UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN ICT FOR INTERNET AND MULTIMEDIA 'CYBERSYSTEMS'

"MULTIMODAL FEDERATED LEARNING IN THE AUTONOMOUS DRIVING SCENARIO"

Relatore: Prof. Pietro ZANUTTIGH

Laureando: Mustafa ALGUN
2049537

ANNO ACCADEMICO 2023 – 2024

Data di laurea 11/04/2024

“”To my family and friends, for your endless support and belief in me —thank you.””

Abstract

This thesis investigates multimodal federated learning (FL) within the domain of semantic segmentation for autonomous driving scenarios. Building upon existing research [1] as the starting point of this work, the study experimentally evaluates the performance of FL when employing multiple modalities of data. Specifically, it uses the Cityscapes [2] dataset, both in its standard RGB format and augmented with geometric information, to evaluate the effectiveness of multimodal FL in improving semantic segmentation accuracy.

Utilizing an encoder-decoder architecture consisting of Mobilenet-v2 as the encoder and Deeplabv3 as the decoder, the study was conducted by performing experiments that covered both multimodal and depth-only scenarios.

The findings in this experimental work present valuable insights into the limitations and challenges linked to integrating multiple modalities within the FL framework for semantic segmentation tasks in autonomous driving settings. The study highlights the importance of empirical analysis, illuminates the practical effects of advanced machine learning techniques, and highlights potential implications for future research aimed at overcoming the observed limitations.

Contents

Listing of figures

Listing of tables

Listing of acronyms

FL	Federated Learning
AI	Artificial Intelligence
ML	Machine Learning
FedAvg	Federated Averaging

1 Introduction

The rise of Artificial Intelligence (AI) is transforming the way complex problems are tackled, especially in autonomous driving, where cutting-edge AI technologies are applied. The quest for developing systems capable of navigating complex urban environments with precision and autonomy has never been more crucial. Federated learning emerges as an important approach and addresses the intrinsic limitations posed by conventional machine learning paradigms. This thesis targets the problems of semantic image segmentation within the context of federated learning as it is a domain where understanding the immediate environment through visual data becomes critical for the safe operation of autonomous vehicles. As cities and urban landscapes become more crowded and complex, the demand for innovative solutions that can process and analyze vast amounts of data while respecting user privacy and system efficiency increases. Federated learning offers a promising and privacy-preserving approach by decentralizing data processing and bridges the gap between these requirements in autonomous driving technologies.

Traditional machine-learning approaches face multiple adversities in effectively addressing the modern world's data-oriented challenges. One notable obstacle stems from the centralized nature of traditional learning models, causing hardship in scalability, privacy, and efficiency. Centralization poses significant scalability challenges with large and complex datasets. The idea of creating a structure that prevents potential bottlenecks and performance limitations while relying on a central information source becomes nearly impossible as such systems would require extensive computational resources and infrastructure. Furthermore, traditional machine learning frameworks often struggle to accommodate the needs of data privacy through such model structures. Another crucial concern lies in the efficiency of these approaches. The centralized model training process results in significant computational overhead, making it difficult to have a short response time in real-time problems.

Tackling these problems requires novel approaches that prioritize data privacy, decentralization, and, therefore, computational efficiency. As a solution to these challenges, an alternative approach is advocated, whereby the training data remains distributed across devices, and a shared model is learned through the aggregation of locally computed updates [9]. This decentralized method is referred to as Federated Learning. Handling a Computer

Vision problem within the scope of Federated Learning, this work dives more into the usage of semantic image segmentation - a critical process for understanding and analyzing complex road scenes in autonomous driving scenarios. Semantic segmentation assigns specific classes, such as cars, pedestrians, and buildings, to each pixel in an image, thereby allowing us to possess a detailed understanding of the environment around autonomous vehicles. This technology stands at the forefront of current Artificial Intelligence technology, among others, offering a new perspective to make sense of our world in ways previously unimagined.

The work presented in this thesis is to harness the power of multimodal data to enhance semantic segmentation. As the first type of input source, Cityscapes dataset [2], a rich repository of urban scene images, is utilized in its standard RGB form. Then, an equivalent set of depth images using a Python script is generated. This conversion process creates Horizontal Disparity to Height (HHA) encoded images, offering a depth perspective alongside the original RGB images. This dual-modality approach—combining RGB and Depth (HHA) images— creates the multimodality of this work. By integrating these various data streams, the objective of this thesis is to explore the effects and raise a more comprehensive understanding of its effect on this nuanced Computer Vision challenge. This research builds upon previous foundational work [1], with the ambition of having an experimental take on top of the current work. It proposes a decentralized collaboration network of clients controlled by a central server under the scope of Federated Learning. This design ensures that data privacy is maintained, as each client's data does not leave its local storage. Instead, model updates are shared and aggregated at the server level, updating the global model without compromising individual data privacy.

A set of experimental configurations is designed to systematically explore the integration of RGB and Depth data within the federated learning framework. Varying combinations of encoders and decoders to process this multimodal data are investigated, aiming to discover optimal architectures for feature fusion and segmentation effectiveness. Through this series of methodologically different experiments, this thesis seeks to contribute to the domain of autonomous driving. The findings yielded from this research are expected to provide important insights into semantic segmentation using federated learning approaches, potentially influencing other domains beyond autonomous driving.

2
Frameworks

2.1 Federated Learning

In today's world, where machine learning has started redefining how many things are, and data has become valuable as currency as also expressed in [10], the concept of Federated Learning has emerged as a revolutionary framework. A team of engineers at Google [11] came up with an approach that allows for the collaborative training of machine learning models without the necessity of storing data at a central repository. This sort of approach can be explained as an algorithm traversing multiple data sources and learning from them, and only the model updates are transmitted back to a central server.

Each client in this scenario possesses their own local data, as seen in Figure 2.1. Clients could range from smartphones to hospitals, and they all contribute to the learning process of the model. Each client independently computes an update to the model based on its local data and sends back to the server only these updates, not the data itself. The server then aggregates these updates using various algorithms such as Federated Averaging (FedAvg). By this, the weighted average of the updates is computed and the model refined global model is redistributed to the clients for further improvement. This iterative process continues until the model converges to a satisfactory level of accuracy.

FedAvg handles federated optimization of models by considering the local counterpart $F_k(w)$ of the the global objective function on each device. In this process, it uses any optimizer (stochastic gradient descent (SGD) is used as an example for explanatory purposes) uniformly across devices and can be described as follows:

1. The server initializes the model and distributes it to a selected subset of clients within the federated network.

2. These clients then execute SGD locally, shown in equation 2.1,for a specified number of epochs E, opti-

mizing their local objective functions.

$$w_{t+1}^k = w_t^k - \eta_t \nabla L_k(w_t^k) \tag{2.1}$$

where w_t^k represents the local model parameters for client k at iteration t, η_t is the learning rate, and ∇L_k is the gradient of the loss with respect to the local dataset.

3. Subsequently, each client sends back their model updates (weights and biases) to the server.
4. The central server aggregates these updates to update the global model as in equation 2.2. This updated model is then redistributed to clients, starting a new round.

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{2.2}$$

where w_{t+1} represents the updated global model parameters, K is the total number of clients, n_k is the number of data samples on client k, and n is the total number of data samples across all clients.

5. This procedure is iteratively repeated until convergence is achieved or a predefined stop condition is met.

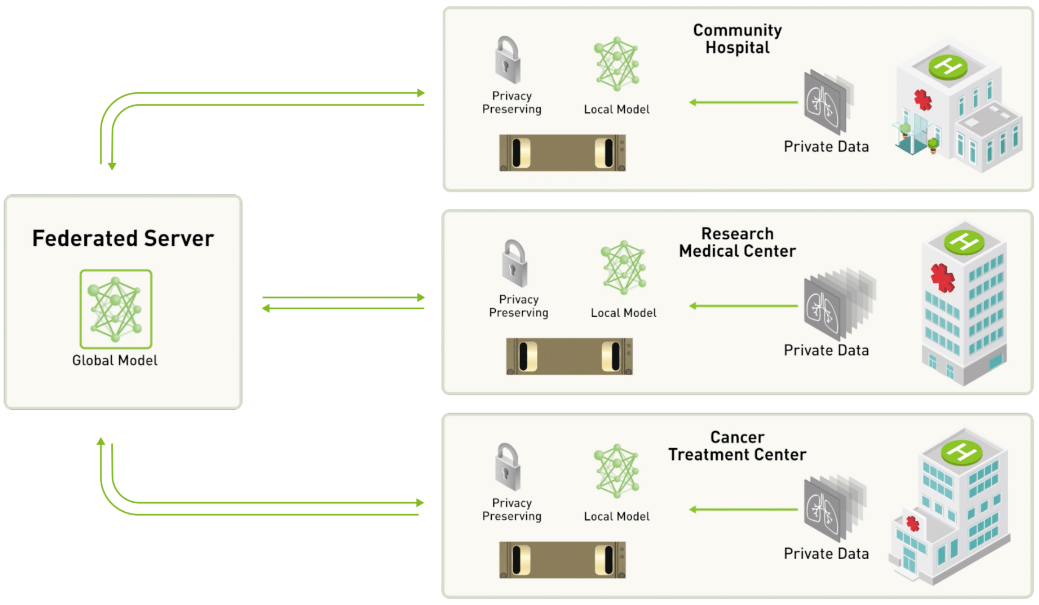


Figure 2.1: Centralized-server approach example to federated learning. Source: NVIDIA [12].

Some of the common assumptions made for the traditional machine learning training approaches are mostly dropped in federated learning. Those assumptions are listed below, also stated in [13]:

- Data is assumed to be sampled as independent and identically distributed (i.i.d) in traditional machine learning algorithms, whereas federated learning assumes different users store various types of data.
- Evenly distributed data assumption does not hold in federated learning, as well. This assumption is violated in almost all real-world problems as it is virtually impossible to achieve.

In this distributed approach, user privacy is preserved by training models directly on remote devices without centralizing data [10]. Moreover, federated learning models may additionally integrate end-to-end encryption to protect against reverse engineering attacks on the raw updates and secure training metadata during transmissions. On top of its privacy advantages, federated learning also stands out with its ability to handle high degrees of systems and statistical heterogeneity [14]. This includes:

- **Systems Heterogeneity:** The clients in the federated learning network might possess a wide range of hardware capabilities and resource availability, such as CPU performance, battery life, and network connectivity. This type of heterogeneity can affect the consistency of participation of devices immensely. A client with a low battery or poor network connectivity may drop out of a given iteration for instance and cause inconsistencies in model training.
- **Statistical Heterogeneity:** Real-life federated learning scenarios don't usually work with independently and identically distributed (i.i.d.) data, unlike traditional centralized ML settings. Clients are likely to have their own unique data distribution depending on their user behavior, geographical location, and other factors. Therefore, the size and shape of data across these devices can vary dramatically. This type of statistical heterogeneity can introduce delays and inefficiencies in the training process and prevent the global model from converging.

Centralized conventional machine learning approaches train the model at the center server or cloud. The information transmitted from clients is used in the server and then stored. This architecture requires a large bandwidth and increases the risk of congestion [15]. Federated learning reduces the latency by eliminating the need to transfer extensive raw data and reduces response time. Low latency and local processing capability that comes with this approach become crucial in scenarios such as autonomous driving, where real-time data analysis and timely response holds paramount importance.

The main objective in federated learning is to minimize a global function formulated as:

$$F(w) = \sum_{k=1}^{m} p_k F_k(w)$$

where p_k defines the relative impact of each device, with $\sum_k p_k = 1$.

The local objective function for the k-th device is typically expressed as:

$$F_k(w) = \frac{1}{n_k} \sum_{j=1}^{n_k} f_j^k(w, x_j^k, y_j^k),$$

where n_k is the number of examples for a given client, f_j^k is the loss of the j-th example on the k-th device, x_j^k and y_j^k are the features and label of the j-th example on the k-th device, respectively. This equation shows the direct relation with local data for global model update

2.2 Semantic Segmentation

Within Computer Vision, semantic segmentation can be considered a crucial domain. It is a subset of artificial intelligence with a focus on analyzing, interpreting, and understanding the visual data by classifying each pixel into a predefined category. As seen in Figure 2.2, the way it differs from classic image classification is by not only identifying objects within an image but also assigning each pixel of the image to a specific class label. This process is called dense prediction, where every pixel intensity value is mapped to a categorical label, making it perfect for analyzing images in a detailed manner. The importance of semantic segmentation lies in its application across numerous fields, such as augmented reality, medical image analysis, video surveillance, three-dimensional reconstruction, and autonomous driving [16].

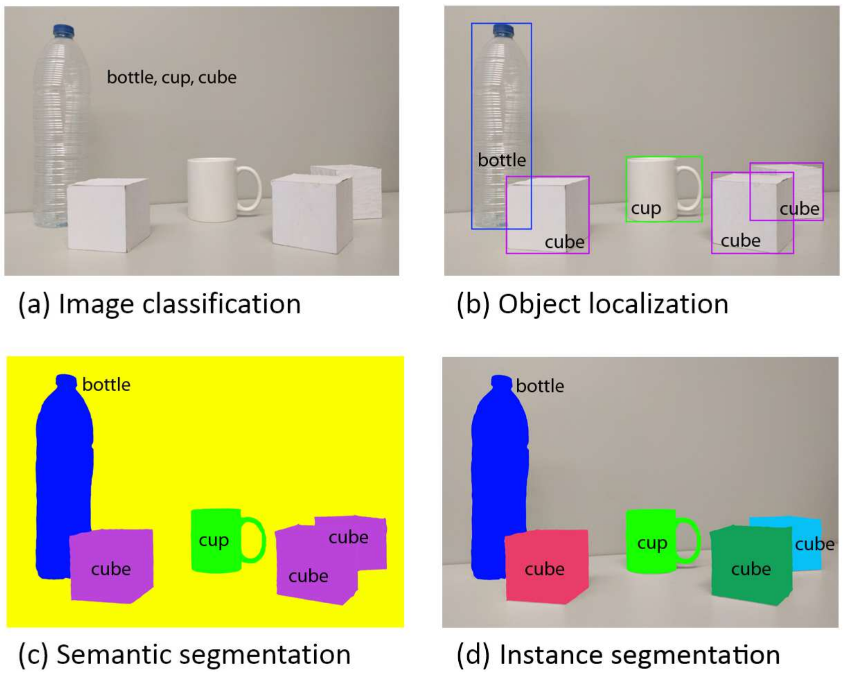


Figure 2.2: Progression in visual comprehension from broad to detailed analysis. Source: [17].

Medical image analysis is also a field where semantic segmentation can be used. Rather than focusing on specific regions for the interpretation, pixel-level and precise explanations of various anatomical structures and pathological regions from medical imaging data are made possible through semantic segmentation [18].

Semantic segmentation also plays a crucial role in augmented reality scenarios. It allows the users the possibility to have more detailed and in-depth experience in the scene , especially in medical environments. It has shown promising use cases in the medical field such as real-time identification of anatomical structures in surgeries.

Another field that's been highly affected by semantic segmentation is video surveillance. It has been heavily used in Unmanned Aerial Vehicles (UAVs), improving the way we monitor and manage environments. Semantic segmentation allows the detailed detection and tracking of objects across a wide range of scenarios by providing a precise pixel-level understanding of video data. Despite its numerous advantages in the field, the scarcity of annotated ground truth data necessary for training robust models makes its application to video surveillance a challenge for researchers. In order to overcome the problem of weakly supervised semantic segmentation, deep learning-based architectures [19] emerge as the technology advances.

Having a good understanding of the surroundings in autonomous driving has always become a crucial task because of the importance of navigation and decision-making processes of self-driving cars. It offers a comprehensive understanding of the environment, helping obstacle detection, route planning, and traffic management by classifying each pixel of an image into categories such as roads, vehicles, and pedestrians. With existing models and the newly emerging ones, the performance of semantic segmentation models is constantly evaluated and being improved [20].

2.3 Multimodality

Complex real-world scenes pose a great challenge that often cannot be mitigated by any single source of data or segmentation architecture when it comes to achieving highly accurate semantic segmentation. Real-world applications usually involve dynamic environments where the volume and the complexity of data can exhaust traditional unimodal strategies [21]. Various data sources have their own unique explanation of the same scenery, as shown in Figure 2.3, aiding overall model accuracy in challenging adverse conditions [21, 22]. Having their own strengths and drawbacks for numerous reasons, using them in a multimodal system leverages their complementary strengths and enriches the perception and understanding of a scene. At its core, multimodality offers integration and synthesis of the information from various sensors and detectors that provide a unique perspective of the same scene. This way, the system captures a more detailed and in-depth representation of the environment without compromising its complexity.

Nonetheless, integration of the multimodal data comes with its own challenges. Each modality presents uncertainty in the fusion process through its own noise levels and characteristics. Therefore, in many multimodal architectures, reducing redundant data while utilizing it in error correction and cross-validation processes becomes highly important. Corresponding strategies are being developed to tackle these challenges while extracting and benefiting from valuable insights from the training data. One of the strategies that has been revolutionizing the field is deep learning, which has come onto the stage for multimodal fusion in semantic segmentation of complex environments. Deep learning models offer vast computational power and the ability to learn from massive datasets [23, 24]. In many multimodal fusion processes, the dataset required for training grows larger, making it difficult for traditional learning strategies. However, deep learning models are able to handle such datasets while taking multiple inputs [25] and understand the given scenery with a level of perceptual depth close to human understanding. Furthermore, deep learning architectures allow for flexible and effective fusion strategies that can precisely gauge the contribution of each modality and optimize the model.

Existing driving scenery datasets often feature a single type of attribute while missing other complementary data [27, 28, 29]. Therefore, designing deep learning models that can harness the features offered by various sensory modalities becomes the primary object in neural network design. In multimodal learning, different data types can be obtained from a range of devices such as 3D LiDARs, depth sensors, RGB cameras, thermal cameras, and various other sensors, providing spatial and contextual data surpassing the richness of unimodal scenarios. These modalities can be utilized individually or together to mitigate the innate uncertainty in interpreting the scene at hand to achieve robust and accurate perception.

For instance, LiDAR sensors are highly precise in the construction of 3D maps of the surroundings. Therefore, this attribute allows them to be beneficial for obstacle detection. When the ambient light conditions are not suitable for traditional visual cameras, LiDAR sensors come in handy as they work reliably under low-light conditions. By nature, LiDAR sensors provide high-resolution point clouds [30] and this way, they offer a much better

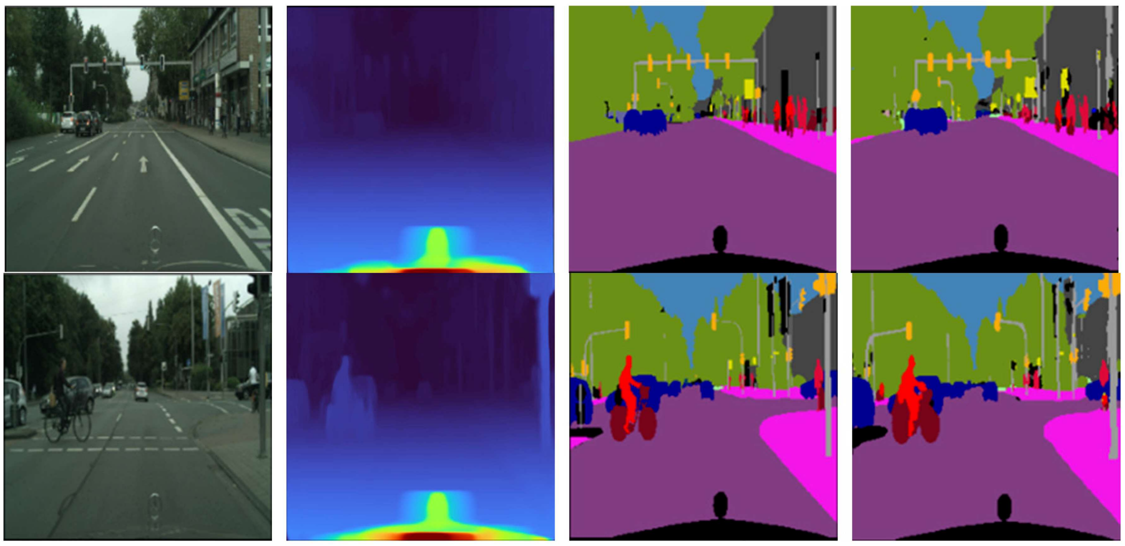


Figure 2.3: (a) RGB images from Cityscapes dataset; (b) Predicted depth map; (c) ground truth semantic segmentation images; and (d) prediction. Source: [26].

understanding of fine features in the scanned area. However, the accuracy precision decreases under challenging weather conditions like rain or fog for these types of sensors. Another challenge to the usage of LiDAR sensors is their high production cost. These systems are often much more expensive compared to other traditional modalities, preventing widespread adoption. However, it's predicted to be cheaper and more available for autonomous vehicles [31]. Their ability to provide high-resolution data can be beneficial in environmental understanding, but they create a heavy load on overall data volume and increase processing power. Lastly, LiDAR sensors are not suitable for all machine learning tasks as they do not provide color information.

In tasks where color information plays an important role, RGB cameras have long been reliable and well-established sensor types. This fact makes an abundance of information from research and development in the field available in scenery interpretation. This type of sensor can be useful in object recognition [32] and tracking tasks. Moreover, modern RGB cameras can yield high-resolution images with the advancement of current technology [33]. In comparison with LiDAR sensors, they are often less expensive to produce, making them more accessible in the autonomous driving field. Contrary to their benefits, RGB cameras lack in some ways, such as dependency on light. By their nature, they require adequate lighting to capture clear images, and this limits their usage in low-light conditions. Furthermore, RGB cameras do not provide in-depth information on the environment, losing an important aspect of data in model training.

Thermal cameras have been a crucial tool for nighttime and low-light condition sensing applications [34]. They can detect heat signatures even in absolute darkness, therefore making them useful in navigation and surveillance. Their ability to perceive through obscuring factors such as smoke, dust, and thin foliage makes them invaluable for operations where a clear weather condition is not present. Moreover, thermal cameras have become incredibly useful for living being detection thanks to their ability to discern the contrast provided by body heat against cooler backgrounds. This capability in itself makes it efficient for wildlife research [35] and security scenarios. Conversely, they fall behind their counterparts in terms of rendering quality and overall output resolution. Similar to LiDAR sensors, they lack color sensitivity and do not provide enough color information.

Depth cameras provide another layer of understanding by delivering important spatial data. This feature allows accurate 3D mapping and enhanced object detection in autonomous vehicles and robotics. They are often used as complementary information sources [36] working in tandem with other sensors as they can provide valuable data across different lighting conditions. Moreover, their real-time processing capabilities make them especially suited for dynamic settings where immediate spatial understanding is paramount. However, like its counterparts, the limitations of depth cameras are noteworthy. Their less extensive range compared to LiDARs, sensitivity to the other infrared-emitting sources in the detecting range, and their less detailed image quality [37] compared to high-resolution RGB cameras prevent them from being the standalone information source in machine learning challenges.

The cumulative evidence from previously conducted research [38, 39, 40, 41] emphasizes that various sensory inputs used together are far more potent than the singular data stream in deep learning model efficiency and accuracy. The integration of multimodal data allows the deep learning model to analyze and understand the scenes in a multidimensional way that single modality systems might not fully capture. The utilization of RGB cameras for capturing visual information about the environment with the complementary usage of depth cameras for gathering more extensive spatial data provides a layered comprehension of the operational environment. This thesis explores a set of encoder-decoder architectures where data is provided in both RGB and Depth formats. This experimental study thrives to illuminate the effects of multimodality in practical autonomous driving scenarios.

3 Dataset

3.1 Dataset Overview: Cityscapes

Cityscapes[2] dataset is one of the most commonly used benchmark suites for evaluating the semantic understanding of urban scenes. It offers 5000 images with high-quality pixel-level annotations and an additional 20,000 with coarse annotations. Its richness for annotations, large size, and scene variability puts it among the top choices for autonomous driving tasks. This dataset has been like a cure for the facilitation of the development and evaluation of algorithms that require a solid understanding of urban environments. Moreover, the Cityscapes dataset stands out with its extensive collection of annotated images that were captured under diverse weather conditions and varying lighting conditions. This feature allows users to have more realistic data for deep learning model training. Another important aspect of this dataset is that it has a diverse set of annotations that cover a wide range of urban scenarios. This diversity helps models to be more robust and less prone to overfitting and have a better overall generalization. The captured imagery within the dataset is from an automobile in transit across seasons to increase the diversity. The data is gathered predominantly across German cities and other areas in their close proximity [2]. Initially, this study employed the standard RGB image format as supplied by the dataset's creators. To further explore the multimodal approach, these RGB images were subsequently transformed into the HHA format through a Python script designed for this conversion task.

3.1.1 Data Recording and Variance

The dataset captures across different seasons and covers a range of city parts—from crowded metropolises to small towns as seen in Figure 3.1

This temporal and spatial diversity is highly important for training models that are robust and generalizable for real-world conditions. The geographic and temporal data distribution is as follows:

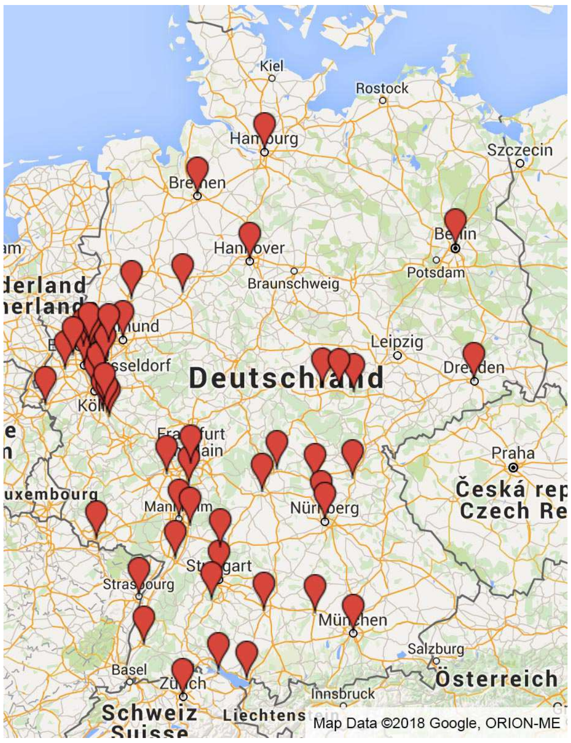


Figure 3.1: Locations of images captured on map. Source: [42].

- **Geographic Spread:** Data are captured from cities located in the western, central, and eastern regions.
- **City Size Variance:** Includes large, medium, and small urban centers.
- **Seasonal Coverage:** Encompasses beginning, middle, and end-of-year scenarios.
- **City-Level Split:** Each city is contained within a single data split to maintain split consistency.

3.1.2 Class Definitions and Annotations

Cityscapes defines 30 visual classes that are further organized into eight categories. These categories range from structural and stationary elements of the urban landscape to moving objects such as vehicles and pedestrians, as well as more miscellaneous elements like the sky and various terrains. A detailed description of the corresponding classes are as follows:

1. Construction: bridge, building†, fence†, guard rail, tunnel, wall†
2. Flat: road†, sidewalk†, parking, rail track
3. Human: person†, rider†
4. Nature: vegetation†, terrain†

5. Object: pole†, pole group, traffic light†, traffic sign†
6. Sky†
7. Vehicle: bicycle†, bus†, car†, caravan, motorbike†, trailer, train†, truck†
8. Void: dynamic, ground, static

First, the unaccounted classes are classified as void. Then, the less frequent classes are disregarded in the evaluation process and narrowed down to the primary 19 classes, i.e., the ones indicated by a dagger symbol. The dataset split was made as follows:

- 2975 images for training
- 500 for validation
- 1525 designated for testing

all uniformly set to a resolution of 1024 × 2048 pixels.

3.1.3 Image Representation

The standard RGB format, which is the format provided by the dataset creators, is used during the initial stages of this experimental work. A conversion was necessary from the conventional RGB to the HHA representation to create a multimodal algorithm. This conversion is manually performed utilizing a custom Python script. This script first computes a depth map, shown in Figure 3.2, from the disparity information available in the Cityscapes dataset. A disparity map quantifies the apparent pixel shift between two images of the same scene captured from different viewpoints resembling human stereoscopic vision.

The computation of a disparity map from stereo images involves correlating each pixel from the left image with its match on the right image. Then, the relative distance is calculated. If the geometric parameters of the stereo camera setup, such as baseline and focal length, are known, the disparity map can be transformed into a precise depth map.

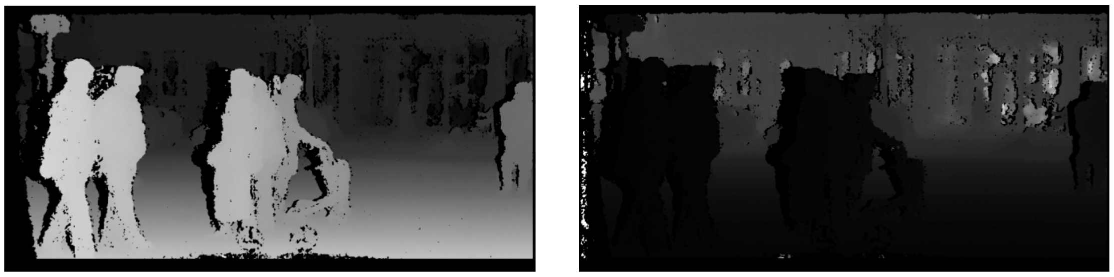


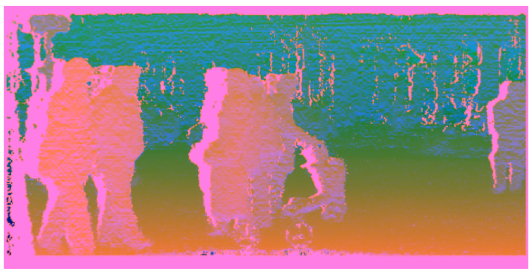
Figure 3.2: Aachen disparity example and the corresponding depth map.

Creating the HHA representation from the depth map involves encoding depth information into three channels at every pixel:

1. **Horizontal Disparity (H):** Corresponds to the horizontal shift and is indicative of the object's distance from the camera.
2. **Height Above Ground (H):** Indicates the vertical position relative to the ground plane.
3. **Angle with Gravity (A):** Reflects the orientation of the surface normals in relation to the direction of gravity.

Then, these channels are linearly scaled to the 0-255 range. Thus, a three-dimensional representation is created from single-channel depth information as illustrated in Figure 3.3.

(a) RGB image


(b) HHA image

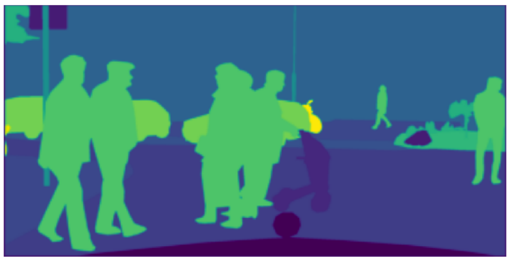

(c) Segmentation mask

Figure 3.3: Result of the conversion procedure

4

Models

Semantic segmentation aims to solve a crucial challenge in the field of computer vision. This task helps models to accurately capture the semantics of image data and understand the complex relationships between various elements within by assigning a semantic label to every pixel in an image. The design choice of encoder-decoder architectures can be seen in many modern semantic segmentation models. This type of architecture was used in autoencoders, which was first introduced in 1986 [43] to learn efficient representations of input data through reconstruction with minimal error.

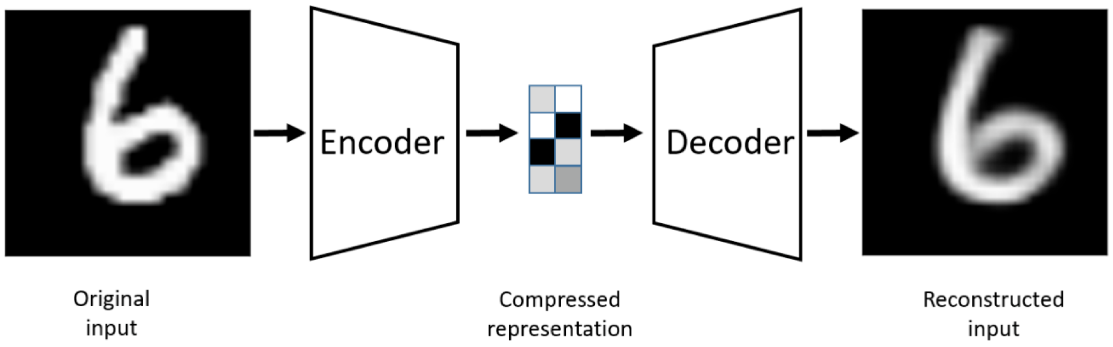


Figure 4.1: An autoencoder example. The input image is encoded to a compressed representation and then decoded. Source: [3].

This fundamental concept of autoencoders has led to the development of more sophisticated encoder-decoder architectures. On top of being the main architecture used for feature extraction, encoder-decoder architectures are now treated as modular components that perform distinct operations tailored to specific applications. Furthermore, encoder-decoder models are now employed in modern machine learning across various domains thanks to their versatile usage in widespread applications, including natural language processing [44], image processing [45], and speech recognition[46].

4.1 Encoder: Mobilenet-v2

Achieving an optimal balance between the limited computational resources and the accuracy of models is crucial, especially on mobile devices. In order to create a model tailored for computer vision applications in resource-constrained environments, the MobileNets series was introduced by Google [4]. The initial model was released in the spring of 2017 [4], then subsequent models with enhanced architectures were released, the most recent one being MobileNetV3 [47]. These models are designed to be lightweight yet effective for a broad range of computer vision tasks, including but not limited to classification, object detection, and semantic segmentation [5].

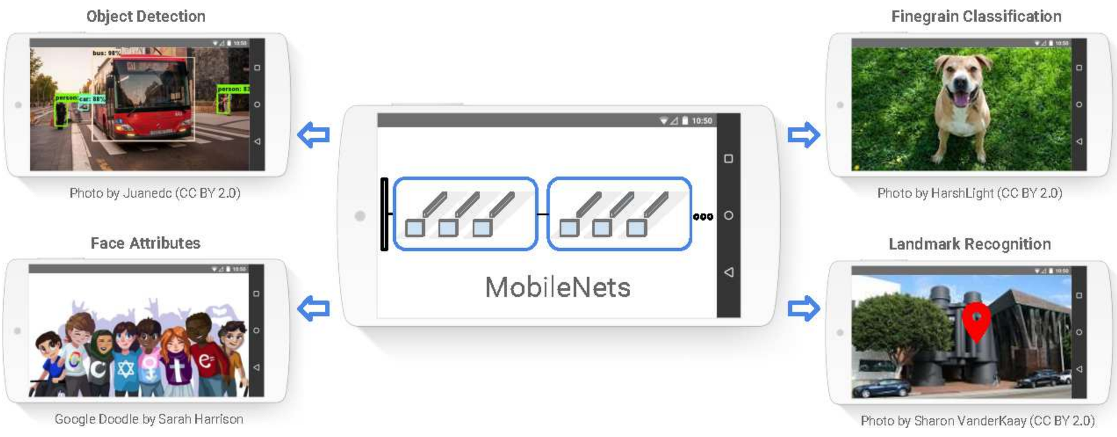

Object Detection
Photo by Juanedc (CC BY 2.0)
Face Attributes
Google Doodle by Sarah Harrison
MobileNets
Finegrain Classification
Photo by HarshLight (CC BY 2.0)
Landmark Recognition
Photo by Sharon VanderKaay (CC BY 2.0)

Figure 4.2: Various recognition tasks of MobileNetV1 architecture. Source: [4].

The initial model MobileNetV1 made a great contribution to minimizing the model size and computational complexity. The reduction in complexity requires a number of parameters and computational operations, such as multiplications and additions. Above all, MobileNetV1 stands out for its diversity across different tasks, as shown in Figure 4.2. For this, the model employs a method that significantly reduces the computational burden. This method is called depthwise separable convolutions. It breaks down the convolutional process into two separate layers, depthwise and pointwise convolution, governed by equations 4.1 and 4.2. By decomposing these steps, filtering and combining steps are done separately and lower the computational requirements drastically.

$$Y_{i,j,m} = \sum_{k=1}^{D} \sum_{l=1}^{K} W_{l,k,m} \cdot X_{i+l,j+k,k} \tag{4.1}$$

where:
$Y_{i,j,m}$ is the output feature map at position (i, j) in channel m,
$W_{l,k,m}$ represents the depthwise filter weights,

$X_{i+l,j+k,k}$ denotes the input feature map at position $(i + l, j + k)$ in channel k.

$$Z_{i,j,p} = \sum_{m=1}^{M} V_{m,p} \cdot Y_{i,j,m} \tag{4.2}$$

where:
$Z_{i,j,p}$ is the final output feature map at position (i, j) in channel p,
$V_{m,p}$ denotes the pointwise filter weights.

Then, a refined version of MobileNetV2 was introduced to improve the performance across various computer vision tasks. This architecture employs linear bottlenecks and shortcut connections as shown in Figure 4.3. Linear bottlenecks serve two purposes: they reduce the dimensionality of data passing through while conserving essential information, and they also act as a compression mechanism between the network's layers. The other update introduced in this model is shortcut connections inspired by residual networks. This addition in the architecture creates a direct pathway for the gradient flow during backpropagation and allows faster convergence and improved model accuracy. They also address the vanishing gradient problem and facilitate the training of deeper network architectures.

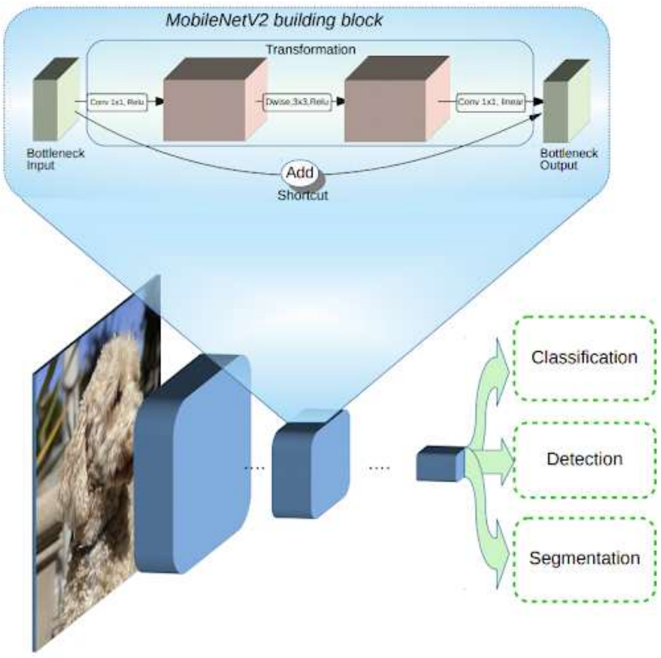


Figure 4.3: Overview of MobileNetV2 architecture. Source: [5].

In MobileNetV2, the input and output of the bottleneck layers are of higher dimensions compared to the intermediate layers. This architecture design allows the network to preserve high-level feature representation and reduces computational resources, which gives it the capacity to transform lower-level features into high-level descriptors efficiently, like image categories.

In summary, MobileNetV2 was preferred over MobileNetV3 thanks to having a crucial balance for real-world applications in the autonomous driving domain given that MobileNetV3 has a larger model size. MobileNetV2 model, which is used in the encoder part in our network, achieves the best performance balancing the trade-offs between accuracy, speed, and computational resource usage and stands out for mobile device applications.

4.2 Decoder: DeeplabV3

The DeepLab architecture, which was developed by Chen et al.[7], has significantly influenced the field of semantic image segmentation through several improved iterations [6]. The first architecture was introduced as DeepLabv1. The model combined Deep Convolutional Neural Networks (DCNNs) with fully connected Conditional Random Fields (CRFs) to tackle inherent challenges in semantic segmentation. It uses a technique called Atrous Convolution or Dilated Convolution, shown in Figure 4.4, to extend the receptive field of convolutional filters without increasing the computational cost. This technique adds spaces between the pixels of the input image for "skipping" over them and expands the range of influence of the filter without introducing additional parameters.

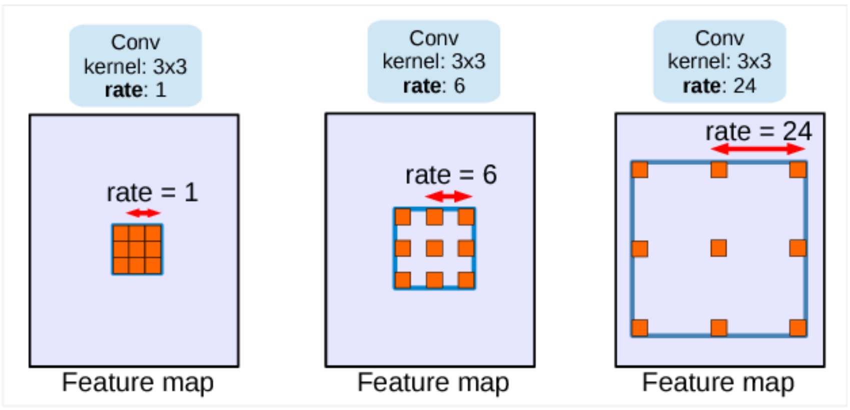


Figure 4.4: Atrous convolution with kernel size 3× 3 and different rates. Source: [6].

In atrous convolution, the spacing between the kernel points of the convolutional filter is dictated by a critical parameter, the dilation rate. This parameter modifies the standard convolution operation, $\mathbf{y}(i) = \sum_k \mathbf{x}(i + r \cdot k)\mathbf{w}(k)$, where $\mathbf{x}$ is the input image, $\mathbf{w}$ is the filter weights, r is the dilation rate, and $\mathbf{y}$ is the output feature map. This way, the model is able to aggregate multi-scale contextual information to capture fine details and maintain spatial resolution in segmentation tasks. After the convolution, fully connected CRFs were employed to enhance the precision of feature localization and reduce segmentation map noise.

In the second model, DeepLabv2, Atrous Spatial Pyramid Pooling (ASPP) aggregates the output of atrous convolutions at different rates to a given input feature map. This approach, illustrated in Figure 4.5, processes the input image at multiple scales concurrently and encodes the multi-scale information to enhance the ability to recognize objects across various sizes of the model.

The model employed in our work, DeepLabv3, was introduced to optimize previous architectures further. This model removes the post-network CRF processing for end-to-end learning. Moreover, it introduces an en-

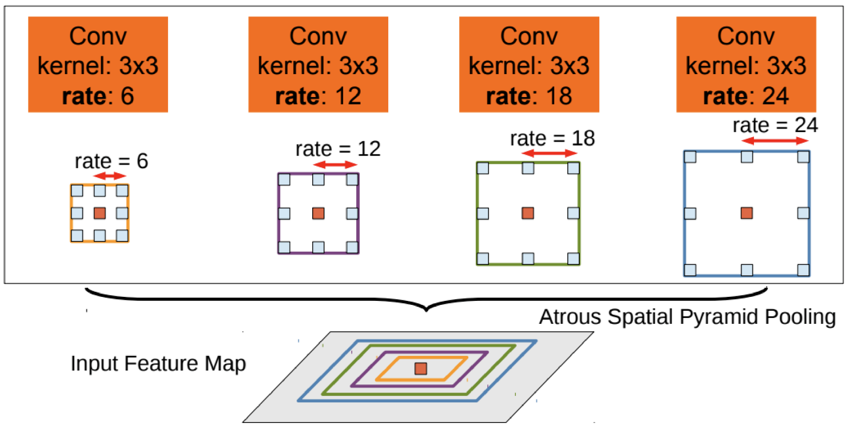


Figure 4.5: Atrous Spatial Pyramid Pooling (ASPP). Source: [7].

hanced version of ASPP, which utilizes atrous convolutions in parallel to expand the model's receptive field [6]. Therefore, it allows for detailed feature extraction while maintaining the spatial dimensions of the input. Moreover, the aspect that made the DeepLabv3 model capture a wide spectrum of image features was the combination of global average pooling followed by feature concatenation and atrous convolution.

5 Experimental Setup

5.1 Baseline Model: Only RGB

The Cityscapes dataset, described in Chapter 3, covers a diverse set of urban scenes from various European cities. The training split organizes images into directories named after the cities they represent, such as Aachen, Stuttgart, and Zurich, each of which contains hundreds of RGB images. The baseline model is configured to process the Cityscapes dataset using only standard RGB color information following the methodology established in [2], which is a Python implementation for supervised learning. This section dives into the training and validation procedures, explains the dataset's composition, and illuminates the data preprocessing steps involved.

In this experimental work, virtual clients are generated for each city. Each of these clients is assigned a unique identifier related to its origin folder. This way is chosen to ensure that each client has images assigned to it from only one city in the dataset. The logic of splitting the dataset into subsets for different clients is illustrated in Figure 4.1 with the Aachen example, where the dataset's 174 total image files are evenly distributed across eight clients.

5.1.1 Image Preprocessing

Images undergo several preprocessing steps to enhance the model's generalization capabilities before being fed to the model for training. The first transformation is *RandomScale*. As the name suggests, it randomly varies the scale of the images within a specified range as an initial step in data augmentation. This is followed by *Random-Crop*. This transformation extracts some portions of the original image while maintaining a specific output size of $[512, 1024]$. The final preprocessing steps include *ToTensor* transformation and normalization. In the final step of preprocessing, we utilize the mean and standard deviation values specified in the DeeplabV3 architecture documentation [6].

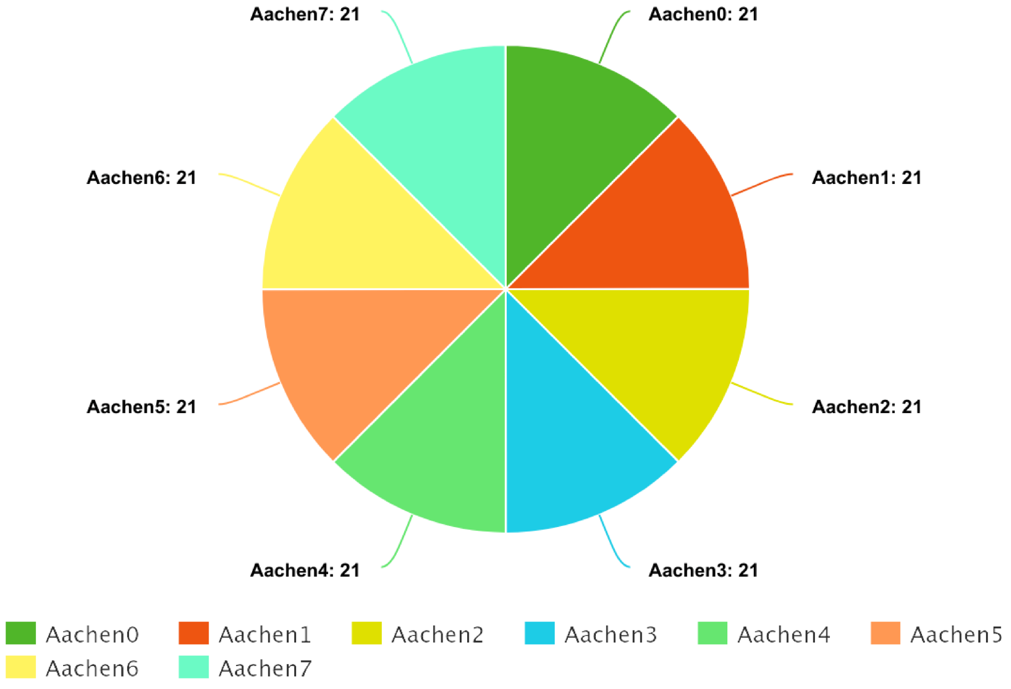


Figure 5.1: Pie chart representation of distributed images across eight clients.

5.1.2 Experiment Configuration

Training is conducted over 5000 rounds. Five clients are randomly selected in each round and trained for a single epoch. The FedAvg algorithm updates the model at the end of each round. Key parameters for the training procedure are outlined as follows:

Standard Training Parameters

- **Optimizer**: Stochastic Gradient Descent (SGD), a widely used optimization algorithm that facilitates the convergence of the model by updating the weights based on the gradient of the loss function.
- **Learning Rate**: Set to 0.05, determining the step size at each iteration while moving toward a minimum of the loss function.
- **Momentum**: A value of 0.9 is used to accelerate the SGD in the relevant direction and dampen oscillations.
- **Weight Decay**: Set to 0, indicating no regularization term is added to the loss function.
- **Batch Size**: 4, determining the number of training examples used in one iteration.
- **Test Batch Size**: 2, specifying the number of examples used in a batch during the evaluation phase.

Federated Learning Parameters

These parameters highlight the decentralized nature of the training process and the strategy for aggregating local updates to improve the global model:

- **Number of Rounds**: 5000 indicates the total number of federated learning rounds to be executed. It allows the model to improve iteratively through local client updates.
- **Clients Per Round**: 5, specifying the number of clients randomly selected in each round to participate in the training, reflecting the federated learning's distributed approach.
- **Number of Epochs**: 1, ensuring that each selected client trains the local model for only one epoch per round to prevent overfitting on local data.
- **Evaluation Interval**: Every 50 rounds, the model's performance is evaluated to monitor progress and adjust strategies as necessary.
- **Test Interval**: Set at every ten rounds, conducting regular testing of the model against the validation set to ensure robustness and generalization capability.
- **Federated Algorithm**: FedAvg, a common federated learning algorithm, manages the aggregation of locally-computed updates to enhance the global model.

A specific evaluation method is employed based on the number of completed training rounds during the training process. When the number of executed rounds is a multiple of the input given parameter ”evaluation interval”, a new client is created. This unique client gathers all training examples into a singular, comprehensive dataset. Moreover, the preprocessing steps are deliberately streamlined to only include *ToTensor* transformation and normalization for this evaluation client. Therefore, images retain their original dimensions of [1024, 2048]. This configuration is used to assess the model's capacity for generalization across diverse urban scenes.

Similarly, a unique test client is instantiated depending on the the ”test interval” parameter following the same logic of evaluation method. The input images for the test client is drawn from evaluation subset of the Cityscapes dataset. This way, the model's performance across standardized urban environments is tested fairly.

Notably, this subset includes:

- Frankfurt, encompassing 267 examples,
- Lindau, with 59 examples, and
- Munster, offering 174 examples.

All images utilized for testing purposes maintain their original resolution of [1024, 2048] to preserve the integrity of the evaluation.

On the server side, a singular model architecture, the encoder-decoder configuration discussed in Chapter 5, is utilized. This model employs a unified aggregation process that sets a foundation for the baseline system for subsequent experiments. This baseline creates a foundational understanding of supervised(oracle) learning with RGB data within urban environments. This section sets the stage for the following experiments that explore multimodal inputs and advanced model architectures.

5.2 Model 2: RGB||D, 2 Encoders & 2 Decoders

In this section, a significant modification is made in the original code [1] and implemented. The concept allows for the flexibility for all potential clients to represent their datasets in either RGB or HHA format. This process is coded in the initial segment of the algorithm in the client selection process, and these modifications are propagated across the entire codebase. Prior to the beginning of the learning procedure, all clients are initialized with a uniform probability. Therefore, either the RGB or HHA data format is selected for each client and then assigned to them.

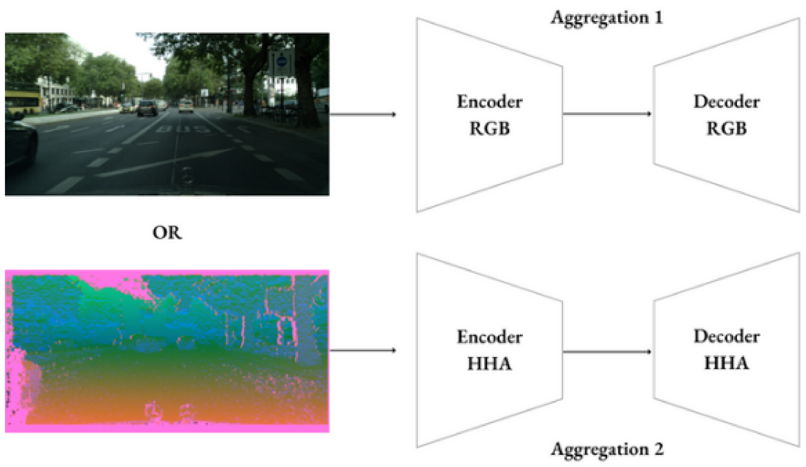


Figure 5.2: RGB || D models on the server side.

The single-client setup discussed in Section 5.1 is introduced for the evaluation phase. However, there will now be one evaluation of the RGB client and one dedicated to the HHA case. A similar strategy is used for the testing phase as well. There will be two test clients, one for each format. The preprocessing steps are the same as those of the baseline case except for the normalization procedure. The mean and standard deviation values are kept identical to the baseline case within the RGB images. For the HHA case, şt requires some computation using a simple Python script iterating over all available examples. To summarize, the following sets will be utilized:

- MeanRGB: (0.485, 0.456, 0.406)
- StdRGB: (0.229, 0.224, 0.225)
- MeanHHA: (0.484, 0.498, 0.471)
- StdHHA: (0.241, 0.063, 0.348)

Two distinct models, as shown in Figure 5.2, are established on the server side. Both models possess the same structure as the baseline experiment's model, with one dedicated to the RGB data and the other to the HHA data. Thus, two distinct update phases for each model are executed. The total number of rounds remains set at 5000. Similarly, a subset of 5 randomly selected clients for a single epoch are selected for training. The additional parameters utilized align with those of the baseline case.

5.3 Model 3: RGB||D, 2 Encoders & 1 Decoder

In the second part of the research, a new model architecture is proposed to streamline the processing of multi-modal data. This approach introduces a single decoder structure on the server side to accommodate two distinct encoders. There are two encoders for each input modality (RGB and Depth), as depicted in Figure 5.3. The decoder component remains consistent with the baseline model and generates a unified output from the divergent inputs.

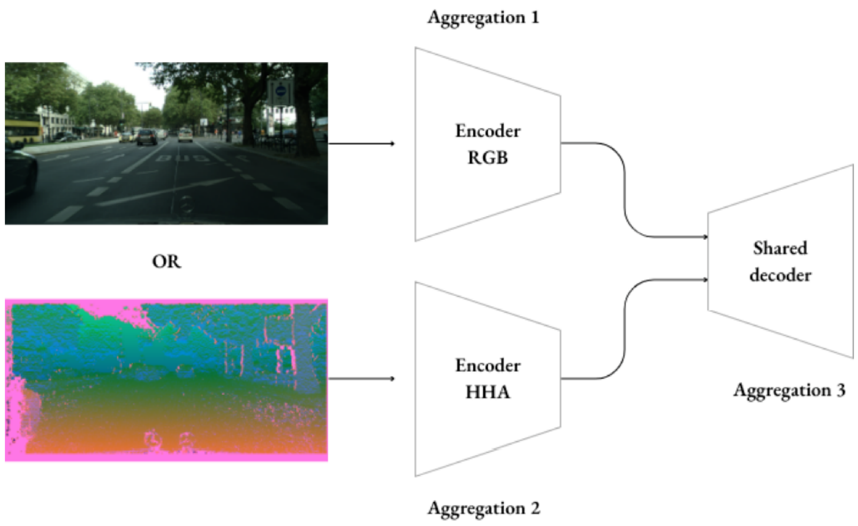


Figure 5.3: Second model architecture with dual encoders for RGB and Depth (HHA) data on the server side.

The client creation process and data preprocessing remain the same as the previous experiments to maintain a uniform approach to data handling and client initialization. Clients are designated a specific modality throughout the learning process, and the model dynamically selects the corresponding encoder for data processing. Moreover, this design approach allows for modality-specific feature extraction.

Unlike prior models, the aggregation process includes modality-specific considerations in the model update phase. Two separate aggregations occur on the encoder side, considering both types of data. On the decoder side, different input representations are merged through a singular aggregation process to generate a cohesive output.

5.4 Model 4: RGB&D, 2 Encoders & 1 Decoder

In the fourth experimental setup, each client is assigned a dataset that features both RGB and HHA (Depth) representations for every frame. This way, the processing of both data types concurrently within a singular model framework was achieved. For instance, if we consider the example distributions for the Aachen clients, we see a balanced presence of RGB and HHA data:

- Aacheno: 21 RGB + 21 HHA examples

- Aachen1: 21 RGB + 21 HHA examples
- Aachen2: 21 RGB + 21 HHA examples
- ⋮
- Aachen7: 21 RGB + 21 HHA examples

There are two primary adaptations to the training protocol in this setup:

1. **Batch Size Adjustment:** The batch size for training is considered halved as each batch concurrently processes RGB and HHA data.
2. **Aggregation Procedure Modification:** The number of samples that affect the weighted sum in the federated averaging is adjusted by a factor of two to account for the dual representation of the dataset.

The model's architecture on the server side is shown in Figure 5.4; separate encoders for RGB and HHA inputs are integrated and then merged before being fed into a shared decoder.

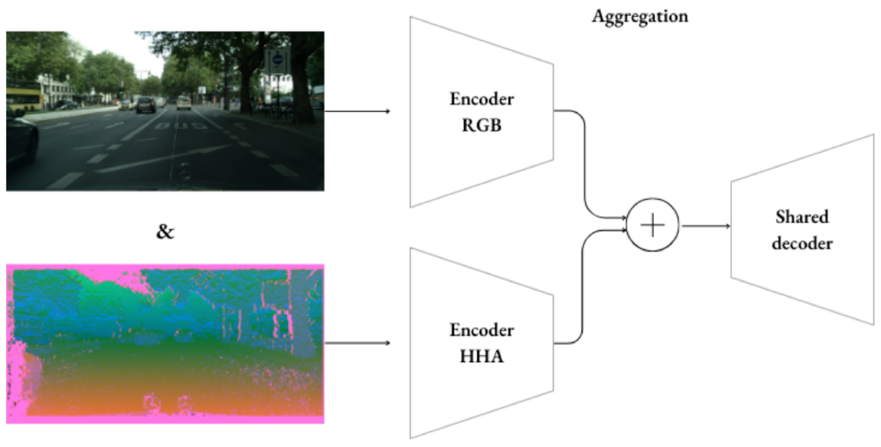


Figure 5.4: The RGB & D integrated model on the server side, which processes both RGB and Depth data simultaneously.

The model utilizes both the RGB and HHA-encoded data at each training step. The outputs from the two encoders are averaged and passed on to the decoder. The aim of this type of aggregation is to enrich the feature representation and, in turn, enhance the predictive quality of the segmentation map. Moreover, a single aggregation procedure is performed once since there is just a single model to update on the server side. The experimental parameters remain consistent with previous experiments to ensure methodological consistency and comparability of results.

5.5 Model 5: Only HHA(Depth)

This experimental setup involves processing the Cityscapes dataset so that the focus is on depth information. The model on the server is virtually identical to the baseline model utilizing only RGB data. The primary difference is the input data format. The model uses the HHA (Depth) version of images rather than employing the standard RGB format. The HHA (Depth) version of the dataset is generated from RGB images in the Cityscapes dataset using a Python script. This model aims to analyze the effectiveness of depth information when applied to the task of semantic image segmentation.

5.5.1 Depth Dataset Processing

The processing of the HHA data mirrors the established pipeline detailed in the baseline model section. Just as with RGB images, preprocessing steps like *RandomScale* and *RandomCrop* are applied to the depth images for data augmentation. The depth data also undergoes the *ToTensor* conversion and normalization steps. The normalization parameters for the depth images differ from those of the RGB data and are computed using the mean and standard deviation of the depth images in the Cityscapes dataset.

5.5.2 Depth Model Configuration

Training for the depth-only model undergoes a similar training process to the RGB baseline model. 5 clients are selected at random for training in each round over the course of 5000 rounds. The FedAvg algorithm, crucial to federated learning, updates the model after every round considering the depth data.

The depth-only model employs the same standard training parameters and federated learning parameters as the baseline model to have consistency in the evaluation of the models.

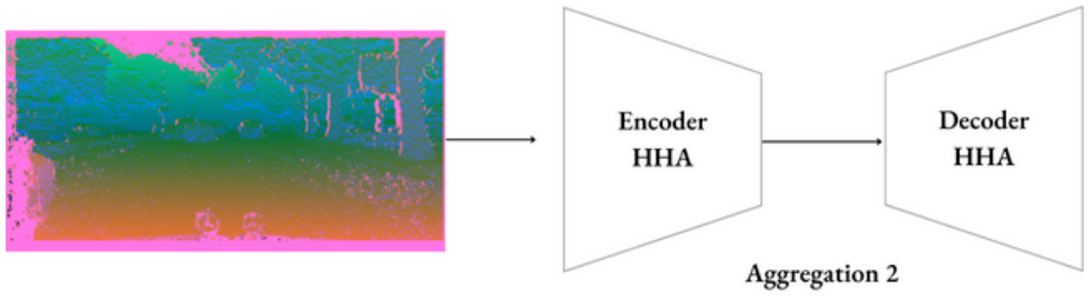


Figure 5.5: Architecture of the depth-only model with preprocessing and training parameters tailored to depth data.

The effectiveness of the depth-only model is an essential part of this research as it provides insights into the capabilities of depth data for autonomous driving applications. This experiment aims to understand the the individual contributions of this modality to semantic segmentation problems through the isolation of depth data.

6 Results

6.1 Metrics

6.1.1 Cross-Entropy Loss

Cross-Entropy loss function [48] is extensively utilized for multi-class classification problems and proves to be efficient for image segmentation tasks. It measures the difference between the predicted probability distributions of the model and the actual distribution presented by the one-hot encoded labels during model training.

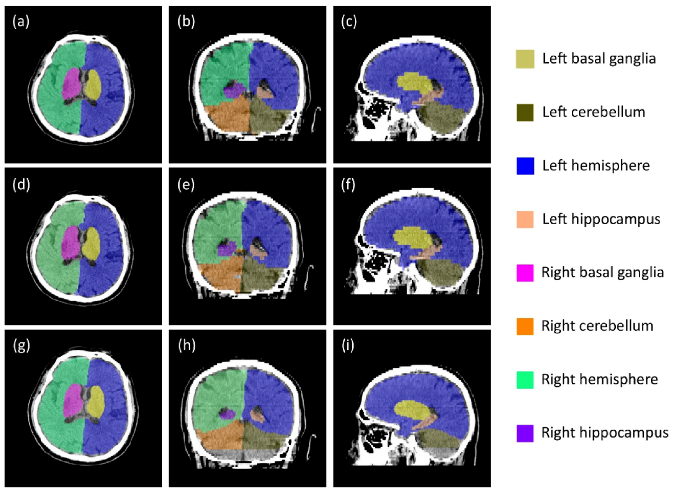


Figure 6.1: An illustrative example of a segmentation mask.Source: [8].

For a set of M images, the Cross-Entropy loss (Loss_{CE}) is formalized as follows:

$$\text{Loss}_{CE} = -\sum_{n=1}^{M} (\text{one\_hot\_label}[n] \cdot \log(\text{output\_softmax}[n])) \tag{6.1}$$

The model's output is typically represented by a tensor with dimensions [batch size, number of classes, height, width] and with entries ranging from $[0, +\infty)$ due to a ReLU activation function in the last layer. Then, a softmax layer that maps the output values into the probability space $[0, 1]$ is employed. Correspondingly, the labels are structured as [batch size, height, width], where each pixel's value corresponds to the class index as illustrated in Figure 6.1. The softmax-transformed model output and the one-hot encoded labels, as shown in Figure 6.2, are then provided as inputs to the Cross-Entropy loss function to evaluate the model's performance.

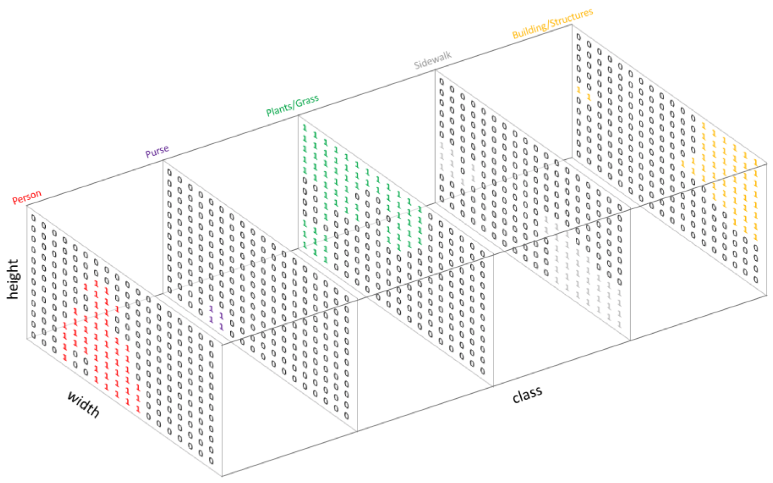


Figure 6.2: Example of one-hot encoding for label masks.Source: [8].

Hard Negative Mining (HNM) is employed within the loss reduction strategy to further optimize the training. HNM, giving priority to those with higher errors, begins by ordering all the pixels according to their associated loss values. Then, the final epoch loss is computed as the mean of high-loss pixels, ensuring that the model focuses on the most challenging examples to improve segmentation accuracy.

6.1.2 L2 Loss Function

L2, least squares error or mean squared error, loss is another common loss function used in regression problems and models where the aim is to predict continuous outputs. It measures the squared differences between the predicted values and the actual values and provides a metric of the performance of a model in terms of error magnitude.

Given a set of predictions $\hat{y}_i$ from the model and the corresponding true values y_i, the L2 loss for a single prediction can be expressed as:

$$L2_i = (\hat{y}_i - y_i)^2 \tag{6.2}$$

For a batch of N predictions, the L2 loss is computed as the average of the squared differences, yielding the Mean Squared Error (MSE):

$$L2_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{6.3}$$

The output of this provides a measure of how well the model is performing across the entire batch of data. Morever, the better the model performs, the lower its score is.

The L2 loss is particularly sensitive to outliers since the errors are squared, and the square of large differences is amplified.

6.1.3 Mean Intersection over Union (mIoU)

One of the most common metrics used to assess the performance of semantic segmentation models is the Intersection over Union (IoU) metric, also known as the Jaccard index. This evaluation metric provides information about the accuracy of the model by calculating the overlap between the predicted segmentation and the ground truth.

The IoU is a ratio ranging between 0 and 1. It is the ratio of the common pixels (intersection) to the total pixels present (union) in both predicted and ground truth masks. For each class, the IoU is computed as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{6.4}$$

where the Area of Overlap and the Area of Union between the predicted segmentation and the ground truth are given by:

$$\text{Area of Overlap} = \text{TP} = \text{Ground Truth} \cap \text{Prediction} \tag{6.5}$$

$$\text{Area of Union} = \text{TP} + \text{FP} + \text{FN} = \text{Ground Truth} \cup \text{Prediction} \tag{6.6}$$

Therefore,

$$IoU = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{6.7}$$

where:

- TP (True Positives): Number of pixels correctly classified for a specific class.
- FP (False Positives): Number of pixels incorrectly labeled as belonging to a class.
- FN (False Negatives): Number of pixels belonging to a class that are missed by the model.

TP, FP, and FN counts can be expressed in terms of pixel-wise comparisons:

$$\text{TP} = \sum_{i,j}[\text{GT}_{ij} \wedge \text{Pred}_{ij}] \tag{6.8}$$

$$\text{FP} = \sum_{i,j} [\neg \text{GT}_{ij} \wedge \text{Pred}_{ij}] \tag{6.9}$$

$$\text{FN} = \sum_{i,j} [\text{GT}_{ij} \wedge \neg \text{Pred}_{ij}] \tag{6.10}$$

The notation $[\cdot]$ represents the Iverson bracket, which evaluates to 1 if the condition inside is true and 0 otherwise; $\wedge$ denotes logical AND; $\neg$ denotes logical NOT; GT_{ij} and Pred_{ij} denote the ground truth and prediction at pixel coordinates (i, j), respectively.

Mean Intersection over Union (mIoU) is an extension of the IoU metric that provides an average measure of performance across multiple classes:

$$\text{mIoU} = \frac{1}{C} \sum_{c=1}^{C} \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c + \text{FN}_c} \tag{6.11}$$

where C is the number of classes, and TP_c, FP_c, and FN_c are the True Positives, False Positives, and False Negatives for class c, respectively.

An example of this can be seen in Figure 6.3, which shows the ground truth and predicted masks, and Figure 6.4 showing the intersection and union regions used for the IoU calculation.

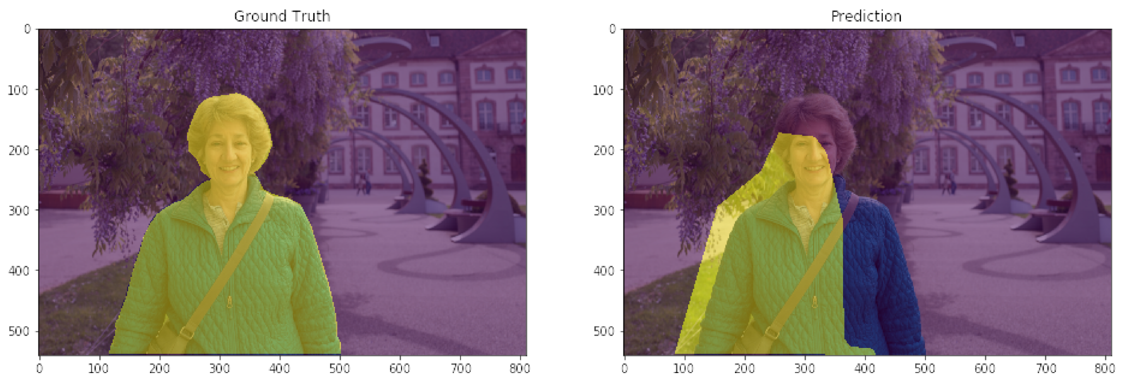


Figure 6.3: Example of ground truth and predicted target masks.Source: [8].

The mean IoU (mIoU) is computed as the average of the IoU scores for each class to consider the model's overall accuracy across all classes. The mIoU provides a comprehensive and useful metric that measures the model's segmentation performance over the entire dataset, effectively summarizing its precision in distinguishing and demarcating various object classes within the images. Moreover, mIoU serves as a critical benchmark to evaluate the accuracy of the semantic segmentation models implemented in this research. Thus, the resulting mIoU scores are crucial in assessing the capability of these models to generalize and accurately perform pixel-wise classification across diverse urban scenes.

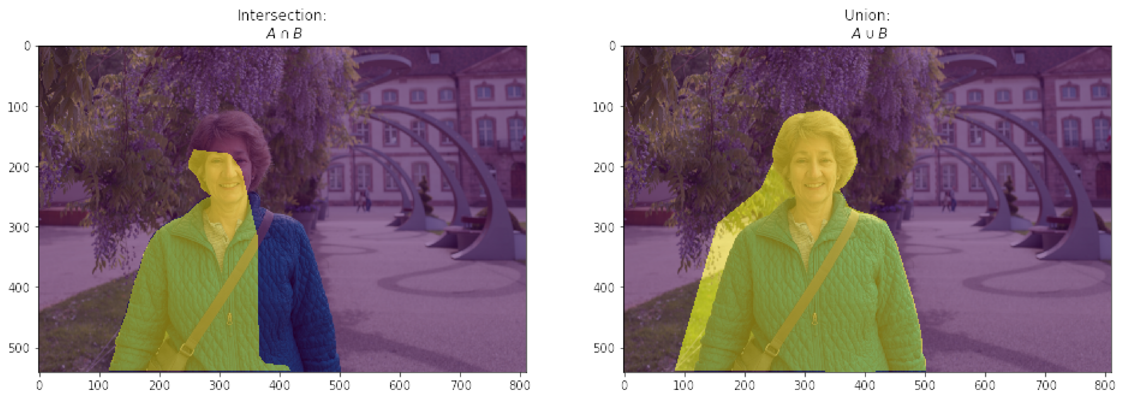


Figure 6.4: Visual representation of the intersection and union operations for IoU computation.Source: [8].

6.2 Quantitative Results

6.2.1 mIoU scores with cross entropy loss

The results shown in Table 6.1 show the Mean Intersection over Union (mIoU) performance of various experimental models trained and tested on the Cityscapes dataset. The loss function for all experiments is set to be ”Cross Entropy”. The baseline model, which only uses RGB images, demonstrates the best performance with a training mIoU of 68.34% and a test mIoU of 59.27%. In contrast, the model using only HHA (Depth) images shows a reduced performance, with training and test mIoU scores at 53.42% and 45.24%, respectively, as illustrated in Figure 6.5.

Setting	Enc.	Dec.	mIoU Train		mIoU Test	
			RGB	D	RGB	D
RGB	1	1	68.34	–	59.27	–
D	1	1	–	53.42	–	45.24
RGB || D	2	2	61.97	43.9	56.25	39.39
RGB || D	2	1	54.26	41.65	49.39	36.99
RGB & D	2	1	62.99		58.33	

Table 6.1: mIoU results using Cross Entropy loss

For the experiment where (RGB || D) with 2 encoders and 2 decoders, there is a noticeable drop in mIoU scores compared to the RGB-only baseline. The training results are 61.97% for RGB and 43.9% for Depth and the testing results are 56.25% for RGB and 39.39% for Depth.

For the experiment where (RGB || D) has two encoders but a single decoder, the model shows further reduced mIoU scores. The training results are 54.26% for RGB and 41.65% for Depth and the testing results are 49.39% for RGB and 36.99% for Depth.

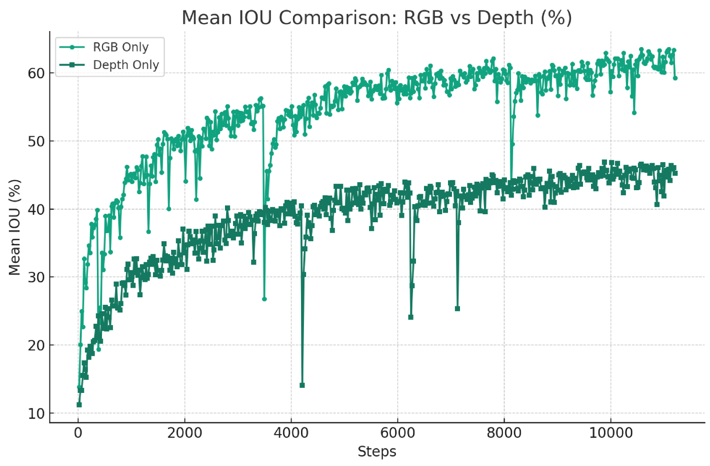


Figure 6.5: Mean IOU Comparison (%).

Although it couldn't improve the baseline model, the most notable improvement compared with other models is observed in the RGB & D setup, which employs two encoders and one decoder. This configuration results in a training mIoU score of 62.99% and a test mIoU score of 58.33%. This represents an advancement over the (RGB || D) modality and shows the potential for further modification.

6.2.2 mIoU scores with L2 + Cross Entropy losses

This section evaluates the case where a combination of L2 and Cross Entropy (CE) losses is used in the (RGB&D) model with two encoders and one decoder. The training and test results were lower compared to the CE-only scenario. The mIoU for training is 58.66%, and the test is 54.43% as in table 6.2. The CE-only scenario presented higher scores of 62.99% for training and 58.33% for testing. It shows that the L2 weighting factor was potentially too significant, and it constricts the model's flexibility during training. Such a constraint may not have allowed the model to explore the feature space effectively. Thus, further experiments with different L2 weighting factor values are suggested to examine the potential for this scenario to generalize and learn from the multimodal data inputs.

Setting	Loss Function	Enc.	Dec.	mIoU Train	mIoU Test
RGB & D	Cross Entropy	2	1	62.99	58.33
RGB & D	L2 + Cross Entropy	2	1	58.66	54.43

Table 6.2: Comparison of Cross Entropy and L2+Cross Entropy losses

7 Conclusion

The study aims to explore ways to enhance semantic segmentation for autonomous driving through multimodal data and federated learning.

The baseline model, using only RGB images, set a high standard with a training mIoU of 68.34% and a test mIoU of 59.27%. However, only the HHA (Depth) model struggled comparatively and achieved training and test mIoU scores of 53.42% and 45.24%, respectively. This highlighted the challenges of learning from depth information alone.

For (RGB || D) with two encoders and two decoders configuration a decline in mIoU scores was observed, suggesting complexity in managing and extracting features from two separate data modalities simultaneously. With this setup, the training mIoU for RGB was 61.97%, and for Depth, it was 43.9%, while testing mIoU for RGB was 56.25%, and for Depth, it was 39.39%.

The (RGB || D) model with two encoders but a single decoder further lowered mIoU scores, which suggests that while a shared decoder provides some computational efficiency, it may not be optimal for feature integration from two modalities.

Conversely, the RGB & D model with two encoders and one decoder marked an improvement against (RGB || D) modality and achieved a training mIoU of 62.99% and a test mIoU of 58.33%. This shows the potential of a multimodal approach that can effectively combine features from both RGB and Depth data for enhanced semantic segmentation.

The study discovered that using both L2 and Cross Entropy losses together gave some valuable insights, but it resulted in lower performance compared to using only the CE model. The research highlighted that the L2 weighting factor has a significant impact on training, which shows that optimizing loss functions can be tricky. Future studies should focus on fine-tuning the training process by carefully adjusting the loss function weights to unlock the model's potential for generalization.

In conclusion, this thesis provides a thorough analysis of multimodal semantic segmentation while also creating opportunities for more exploration. The results help to improve model architectures and loss functions.

Future work may include conducting broader experiments with different weighting factors or implementing advanced techniques to identify the most effective model structures. The ultimate goal is to refine and advance semantic segmentation methods to achieve the highest levels of accuracy and reliability required for autonomous driving and beyond.

References

[1] D. Shenaj, E. Fanì, M. Toldo, D. Caldarola, A. Tavera, U. Michieli, M. Ciccone, P. Zanuttigh, and B. Caputo, "Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning," 2022.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2021.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019.

[6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," 2017.

[8] J. Jordan, "Evaluating image segmentation models," https://www.jeremyjordan.me/evaluating-image-segmentation-models/, 2017, accessed: 2024-04-01.

[9] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2023.

[10] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[11] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. [Online]. Available: https://arxiv.org/abs/1610.05492

[12] NVIDIA. (2019) What is federated learning? Accessed: 2024-01-21. [Online]. Available: https://blogs.nvidia.com/blog/what-is-federated-learning/

[13] M. Asad, A. Moustafa, and T. Ito, "Federated learning versus classical machine learning: A convergence comparison," *CoRR*, vol. abs/2107.10976, 2021. [Online]. Available: https://arxiv.org/abs/2107.10976

[14] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016. [Online]. Available: http://arxiv.org/abs/1610.05492

[15] C. Chang, S. N. Srirama, and R. Buyya, "Internet of things (iot) and new computing paradigms," *CoRR*, vol. abs/1812.00591, 2018. [Online]. Available: http://arxiv.org/abs/1812.00591

[16] D. L. S. H. V. S. S. Jieren Cheng, Hua Li, "A survey on image semantic segmentation using deep learning techniques," *Computers, Materials & Continua*, vol. 74, no. 1, pp. 1941–1957, 2023. [Online]. Available: http://www.techscience.com/cmc/v74n1/49879

[17] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez, "A review on deep learning techniques applied to semantic segmentation," *CoRR*, vol. abs/1704.06857, 2017. [Online]. Available: http://arxiv.org/abs/1704.06857

[18] W.-K. L. Rizwan Ali Naqvi, Dildar Hussain, "Artificial intelligence-based semantic segmentation of ocular regions for biometrics and healthcare applications," *Computers, Materials & Continua*, vol. 66, no. 1, pp. 715–732, 2021. [Online]. Available: http://www.techscience.com/cmc/v66n1/40476

[19] B.-C.-Z. Blaga and S. Nedevschi, "Weakly supervised semantic segmentation learning on uav video sequences," in *2021 29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 731–735.

[20] S. Cakir, M. Gauß, K. Häppeler, Y. Ounajjar, F. Heinle, and R. Marchthaler, "Semantic segmentation for autonomous driving: Model evaluation, dataset generation, perspective comparison, and real-time capability," 2022.

[21] S. Dong, Y. Feng, Q. Yang, Y. Huang, D. Liu, and H. Fan, "Efficient multimodal semantic segmentation via dual-prompt learning," 2023.

[22] T. Brödermann, D. Bruggemann, C. Sakaridis, K. Ta, O. Liagouris, J. Corkill, and L. V. Gool, "Muses: The multi-sensor semantic perception dataset for driving under uncertainty," 2024.

[23] S. Sohangir, D. Wang, A. Pomeranets, and T. Khoshgoftaar, "Big data: Deep learning for financial sentiment analysis," *Journal of Big Data*, vol. 5, 2018.

[24] Q. Zhang, L. Yang, and Z. Chen, "Deep computation model for unsupervised feature learning on big data," *IEEE Transactions on Services Computing*, vol. 9, pp. 161–171, 2016.

[25] J. Liu, T. Li, P. Xie, S. Du, F. Teng, and X. Yang, "Urban big data fusion based on deep learning: An overview," *Inf. Fusion*, vol. 53, pp. 123–133, 2020.

[26] S. Chen, M. Tang, R. Dong, and J. Kan, "Encoder–decoder structure fusing depth information for outdoor semantic segmentation," *Applied Sciences*, vol. 13, no. 17, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/17/9924

[27] Y. Choi, N. Kim, S. Hwang, K. Park, J. S. Yoon, K. An, and I. S. Kweon, "Kaist multi-spectral day/night data set for autonomous and assisted driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 934–948, 2018.

[28] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," 2020.

[29] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth, "A2d2: Audi autonomous driving dataset," 2020.

[30] K. Lim, P. Treitz, M. Wulder, B. St-Onge, and M. Flood, "Lidar remote sensing of forest structure," *Progress in Physical Geography*, vol. 27, pp. 88–106, 03 2003.

[31] E. Ackerman, "Lidar that will make self-driving cars affordable [news]," *IEEE Spectrum*, vol. 53, pp. 14–14, 2016.

[32] S. Pillai and J. Leonard, "Monocular slam supported object recognition," *ArXiv*, vol. abs/1506.01732, 2015.

[33] Y. Fu, Y. Zheng, H. Huang, I. Sato, and Y. Sato, "Hyperspectral image super-resolution with a mosaic rgb image," *IEEE Transactions on Image Processing*, vol. 27, pp. 5539–5552, 2018.

[34] M. Hu, G. Zhai, D. Li, Y. Fan, H. Duan, W. Zhu, and X. Yang, "Combination of near-infrared and thermal imaging techniques for the remote and simultaneous measurements of breathing and heart rates under sleep situation," *PLoS ONE*, vol. 13, 2018.

[35] D. Mota-Rojas, A. Pereira, J. Martínez-Burnes, A. Domínguez-Oliva, P. Mora-Medina, A. Casas-Alvarado, J. Rios-Sandoval, A. de Mira Geraldo, and D. Wang, "Thermal imaging to assess the health status in wildlife animals under human care: Limitations and perspectives," *Animals : an Open Access Journal from MDPI*, vol. 12, 2022.

[36] C.-C. Sun, Y.-H. Wang, and M. Sheu, "Fast motion object detection algorithm using complementary depth image on an rgb-d camera," *IEEE Sensors Journal*, vol. 17, pp. 5728–5734, 2017.

[37] I. C. F. S. Condotta, T. Brown-Brandl, S. K. Pitla, J. P. Stinn, and K. O. Silva-Miranda, "Evaluation of low-cost depth cameras for agricultural applications," *Comput. Electron. Agric.*, vol. 173, p. 105394, 2020.

[38] D. Ramachandram and G. W. Taylor, "Deep multimodal learning: A survey on recent advances and trends," *IEEE Signal Processing Magazine*, vol. 34, pp. 96–108, 2017.

[39] J. Gao, P. Li, Z. Chen, and J. Zhang, "A survey on deep learning for multimodal data fusion," *Neural Computation*, vol. 32, pp. 829–864, 2020.

[40] Z. Guo, X. Li, H. Huang, N. Guo, and Q. Li, "Deep learning-based image segmentation on multimodal medical imaging," *IEEE Transactions on Radiation and Plasma Medical Sciences*, vol. 3, pp. 162–169, 2019.

[41] C. Zhang, Z. Yang, X. He, and L. Deng, "Multimodal intelligence: Representation learning, information fusion, and applications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, pp. 478–493, 2019.

[42] "Cityscapes Dataset Overview," https://www.cityscapes-dataset.com/dataset-overview/#features, Cityscapes Team, 2024, accessed: 10/02/2024.

[43] D. Rumelhart and J. L. McClelland, "Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations," 1986.

[44] K. Janod, M. Morchid, R. Dufour, G. Linarès, and R. Mori, "Denoised bottleneck features from deep autoencoders for telephone conversation analysis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, pp. 1505–1516, 2017.

[45] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," *ArXiv*, vol. abs/1703.00395, 2017.

[46] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, pp. 2041–2053, 2019.

[47] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019.

[48] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020.