ground truth for FedProx MAN non-uniform model 32

List of Tables

4.1 Training accuracy of Federated Algorithms with MAN Regularizer for uniform sampling and non-uniform sampling 24
4.2 Validation Accuracy of Federated Algorithms with MAN Regularizer for uniform sampling and non-uniform sampling 24

Chapter 1

Introduction

Computer vision has emerged as a vital scientific field that aims to interpret images using specialized techniques used in numerous applications, such as object detection, semantic segmentation, and image recognition. Semantic segmentation's goal is to categorize each pixel of the image into a class or object, with numerous implications such as biomedical imaging, robotics, self-driving cars, agriculture and many more [5]. Partitioning image into multiple regions, can help to get a more detailed examination of each region, therefore making it more effective to identify attributes present in the image.

In recent times, data has become a vital catalyst for innovation in the modern era. As a result, the need to protect the security and confidentiality of data has become more paramount. Simultaneously, working with data and improving the machine learning models has been a crucial task. To address these challenges, federated learning (FL) setup has emerged as a promising approach which solves the problem of data privacy, enabling models that can be trained on decentralized data, while reducing the computational burden on edge devices. Federated Learning's ability to preserve user data privacy has led to its increased popularity in diverse fields, including the healthcare industry, IoT devices, and mobile phones.

The growing proliferation of intelligent devices has highlighted the underlying need for efficient machine learning models that are optimized for resource-constraint devices which have restricted computational abilities. However, most of the available pre-trained models for semantic segmentation task need huge computational power, making them unsuitable for real-time applications on portable devices. In this project, we focus on lightweight pre-trained models such as DeepLabV3 with MobileNetV2 as an encoder, which has fewer parameters and are more suitable for deployment on portable devices. This approach finds a balance between complexity of model and its performance in federated learning setup.

Federated Averaging (FedAvg) is the most common FL algorithm used. FedAvg often exhibits convergence issues, particularly when dealing with non-iid data. So in order to tackle non-iid data, other algorithms have been developed such as Federated Dynamic Regularization (FedDyn), Federated Optimization in Heterogeneous Networks (FedProx) and many more. In this project, the above-mentioned two algorithms along with FedAvg are used to test out the performance of the chosen model for the segmentation task.

Improvement for generalization has been a concern even with many federated learning algorithms present. For this purpose, Minimizing Activation's Norm (MAN) regularizer has been introduced. This regularizer aims to create a smoother model landscape which guides the federated learning objectives towards more stable solutions. MAN regularizer has proven to be effective in the case of classification task for model generalization [6].

Overall, this project aims to investigate the performance of semantic segmentation in a federated learning setup using the cityscapes dataset. Exploring pre-trained modes with reduced parameters such as DeepLabV3 with MobileNetV2 as encoder, to enable deployment on edge devices. Specifically, we utilized above mentioned various federated algorithms, experimented with uniform number of samples and non-uniform number of samples data distribution and on top of that, used Minimizing Activations Norm (MAN) regularizer which has been proven to be effective in classification tasks. We aim to explore its impact on segmentation tasks., thereby enhancing the development of accurate and reliable models in real-world applications.

Chapter 2

Theoritical Background

This section aims to provide an overview of concepts and principles that covers this project, focusing on the technical foundations that span various critical domains like image segmentation, federated learning.

Computer vision is a field of study that applies machine learning and neural network techniques to enable computers and systems to derive meaningful information from digital images, videos and other visual inputs and to make recommendations or take actions when they see defects or issues in the broader field of artificial intelligence (AI). Understanding scenes recently has become a thriving area of investigation in the field of computer vision. It involves interpreting the contents of a scene on different levels of detail. Despite many significant advancements, understanding visual scenes still remains a complex task, particularly in resource-constrained edge device environments. The challenges associated with processing and analyzing visual data in these environments necessitate innovative solutions that balance computational efficiency with performance.

2.1 Semantic Segmentation

As discussed earlier, computer vision has been a crucial field of AI and involves various tasks such as object detection, image classification, segmentation, pose estimation and many more.

An image is a collection or set of various pixels. Image segmentation is a task that divides image into distinct pixel groupings, known as image segments to aid with object detection and other comparable tasks. This involves classifying each pixel in the image to a certain label, where pixels with the same label share common properties, like color, brightness or texture.[7]

The field of image segmentation has distinct approaches named as instance segmentation, panoptic segmentation and semantic segmentation. Semantic segmentation is an approach that identifies the belonging class or category of every pixel. Going in this direction, instance segmentation takes

semantic segmentation a step further by providing different labels for separate instances of objects belonging to the same object class. This technique distinguishes between individual objects within the same class. Panoptic segmentation brings together the benefits of both semantic and instance segmentation, enabling the identification of a class for every pixel and the differentiation of individual instances within the same class. [8, 9, 10]

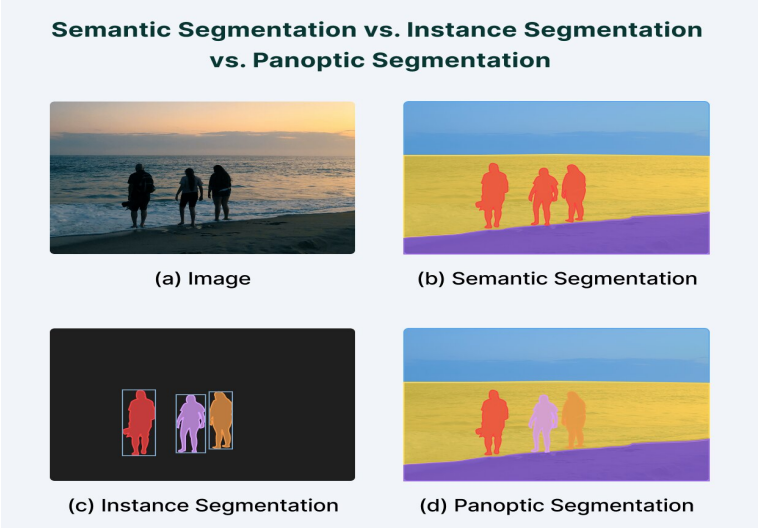


Figure 2.1: Different types of segmentation [1]

Image segmentation has been used in various fields such as

- **Medical imaging**: radiography, magnetic resonance imagery (MRI), computer tomography.
- **Autonomous vehicles**: lane detection, traffic signs navigation and many more.
- **Satellite imaging**: identification of different terrain and topographical features
- **Agriculture**: Helps farmers estimate crop yields and detect weeds from removal.

When it comes to assessing the semantic segmentation tasks, there has been various metrics used such as mean intersection over union (mIOU), precision, recall and pixel accuracy. In this project, mIOU has been used for evaluation, which is also called Jaccard index. Mean intersection over union (mIOU) calculates the ratio of intersection area between segmentation mask and ground truth to the union area of the two, considering each and every pixel of the image.

$$\text{mIoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$$

where, TP: True Positive, FP: False Positive, FN: False Negative

2.1.1 Deep learning models

Neural networks in deep learning, when trained on annotated image datasets, can discover underlying patterns in visual data and identify the salient patterns most relevant to tasks such as classification, segmentation and detection. A common architecture employed in these networks is the encoder-decoder combination to obtain the generation of segmentation masks.

There are prominent deep learning models used for segmentation task, which include

- **Fully convolutional networks (FCNs)**: These are similar to convolutional neural networks with no fixed layers. The encoder network plays a crucial role in segmentation tasks by processing visual data through convolutional layers to extract most relevant information. The relevant data is then compressed and fed into decoder layers, which remove non-essential information and upsample the features to create segmentation masks.[11]

- **U-Nets**: These modify FCN architecture by incorporating skip connections to reduce loss function during downsampling. Skip connections concatenate the encoder feature map with the decoder, improving the backward flow of gradients for enhanced training.[12]

- **DeepLab**: These also modifies FCN architecture. The model uses atrous convolution (dilated) combined with skip connections to capture larger contextual information without the need of additional computational power. [13]

- **Mask R-CNNs**: These are prominent for instance segmentation task. These combine two components: a region proposal network (RPN) that identifies objects and create bounding box around them and FCN-based mask head which refines identified regions and create segmentation masks within each bounding box.

- **Transformers**: Vision Transformer (ViT) using attention mechanism in place of convolution layers, have demonstrated superior performance for computer vision tasks.

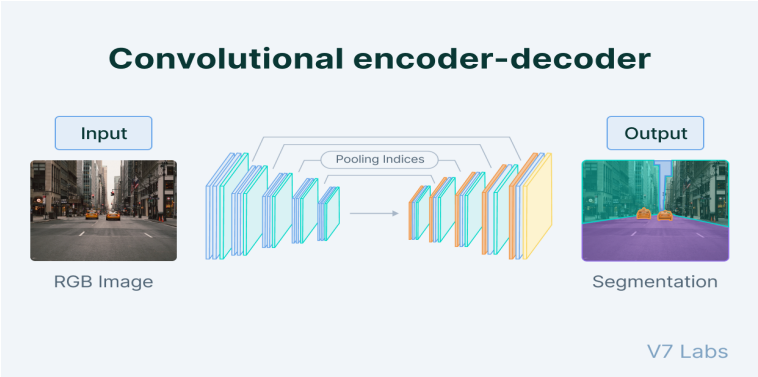


Figure 2.2: Convolutional Encoder-Decoder structure [2]

These deep learning models have revolutionized image segmentation, leading to major improvements in accuracy and efficient algorithms. Even with such advancement of models for the segmentation task, there has been numerous challenges included such as computational complexity, image quality, data annotation and handling diverse data sources.

2.2 Federated Learning

Modern world is characterized by many distributed networks ranging from mobile phones to autonomous vehicles that generate vast amounts of data. As the processing power of these devices increased, along with rising concerns over data privacy, there has been growing interest in keeping the data local and shifting network computation to the edge.

Edge computing isn't a noval concept. It has been a research focus for decades, particularly in the context of distributing computational tasks across low-powered edge devices. The expanding storage and processing capabilities of these devices made it practical to use local resources. This has led to a growing interest in federated learning, a paradigm which explores training statistical models directly on remote devices.[14]

Figure 2.3: Basic Idea of federated learning

There have been numerous applications of federated learning

- *Smartphones*: Smartphones are a prime example of devices that can benefit from federated learning. By jointly learning the user behaviour using a pool of smartphones, statistical models can power applications such as face detection, and voice recognition. However, users are not willing to share their data to protect their privacy. Federated learning can achieve this on edge devices without diminishing user experience or leaking data.

-*Internet of Things (IoT)*: Modern IoT networks, including autonomous vehicles, smart homes and many more, are often equipped with many sensors which collect, process and adapt to data in real time. However, building aggregate models can be challenging in these scenarios, considering the private nature of the data on different devices. Federated learning can help to train models and adapt efficiently to changes in these systems maintaining data privacy.

2.2.1 Core Challenges

We describe here some of the core challenges posed by federated learning. These challenges differentiate the federated setting from other classical problems, such as traditional private data analyses.

Communication and Systems heterogeneity. Communication is a significant bottleneck in federated networks, where raw data must remain local due to privacy concerns. Federated networks can involve millions of devices, such as smartphones, where communication speed is often slower compared to local computation. To address this, strategies include reducing the number of communication rounds or minimizing the size of messages in each round. Additionally, devices in federated networks exhibit significant differences in terms of storage, processing power, and communication capabilities. This heterogeneity requires federated learning systems to be designed to handle low participation rates, accommodate diverse hardware, and remain robust to device dropouts.

Statistical heterogeneity and Privacy concerns. In federated networks, data is often non-identically distributed (non-iid) across devices, deviating from the independent and identically distributed (iid) assumptions commonly used in distributed optimization. This statistical heterogeneity increases the likelihood of stragglers and adds complexity to modeling. Furthermore, privacy is a major concern in federated learning applications, as model updates can potentially reveal sensitive information to third parties or the central server. While techniques like multiparty computation or differential privacy can bolster privacy but can lead to reduced accuracy, model performance or increased computational overhead. Balancing these trade-offs remains a significant challenge in developing effective and private federated learning systems. [14]

To tackle these challenges, there are many federated learning algorithms developed which could mitigate some issues above. In this project, to tackle mainly the challenge of statistical heterogeneity, these algorithms have been implemented and performance analysis has been done.

Some of the federated learning algorithms that have been developed are:
- FedAvg: Federated averaging.
- FedDyn: Federated learning with dynamic regularization.
- FedProx: Federated optimization in heterogeneous networks.
- FedDC: Federated learning on non-IID data with local-drift decoupling and correction.
and many more.

2.2.2 Federated Learning Algorithms

In this project, among the above mentioned federated learning algorithms, three of them are implemented. They are federated averaging (FedAvg), federated dynamic (FedDyn) and federated optimization in heterogeneous networks (FedProx).

2.2.2.1 Federated Averaging (FedAvg)

One of the most common federated learning algorithms used is federated averaging (FedAvg). In a traditional machine learning setup, training data is centralized, meaning all data is gathered in a single location where the model is trained. However, this approach is not always desirable, especially when dealing with sensitive or distributed data. Federated learning addresses these concerns and to improve the efficiency of tasks, specific algorithms are employed.

FedAvg, which is proposed by McMahan, is the most commonly used algorithm because of its straight forward implementation and demonstrated efficacy. The main idea behind FedAvg is to iteratively aggregate locally trained models from multiple clients (devices) to produce a global model that can generalize well across all clients. A *communication round* in federated learning refers to the process in which local models are trained on client devices, then sent to a central server for aggregation into a global model, followed by sending back the updated global model to the clients for the next round. Now, this process is repeated several times until global model converges.

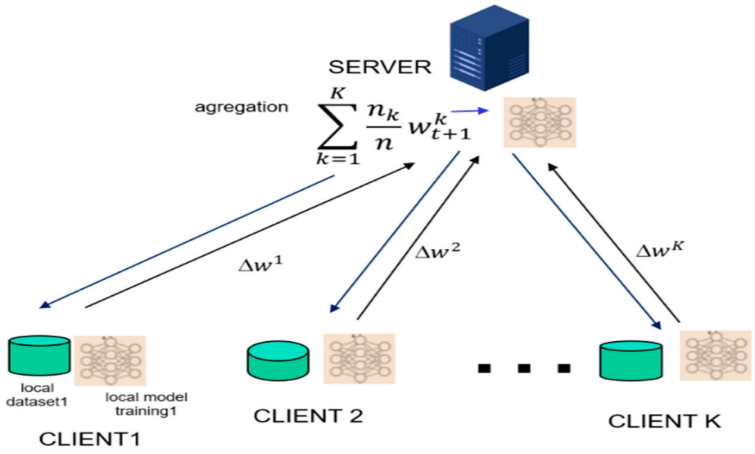


Figure 2.4: Overview of Federated Averaging [3]

How federated averaging works:

- Initialization: The central server initializes a global ML model and distributes it to each participating clients or edge devices. Each client receives a copy of this model.
- Local training: Each client uses its local data to train the model received from the server. Training is done independently, meaning data privacy is secured. This training involves multiple

local iterations to update global parameters based on clients local data

- Model upload: After training locally, each client sends the model parameters back to the server.
- Global aggregation: The server aggregates the models received from all the clients by computing a weighted average of all the model parameters. This is the federated averaging step, where contribution of each client to global model is proportional to size of the dataset it has i.e. weighted averaging. Again, updated global model is then sent back to the clients for next round of local training.
- Iterative process: The above steps are repeated for several communication rounds until the global model converges or achieves necessary efficiency. [15]

The challenges of the federated averaging (FedAvg) include:

-**Non-IID Data**: Variability in data distribution across clients can lead to poor convergence and reduced model performance.
-**Model heterogeneity**: Clients with differing computational resources and data sizes may lead to imbalanced contributions and slow convergence.
-**Robustness to local model drift**: local model updates might drift significantly from the global model if local data is highly variable.
- **Slow convergence**: FedAvg can experience slow convergence, particularly in the presence of non-IID data or heterogeneous client environments.

To tackle some of the above challenges, some algorithms have been developed. Among them, federated learning with dynamic regularization (FedDyn), federated optimization in heterogeneous networks (FedProx) are discussed below.

2.2.2.2 Federated Dynamic Regularization (FedDyn)

Federated dynamic (FedDyn) is an advanced algorithm designed to address some of the limitations of the traditional federated averaging (FedAvg) approach in federated learning. It introduces dynamic regularization technique to improve model performance, particularly in challenging scenarios involving non-independent and identically distributed (non-IID) data.

FedDyn, which is proposed by Durmus Alp Emre Acar, extends FedAvg by incorporating dynamic regularization to enhance robustness and efficiency of federated learning process. This reduces local model variations by adjusting the regularization terms dynamically throughout the training process.

How FedDyn Works: All the processes almost remain similar to FedAvg, so only the changes are mentioned here.

- Dynamic adjustments: FedDyn modifies the regularization term applied during local model training based on the current state of the global model and the local data distribution. This

is designed to penalize large deviations from the global model, reducing the impact of data heterogeneity on the training process.

- Training loss: Each client trains the local model while applying dynamic regularization to guide the training process. This regularization term includes computing difference between local parameters and global parameters which is multiplied by a hyper parameter and added to loss function.
- Server aggregation: Now, the server aggregates the client's model parameters similar to FedAvg by weighted average of all the model parameters.

The use of FedDyn improves the training process by increasing both the speed and stability of global model convergence. Also, improves the robustness of the global model against variations in local data distributions and computational discrepancies among clients. [16]

2.2.2.3 Federated Optimization in Heterogeneous Networks (FedProx)

FedProx algorithm is designed to mitigate the challenges arising from heterogeneous data distributions and diverse computational resources among clients. This introduces a proximal term to the loss function, encouraging local models to stay close to a global model.This term penalizes the deviation of local model updates from global model. This helps to reduce the impact occurred from non-IID data and achieve better performance.

Formulation of the clients loss function:

$$\text{Client objective: } \min_{\theta} \mathcal{L}_k(\theta) + \frac{\mu}{2}\|\theta - \theta_{t-1}\|^2$$

where: $\mathcal{L}_k(\theta)$ represents the local loss function, θ_{t-1} denotes the global model parameters from the previous round, θ represents clients model parameters.

FedProx, also works in a similar way to FedDyn, the only change is addition of proximal term to the local loss function and other steps remain the same to FedDyn. Server also uses weighted average for the clients model parameters. [17]

FedDyn and FedProx are both advanced algorithms designed to address specific challenges, but they approach these challenges differently. They both tackle *data heterogeneity (Non-IID Data)*, *robustness to noisy updates*, *client updates* and many more.

2.3 Model Architecture and Optimization Framework

Many deep learning models have been developed and employed for the task of semantic segmentation, as discussed in subsection 2.1.1. In this project, we have selected DeepLabV3, utilizing MobileNetV2 as the encoder for the segmentation model.

The primary objective of this project is to employ semantic segmentation models on edge devices, where computational limitations are a key concern. Given the potential constraints in computational power on such devices, it is crucial to select a model that minimizes parameter complexity. This ensures that the model remains computationally efficient, addressing our primary requirement. However, model performance is greatly affected when reducing number of parameters, necessitating a careful balance between maintaining high performance and meeting computational efficiency. Considering these into detail, the above chosen model requires only 2 million parameters, significantly fewer than the 23 million parameters required by ResNet50 encoder.

In this section, we will discuss the architecture of DeepLabV3 model in detail. We consider challenges in applying deep convolutional neural networks (DCNNs) to dense prediction tasks. Consecutive pooling operations or convolution striding in DCNNs lead to reduced feature resolution, impeding the model's ability to capture detailed spatial information. To overcome this limitation, we advocate the use of atrous (dilated) convolution. Atrous convolution omits downsampling operations from last layers and upsamples corresponding filter kernels which facilitates the extraction of feature maps with increased density. This technique, analogous to inserting holes between filter weights, offers control over the resolution of computed features within DCNNs without increasing the number of parameters to be learned.

This model consists of two primary components:
a) **Encoder**: The encoder gradually reduces the spatial dimensions of feature maps, capturing long range contextual information at deeper layers. In this project, MobileNetV2 is utilized as an encoder, a lightweight and efficient architecture
b) **Decoder**: The decoder recovers object details and spatial dimensions, refining the segmentation output to align with the input image resolution. Here, the decoder is simplified compared to its extended version, DeepLabV3+, which includes a more elaborate decoding process.

This model employs **atrous spatial pyramid pooling (ASPP)** which captures multi-scale contextual information. Atrous spatial pyramid pooling (ASPP) achieves this by applying parallel atrous convolution which have their unique dilation rates. With these varying dilation rates, effectively sampling the input features at different scales. Dilation rate determines the spacing between kernel elements. So, by this ASPP captures context at both fine-detailed as well as broad information, fuses the information when making segmentation masks.

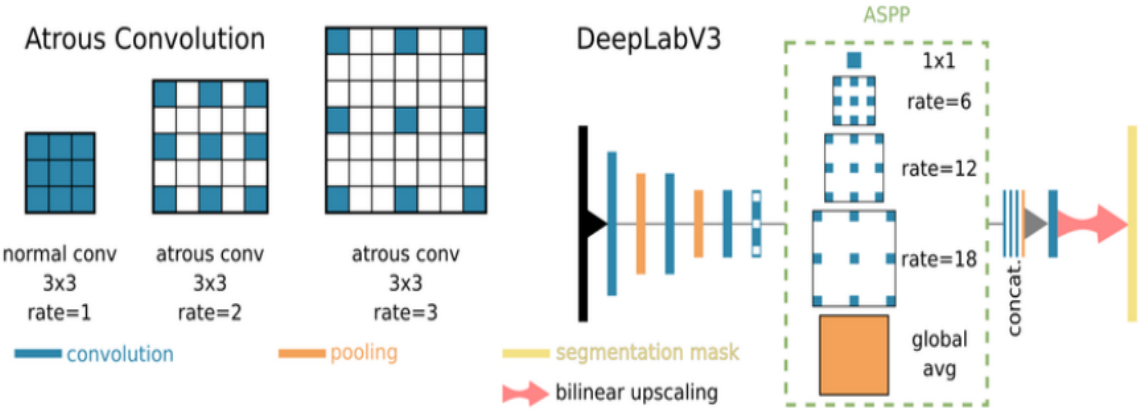


Figure 2.5: Overview of Atrous convolution and ASPP within DeeplabV3 architecture [4]

Now, the encoder employed in this model is MobileNetV2. Instead of the standard residual blocks found in ResNet, MobileNetV2 introduces inverted residual blocks with linear bottlenecks. This architecture helps in reducing memory usage and computational overhead, making it suitable for environments where resources are limited. Inverted residual block begins by expanding the input to a higher-dimensional space, followed by depth-wise separable convolution, and then projected back to a lower-dimensional space. The idea of depth-wise separable convolution is splitting convolution into two separate layers replacing a full convolutional operator. The initial layer is called a depthwise convolution, employing individual convolutional filter to each input channel, reducing computational complexity. The second layer is a 1×1 convolution, called a pointwise convolution, combines the outputs from the depthwise convolution using a linear combination of the input channels. This helps to build new features and significantly reduces the parameters and computations. After expanding and applying depthwise convolution, the block applies a linear bottleneck without using non-linear activation at the final layer to reduce dimensionality. This helps to reduce information loss and maintain the expressiveness of the network. Due to its lightweight design and use of depthwise separable convolutions, MobileNetV2 has faster inference times compared to ResNet, making it perfect for deploying real-time on devices which have restraint on computational power.[18]

While DeepLabV3 itself doesn't have an explicit decoder, the model incorporates a segmentation head to transform the encoded multi-scale features into pixel-wise class predictions. This includes:
1*1 Convolution layer: This layer reduces the number of channels from the concatenated ASPP output to the number of target classes, producing a dense score map for each class at each pixel.
Final upsampling: After the 1x1 convolution, a bilinear upsampling is applied to match the resolution of the input image. This process involves interpolating the feature maps to the original input size, ensuring that the final segmentation map is accurately aligned.

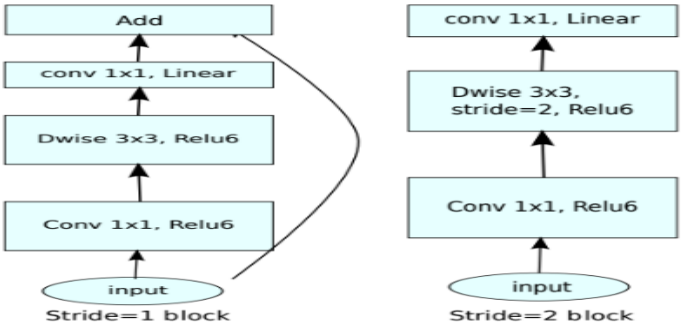


Figure 2.6: MobileNetV2 Architecture

Optimizer

Optimizers and schedulers play crucial roles in ensuring effective learning and convergence of the model. Optimizers are algorithms designed to identify the optimal parameters of model such as weights and biases during model training. Optimizers control how the model learns by determining the step size (learning rate) and direction of the updates. There are many optimizers present, chosen based on the task. In this project, we are using **AdamW** optimizer.

AdamW optimization is a stochastic gradient descent method which incorporates weight decay mechanism to further enhance training performance. This leverages adaptive moment for both first and second order moments. Instead of adding weight decay directly to the loss function, AdamW decouples the weight decay from the gradient-based parameter updates.

2.4 Dataset

The dataset chosen for this project is **cityscapes**. The cityscapes dataset is a large-scale, high-resolution dataset designed for various tasks in computer vision, particularly for semantic segmentation, instance segmentation, and object detection in urban street scenes. This dataset has become a benchmark for evaluating the performance of computer vision algorithms.

This dataset consists of two types of annotations: fine and coarse. There are 20000 coarse annotations and 5000 fine annotations with high resolution of 2048*1024. Dataset has been created by capturing images from 50 different cities in germany and neighboring countries, across several months (spring, summer, fall), in various weather conditions. The 5,000 finely annotated images are split into 2,975 images for training, 500 for validation, and 1,525 for testing.

After images have been captured, for the segmentation task, all images have been segmented to contain total 30 classes that are grouped into eight categories: flat surfaces (road, sidewalk), human (person, rider), vehicles (car, truck, bus, etc.), construction (building, wall), objects (pole, traffic light), nature (vegetation, terrain), sky, and void (areas that don't belong to any other category).

Challenges and Features

- High complexity: The dataset contains complex urban scenes with numerous objects and detailed occlusions, making it challenging for semantic segmentation algorithms to correctly classify every pixel.

- Fine-Grained Annotations: Has highly detailed, with precise object boundaries, making it suitable for tasks that require accurate spatial understanding of scenes, such as in autonomous driving and robotics.

In this project, the original 30 classes of the dataset has been reduced to 20 classes. This reduction was achieved by consolidating all void classes, some classes from flat surfaces, construction, object and vehicle categories are considered into a single class. This modification was made to simplify the task and focus on the most relevant classes for the project. [19]

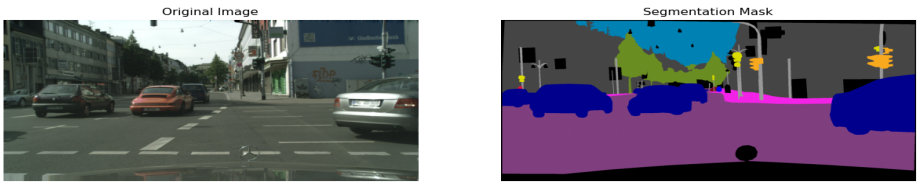


Figure 2.7: Sample Image and Segmentation Mask of Cityscapes dataset

Chapter 3

Methodology

This section outlines the approach for handling data distribution under both IID and non-IID conditions, the problem formulation for the MAN regularizer, the necessary model modifications to integrate the regularizer, and the pseudo-code for all federated learning algorithms utilized in this project.

3.1 Data Handling

In this project, we wanted to perform the analysis of how the pre-trained model behaves when both identical and non-identical distribution of data across clients is done. In-order to replicate IID distribution, we assigned such that each client gets equal number of samples (uniform sampling). For non-IID, we experiment on assigning varying number of samples to the clients based on Dirichlet distribution (non-uniform sampling). We performed experiments based on samples distribution, so we refer to them as uniform sampling and non-uniform sampling.

The Dirichlet distribution is a family of continuous multivariate probability distributions parameterized by a vector of positive real numbers, often denoted as $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_K)$. This models the probabilities of K different categories. In the context of federated learning, this distribution is frequently employed to simulate non-iid data distributions across clients. A smaller concentration parameter α results in a more skewed distribution, leading to higher data non-IIDness (non-uniform number of samples) among clients, while a larger α generates a more uniform data distribution.

The probability density function (PDF) p of the Dirichlet distribution for a K-dimensional random vector $\mathbf{x} = (x_1, x_2, \ldots, x_K)$ is given by:

$$p(\mathbf{x}; \alpha) \propto \prod_{i=1}^{K} x_i^{\alpha-1} \quad \text{for } x_i > 0 \text{ and } \sum_{i=1}^{K} x_i = 1 \tag{3.1}$$

where α can be a vector containing the positive concentration parameters or a scalar.

- If $\alpha \ll 1$, the distribution becomes more skewed, resulting in some clients having much more data than others.
- If α = 1, the distribution becomes uniform.
- If $\alpha > 1$, the distribution becomes more balanced but less uniform, with all clients receiving similar amounts of data.

3.2 MAN Regularizer

This is the introduction of MAN (Minimizing the Activation's Norm) regularizer. Through theoretical analysis, we demonstrate correlation between eigen values of layer wise hessian and activation norms, providing a foundation for methods success. Based on our theoretical insights, we minimize activation norm and dice-loss to attain flatness. This method has been proven effective with the classification task, performed by [6]. Our federated learning setup involves central server which is interacting with n edge devices. It is assumed that each client k has m_k training samples drawn iid from data distribution $D_k(x, y)$ with k from 1 to n. Minimizing the objective function is the target to make the model efficient.

$$\arg\min_{w} \ell(w) \tag{3.2}$$

where

$$\ell(w) = \frac{1}{n} \sum_{k=1}^{n} L_k(w) \tag{3.3}$$

Here, ℓ_k is Dice loss, $L_k(w)$ is the client specific objective function and w denotes model parameters. Now, optimizing $L_k(w)$ directly i.e. FedAvg leads to solutions in sharp minimum. Local models do not generalize well. In-order to avoid this, a regularizer has been introduced that induces flatness to global model. In-order to introduce flatness constraint, minimizing top eigen value of training loss Hessian of global model. This approach builds on previous research, which states that Hessian characteristics, such as its trace and largest eigenvalue, reliably indicate flatness.

3.2.1 Problem Formulation

Now, we construct the problem of FL as a constrained optimization problem with the flatness constraint. We rewrite 3.2 with constraint as below

$$\lambda_{\max}(H(\ell(w))) < \tau \tag{3.4}$$

where $\lambda_{\max}(H(\ell(w)))$ denotes the top eigen value of the Hessian of the loss $l(w)$. Hessian refers to the

square matrix of 2nd order partial derivatives of the loss function with respect to model's parameters. We now use the above constraint and add it to the loss with hyper-parameter $\zeta > 0$ balancing loss and flatness.

$$\arg\min_{w} \hat{\ell}(w) = \ell(w) + \zeta \lambda_{\text{max}}(H(\ell(w))) \tag{3.5}$$

It is hard to optimize the term $\lambda_{\text{max}}(H(\ell(w)))$ for two reasons. Primarily, it cannot access loss function as it is local to each client in FL setup. Another is the computational complexity of calculating eigen value of Hessian. So, we simplify optimization in Eq.3.5 by upper bounding the $\lambda_{\text{max}}(H(\ell(w)))$.

As Hessian is Jacobian of the gradient, we get

$$\mathbf{H}(\ell(\mathbf{w})) = \frac{1}{n} \sum_{k} \mathbf{H}(L_k(\mathbf{w})) \tag{3.6}$$

We use the above equation in 3.5, to get

$$\lambda_{\text{max}}(\mathbf{H}(\ell(\mathbf{w}))) \leq \frac{1}{n} \sum_{k} \lambda_{\text{max}}(\mathbf{H}(L_k(\mathbf{w}))) \tag{3.7}$$

Using equations (3.7), (3.3) ,we upper bound the loss $\hat{\ell}(w)$ by the following

$$\tilde{l}(\mathbf{w}) = \frac{1}{n} \sum_{k} (L_k(\mathbf{w}) + \zeta \lambda_{\max}(\mathbf{H}(L_k(\mathbf{w})))) \tag{3.8}$$

i.e. $\tilde{l}(\mathbf{w}) \leq \hat{\ell}(w)$. We now minimize the $\tilde{l}(\mathbf{w})$.

$$\arg\min_{\mathbf{w}} \ \tilde{l}(\mathbf{w}) = \frac{1}{n} \sum_{k} (L_k(\mathbf{w}) + \zeta \lambda_{\max}(\mathbf{H}(L_k(\mathbf{w})))) \tag{3.9}$$

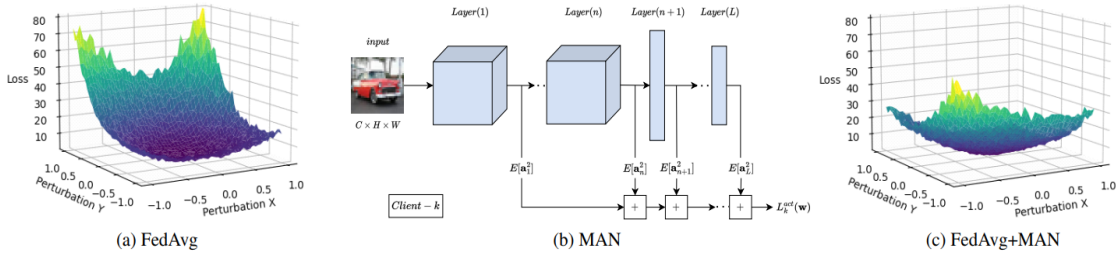
Minimizing upper bound $\tilde{l}(\mathbf{w})$ will minimize $\hat{\ell}(w)$, which is our true goal. So, we modified the problem of 3.2 by including the flatness constraints in 3.9. Instead of just minimizing the $L_k(w)$, each client now minimizes the $L_k(\mathbf{w}) + \zeta \lambda_{\max}(\mathbf{H}(L_k(\mathbf{w})))$. The ζ trades off between the flatness and training loss.

The highest eigen value of layer-wise Hessian scaled with a constant upper bounds total eigen values of overall Hessian $\lambda(H)$. So, now we need to minimize layer-wise $\lambda_{\max}(H)$. Top eigen value $\lambda_{\max}(H)$ of the layer-wise Hessian is upper-bounded by the activation norm of inputs to that layer. Thus, top eigen value of Hessian can be reduced by minimizing the activation norm.

The MAN regularizer $\lambda_{\max}(\mathbf{H}(L_k(\mathbf{w}))$ calculates variance of each layer ℓ, designated as a_l, after non-linearity and accumulates (sums) these values across all the L layers.

$$L_k^{act}(\mathbf{w}) \triangleq \sum_{l=1}^{L} \mathbb{E}\left[\|\mathbf{a}_l\|^2\right] \tag{3.10}$$

Now, combining this regularizer to the federated learning algorithms gave better results for classification task performed by [6].



(a) FedAvg (b) MAN (c) FedAvg+MAN

Figure 3.1: (a) Loss surface of global model with FedAvg ,(b) MAN regularizer ,(c) FedAvg with MAN loss surface

3.3 Model Modifications for MAN Regularization

In order to incorporate the Minimizing Activation's Norm (MAN) regularizer into the pre-trained model, specific modifications were necessary to efficiently calculate and integrate the variance of activations into the loss function. As the MAN regularizer aims to enhance model generalization by minimizing the variance of activations, introducing a flatness constraint effective in federated learning setups.

Given this project utilizes a a pre-trained model (DeepLabV3) architecture with distinct encoder, decoder and segmentation head components, the approach to integrate the MAN regularizer was adapted accordingly. Instead of computing the variance of activations after every layer of the model (as might be done in a model trained from scratch), the MAN regularizer was applied only after the encoder, decoder, and segmentation head parts of the model.

1. The variance of activations was computed for the outputs of encoder. As the encoder is crucial for extracting meaningful features from the input data, and controlling the variance here helps maintain a consistent feature distribution across different clients in the federated learning setup.

2. The decoder in DeepLabV3 is designed to upsample and refine the feature maps obtained from the encoder. After the final layer of the decoder, the variance of the activations was computed again.

3. The segmentation head of DeepLabV3 is responsible for producing the final pixel-wise class predictions. After the segmentation head, the variance of activations was computed one more time. This final computation ensures that the output predictions achieve improved resilience against minor disturbances in input features.

So, by selectively applying the MAN regularizer after the encoder, decoder, and segmentation head, the model benefits from reduced variance across key components without significantly increasing computational complexity, leveraging pre-trained weights, enhancing performance on edge devices as the model remains computationally efficient.

Loss function:

The variances calculated from the encoder, decoder, and segmentation head were averaged to derive a single scalar value representing the overall variance of activations across key model components. This value is then combined as an additional regularization term to the primary segmentation loss (dice loss + crossentropy loss).

$$Totalloss = Segmentationloss + \zeta.MeanVarianceofActivations \tag{3.11}$$

3.4 Federated Learning Algorithms

As per the above mentioned FL algorithms in the theoritical background section 2.2.2. Here, pseudo codes of all the federated learning algorithm which have been applied are discussed.

Firstly, the pseudo code for federated averaging (FedAvg) algorithm 2.2.2.1, [15]

Algorithm 1 Federated Averaging: The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, w is the weights and η is the learning rate, C is the fraction of clients that perform computation on each round

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  end for
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end for

ClientUpdate(k**,** w**):** // Run on client k
$B \leftarrow$ (split P_k into batches of size B)
for each local epoch i from 1 to E **do**
  for each batch $b \in B$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
  end for
end for
return w to server

Next, the federated dynamic regularizer pseudo code 2.2.2.2, [16]

Algorithm 2 Federated Dynamic Regularizer - (FedDyn)

Input: T, θ^0, $\alpha > 0$, $\nabla L_k(\theta_k^0) = 0$
for $t = 1, 2, \ldots, T$ **do**
  Sample devices $P_t \subseteq [m]$ and transmit θ^{t-1} to each selected device
  for each device $k \in P_t$, in parallel **do**
    Set $\theta_k^t = \arg\min_\theta L_k(\theta) - \langle \nabla L_k(\theta_k^{t-1}), \theta \rangle + \frac{\alpha}{2}\|\theta - \theta^{t-1}\|^2$
    Set $\nabla L_k(\theta_k^t) = \nabla L_k(\theta_k^{t-1}) - \alpha(\theta_k^t - \theta^{t-1})$
    Transmit device model θ_k^t to server
  end for
  for each device $k \notin P_t$, in parallel **do**
    Set $\theta_k^t = \theta_k^{t-1}$
    Set $\nabla L_k(\theta_k^t) = \nabla L_k(\theta_k^{t-1})$
  end for
  Set $h^t = h^{t-1} - \alpha \frac{1}{m}(\sum_{k \in P_t} \theta_k^t - \theta^{t-1})$
  Set $\theta^t = (\frac{1}{|P_t|} \sum_{k \in P_t} \theta_k^t) - \frac{1}{\alpha} h^t$
end for

Finally, the FedProx pseudo code2.2.2.3, [17]

Algorithm 3 Federated Optimization in Heterogeneous Networks (FedProx)

Input: K, T, μ, γ, w^0, N, p_k, $k = 1, \ldots, N$
for $t = 0, \ldots, T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
  Server sends w^t to all chosen devices
  Each chosen device $k \in S_t$ finds w_k^{t+1} which is a γ_k^t-inexact minimizer of:

$$w_k^{t+1} \approx \arg\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$$

  Each device $k \in S_t$ sends w_k^{t+1} back to the server
  Server aggregates the w's as:

$$w^{t+1} = \frac{1}{|K|} \sum_{k \in S_t} w_k^{t+1}$$

end for

These are the algorithms that have been used for the federated learning framework in this project. These algorithms have been used for performing segmentation task and on top of that MAN regularizer has been added.

Chapter 4

Experiments

This chapter presents a thorough quantitative evaluation of the simulated federated learning (FL) environment, aimed at analyzing the performance of a semantic segmentation model under both uniform samples and non-uniform number of sample distribution. The experiments are run using the cityscapes dataset. For all the experiments, DeepLabV3 + MobileNetV2 is employed, implemented in Pytorch. The model's performance is assessed in both federated learning settings and centralized training for comparison. The analysis offers valuable understanding into the impact of client-side data distribution on the model's ability to accurately segment data in a federated learning setting.

4.1 Central Training

Model is trained on cityscapes dataset (reducing the number of classes to 20) using learning rate as 0.001 and the AdamW optimizer for 250 epochs. The loss function used is a combination of dice loss and cross-entropy loss (dice loss + cross-entropy loss). Several image augmentations are applied during training, including resizing, horizontal flipping, ColorJitter, and normalization. For validation, a simpler preprocessing pipeline consisting of resizing and normalization is employed.

The mean intersection over union (mIoU) is used as the primary performance metric, yielding a train mIoU of 68.42% and a validation mIoU of 49.32%. The mIoU vs epochs graph, along with visual comparisons of the segmented mask predictions against the ground truth, is provided in the subsequent figures.

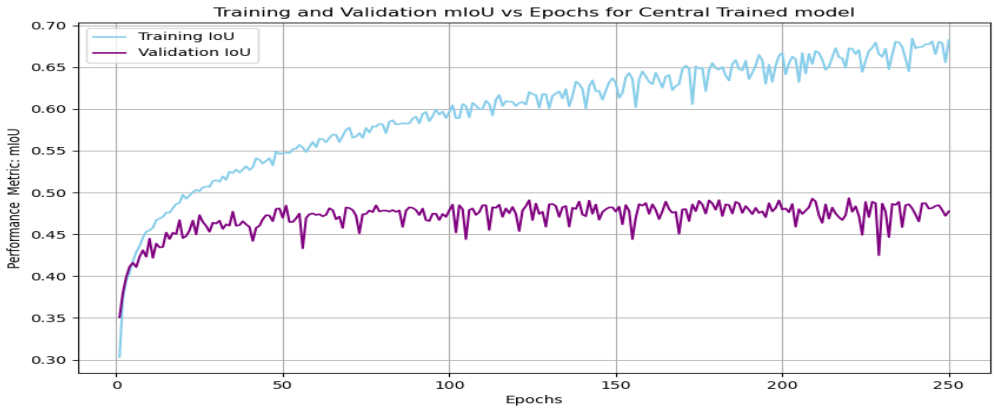


Figure 4.1: mIoU vs Epochs for Centrally Trained model

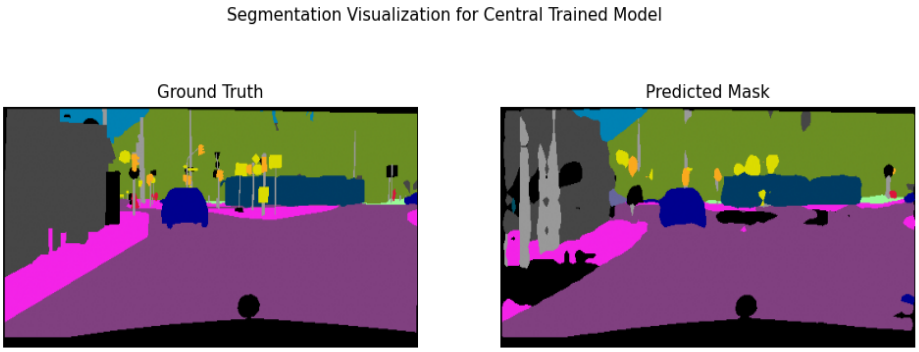


Figure 4.2: Segmented mask vs Ground truth for Centrally Trained model

4.2 Federated Learning Model Training

In the federated learning setup, we experimented with three algorithms: FedAvg, FedDyn and FedProx. All the experiments are conducted with taking 10 clients, 100 communication rounds, 10 local epochs per communication round before transmitting its model updates to the server.

As in central training, learning rate is set to 0.001. We use a combination of dice loss and cross-entropy loss (dice-loss + cross-entropy loss) as a loss function, AdamW as an optimizer with weight decay of 0.01.

For all the federated learning algorithms, both uniform sample and non-uniform sample data distributions are explored. Dirichlet distribution with a concentration parameter (α) is employed to distribute

data unevenly across clients. The value of α=0.6 was used, allowing each client to receive varying amounts of training data.

To implement MAN regularizer on these combinations, model modification has been done to accommodate MAN regularizer, which has been explained in 3.3. We have a hyper-parameter involved for this regularizer ζ which has been set to 0.1 for cityscapes dataset. With all these combinations, each algorithm has been trained and validated on both uniform data sampling and non-uniform sampling and comparatively analysis is done with the centrally trained model (where there is no data distribution among clients)

4.3 Results of Federated learning Algorithms

The results of all the aforementioned combinations are displayed in the table below.

mIoU Training Accuracy:

Algorithm	α = 0.6 (non-uniform sampling)	uniform sampling
FedAvg	74.11%	70.38%
FedAvg + MAN	75.19%	71.61%
FedDyn	71.03%	70.29%
FedDyn + MAN	73.8%	71.62%
FedProx	69.79%	68.09%
FedProx + MAN	68.5%	68.77%

Table 4.1: Training accuracy of Federated Algorithms with MAN Regularizer for uniform sampling and non-uniform sampling

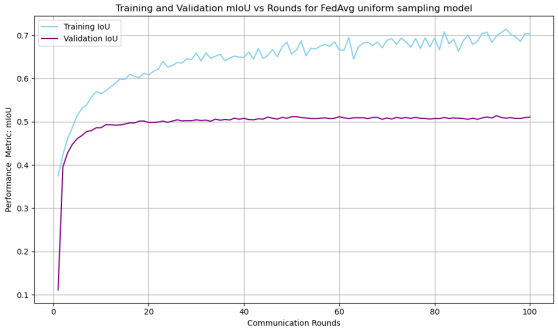
mIoU Validation Accuracy:

Algorithm	α = 0.6 (non-uniform sampling)	uniform sampling
FedAvg	51.26%	51%
FedAvg + MAN	52.04%	51.8%
FedDyn	50.26%	50.9%
FedDyn + MAN	50.5%	51.2%
FedProx	50.8%	50.12%
FedProx + MAN	50.89%	50.65%

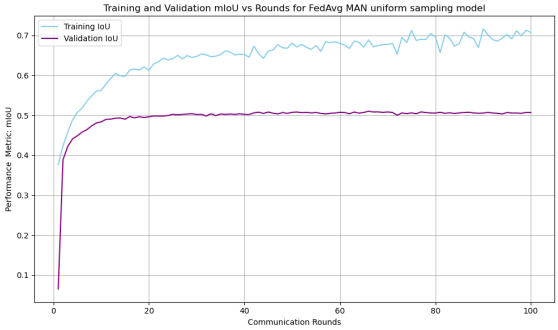
Table 4.2: Validation Accuracy of Federated Algorithms with MAN Regularizer for uniform sampling and non-uniform sampling

4.3.1 FedAvg results

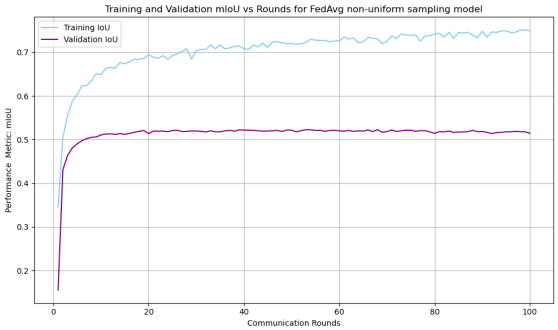
To evaluate the performance of model, mean intersection over union (mIOU) metric was used. For each experimental setup, the mIoU values at various communication rounds are plotted, allowing for a comprehensive comparison of the model's convergence and performance across these combinations.



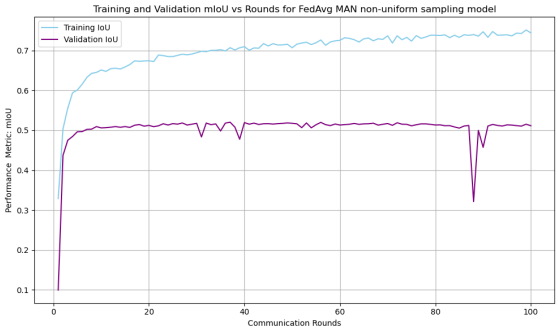
((a)) mIoU vs Rounds for FedAvg uniform model



((b)) mIoU vs Rounds for FedAvg MAN uniform model



((c)) mIoU vs Rounds for FedAvg non-uniform model



((d)) mIoU vs Rounds for FedAvg MAN non-uniform model

Figure 4.3: FedAvg algorithm with various experimental setups

To further analyze the performance of each experimental setup, visual comparisons between the ground truth segmented images and the predicted segmentation maps from the models trained using FedAvg algorithms are made. These visual comparisons offer insights into how effectively each setup captures fine-grained details and boundary accuracy in the predicted segmentation masks. Such visualizations are particularly useful for assessing the practical applicability of the models in real-world scenarios where precise segmentation is critical.

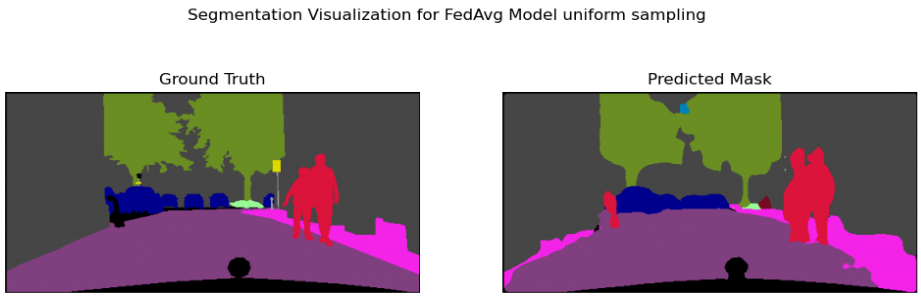


Figure 4.4: Segmented mask vs Ground truth for FedAvg model

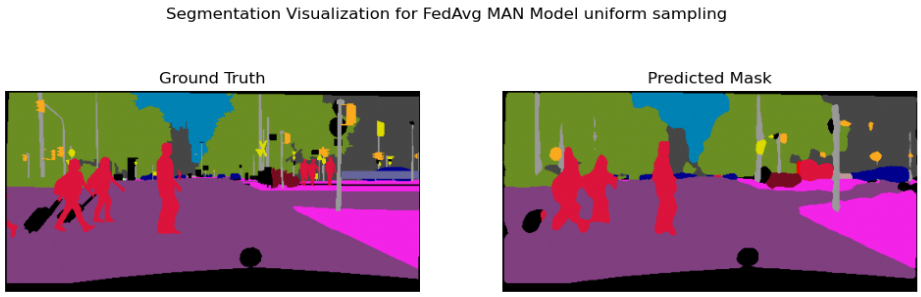


Figure 4.5: Segmented mask vs Ground truth for FedAvg MAN model

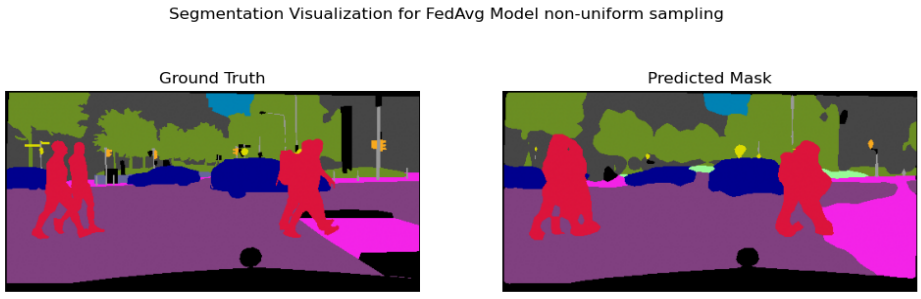


Figure 4.6: Segmented mask vs Ground truth for FedAvg non-uniform model

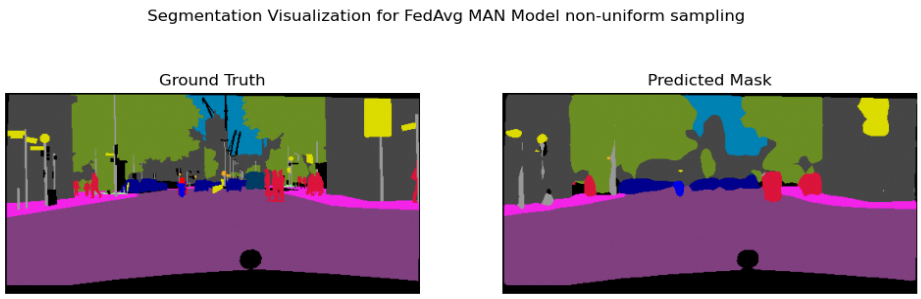


Figure 4.7: Segmented mask vs Ground truth for FedAvg MAN non-uniform model

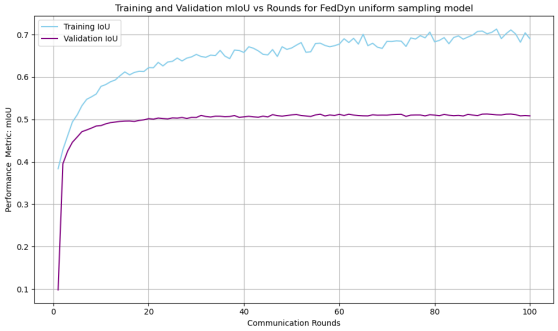
From the above 4 visualization results, it can be observed that FedAvg algorithm with MAN regularizer on non-iid distribution worked best. Also, incorporating MAN regularizer has improved the performance metrics in all combinations of FedAvg algorithms.

4.3.2 FedDyn results

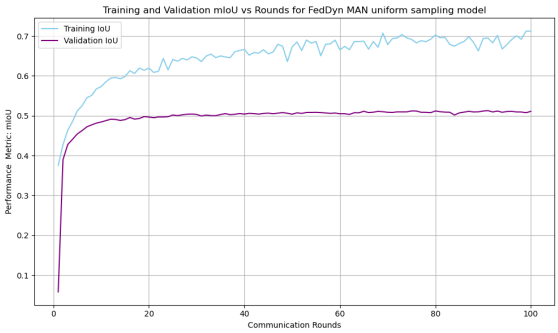
For the FedDyn algorithm, α parameter is set as 0.01, which provides better performance after fine-tuning. Similarly here too mIOU metric is used for evaluation of models. The mIoU values for each Algorithm of FedAvg at various communication rounds are plotted (see 4.8)

Similarly, to above FedAvg, visual comparisons between the ground truth segmented images and the predicted segmentation maps from the models trained using FedDyn algorithms are done (see 4.9, 4.10 ,4.11, 4.12)

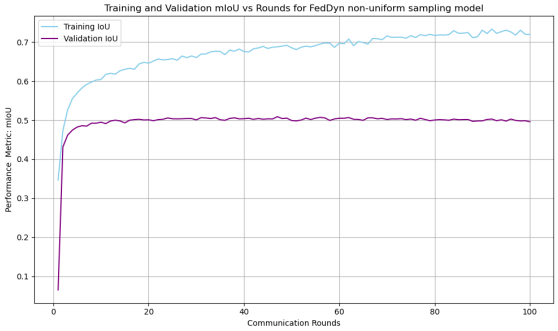
The MAN regularizer worked as expected with the FedDyn algorithm, resulting in improved performance in both the IID and non-IID setups. All the visual comparisons, along with the mIoU vs communication rounds plots for the FedDyn algorithm, are presented below.



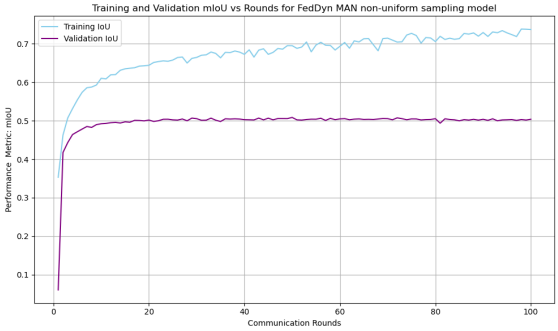
((a)) mIoU vs Rounds for FedDyn uniform model



((b)) mIoU vs Rounds for FedDyn MAN uniform model



((c)) mIoU vs Rounds for FedDyn non-uniform model



((d)) mIoU vs Rounds for FedDyn MAN non-uniform model

Figure 4.8: FedDyn algorithm with various setup combinations

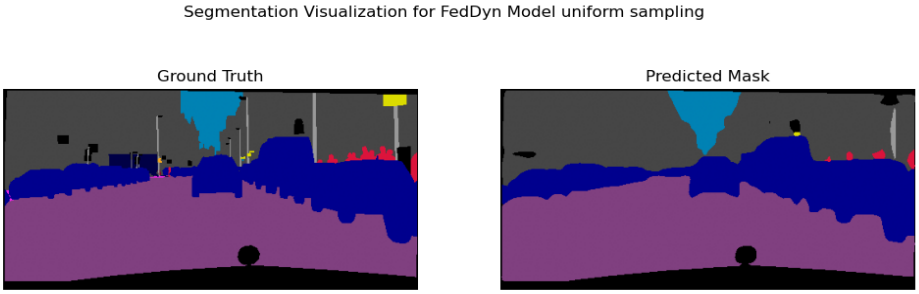


Figure 4.9: Segmented mask vs Ground truth for FedDyn model

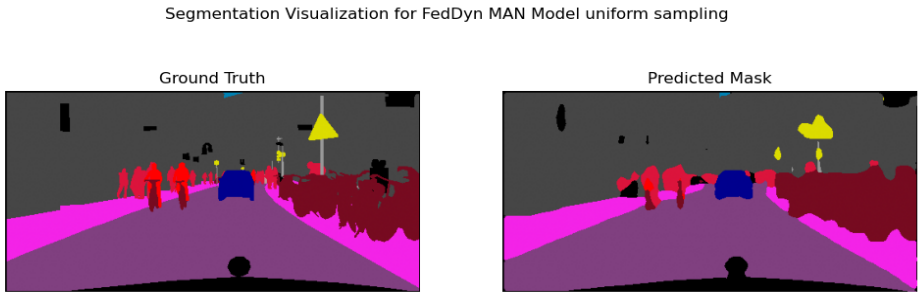


Figure 4.10: Segmented mask vs Ground truth for FedDyn MAN model

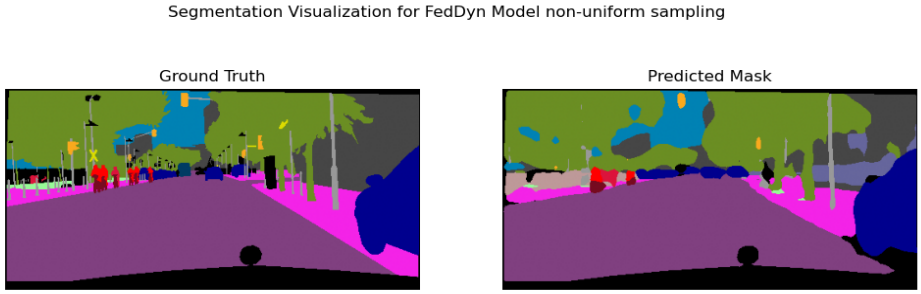


Figure 4.11: Segmented mask vs Ground truth for FedDyn non-uniform model

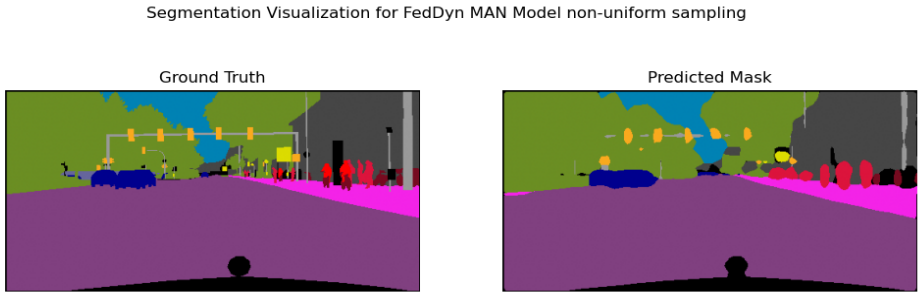
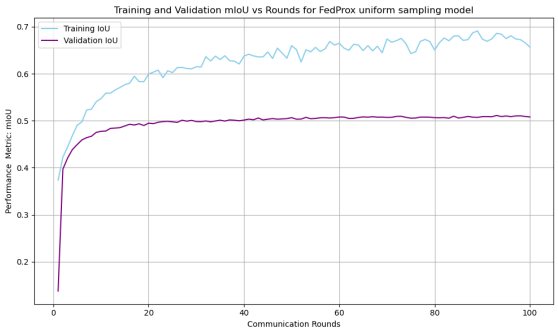


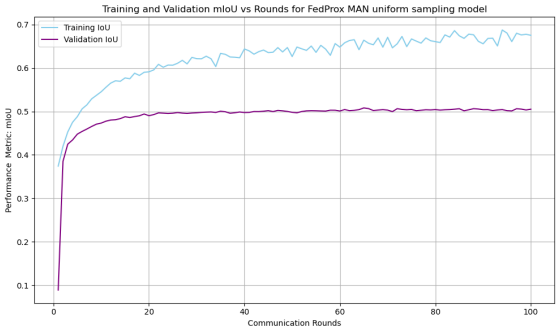
Figure 4.12: Segmented mask vs Ground truth for FedDyn MAN non-uniform model

4.3.3 FedProx results

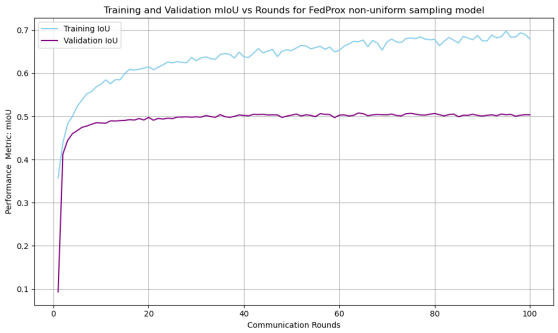
For the FedProx algorithm, μ parameter is set as 0.001, which provides better performance after fine-tuning. The mIoU values for each setup of FedProx at various communication rounds are plotted.



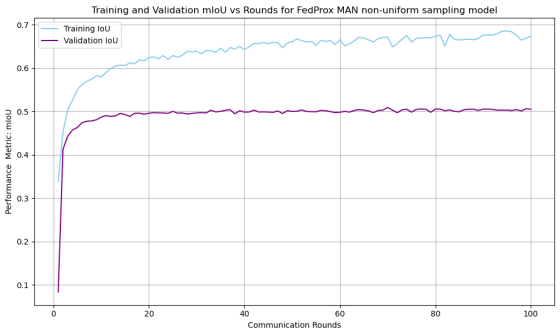
((a)) mIoU vs Rounds for FedProx uniform model



((b)) mIoU vs Rounds for FedProx MAN uniform model



((c)) mIoU vs Rounds for FedProx non-uniform model



((d)) mIoU vs Rounds for FedProx MAN non-uniform model

Figure 4.13: FedProx algorithm with various setup combinations

Visualization Comparisons between Segmented mask and Ground truth for all FedProx algorithms are presented below. (see 4.14, 4.15,4.16, 4.17)

Applying MAN regularizer to FedProx non-iid Model did not yield the expected performance improvement. However, the FedProx IID model showed a positive response to the MAN regularizer, resulting in better performance

.

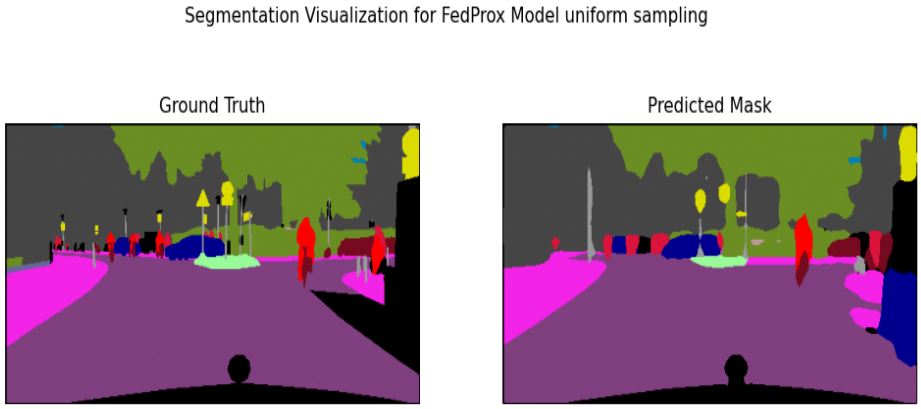


Figure 4.14: Segmented mask vs Ground truth for FedProx uniform model

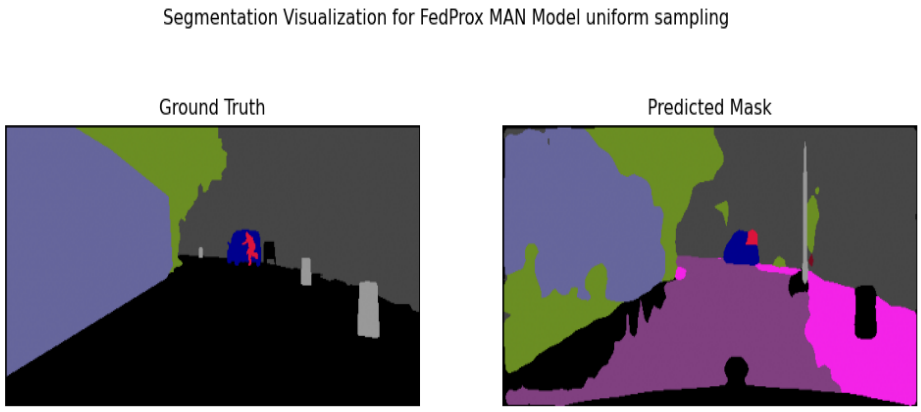


Figure 4.15: Segmented mask vs Ground truth for FedProx MAN uniform model

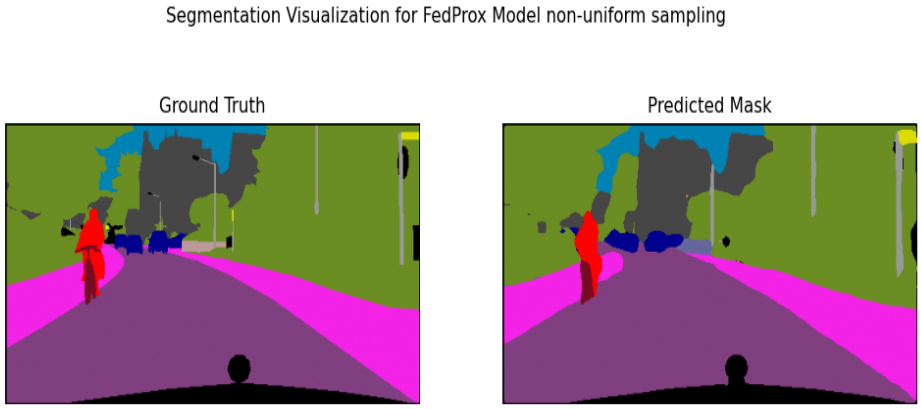


Figure 4.16: Segmented mask vs Ground truth for FedProx non-uniform model

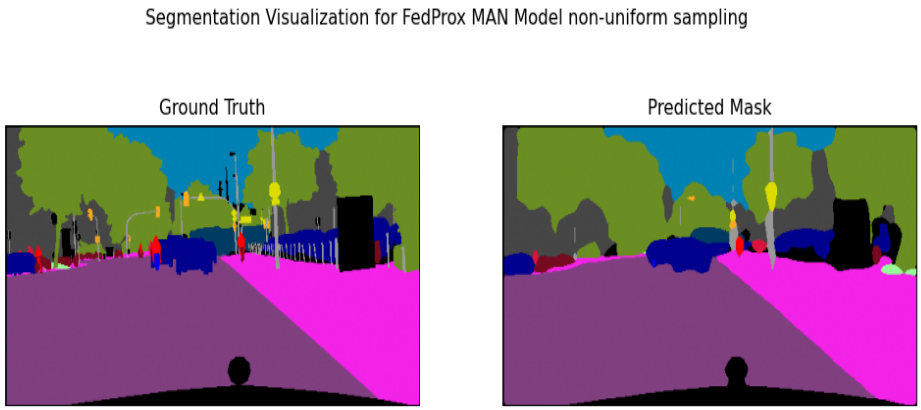


Figure 4.17: Segmented mask vs Ground truth for FedProx MAN non-uniform model

Chapter 5

Summary and Conclusion

In this project, we attempted to tackle how semantic segmentation task performs with model DeepLabV3 and MobileNetV2 as backbone for deployment on edge devices. Task is carried out in federated learning setup and deploying various federated learning algorithms such as FedAvg, FedDyn and FedProx. To address the sharp minimum problem, I also apply the MAN (Minimizing Activation Norm) regularizer. The experiments are performed using both uniform samples and non-uniform sample data distributions, simulating diverse data conditions across clients in a federated learning environment. With this MAN regularizer, we have shown theoretically that this approach minimizes the layer-wise eigen values of Hessian of client's loss, which leads to minimizing top eigen value of overall Hessian of the loss.

In conclusion, this project successfully concludes the possibility of deploying a lightweight DeepLabV3 + MobileNetV2 model for semantic segmentation within a federated learning environment. FedAvg and FedDyn, when combined with the MAN regularizer, improve segmentation accuracy and generalization. However, the FedProx algorithm in non-uniform sample distribution settings does not benefit from the MAN regularizer, suggesting that additional analysis is required to characterize its constraints in this context. The MAN regularizer proves valuable in enhancing model performance across most setups, offering key insights for future work in distributed deep learning systems.

Bibliography

[1] Hmrishav Bandyopadhyay. An introduction to image segmentation: Deep learning vs. traditional. https://www.v7labs.com/blog/image-segmentation-guide, 2021.

[2] Nilesh Barla. The beginner's guide to semantic segmentation. https://www.v7labs.com/blog/semantic-segmentation-guide, 2024.

[3] Krista Rizman Žalik and Mitja Žalik. A review of federated learning in agriculture. *Sensors*, 23(23), 2023.

[4] Thorsten Hoeser and Claudia Kuenzer. Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends. *Remote Sensing*, 12, 05 2020.

[5] Ritu John. Image segmentation: Applications, techniques tools. https://www.docsumo.com/blogs/data-extraction/image-segmentation, 2024.

[6] H. Rangwani A. Singh R. V. Babu A. Chakraborty M. Yashwanth, G. K. Nayak. Minimizing layerwise activation norms improves generalization in federated learning. In *Winter Conference on Applications of Computer Vision (WACV)*, 2024. WACV 1253.

[7] Pulkit Sharma. A step-by-step guide to image segmentation techniques. https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python, 2024.

[8] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. A survey on instance segmentation: state of the art. *International Journal of Multimedia Information Retrieval*, 9(3):171–189, July 2020.

[9] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation, 2019.

[10] Gabriela Csurka, Riccardo Volpi, and Boris Chidlovskii. Semantic image segmentation: Two decades of research, 2023.

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2017.

[14] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, May 2020.

[15] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023.

[16] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization, 2021.

[17] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[19] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.