University of Padova

Department of Information Engineering

Master Thesis in ICT for internet and multimedia

Multimodal Domain Adaptation for Point Cloud Semantic Segmentation

Supervisor
Prof. Pietro Zanuttigh
University of Padova

Co-supervisors
Dr. Francesco Barbato
Dr. Giulia Rizzoli
University of Padova

Master Candidate
Gianpietro Nicoletti

Student ID
2053042

Academic Year
2022-2023

Graduation Date
26-10-2023

"Working hard is important, but there's something that matters even more. Believing in yourself." - "Lavorare sodo è importante, ma c'è una cosa che conta di più. Credere in voi stessi."
— Harry Potter

Abstract

Semantic segmentation, thanks to multimodal datasets, can be made more reliable and accurate by mixing heterogeneous data, e.g. RGB images with LIDAR sequences or depth maps. However, it requires a large amount of annotated data to train neural networks able to solve the task effectively. Obtaining accurate and complete annotations for every image and LIDAR sample in a dataset (i.e. a label for each pixel or 3D point) can be a significant challenge, particularly when dealing with complex problems. To tackle the lack of annotation, Unsupervised Domain Adaptation (UDA) was studied and developed in recent years assuming labels are not available for the target dataset and supervising the training of the model using the labels of the source dataset. Using multimodal data in the UDA context is challenging since the different domains can be impacted differently by domain shifts. In this work, the exploration of autonomous driving datasets is undertaken to address the challenge of developing UDA techniques with a specific emphasis on adapting between RGB and Depth (RGB-D) and RGB-Lidar data. For all the datasets considered, the information from 2D images and 3D data (RGB-D or LIDAR-RGB) were used. The results demonstrate the effectiveness of adapting between different datasets and highlight how the accuracy of 3D data can be enhanced by leveraging the information provided by the images.

Contents

1 Introduction

The main tasks of computer vision are a set of problems aimed at teaching computers to understand, analyze, and interpret visual information obtained from images or videos. Nowadays, machine learning and deep learning models are frequently used to improve performance in various tasks.

Computer vision techniques have come a long way, allowing us to perform recognition tasks with ease, such as image classification and object detection. Thanks to these advancements, we can now apply more complex techniques, such as semantic segmentation, to understand scenes better. Moreover, thanks to the presence of various sensors, machine learning techniques can be implemented using multimodal data, which provide different information and are, therefore, more challenging to process. However, with a proper design of such algorithms, they can lead to more reliable results. Semantic segmentation finds practical applications in diverse fields, including autonomous driving, medical image analysis, robotics, and augmented reality. In this thesis, the focus lies within the context of autonomous driving, where semantic segmentation is employed using multimodal data containing both 2D and 3D information.

However, one of the major challenges in applying these techniques to real-world scenarios is the requirement for large amounts of labeled data. Collecting and annotating such data can be time-consuming and expensive. This is where Unsupervised Domain Adaptation (UDA) comes into play. UDA is a subfield of machine learning and computer vision that address the problem of adapting models trained on a source domain to work well in a target domain where labeled data is scarce or unavailable. In other words, it focuses on transferring knowl-

edge learned from one domain (*Source*) to another domain (*Target*) without requiring labeled data in the target domain. In recent years, UDA has become an active area of research within the computer vision community. Various methods, including domain adaptation neural networks, adversarial training, and domain-invariant representations, have been proposed to tackle the challenges of domain shift and reduce the need for labeled target domain data.
Focusing on these concepts, the aim of this thesis is to adapt a model trained on RGB-D information to data containing LIDAR-RGB information. Specifically, a data preprocessing step is introduced to transform the RGB-D information in the source dataset into LIDAR-like data that is the format of the target dataset. The segmentation is based on a modified version of UNet architecture, allowing the implementation of Sparse Convolution, and a feature extraction method using the VGG16 network is also employed. As source both real and synthetic datasets are considered in order to test the framework in different scenarios, while as target a real dataset is taken into account.

2

Background

2.1 Semantic Segmentation

Computer Vision plays a crucial role in enabling machines to perceive and understand the world through visual data, much like how our own eyes process information. By equipping machines with the ability to analyze images and videos, we empower them to perform tasks that were once limited to human perception. This has significant implications across various industries, including healthcare, automotive, surveillance, robotics, and entertainment.

Image classification is one of the foundational tasks in Computer Vision, and it involves teaching machines to recognize and categorize objects in images. For instance, a trained model can distinguish between cats and dogs or identify different types of fruits. By building on the success of image classification, researchers and engineers have ventured into more complex challenges, paving the way for applications like object detection, where machines can not only identify objects but also locate them within an image.

Semantic segmentation represents a remarkable advancement in this field, taking image analysis to a pixel-level understanding. Instead of just recognizing objects in their entirety, semantic segmentation delves into the finer details of an image, outlining the boundaries and regions of each object. This capability has found practical uses in various real-world scenarios. For example, in autonomous vehicles, semantic segmentation assists in road scene understanding, enabling the car to differentiate between roads, pedestrians, other vehicles, and obstacles.

In practice, the main objective of semantic segmentation is to take an image and produce an output that includes a segmentation map. In this map, the initial pixel values are transformed into specific class label values, such as [0, 1,..., N-1], where N is the number of classes. In Figure 2.1 an example of an image with its ground truth, representing the semantic segmentation, is reported.

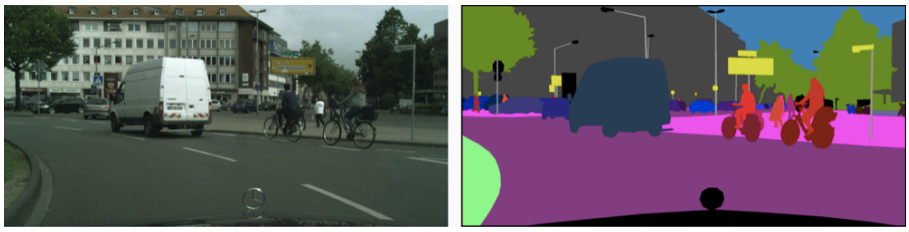


Figure 2.1: Example of semantic segmentation in Cityscapes dataset.

Formally, if we have an image of size HxW pixels $P_{(i,j)}$ with in general k channels (e.g $k = 3$ in case of RGB images), $\forall i,j : 0 \leq i \leq H-1, 0 \leq j \leq W-1$:

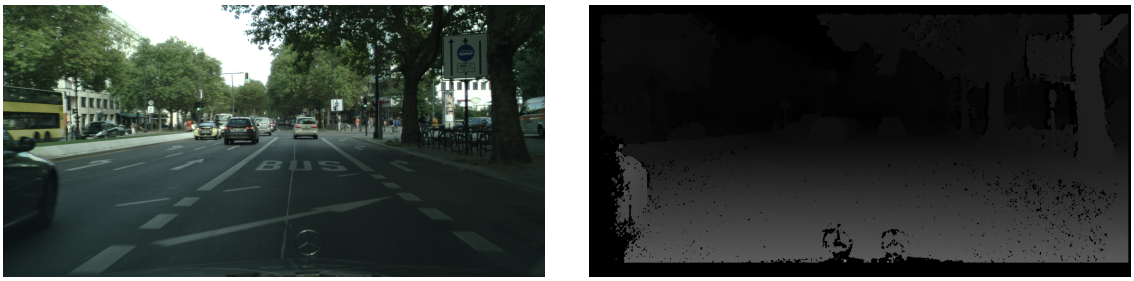
$$P_{(i,j)} = (x_0, x_1, ..., x_{k-1}) \rightarrow l_{(i,j)},\ l_{(i,j)} \in [0, N-1] \tag{2.1}$$

In general, we can extend the definition to different types of data, such as a series of 3D points in a pointcloud.

2.2 Multimodal

Multimodal data refers to information that is captured through multiple sensory modalities, such as images, audio, text, or sensor readings. Each modality carries unique insights and context about the data, making it richer and more comprehensive. For instance, a video contains both visual information and corresponding audio, which, when combined, provide a more profound understanding of the content.
The use of multimodal data has become increasingly popular in the field of machine learning. Traditional machine learning models typically worked with data that had only one type of information, known as unimodal data. However, it's worth mentioning that sometimes these models also encountered data with multiple types of information, although they weren't very effective at handling it. However, with the development of more advanced neural networks, data processing has improved significantly. This improvement allows the models to better use

the strengths of each type of information and combine them effectively, resulting in more accurate predictions.
Applications of multimodal data in machine learning are abundant. In natural language processing, combining text with images can lead to better sentiment analysis or image captioning. In autonomous vehicles, integrating data from cameras, LIDAR, and other sensors enables a more robust perception of the environment, ensuring safer and more reliable driving.
The use of multimodal data is highly advantageous in the field of machine learning and neural networks. By incorporating information from different sensory modalities, we can create more adaptable and intelligent systems that can better understand the complexities of our world. As we further explore the possibilities of multimodal learning, we can anticipate significant breakthroughs in AI and its revolutionary impact on various industries. An example of multimodal data is the disparity map associated with an RGB image, Figure 2.2.



(a) RGB image.

(b) Disparity map.

Figure 2.2: Example of disparity map.

2.3 Domain Adaptation

In the realm of machine learning and artificial intelligence, one of the critical challenges we face is adapting models to new and unseen data distributions. This process is known as domain adaptation, and it plays a crucial role in building robust and generalizable AI systems.
Domain adaptation refers to the process of using knowledge acquired from a source domain, where there is labeled data available for training, to enhance the performance of a model in a target domain, where labeled data may be scarce or not exist. The main goal of domain adaptation is to reduce the differences between the source and target domain distributions, which allows the model to perform effectively in the target domain, even with limited labeled samples.
Domain adaptation becomes an essential topic to explore, as it addresses the challenges faced

when deploying AI models in real-world scenarios. For instance, in computer vision, a model trained on images from a specific dataset might not generalize well to different lighting conditions, camera angles, or environmental variations. Domain adaptation techniques seek to mitigate such discrepancies, allowing the model to adapt and maintain high performance in diverse operational environments.
One of the prominent subfields of domain adaptation is unsupervised domain adaptation (UDA). While traditional domain adaptation assumes access to labeled data in the target domain, UDA takes a more challenging scenario where only unlabeled data is available in the target domain. This setting is particularly relevant in situations where manually labeling data in the target domain is expensive, time-consuming, or impractical.

2.3.1 Test Time Adaptation

Test-time adaptation is a powerful technique used in artificial intelligence to help models perform effectively on new and unseen data in real-world scenarios. It allows the model to continually learn and adapt, much like how humans learn from experiences and adjust their understanding accordingly. One common approach to test-time adaptation is fine-tuning, where a pre-trained model is further trained with new data related to the specific task at hand. However, it is essential to note that fine-tuning cannot always be done using the same loss function used during the initial training. Using the same loss function often leads to catastrophic forgetting, a phenomenon where the model forgets its previous knowledge when adapting to new data. Moreover, during fine-tuning, there may be limitations in accessing the original training data due to privacy or proprietary concerns. This makes it challenging to maintain a balance between incorporating new information and preserving the knowledge obtained during the initial training. Furthermore, test-time adaptation becomes particularly difficult when done in an unsupervised manner, such as in the context of unsupervised domain adaptation. In unsupervised domain adaptation, the target domain lacks labeled data, and the model must adapt to the new domain without explicit guidance from labeled samples. This adds an extra layer of complexity to the adaptation process, as the model must learn to generalize effectively from the source domain to the target domain without access to explicit target labels. In conclusion, test-time adaptation is a critical aspect of AI research, allowing models to adapt and perform well on new and diverse data in real-world scenarios. While fine-tuning is a common approach, it requires careful consideration to prevent catastrophic forgetting and handle limitations in

accessing training data. Additionally, performing test-time adaptation in an unsupervised context, such as unsupervised domain adaptation, presents further challenges that researchers are actively addressing to improve the adaptability and performance of AI models.

3
Related works

3.1 3D Semantic Segmentation

3D semantic segmentation methods aim to assign semantic labels to individual points in a 3D point cloud. These methods can be broadly categorized into traditional methods and deep learning methods.

Traditional methods typically follow a process of over-segmenting the point cloud into smaller regions, extracting features from these regions, and then classifying them into semantic categories. Representative methods using this process include [1], [2], [3]. Another approach in traditional methods is to directly design feature vectors for each point without prior over-segmentation, as seen in [4]. Some traditional methods, such as [5] use Conditional Random Fields (CRF) to aggregate contextual information.

On the other hand, deep learning methods utilize deep neural networks to learn feature representations and directly map input data to semantic labels. These methods can be further divided into four groups based on the formats of input data:

Point-based methods: Methods like Pointnet[6] take raw point clouds as input and output point-wise labels. They can process arbitrary unstructured point clouds. The main challenge in processing raw point clouds is extracting local contextual features from the unstructured data.

Image-based methods: These methods project 3D LiDAR data onto a surface to generate 2D

images, which are then used as inputs for deep models. The output predictions with pixel-wise labels are reprojected to the original 3D LiDAR points. Multi-view segmentation and range image segmentation [7] are two common strategies in image-based methods.

Voxel-based methods: These methods divide the 3D space into regular grids called voxels converting the pointcloud into a voxel grid representation. The voxel grid is then fed into a 3D convolutional neural network (CNN) for semantic segmentation. One example is the method proposed by Choy et all.[8].

Graph-based methods: These methods represent the point cloud as a graph, where each point is a node and the edges represent the spatial relationships between points. Graph convolutional networks (GCNs) or graph neural networks (GNNs) are used to process the graph structure and perform semantic segmentation. An example is RG-GCN [9].

3.2 Multimodal Semantic Segmentation

In recent years, there's been a growing interest in creating multi-modal 3D semantic segmentation models for autonomous driving. Different techniques have been proposed to tackle this challenge, like combining data from various sensors, using prior knowledge, and transferring learning from one domain to another.

Among the models for LiDAR point cloud segmentation, the Randla-Net[10] focuses on 3D information but ignores valuable 2D information from RGB images.

To overcome these limitations, Yan et al. [11] proposed a method that uses 2D information as priors to improve semantic segmentation accuracy on LiDAR data. It performs better than other techniques that combine data or use prior knowledge while being faster than domain transfer methods.

In the broader context, other works have also been proposed, achieving good performance on certain benchmarks but still facing challenges in handling different sensor data effectively.

To address these issues MSeg3D[12] was proposed, a new multi-modal 3D semantic segmentation model. It generates pseudo-camera views from LiDAR and multi-camera images using an asymmetric data augmentation technique. The model combines features from different sensors and generates pixel-wise semantic labels. Experiments on benchmarks show that MSeg3D performs exceptionally well, especially in challenging scenarios like occlusions and limited camera field of view.

Additionally, in an early work Eitel et al.[13] proposed a fusion method based on deep neural networks.

Other works like JSIS3D[14] and MFNet[15] also tackled multimodal segmentation but were limited to a specific number of modalities and didn't address the challenge of fusing diverse information.
To tackle these challenges, CMNEXT[16] model was introduced, using an asymmetric fusion structure and a universal fusion model. It can handle multiple modalities flexibly and achieve state-of-the-art performance on various benchmarks, including the DELIVER benchmark (proposed by the same authors [16]) with multiple modalities and adverse weather conditions. Some works, like the one proposed by Barbato et al.[17], used as modality the depth instead of LIDAR. An exhaustive survey written by Rizzoli et al. [18] presented a series of proposed methods in the context of semantic segmentation in autonomous driving.

3.3 Domain Adaptation

In the realm of computer vision, particularly in the context of complex perception tasks such as object detection and semantic segmentation, the domain of unsupervised domain adaptation (UDA) has garnered significant attention in recent years. This surge in interest has prompted the exploration and development of various UDA techniques, exemplified by methods like the one proposed by Luo et al.[19].
One notable avenue in pursuing UDA involves adversarial training, a strategic approach aimed at minimizing the distributional discrepancies between source and target domains. These discrepancies may manifest across different levels, ranging from subtle pixel-level variations to more pronounced feature-level disparities and even extending to differences in output spaces. Remarkable UDA methods like Deep Domain Adaptation [20], Generalized adversarial adaptation [21], and the algorithm introduced by Tsai et al.[22] highlight the effectiveness of leveraging adversarial alignment to address these challenges.
Within this dynamic landscape, an essential contribution comes from the work of Wu et al., who introduced SqueezeSegV2[23]. This pioneering work not only presented an improved model structure but also demonstrated the potential of unsupervised domain adaptation for road object segmentation using LIDAR point clouds.
Taking a significant step forward, a recent breakthrough has led to the emergence of xMUDA[24]. This approach represents a paradigm shift by capitalizing on the inherent synergies between 2D images and 3D point clouds. By incorporating insights from both modalities, xMUDA applies UDA to multimodal data, with the potential to significantly enhance the accuracy and robustness of segmentation models, even in the presence of substantial domain differences.

While xMUDA marks an important stride, it is crucial to recognize its limitations. Specifically, xMUDA's ability to fully harness the complementary information from different modalities remains underexplored, leading to performance constraints, particularly when domain disparities are significant. In response, researchers have introduced Mx2M[25], an innovative method that employs masked cross-modality modeling to effectively bridge domain gaps. Comprising components such as cross-modal removal and prediction (xMRP) and dynamic cross-modal filter (DxMF), Mx2M dynamically aligns and matches features from diverse modalities, ultimately enhancing model performance.

Amidst this landscape of advancements, the transformative influence of Transformers stands out. These models, initially harnessed for natural language processing, have transcended disciplinary boundaries, leaving an indelible impact on diverse domains, including computer vision. The works of Wolf et al.[26] and Rizzoli et al. [27] serve as proof of the versatility and state-of-the-art performance that Transformers bring to the table. Their integration has paved the way for researchers to explore novel avenues within the realm of UDA, fostering innovative strategies to address challenges posed by domain adaptation.

Further augmenting this evolution is the emergence of the method proposed by Xing et al.[28]. This method introduces a novel cross-modal contrastive learning scheme, leveraging correspondences between 2D pixel features and 3D point features. By fostering interaction between these modalities, the approach enhances feature representations within both the labeled source domain and the unlabeled target domain. Additionally, the method introduces a neighborhood feature aggregation module to further enrich pixel features, effectively utilizing 2D contextual information for domain adaptation through cross-modal learning. Furthermore, it's important to mention that Toldo et al.[29] have presented a comprehensive survey that analyzes many of the known UDA techniques, also considering many of those mentioned earlier.

3.3.1 Test time adaptation

In recent years, deep learning has made remarkable strides in various domains, such as image classification, object detection, and natural language processing. However, one of the major challenges in deploying deep learning models in real-world applications is their limited ability to generalize to new and different data distributions, a problem known as dataset shift.

To address this issue, several methods have been proposed, including domain adaptation, transfer learning, and meta-learning. Domain adaptation aims to generalize a model to a new domain by leveraging knowledge from a source domain, while transfer learning initializes a model's

weights using a pre-trained model on a large dataset for a specific task. Meta-learning focuses on enabling a model to quickly adapt to new tasks with limited data.

Despite the success of these methods, they have limitations in fully test-time adaptation, where the model must adapt to new data distributions during inference without access to source data or supervision. To tackle this challenge, the TENT[30] method is proposed.

TENT optimizes a model for confidence during testing by minimizing the entropy of its predictions. It estimates normalization statistics and optimizes channel-wise affine transformations during test-time adaptation. The proposed method has achieved state-of-the-art results on various benchmarks for image classification and domain adaptation tasks, providing a promising solution to fully test-time adaptation in deep learning. By optimizing for confidence during testing, TENT can adapt to new data distributions without access to source data or supervision, potentially improving the accuracy, efficiency, and availability of deep learning models in real-world applications.

On the other hand, the EcoTTA[31] addresses the problem of performance degradation in deep neural networks due to domain shifts between train and test distributions. This issue has been tackled by research fields like unsupervised domain adaptation and domain generalization, and various methods have been proposed, such as [32], [33], [34] for unsupervised domain adaptation, and [35], [36], [37] for domain generalization.

To address this issue, EcoTTA introduces a novel approach to continual test-time adaptation (TTA) that outperforms other state-of-the-art methods. It utilizes lightweight meta-networks trained on the source domain and adapted to the target domain during TTA, minimizing memory consumption by using frozen original networks and discarding intermediate activations. The proposed approach achieves stable performance even in long-term TTA.

Additionally, the authors propose self-distilled regularization to prevent overfitting and catastrophic forgetting during long-term adaptation with unsupervised loss. This regularization uses the architecture of the lightweight metanetworks to preserve well-trained knowledge from the source domain.

4

Pipeline

The proposed framework for this research work is based on a series of key steps aimed at leveraging multimodal data for semantic segmentation and test-time adaptation. The main objective is to demonstrate that using multimodal data can improve performance both during training on source data and during the test-time adaptation phase, following the approach outlined by the authors of TENT [30]. While in this chapter a high-level description of the pipeline is presented, in Chapters 6 and 7 a better description of the models used and the various steps, along with the results, are reported.

4.1 Preprocessing

The first step in our framework is data preprocessing as explained in Section 5.3. RGB images and LiDAR-like data from various sources, for example, Cityscapes as the source dataset and SemanticKITTI as the target dataset are preprocessed to standardize the data format to make them compatible with each other. This phase includes transforming images into LiDAR-like data, ensuring that information is represented in a common format.

4.2 Network Training

In the next phase, preliminary training of the neural network is performed using only the 3D data from the source dataset. Subsequently, RGB information is introduced into the segmentation process. Initially using raw RGB information to evaluate performance improvement. Additionally, further preprocessing to RGB images was applied, using the first two layers of the VGG16 network to extract features. This feature extraction can be done both without further training and by training the VGG16 layers, depending on the cases. In practice, during the backpropagation phase, the update of the layers responsible for feature extraction was blocked, even if those layers were part of the networks.

4.3 Selection of Optimal Features

After incorporating RGB information into the segmentation process, it is necessary to determine which features are most useful for performance improvement. In this context, several options are considered:

- RGB Raw (3D+RGB): Direct use of RGB information without further processing.
- Feature Layer 1 without training (*3D + Feats1(PT)*): Using features extracted from the first layer of VGG16 without further training.
- Feature Layer 1 with training (*3D + Feats1(T)*): Using features extracted from the first layer of VGG16 with additional training.
- Feature Layer 2 without training (*3D + Feats2(PT)*): Using features extracted from the second layer of VGG16 without further training.
- Feature Layer 2 with training (*3D + Feats2(T)*): Using features extracted from the second layer of VGG16 with additional training.

These steps allow us to explore which types of RGB information or features extracted from VGG16 are most effective in improving semantic segmentation on source data and on target data before the TTA.

4.4 Test Time Adaptaion

To complete the framework, TTA is performed on the target data. The method used for TTA is based on the approach presented in [30]. Prior to TTA, the data is preprocessed and the same features selected in the previous step are used. Then, since in the context of TTA the target labels are not available, the model, previously trained on the source dataset, is retrained on target data in an unsupervised manner using the Shannon entropy loss.

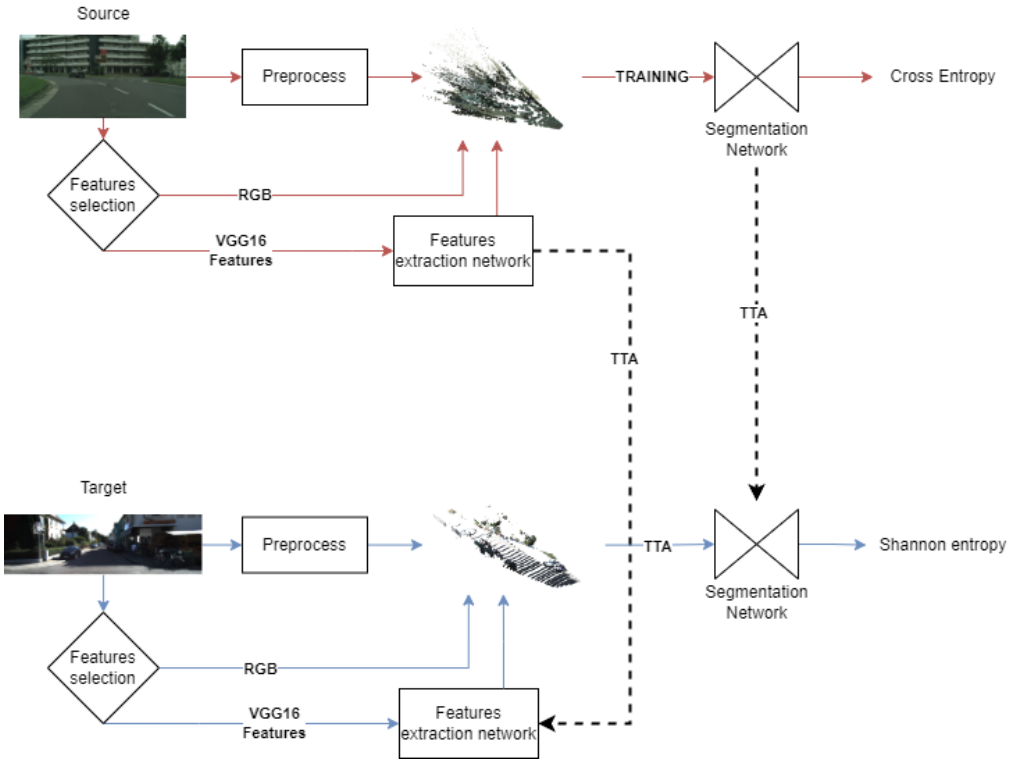


Figure 4.1: Pipeline: first of all both the source and target dataset are preprocessed as defined in Section 5.3, then the model is trained using a specific set of features using the Cross-Entropy loss computed on the ground truth of the source dataset: if the raw RGB data was used, they are retrieved directly from the image, otherwise the image is passed through the features extraction network (VGG16) in order to create a series of features for each point of the pointcloud. When the model is trained, TTA is applied to the target dataset using the entropy minimization on the Shannon entropy.

5 Dataset

This chapter focuses on the datasets used in this thesis for working with multimodal data and unsupervised domain adaptation. Three datasets containing both RGB and three-dimensional information were carefully selected and considered for the study.

5.1 RGBD

5.1.1 Real dataset: Cityscapes

The Cityscapes dataset [38] is a widely-used benchmark in computer vision for understanding urban scenes. It contains high-resolution images taken from a car's perspective while driving through different cities in Germany. The dataset includes various urban environments, such as city centers, neighborhoods, and suburbs, with complex traffic scenarios, diverse buildings, pedestrians, and different weather conditions.

- **Semantic Segmentation Annotations:** The dataset provides pixel-level annotations for object classes, such as pedestrians, vehicles, roads, buildings, traffic signs, and vegetation. This allows for tasks like object detection and semantic segmentation.
- **Instance-level Annotations:** Cityscapes includes annotations that distinguish individual instances within the same class. This information is useful for tasks like instance segmentation and tracking.

- **Fine and Coarse Annotations:** The dataset offers both fine and coarse annotations. Fine annotations have highly detailed pixel-wise labels for a subset of images, while coarse annotations have lower-resolution labels for the entire dataset, allowing for less computationally intensive experiments.
- **Stereoscopic Information:** In some scenes, Cityscapes includes a stereo pair of images, providing depth information for tasks like depth estimation and 3D scene reconstruction.
- **Training, Validation, and Test Sets:** To ensure fair evaluation, the dataset is split into training, validation, and test sets. The training set is used for model training, the validation set for hyperparameter tuning, and the test set for unbiased evaluation.

In conclusion, the Cityscapes dataset is a valuable resource for researchers in computer vision, enabling them to develop and evaluate algorithms for urban scene analysis effectively in German cities.

5.1.2 Synthetic dataset: Synthia

The SYNTHIA dataset[39] stands as a valuable asset in the realm of computer vision, utilized for gaining enhanced insights into urban scenes. It encompasses synthetic images simulating realistic urban landscapes and has been formulated to train and assess computer vision algorithms effectively.

- **Synthetic Images:** The dataset comprises images that resemble real city scenes. These images are generated using specialized software that imparts them with detailed and convincing attributes.
- **Semantic Labels:** Each image is equipped with labels that indicate the presence and location of objects. This aids in teaching computers to recognize objects and features within urban settings.
- **Depth Information:** Certain images within the dataset include depth information. This information proves useful for estimating distances and understanding the three-dimensional arrangement of objects within the scene.

The dataset's images are generated through advanced rendering software. This process facilitates the creation of realistic three-dimensional scenes, rendering synthetic images akin to those found in the real world. Annotations, which are labels describing objects within the images, are accurately added.

A noteworthy feature of the SYNTHIA dataset is its compatibility with Cityscapes. This compatibility was devised to streamline comparisons between the two datasets and enable algorithms trained on SYNTHIA to function seamlessly on images from Cityscapes.

This compatibility was achieved by aligning categories and annotations across the two datasets. Practically, it means that models trained on SYNTHIA can be employed on Cityscapes. This proves advantageous, as it allows researchers to test their algorithms on both synthetic and real-world scenarios using the same models.

In essence, the SYNTHIA dataset constitutes a vital resource for computer vision, offering synthetic data useful for training algorithms. Its compatibility with Cityscapes expands research possibilities, enabling the testing of ideas across diverse scenarios. This advancement continually propels the understanding of computers' visual perception capabilities forward.

5.2 RGB + LIDAR

The Semantic KITTI dataset [40] is a widely used dateset in computer vision for semantic segmentation tasks in autonomous driving scenarios. It is an extension of the original KITTI dataset, with additional semantic labels for urban street scenes captured from a moving vehicle in Germany.

- **Dataset Extension:** Semantic KITTI extends the original KITTI dataset by providing pixel-level semantic labels for each captured LiDAR point. This semantic information enhances the dataset's usefulness for various scene-understanding tasks, particularly semantic segmentation.
- **LiDAR-based Annotations:** Unlike traditional RGB-based semantic segmentation datasets, SemanticKITTI utilizes LiDAR sensors to capture 3D point cloud data. Each point in the point cloud is associated with a semantic label, allowing for the study of 3D semantic understanding and object recognition in urban environments.
- **Outdoor Driving Scenarios:** The dataset consists of urban street scenes captured from a moving vehicle equipped with LiDAR and cameras in Germany. It includes a wide range of driving scenarios, such as busy city streets, highways, and residential areas, enabling comprehensive analysis of autonomous driving challenges.
- **Large-scale and Diverse:** Semantic KITTI comprises sequences of point clouds, with each sequence containing thousands of LiDAR sweeps. The dataset covers several urban areas in Germany, making it diverse and suitable for training and evaluating large-scale models.

- **Training, Validation, and Test Sets:** The dataset is divided into training, validation, and test sets, following the original KITTI dataset split.

In conclusion, the Semantic KITTI dataset is a valuable resource for researchers in computer vision and autonomous driving, providing rich LIDAR-based annotations for urban street scenes captured in Germany. It enables the development and evaluation of algorithms for semantic segmentation and other related tasks in the context of autonomous vehicles.

5.3 Datasets preprocess

5.3.1 Cityscapes

The main objective of the preprocessing pipeline is to obtain LiDAR-like data from stereo images. This involves simulating 3D point clouds from the provided disparity information and camera calibration. By leveraging the disparity maps and intrinsic camera parameters, a representation similar to LiDAR data is achieved. The approach utilizes only the left image and computes 3D points to generate LiDAR-like data, enabling further processing and analysis for various computer vision applications.

- **Disparity Calculation**: For the given stereo images, we used the Cityscapes dataset, which already provides pre-calculated disparity maps. A disparity map is a visual representation used in the field of computer vision and stereoscopy. It is obtained by comparing two images captured by cameras with slightly different angles. Disparity refers to the difference in the position of a point in one of these images compared to its counterpart in the second image. The disparity map visualizes these differences as a map, where each point represents a disparity. This map is commonly used to determine the depth of objects in the scene because the disparity between corresponding positions in the two images can be used to calculate the distance from objects in a three-dimensional scene.
  In the absence of pre-calculated disparities, stereo-matching techniques could be employed. Stereo matching aims to find corresponding points between the left and right images and calculate the disparity as the horizontal difference between these corresponding points.
  Common stereo matching algorithms, such as Sum of Absolute Differences (SAD) or Normalized Cross-Correlation (NCC), involve sliding a small window or block across the left image and searching for the best matching block in the right image. The disparity is then determined based on the shift between the two matching blocks.

- **Conversion of Disparity to Depth**: Once the disparity map is available, the depth of a 3D point from its disparity is calculated using the following formula:

$$Z = \frac{b \cdot f}{d} \tag{5.1}$$

Where:

  - Z represents the depth of the 3D point concerning the camera.
  - b denotes the distance between the two cameras (baseline), representing the horizontal distance between their viewpoints.
  - f signifies the focal length of the camera, which is the distance between the camera's focal point and the image plane. In this preprocessing, we only consider the focal length for converting disparity to depth.
  - d represents the disparity value obtained from the stereo images using stereo matching techniques or provided in the Cityscapes dataset.

- **Projection from Image to 3D**: To project a point from a 2D position in the left image to a 3D position in space, we utilize the camera's intrinsic parameters. The 2D coordinates of the point in the left image are denoted as (u, v), and its 3D coordinates are represented by (X, Y, Z).

$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} I|0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.2}$$

Where:

  - (u, v) represents the coordinates of the point in the left image.
  - (X, Y, Z) represents the 3D coordinates of the point in space.
  - K is the intrinsic parameters matrix, which includes the focal length and other intrinsic parameters. In this preprocessing, we only consider the focal length for the projection.

In the context of the 3D reconstruction process, it is important to note that the generated data results in a dense representation of the scene. Unlike sparse LIDAR data, which contains discrete points obtained from laser measurements, the 3D reconstruction yields a high density of points densely covering the surfaces of the scene. This difference in data density between 3D reconstruction and LIDAR data is a crucial consideration during the subsequent stages of processing.

Generation of Lidar-like Data

The objective of the following steps is to generate LIDAR-like data starting from the dense 3D reconstruction. This means that a subsampling of the points must be performed.

The 3D points are first of all converted into spherical coordinates using the following formulas:

- The horizontal distance $d_{x,y}$ from the point $P(x, y, z)$ to the origin in the xy plane is given by:
$$d_{x,y} = \sqrt{x^2 + y^2} \tag{5.3}$$
- The radial distance rad from P to the origin in 3D space is given by:
$$d_{radial} = \sqrt{d_{x,y}^2 + z^2} \tag{5.4}$$
- The polar angle θ, representing the angle between the position vector P and the z-axis, is calculated as:
$$\theta = \arccos \frac{z}{\text{rad}} \tag{5.5}$$
- The azimuthal angle φ, representing the angle between the projection of P onto the xy plane and the x-axis, is calculated as:
$$\varphi = \arccos \frac{x}{d_{x,y}} \tag{5.6}$$

Next, we define a set of horizontal planes with quantized polar angles. We call this set thetas, and the values of the quantized polar angles are given by:

$$S_\theta = \text{linspace}\left(\frac{\pi}{2} - \frac{\pi}{8}, \frac{\pi}{2} + \frac{\pi}{8}, 64\right) \tag{5.7}$$

Finally, we select the points whose absolute difference between the polar angle θ and the quantized values t in the set of thetas is less than a certain threshold. Formally, we have:

$$|\theta - t| < \text{threshold}, \ t \in S_\theta \tag{5.8}$$

This process enables the generation of 3D points with information about their spatial locations in spherical coordinates, as well as the associated RGB color. By selecting points based on the polar angle condition, a sparse LiDAR-like representation enriched with color information is achieved.

(a) RGB image

(b) Pointcloud

Figure 5.1: Image and pointcloud at the end of preprocessing, Cityscapes,

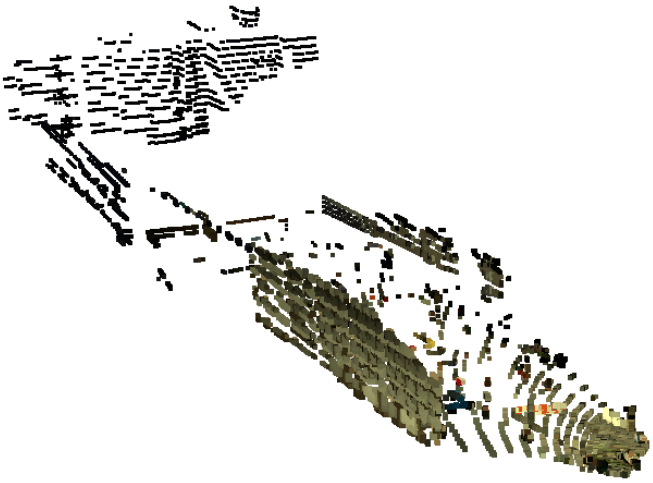
5.3.2 SYNTHIA

The preprocessing for the SYNTHIA dataset was similar to Cityscapes, with one difference. SYNTHIA already had disparity maps converted into depth maps, making the preprocessing smoother as depth information was readily available without extra steps.

5.3.3 SEMANTICKITTI

In the context of the SemantiKITTI dataset, a valuable collection of 3D points captures a detailed representation of the environment. To effectively correlate this 3D data with the associated LiDAR measurements and RGB information, a crucial step is required: projecting the 3D points into the 2D image space and selecting only the points that lie

(a) RGB image.



(b) Pointcloud.

Figure 5.2: Image and pointcloud at the end of preprocessing, Cityscapes.

within the boundaries of the corresponding 2D image.
The camera calibration parameters are utilized for the projection of each 3D point onto the image plane, establishing a direct correspondence between the 3D points and their corresponding 2D pixel coordinates in the image. By aligning the 3D points with the 2D image, the transfer of RGB color information from the image to the associated 3D points is achieved in a straightforward manner.
Following the projection, a careful selection process is performed to retain only the 3D points that fall within the dimensions of the image. This refined selection ensures that the 3D points are spatially aligned with the captured LiDAR data and RGB color information from the image.

(a) RGB image.

(b) Pointcloud.

Figure 5.3: Image and pointcloud at the end of the preprocess, SemanticKITTI.

5.3.4 Classes re-mapping

Class remapping is important because the SemantiKitti and Cityscapes (or Synthia) datasets use different class labels, although, in most cases, they represent the same element.
A mapping (Table 5.1) was created to establish clear connections between class names and their corresponding numeric IDs in the SemantiKitti and Cityscapes datasets. This process ensures precise alignment of each class name and numeric ID in one dataset with

their counterparts in the other.
In cases where certain classes are not suitable for the LiDAR-like data format, such as "sky," these classes and their corresponding numeric IDs were intentionally mapped to a class considered "unlabeled."

Table 5.1: Cityscapes/Synthia-SemanticKITTI class re-mapping

City Name	City Raw ID	Kitti Name	Kitti Raw ID	Join Name	Join ID
Unlabeled	0	Unlabeled	0	Unlabeled	-100
		Outlier	1		
Ego Veh.	1				
Rect. Bor.	2				
Out of ROI	3				
License P.	34				
Sky	23				
Guard Rail	14				
Bridge	15				
Tunnel	16				
Rail Track	10	Other Struct.	52		
Car	26	Car	10	Vehicle	0
		Mov. Car	252		
Truck	27	Truck	18		
		Mov. Truck	258		
Bus	28	Bus	13		
		Mov. Bus	257		
Caravan	29	Other Veh.	20		
Trailer	30				
		Mov. Other.	259		
Train	31	On Rails	16		
		Mov. On Rails	256		
Motorcycle	32	Motorcycle	15	Two-wheels	1
Bicycle	33	Bicycle	11		
Rider	25	Bicyclist	31		
		Mov. Bicycle.	253		
		Motorcyclist	32		
		Mov. Motorc	255		
Person	24	Person	30	Person	2
		Mov. Person	254		
Road	7	Road	40	Road	3
		Lane Mark.	60		

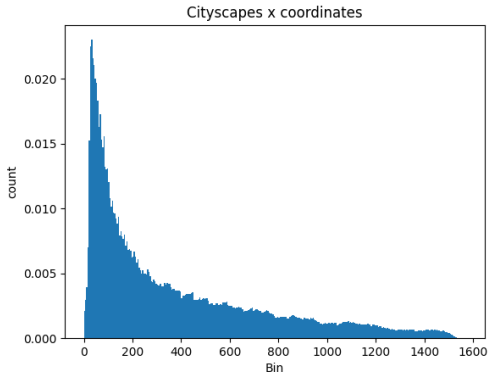
Continues on the next page

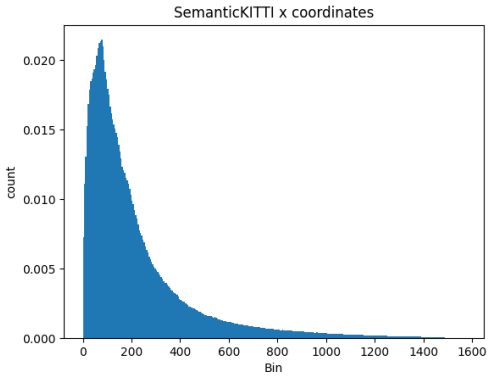
Table 5.1 – Continuation from previous page

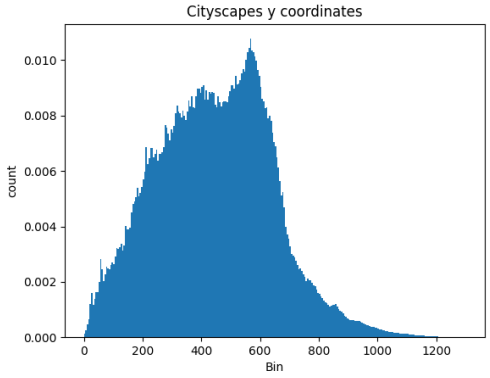
City Name	City Raw ID	Kitti Name	Kitti Raw ID	Join Name	Join ID
Parking	9	Parking	41		
Sidewalk	8	Sidewalk	48	Sidewalk	4
Building	11	Building	50	Building	5
Vegetation	21	Vegetation	70	Nature	6
		Trunk	71		
Terrain	22	Terrain	72		
Ground	6	Other Ground	49		
Pole	17	Pole	80	Pole	7
Pole Group	18				
Fence	13	Fence	51	Structures	8
Wall	12				
T. Sign	20	T. Sign	81	T. Sign	9
Static	4	Other Obj.	99	Objects	10
Dynamic	5				
T. Light	19				

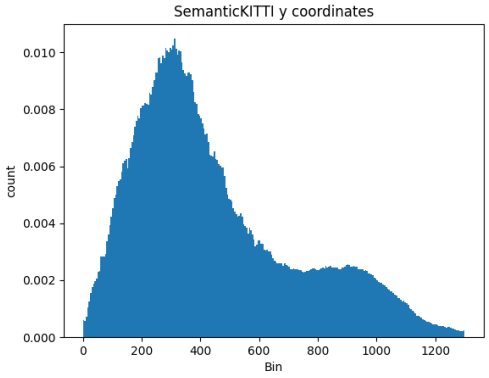
5.3.5 Distribution of the 3D coordinates

To ensure a consistent 3D scene representation, a threshold of 0.00025 was chosen (referring to the Equation 5.8). This value was determined after analyzing the results, as explained in Chapter 7. Using this threshold, the 3D coordinates in both the SemanticKITTI and Cityscapes datasets appear very similar. This similarity is important for making the generated LiDAR-like data compatible. The examination encompassed all three spatial coordinates as well as the distance from the origin (Figure 5.5).


Cityscapes x coordinates
count
Bin
0.000
0.005
0.010
0.015
0.020
0 200 400 600 800 1000 1200 1400 1600


SemanticKITTI x coordinates
count
Bin
0.000
0.005
0.010
0.015
0.020
0 200 400 600 800 1000 1200 1400 1600


Cityscapes y coordinates
count
Bin
0.000
0.002
0.004
0.006
0.008
0.010
0 200 400 600 800 1000 1200


SemanticKITTI y coordinates
count
Bin
0.000
0.002
0.004
0.006
0.008
0.010
0 200 400 600 800 1000 1200

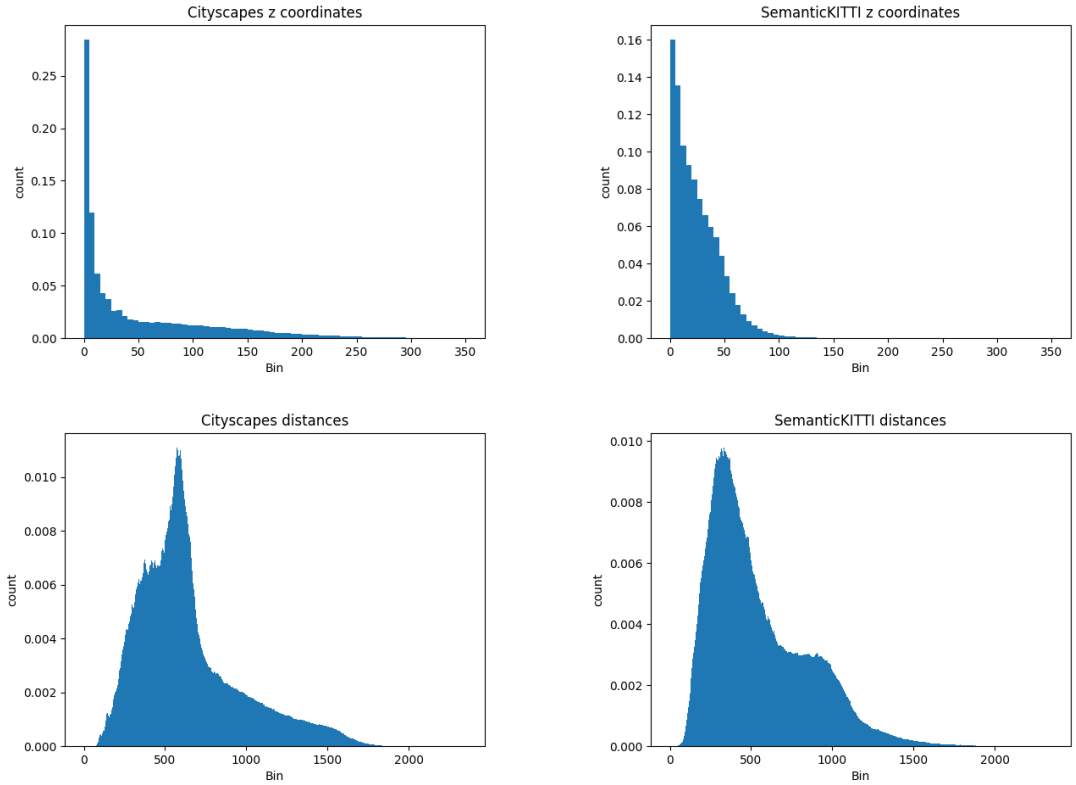
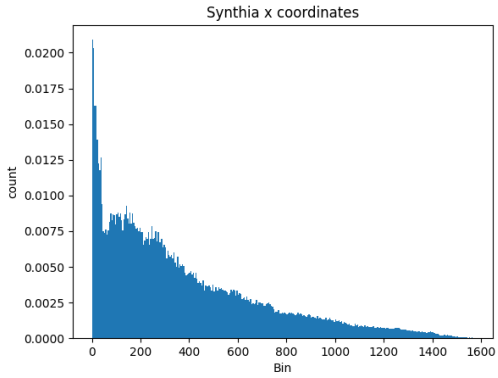
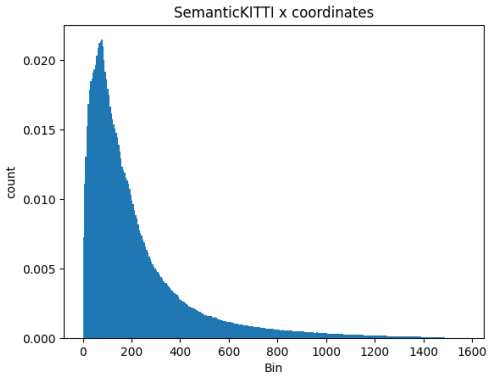
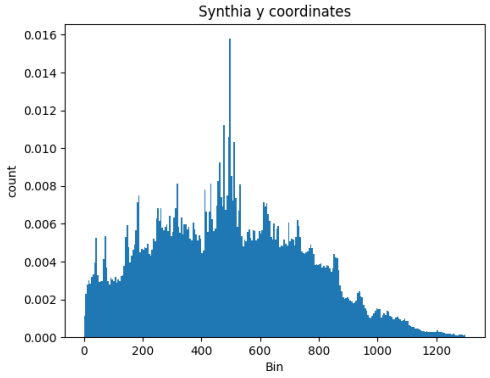
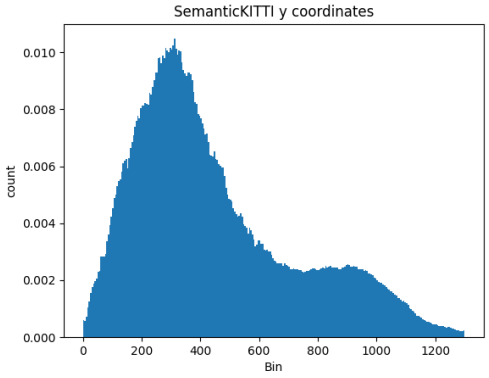


Figure 5.5: Comparison between the spatial coordinates and distances in Cityscapes (left column) and SemanticKITTI datasets (right column): these plots show how the prepress helps to obtain a similar PDF for the x,y,z coordinates and the PDF of the distances (computed as the norm of the x,y,z coordinates of the points in the point clouds).


Synthia x coordinates
count
Bin
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200
0
200
400
600
800
1000
1200
1400
1600


SemanticKITTI x coordinates
count
Bin
0.000
0.005
0.010
0.015
0.020
0
200
400
600
800
1000
1200
1400
1600


Synthia y coordinates
count
Bin
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
0
200
400
600
800
1000
1200


SemanticKITTI y coordinates
count
Bin
0.000
0.002
0.004
0.006
0.008
0.010
0
200
400
600
800
1000
1200

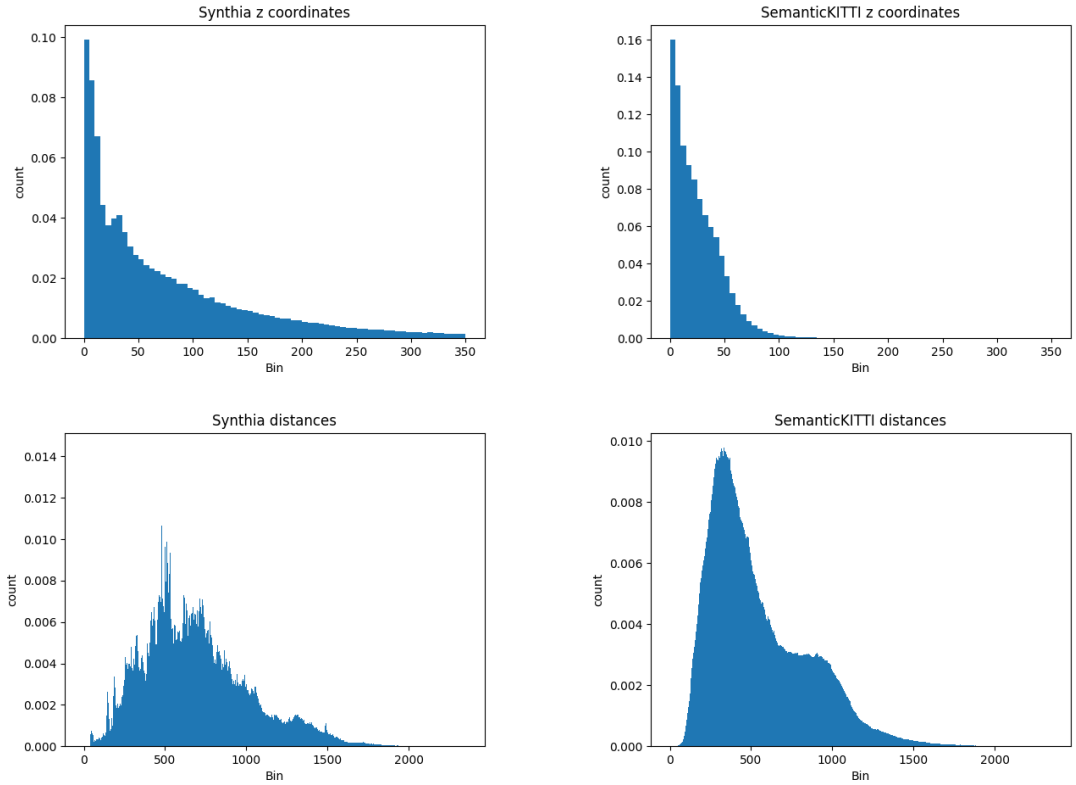


Figure 5.7: Comparison between the spatial coordinates and distances in Synthia (left column) and SemanticKITTI datasets (right column): these plots show how the prepress helps to obtain a similar PDF for the x,y,z coordinates and the PDF of the distances (computed as the norm of the x,y,z coordinates of the points in the point clouds).

5.4 Conversion of Point Cloud to Voxel Representation

The three-dimensional point cloud data has been transformed into voxels using a voxel size of 5 cm and a maximum voxel count of 4096 along each axis. This conversion was performed to simplify the spatial representation of the data and prepare it for further processing.
To achieve the voxelized representation, a scaling operation was applied to the points, where each measurement unit in the points corresponds to 5 cm in the voxels. Additionally, the entire point cloud underwent a translation so that the minimum coordinate value for each axis was shifted to the origin point (0,0,0). This translation was carried out to ensure proper alignment of the point cloud with the voxel grid.
The outcome is a voxelized representation of the point cloud, where the data is aligned with a regular grid of voxels, aiding in subsequent analysis and manipulations. This data preparation is essential for applications that require a structured and uniform representation of three-dimensional data, such as 3D image processing, machine learning, and other applications in the field of computer vision.

6

Models

In the field of computer vision, segmenting images to understand them better is crucial. This chapter dives deep into the segmentation design used in this research. We'll explore the main features of the network used, with a focus on why we chose to use an existing network and added a special feature called "sparse convolution" to the UNet structure. We'll also look at the challenges in creating a new network design and why certain choices were made.

6.1 Introduction to Neural Networks and Convolutional Neural Networks (CNNs)

In the vast landscape of signal processing and machine learning, artificial neural networks (ANNs) stand as one of the foundational pillars. These models, inspired by the structure of biological neural networks, provide a potent toolset for addressing a wide range of prediction and classification problems. At the heart of ANNs lies the perceptron, which performs a weighted sum of inputs x_i with weights w_i considering a specific number n of neurons, followed by an activation function f:

$$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right) \tag{6.1}$$

6.2 Convolutional Neural Networks (CNNs) and Convolution

Convolutional Neural Networks (CNNs) constitute a pivotal stage in the evolution of structured data analysis, particularly in the realm of images. The hallmark feature of CNNs is their ability to harness the convolution operation. This process entails the application of a filter (kernel) to the input, enabling the detection of local patterns. The output $C(i,j)$ is computed using the formula:

$$C(i,j) = \sum_{m=1}^{M} \sum_{n=1}^{N} X(i+m, j+n) \cdot W(m,n) \tag{6.2}$$

Where $C(i,j)$ represents the output element at position (i,j), X is the input, W is the filter, and M and N are the filter dimensions. This operation could highlight local features like edges and textures in images, but in general, permit to analysis the images, maintaining the spatial relation between the input pixels.

6.3 Stride and Padding

In the convolution operation, one can apply a stride S and padding P to influence the output dimension. Stride represents the step by which the filter moves across the input, while padding extends the area of the input (in most of the case cases, adding zeros) to maintain dimensions.

6.4 Formula for Output Calculation

The formula to compute the output dimension O of the convolution operation, accounting for stride and padding, is given by:

$$O = \frac{I - F + 2P}{S} + 1 \tag{6.3}$$

Where I is the input dimension, F is the filter dimension (kernel), P is the padding, and S is the stride. This formula helps determine the output dimension based on the stride and padding configurations used.

6.5 Role of Pooling Layers

In addition to the convolution operation, pooling layers play a significant role in CNNs. These layers focus on reducing the input dimensions, contributing to efficiency and control over overfitting. The pooling operation involves aggregating information within local regions, extracted through max pooling, which selects the maximum value within a pooling window. Pooling layers have several key roles:

1. **Dimension Reduction:** Gradually reduce input dimensions, simplifying the network and optimizing computation.
2. **Increase of the receptive field:** reduce input dimensions allows subsequent neurons to cover a larger area of the original input.
3. **Feature Extraction:** Highlight significant features from local regions, enabling the network to focus on relevant aspects of the image.
4. **Translation Invariance:** Enhance robustness to small translations in the input, enabling the network to recognize patterns regardless of their position.

6.6 Batch Normalization Layer

In addition to the convolution, pooling operations, and considerations of stride and padding, Batch Normalization (BatchNorm) layers play a fundamental role in neural network architectures. These layers normalize inputs for each layer within a mini-batch, calculating means μ and standard deviations σ:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \tag{6.4}$$

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2} \tag{6.5}$$

Normalized inputs $\hat{x}$ are then scaled and shifted using learnable parameters γ and β:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}} \tag{6.6}$$

$$y = \gamma\hat{x} + \beta \tag{6.7}$$

These parameters contribute to better fitting the model to the data and improving training stability.

The interplay between convolution, pooling operations, and the use of BatchNorm layers presents a powerful approach for image analysis through CNNs and other neural network architectures. This combination of techniques empowers neural networks to recognize complex patterns, reduce the risk of overfitting, and effectively adapt to data. By acknowledging the importance of each component, from convolution operations to data normalization, a solid foundation is established for creating neural models capable of learning meaningful representations and generating accurate predictions.

6.7 Common Regularization Techniques

Regularization techniques aim to strike a balance between fitting the training data well while preventing the model from becoming overly complex. Some common regularization methods include:

1. **L2 Regularization (Weight Decay):** This method involves adding a penalty term to the loss function based on the squared magnitudes of the weights. This encourages the model to use smaller weights, leading to a simpler model.
2. **Dropout:** Dropout involves randomly setting a fraction of the neurons' outputs to zero during training. This prevents the network from relying too heavily on any single neuron and encourages the network to learn more robust features.
3. **Early Stopping:** Early stopping involves monitoring the validation error during training and stopping the training process once the validation error starts to increase, thus preventing overfitting.
4. **Data Augmentation:** Data augmentation involves applying random transformations to the training data, such as rotations, flips, and translations. This artificially increases the size of the training dataset and improves the model's ability to generalize.

6.8 Skip Connections

Skip connections are a distinctive type of architectural design that helps mitigate the vanishing gradient problem and improve information flow in deep networks. They involve creating shortcut connections that bypass one or more layers in the network. The most famous example is the "residual block," introduced by ResNet. In a residual block, the

output of one layer is added to the output of a deeper layer, effectively creating a "shortcut" for the gradient during backpropagation. This addresses the vanishing gradient issue and enables easier training of deep networks. The formula for the skip connection is:

$$\text{Output} = \text{Input} + \mathcal{F}(\text{Input}) \tag{6.8}$$

Where $\mathcal{F}$ represents the residual mapping introduced by the skipped layers. This innovation has enabled the training of extremely deep networks with hundreds of layers, leading to substantial performance improvements. Skip connections not only facilitate training but also enable the network to learn hierarchical features effectively. These connections allow the network to retain and propagate essential information while learning incremental transformations. As a result, skip connections have become a fundamental component of modern architectures, contributing to their success in various tasks such as image classification, object detection, and semantic segmentation.

6.9 Sparse Convolution and Submanifold Sparse Convolution

In traditional neural networks based on classic convolution, the convolution operation is applied across the entire input. This approach can be computationally expensive and inefficient when dealing with sparse data or data with many irrelevant regions, such as 3D models where relevant information may be concentrated only in specific parts.
To address this issue, sparse convolution was introduced. In this case, a mask is used or it is assumed that empty regions have a value of zero in the input. The formula for sparse convolution is as follows:

$$\text{Output}(i,j,k) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\sum_{o=0}^{O-1} \text{Input}(i+m,j+n,k+o) \times \text{Kernel}(m,n,o) \times \text{Mask}(i+m,j+n,k+o) \tag{6.9}$$

where:

- $\text{Output}(i,j,k)$ is the value of the output feature map at position (i,j,k).
- $\text{Input}(i+m,j+n,k+o)$ is the input value at position $(i+m,j+n,k+o)$ in the feature map.
- $\text{Kernel}(m,n,o)$ is the weight (or filter coefficient) at position (m,n,o) in the convolutional kernel.

- Mask$(i+m, j+n, k+o)$ is the sparse mask, taking the value of 1 if the input at position $(i+m, j+n, k+o)$ is non-empty, and 0 otherwise.
- M, N and O are the dimensions of the convolutional kernel.

Sparse convolution computes the output only when the kernel is centered on a non-empty region, avoiding unnecessary computations on empty regions and making the operation more efficient.

6.9.1 Submanifold Sparse Convolution

The key innovation of Submanifold Sparse Convolution (SSC) [41] lies in its extension of the efficiency and accuracy of sparse convolution to 3D data by leveraging the intrinsic geometric properties of 3D data.
In the context of 3D data, SSC operates under the assumption that the data lies on a continuous submanifold within the 3D space. A submanifold is a lower-dimensional structure embedded within a higher-dimensional space, in this case, the 3D space. The fundamental idea behind Submanifold Sparse Convolution is to perform the convolution operation exclusively on this lower-dimensional submanifold rather than considering the entire 3D space.
Mathematically, Submanifold Sparse Convolution can be defined as follows:

$$\text{Output}(i,j,k) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{o=0}^{O-1} \text{Input}(i+m, j+n, k+o) \times \text{Kernel}(m,n,o) \quad (6.10)$$

The key distinction is in the way the input feature map is sampled. In Submanifold Sparse Convolution, the input is sampled solely from the points that lie on the submanifold, rather than considering all possible positions in the 3D space.
By focusing only on the points residing on the submanifold, Submanifold Sparse Convolution achieves a more targeted and effective computation, concentrating solely on the relevant parts of the 3D data. This preserves the geometric structures and relationships within the data, leading to enhanced performance in 3D semantic segmentation tasks, particularly when dealing with sparse 3D datasets.
In conclusion, Submanifold Sparse Convolution represents a significant advancement, as it efficiently handles 3D data by operating exclusively on the lower-dimensional submanifold, capitalizing on the geometric properties of the data and achieving state-of-the-art performance in 3D semantic segmentation tasks.

6.10 Challenges in Making a New Design

Designing a network for segmentation from scratch is a tough job. It requires a good understanding of how machines learn and how images work. It can be complicated and risky because every design choice affects how well the network works. Making the network work well takes a lot of time and resources.
In this specific thesis work, it is essential to address challenges related to handling multimodal data (3D + RGB) and tackle the issue of domain shift introduced by the presence of data from both Lidar sensors and depth maps. Hence, opting for a pre-designed segmentation network provides a robust foundation for applying UDA techniques between the source dataset and the target dataset.

6.11 Benefits of Using Ready-Made Networks

Using networks that are already trained, like the UNet, has some big advantages for segmentation. These networks have learned from many different pictures, so they know how to find important parts of images automatically. This means we don't have to start from scratch, which makes building the network faster. Even if we don't have a lot of pictures to train on, using a pre-trained network can still give us good results. These networks also start with some good settings, which helps them learn faster when we teach them new things.

6.12 Segmentation network

6.12.1 The UNet Structure: Combining Encoder and Decoder

The core of our segmentation network is the UNet architecture[42], which looks like a "U." This design brings together two essential parts: the encoder and the decoder. The encoder, located at the top of the "U," extracts complex details from the initial image using layers that do special calculations. This helps us capture important details of different sizes, so we understand both the overall picture and specific parts.

On the other hand, the decoder is responsible for putting together the segmented image. It does this by using special layers that work oppositely to the encoder. These layers help mix the details from the encoder and recreate the final segmented image. The "U" shape is chosen to make sure the encoder and decoder work well together, creating an accurate segmentation result that doesn't lose important details.

Feature Concatenation

A distinctive aspect of the UNet architecture is the use of feature map concatenation from different depths. During the up-sampling process in the decoder, feature maps are combined with those corresponding to the encoder, creating a fusion of details at various scales and contexts. This mechanism allows the network to maintain an awareness of lower-level details and higher-level semantic features, optimizing segmentation accuracy.

Activation Functions and Finalization

In each convolutional layer, the UNet architecture employs activation functions, often using the Rectified Linear Unit (ReLU), to introduce non-linearity in the processing. This helps capture the complex relationships present in the data and enhances the network's discrimination ability.
For the final layer, it's common to use an appropriate activation function based on the nature of the segmentation problem. For instance, in binary segmentation applications, the sigmoid function can be employed to produce an output between 0 and 1 for each pixel, representing the probability of belonging to the target class.

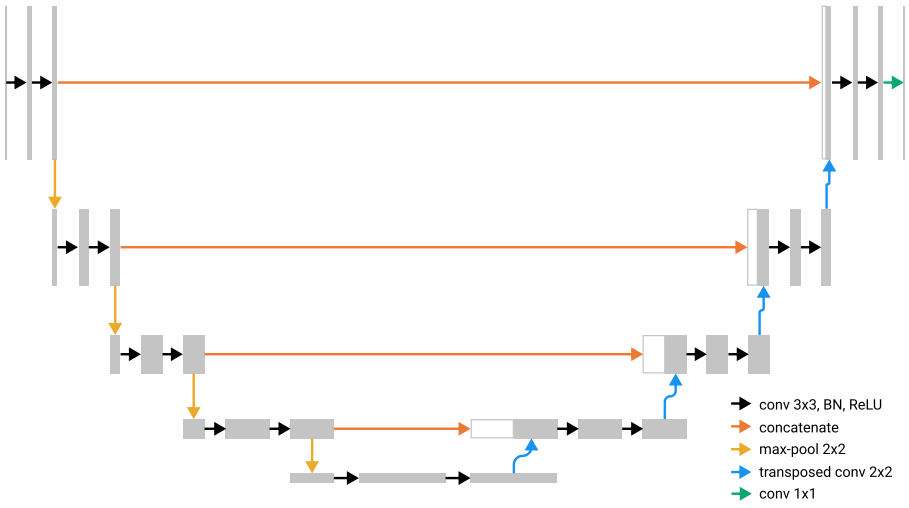


Figure 6.1: UNet architecture.

6.13 Features extraction network

Within the framework of the conducted research, the VGG16 (Simonyan et al.[43]) architecture has proven to be of paramount importance for accurate image analysis and for

linking the extracted features with their corresponding three-dimensional (3D) points. The VGG16 convolutional neural network, developed by Karen Simonyan and Andrew Zisserman, was selected due to its depth and strategic use of convolutional filters.
The primary focus of the thesis was the utilization of the initial two convolutional layers of the VGG16 network. These early layers are responsible for capturing fundamental features from images, such as edges, angles, and textures. This serves as a fundamental basis for subsequent analysis and interpretation.
The objective was to extract pertinent information from the original images and establish a connection between these features and their corresponding points in three-dimensional space. To facilitate this linkage, maintaining the output's (feature map) dimensions identical to the input's (original image) dimensions was essential.
To achieve this goal, a convolution technique with a stride of 1 was employed, ensuring a one-to-one mapping between the input and output pixels. This strategy preserved the image's dimensions during convolution, providing a direct correspondence between the extracted features and the corresponding 3D points in the real environment.
This approach had the purpose of effectively linking the features extracted from the images with the respective 3D points, thus paving the way for enhanced analysis and comprehension of three-dimensional scenes. The methodology grounded in the VGG16 architecture demonstrated its effectiveness in extracting significant information and establishing a bridge between the two-dimensional world of images and the three-dimensional realm of reality.

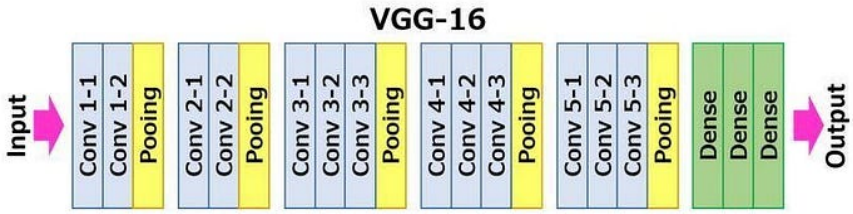


Figure 6.2: VGG16 architeture.

6.14 Intersection over Union (IoU) in Semantic Segmentation

IoU, or Intersection over Union, is a critical metric in the field of computer vision, especially for segmentation and detection tasks. This metric provides an accurate way to evaluate how well machine learning models are able to segment objects of interest in an image. Its significance arises from its ability to consider the spatial overlap between

model predictions and ground truth, overcoming the limitations of simpler metrics like accuracy.

The formula to calculate IoU is as follows:

$$IoU = \frac{Area_{\text{Intersection}}}{Area_{\text{Union}}}$$

Where:

- $Area_{\text{Intersection}}$ represents the area where model predictions and ground truth overlap.
- $Area_{\text{Union}}$ represents the total area covered by both model predictions and ground truth.

The result is a value ranging from 0 to 1, where 0 indicates no overlap and 1 indicates a perfect overlap between the segmented areas.

Let's consider an example where a segmentation model identifies a car in an image. Suppose the area segmented by the model and the ground truth area of the car have an intersection of 300 square pixels and a union area of 500 square pixels. The calculation of IoU would be as follows:

$$IoU = \frac{300}{500} = 0.6$$

In this case, the IoU would be 0.6, indicating that 60% of the segmented area coincides with the ground truth area.

The IoU's ability to evaluate the overlap between segmented areas makes it preferable over simple accuracy when assessing segmentation models. Accuracy could be misleading in situations where small details or objects are missed due to their size or position. IoU, on the other hand, considers the precision in object localization and penalizes incorrect or dispositioned predictions.

7

Expreriments and results

In this chapter, the procedure for selecting the optimal threshold will be explained, and a series of experiments conducted to evaluate the impact of integrating RGB image information into semantic segmentation processes will be presented. This section is divided into two main parts: the first part describes the experiments conducted using only 3D information without the inclusion of RGB images, and then introduces RGB visual information demonstrating how it has improved performance in both Oracle training and source-only scenarios. In the second part, test time adaptation is applied to the selected model to be adapted to the target dataset.

To conduct these experiments, two different source datasets were utilized. Initially, reliance was placed on a real-world dataset, Cityscapes, to assess the model's performance in an authentic urban context. Subsequently, a synthetic dataset, Synthia, was introduced to evaluate the model's robustness in virtual scenarios. In both cases, the ultimate goal was an adaptation on target data from SemanticKITTI.

7.1 Threshold selection

During the initial experiment, the primary aim was to determine the most suitable threshold value for producing a dataset with LIDAR-like characteristics using Cityscapes images (see section 5.3).
To tackle this challenge, the approach used the Intersection over Union (IoU) calculation in the SemanticKITTI dataset, which was the reference. We examined several threshold values, such as 0.0005, 0.00025, 0.0001, 0.000075, and 0.00005. Our primary objective was to identify the threshold that maximized the IoU value on the target dataset. To achieve this, we trained five models using five source datasets (Cityscapes)

created with these thresholds. These models were then tested on the target dataset to evaluate their performance.
It's important to mention that we chose the threshold directly by considering data that includes RGB information because our main focus in this study is to use data that combines different types of information.

7.2 Semantic Segmentation

7.2.1 Unimodal segmentation: 3D only

Utilizing the dataset generated with the designated threshold and training the network solely with 3D point cloud data, a foundational benchmark was established. This approach provided a basis for evaluating the benefits derived from the integration of various data sources, encompassing both 3D and RGB modalities.
Since the network utilizes the 3D information only to capture the spatial relationships between points and the absolute position of each point is not directly utilized in the convolution operations, a dummy feature was introduced for all points, in particular, following the methodology employed by the authors of xMUDA [24], a value of 1 was assigned to all points.

7.2.2 Multimodal segmentation: 3D + RGB

Incorporating RGB data into the semantic segmentation process has played a crucial role in this study, leading to significant enhancements in various aspects of model performance. Notably, the use of RGB information has not only resulted in small but meaningful improvements in mean Intersection over Union (mIoU) scores during the Oracle training, but it has also proven effective in boosting the model's performance on both the source and target datasets in source-only runs.

The inclusion of RGB data represents a significant advancement in multimodal segmentation. It allows the model to leverage the rich visual information found in RGB images, complementing the 3D data. This integration of modalities facilitates a more comprehensive understanding of the environment, resulting in improved segmentation accuracy.

7.2.3 Features extracted using VGG16

An additional experiment was conducted involving the utilization of features extracted from the first and second layers of the VGG16 network. The results of these experi-

ments reveal that, although these features did not surpass the results achieved using RGB information directly, this technique nonetheless led to an enhancement in model performance.

Specifically, two experiments were carried out for each of the aforementioned layers. In the first experiment, a pre-trained network available in the PyTorch library was employed, without updating the weights of the layers. In the second experiment, the layers were updated, starting from the pre-trained ones, during the training to adapt feature extraction to the source dataset used for training.

While the improvements observed using VGG16 features were not as substantial as those achieved with RGB data, the potential of leveraging feature extraction models for feature extraction was demonstrated.

In this case, the images used underwent the same preprocessing steps required by VGG16. They were normalized to the range of 0-1, followed by subtracting the mean of ImageNet [44] and dividing by its standard deviation, with the following values for each RGB channel:

- Mean (R, G, B): (0.485, 0.456, 0.406)
- Standard Deviation (R, G, B): (0.229, 0.224, 0.225)

7.3 Test time adaptation

In the context of this thesis work, three distinct scenarios have been considered for test time adaptation. The choice was made to employ the model trained with the features yielding the highest mIoU on the target dataset. In the case of the source dataset being Cityscapes, TTA was performed using the raw RGB data.

7.3.1 Case 1 (BN): Forward Pass Without Backward Loss

In the first case, a straightforward forward pass was implemented without computing the gradient of the loss function concerning the network's parameters. The focus was solely on updating the parameters of the batch normalization layers. This approach aims to maintain data distribution stability through batch normalization, enhancing the model's generalization capability during testing.

7.3.2 Case 2 (BN + EM): Using Shannon Entropy Loss with Backward on Batch Normalization

In the second case, the utilization of Shannon entropy loss during test time adaptation was introduced, following the approach proposed by the authors of TENT [30]. This loss function was employed to minimize model uncertainty in an unsupervised fashion. However, gradient computation was performed exclusively concerning the batch normalization parameters. This approach was designed to optimize class probability calibration, enabling the model to express uncertainty in predictions.

7.3.3 Case 3 (EM): Backward of Shannon Entropy Loss with Full Weight Update

In the third case, a more aggressive strategy was adopted, where the gradient computation of Shannon entropy loss was executed across all neural network weights. This encompassed both the batch normalization layer and convolutional layer weights during test time adaptation. The aim was to maximize prediction accuracy during testing, allowing the model to fully adapt to the specifics of the target input.

7.3.4 Fine-Tuning and Validation

Furthermore, to assess the effectiveness of the various test time adaptation strategies, fine-tuning processes were conducted. During fine-tuning, cross-entropy loss was used in conjunction with ground truth target data to retrain the model. Initially, fine-tuning involved updating all network weights. Subsequently, a second phase of fine-tuning was carried out, limiting weight updates to only the batch normalization layers. This approach facilitated an evaluation of the specific role of batch normalization in the model adaptation process.

7.4 Threshold selection results

The experiment demonstrated that the best threshold value was 0.00025 (Table 7.1). This showed that this specific threshold effectively optimized the alignment between the generated data and the SemanticKITTI dataset.

Table 7.1: Horizontal Channel Error ablation: the error defines how far from the "theoretical" angle, a point is considered valid (i.e. the smaller the margin the fewer are the valid points).

Ho.-Channel Error	Steps (best/final)	Train (best/final)	Val (best/final)	Test	Target
5e-4	71k/80k	57.58/57.80	56.20/55.80	52.65	22.28
2.5e-4	67k/80k	56.88/55.83	55.41/55.07	52.08	**22.38**
1e-4	77k/80k	52.05/51.91	51.19/49.42	47.28	21.60
7.5e-5	78k/80k	50.36/49.73	48.87/50.06	46.54	22.36
5e-5	78k/80k	47.92/47.73	47.33/47.56	44.34	19.94

7.5 Real dataset results

Table 7.2 reports the IoU values for all the classes and the final mIoU values for all the experiments. In conclusion, when using the Cityscapes dataset as the target, it has been observed that the most significant improvement is achieved using raw RGB data (Table 7.3). During the Test Time Adaptation (TTA) phase, all three cases yield similar results, both with a substantial increase of about 8-9% in the target data and the values are not too far from the fine-tuning results (Table 7.2), which can be considered the maximum value we can reach using the entropy minimization technique on this model. It is worth noting that, to ensure greater stability of the model, especially in the source data, the preferable solution is undoubtedly to update only the batch normalization layers as shown in Table 7.4.

Some examples of comparison in the outputs between ground truth and Oracle training and between source-only and TTA are provided in Figure 7.1 and Figure 7.2.

Table 7.2: Adaptation Cityscapes → SemanticKITTI. FT = fine-tuning, BN = batch norm., EM = entropy minimization.

Method	Mode	road	side.	build.	nature	struct.	pole	sign	pers.	vehic.	2-wheel	obj.	mIoU
Oracle	3D	94.05	82.93	92.11	88.94	53.33	58.88	44.24	34.06	93.91	51.65	30.48	65.87
Oracle	3D+RGB	93.84	82.25	92.78	90.59	58.27	60.04	45.16	36.71	93.98	47.72	27.85	66.29
Oracle	3D+Feats1(PT)	94.10	83.00	92.48	89.59	53.36	59.26	44.50	37.35	94.37	53.18	31.62	66.62
Oracle	3D+Feats1(T)	93.85	82.55	92.37	89.88	54.72	58.40	43.28	34.03	94.16	52.24	32.70	66.20
Oracle	3D+Feats2(PT)	93.89	82.59	92.95	89.72	54.64	60.02	45.21	33.50	94.24	48.09	30.82	65.97
Oracle	3D+Feats2(T)	94.48	83.74	92.53	90.14	55.25	58.65	47.86	41.98	94.26	49.62	33.10	67.42
Source-only	3D	16.74	10.29	36.99	19.24	12.38	12.69	3.50	13.02	29.78	7.19	3.42	15.02
Source-only	3D+RGB	67.81	29.77	19.07	27.98	11.04	18.29	11.30	3.44	41.83	5.64	10.03	**22.38**
Source-only	3D+Feats1(PT)	58.28	23.87	27.07	13.29	11.14	15.47	4.20	1.58	27.64	3.20	5.52	17.39
Source-only	3D+Feats1(T)	64.04	9.90	22.38	22.31	4.68	20.55	4.57	1.19	31.42	2.49	5.12	17.17
Source-only	3D+Feats2(PT)	61.46	17.78	18.45	20.85	11.53	19.73	5.97	1.96	26.97	3.28	4.35	17.47
Source-only	3D+Feats2(T)	62.53	21.61	30.65	41.22	12.06	25.47	4.12	1.43	35.99	1.51	9.35	22.36
FT	3D+RGB	93.51	81.59	92.68	90.86	56.96	59.91	41.20	42.62	94.60	58.47	31.09	67.59
FT (only BN)	3D+RGB	74.57	38.80	59.42	68.99	21.82	27.09	15.56	8.91	70.71	4.21	2.35	35.68
BN update	3D+RGB	75.40	40.67	49.39	66.26	24.92	19.67	2.31	4.91	58.97	2.52	4.86	31.81
BN update + EM	3D+RGB	74.43	39.77	38.73	50.88	23.70	30.19	6.10	7.24	55.83	2.15	4.88	30.36
EM	3D+RGB	76.71	33.52	75.39	69.50	2.20	2.56	0.70	11.14	78.49	0.00	0.52	31.89

Table 7.3: Ablation on different RGB feature integrations. PT = pre-trained, T = trained.(Cityscapes-Semantickitti)

	Mode	mIoU Source	mIoU Target
Plain	3D	45.09	15.02
	3D+RGB	52.08	**22.38**
Layer 1	PT	51.20	17.39
	T	53.89	17.17
Layer 2	PT	51.87	17.47
	T	55.31	22.36

Table 7.4: Comparison of the UDA approaches in source and target datasets (Cityscapes → SemanticKITTI, 3D+ RGB)

UDA method	mIoU Source	mIoU Target
BN	42.18	31.81
BN+EM	47.49	30.36
EM	28.93	31.89

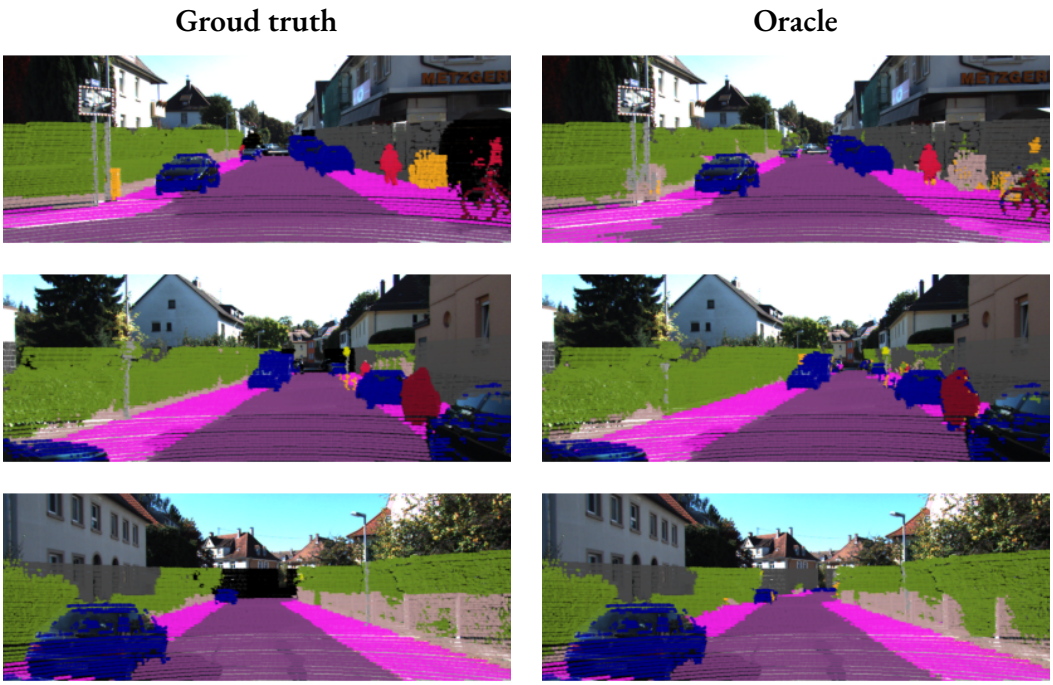


Figure 7.1: Comparison between ground truth and the oracle training in the semanticKITTI dataset with the raw RGB information.

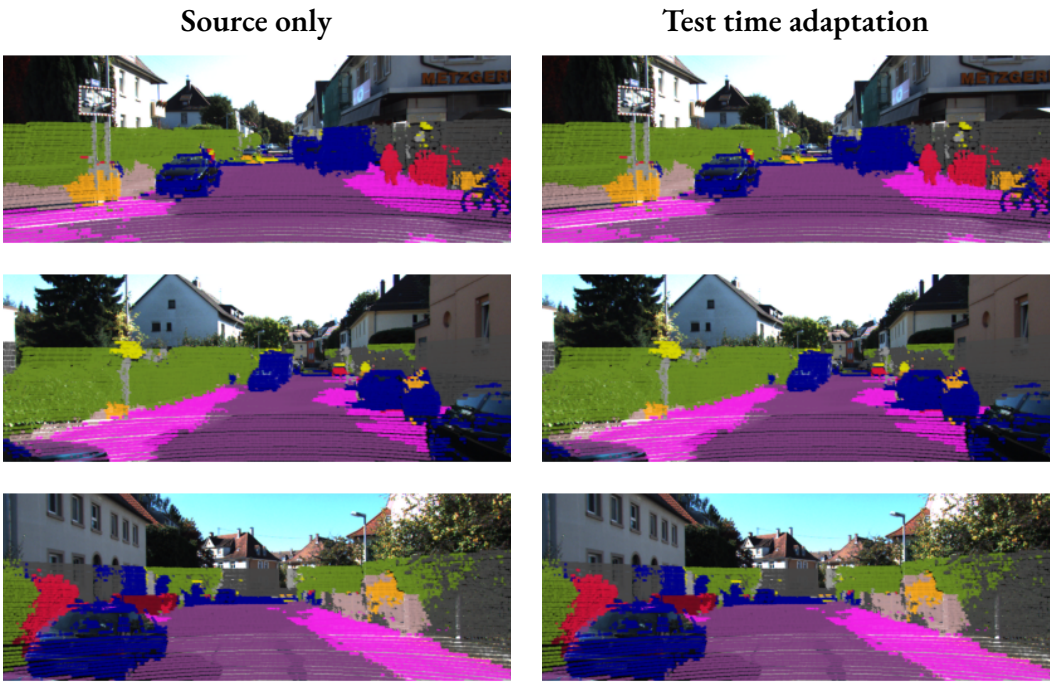


Figure 7.2: Comparison between source only and TTA using Cityscapes dataset as source and semanticKITTI dataset as target with the raw RGB information.

7.6 Syntethic dataset results

The same procedures were applied using a synthetic dataset, Synthia, generated with the same threshold (0.00025) used for the Cityscapes dataset. In this case, the best features were found to be those derived from the first layer of the VGG16 network without updating its respective weights during training, excluding cases where the data contained only 3D information. It is noteworthy that, even in this scenario, the use of RGB information contributes to the improvement of mIoU in the source dataset (Table 7.6).

As for testing on target dataset (without any form of test-time adaptation), it is highly likely that due to the significant domain shift present in RGB images, which is not compensated for by the method utilizing the threshold to generate LIDAR-like data, this may result in inferior performance. Infact, looking at the Table 7.6, the results mIoU on the source-only training with the integration of the RGB information, leads to lowers values w.r.t. the result obtained with the 3D only information.

This is further confirmed by the fact that when applying Test Time Adaptation (TTA) techniques, the performance improvements are found to be quite limited compared to the "source-only" results. Specifically, it has been observed that by applying TTA to both the best-identified feature (i.e., the first layer of the VGG16 network) and the raw RGB information, the performance gains were modest. Considering also the fine-tuning case (Table 7.5) the results are lower w.r.t. the results obtained in the adaptation Cityscapes $\rightarrow$ SemanticKITTI (Table 7.2) confirming the fact that the domain shift in this case is higher.

In this case, even if the results are limited, it can be observed that the TTA applied to the model trained with the best features had a performance increase bigger than the model trained with the raw RGB data. In particular, in the first case, the gain in mIoU is equal to $26.87 - 25.40 = 1.4\%$, while in the second case is $25.54 - 24.47 = 1.07$ (Table 7.5). This showed how the utilization of the VGG16 features is a good starting point to increase the model performance on target data and reduce the domain shift in the RGB space.

Moreover, the results in the TTA confirmed the fact that the best way to proceed is to update only the batchnorm layers during the training, since, updating the entire network reduces the performance of the model in both source and target datasets (Table 7.7).

These findings highlight the complexity of the challenge posed by the synthetic domain in contrast to the real LIDAR world, suggesting that the intrinsic differences between synthetic RGB-D data and real LIDAR-RGB data may require additional domain adaptation efforts to achieve better performance on the target dataset.

Some examples of comparison in the outputs between ground truth and oracle training and between source-only and TTA are provided in Figure 7.3 and Figure 7.4.

Table 7.5: Adaptation Synthia → SemanticKITTI. FT = fine-tuning, BN = batch norm., EM = entropy minimization.

Method	Mode	road	side.	build.	nature	struct.	pole	sign	pers.	vehic.	2-wheel	obj.	mIoU
Oracle	3D	94.05	82.93	92.11	88.94	53.33	58.88	44.24	34.06	93.91	51.65	30.48	65.87
Oracle	3D+RGB	93.84	82.25	92.78	90.59	58.27	60.04	45.16	36.71	93.98	47.72	27.85	66.29
Oracle	3D+Feats1(PT)	94.10	83.00	92.48	89.59	53.36	59.26	44.50	37.35	94.37	53.18	31.62	66.62
Oracle	3D+Feats1(T)	93.85	82.55	92.37	89.88	54.72	58.40	43.28	34.03	94.16	52.24	32.70	66.20
Oracle	3D+Feats2(PT)	93.89	82.59	92.95	89.72	54.64	60.02	45.21	33.50	94.24	48.09	30.82	65.97
Oracle	3D+Feats2(T)	94.48	83.74	92.53	90.14	55.25	58.65	47.86	41.98	94.26	49.62	33.10	67.42
Source-only	3D	60.94	32.73	16.71	71.74	2.06	31.22	5.35	12.43	43.76	24.64	0.06	27.42
Source-only	3D+RGB	61.42	22.84	17.11	61.38	5.72	55.03	9.74	8.68	39.96	13.16	0.06	24.47
Source-only	3D+Feats1(PT)	52.37	22.80	21.73	71.50	2.60	35.85	6.85	7.52	43.39	14.79	0.02	**25.40**
Source-only	3D+Feats1(T)	60.65	21.43	18.28	69.09	1.37	34.89	8.34	4.64	39.48	13.83	0.06	24.73
Source-only	3D+Feats2(PT)	60.07	18.39	9.61	66.05	0.89	33.32	3.71	13.99	39.96	13.22	0.12	23.58
Source-only	3D+Feats2(T)	42.27	23.17	33.41	67.74	1.21	31.29	6.87	6.86	33.28	13.17	0.13	23.58
FT	3D+Feats1(PT)	93.18	80.85	92.60	89.66	53.80	58.29	47.66	33.21	93.84	40.67	31.50	65.02
FT (only BN)	3D+Feats1(PT)	71.93	23.02	47.66	68.94	12.68	30.43	12.33	14.17	60.38	4.94	1.75	31.66
BN update	3D+Feats1(PT)	64.07	28.29	31.15	69.29	4.06	30.80	7.14	5.20	43.24	3.84	0.12	26.11
BN update + EM	3D+Feats1(PT)	67.00	27.51	30.85	69.83	1.56	32.77	9.45	6.94	44.45	5.19	0.01	26.87
EM	3D+Feats1(PT)	71.40	2.21	30.37	47.84	0.00	0.03	0.00	6.40	24.78	0.07	0.00	16.65
BN update	3D+RGB	58.82	25.13	39.77	63.33	6.95	24.00	2.08	3.79	52.76	2.17	0.06	25.35
BN update + EM	3D+RGB	63.39	26.14	25.97	61.24	4.43	31.97	8.81	8.21	46.68	4.04	0.00	25.54
EM	3D+RGB	67.76	4.81	48.61	51.40	0.00	0.72	0.16	3.66	48.65	0.00	0.00	20.53

Table 7.6: Ablation on different RGB feature integrations. PT = pre-trained, T = trained.(Synthia-Semantickitti)

	Mode	mIoU Source	mIoU Target
Plain	3D	63.63	27.42
	3D+RGB	66.35	24.47
Layer 1	PT	65.50	**25.40**
	T	67.01	24.73
Layer 2	PT	66.75	23.58
	T	68.70	23.58

Table 7.7: Comparison of the UDA approaches in source and target datasets (Synthia → SemanticKITTI, 3D+Feats1(PT))

UDA method	mIoU Source	mIoU Target
BN	58.05	26.11
BN+EM	54.71	26.87
EM	26.89	16.65

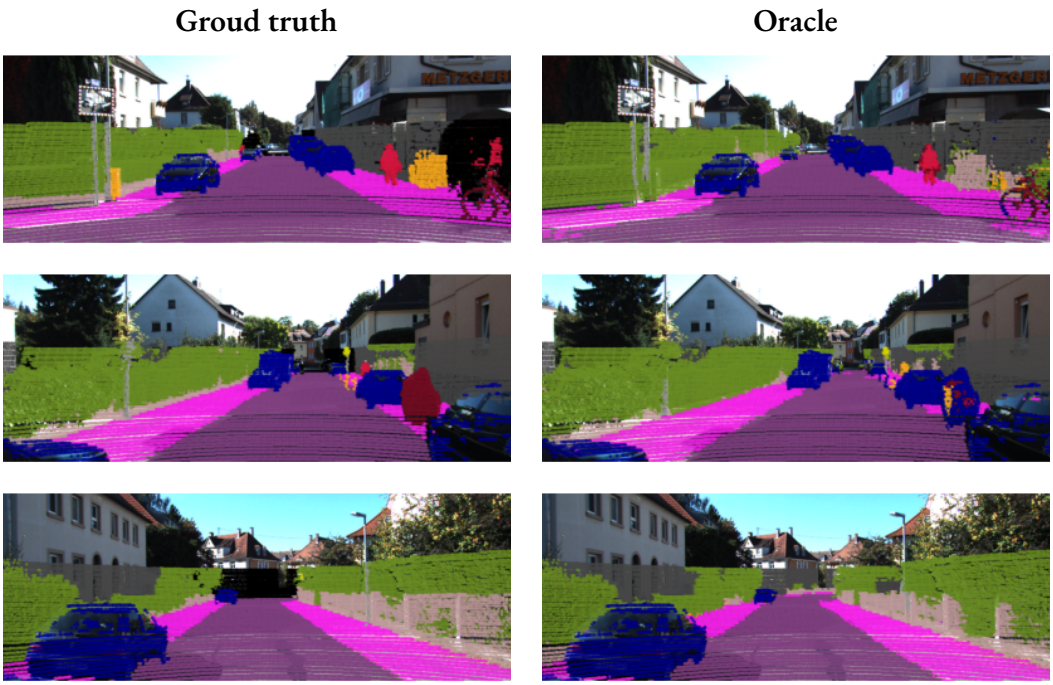


Figure 7.3: Comparison between ground truth and the oracle training in the semanticKITTI dataset with the features extracted from the first layer of VGG16 without training.

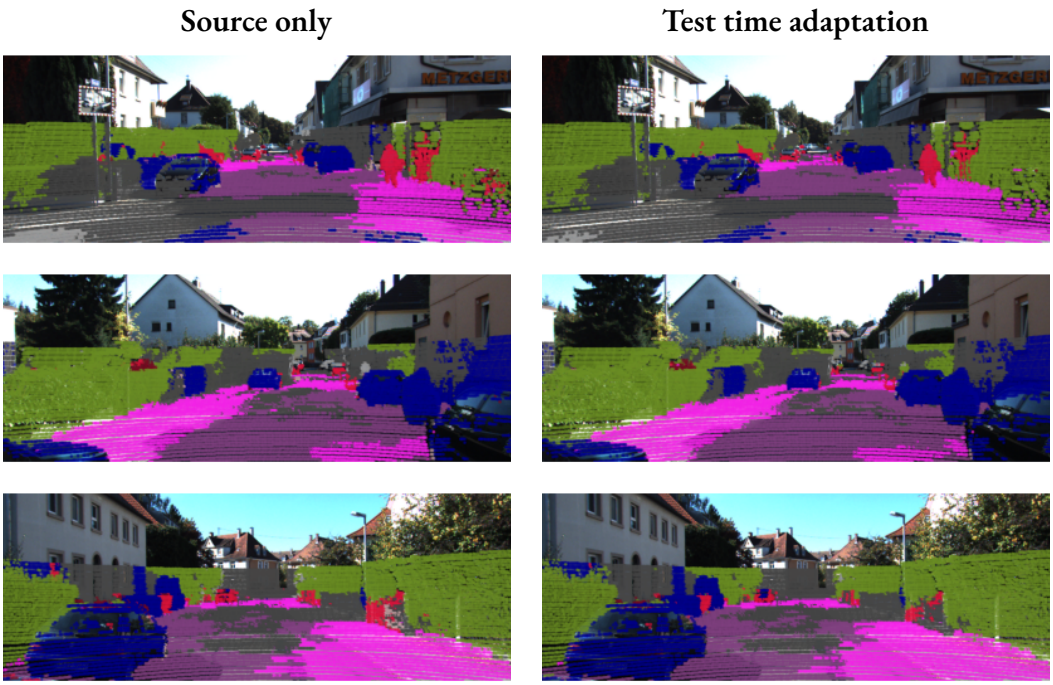


Figure 7.4: Comparison between source only and TTA using Synthia dataset as source and semanticKITTI dataset as target with the raw RGB information.

8

Conclusion

In the research conducted, we explored the integration of various multimodal data sources, such as RGB and 3D point clouds, alongside the application of Unsupervised Domain Adaptation (UDA) techniques to enhance the accuracy of semantic segmentation models on target dataset. This approach proves particularly advantageous when dealing with complex issues that may make the acquisition of labeled data challenging.
The objectives were achieved through a series of experiments employing two distinct datasets: Cityscapes, a real-world dataset representing an authentic urban context, and Synthia, a synthetic dataset that allowed for the evaluation of the model's robustness in virtual scenarios. Both datasets served as the foundation for assessing the effectiveness of the deep-learning techniques employed in the research.
One of the primary steps in the framework involved data preprocessing. RGB images and LiDAR-like data from various sources, such as Cityscapes as the source dataset and SemanticKITTI as the target dataset, were subjected to preprocessing to standardize their data formats, ensuring compatibility. This phase encompassed the transformation of images into LiDAR-like data, thereby ensuring uniform representation of information. In particular, the proposed preprocessing proved to be effective in establishing an initial alignment between the source data and the target data PDFs.
In the subsequent phase, preliminary training of the neural network was initiated using exclusively the 3D data from the source dataset. Subsequently, RGB information was incorporated into the segmentation process. Initially, raw RGB data was utilized to assess performance improvements. Furthermore, additional preprocessing was applied to RGB images, and feature extraction was performed using the first two layers of the VGG16 network. Depending on the context, this feature extraction could be executed with or without further training of the VGG16 layers.
To establish an effective link between the features extracted from images and their corre-

sponding 3D points, a convolution technique with a stride of 1 was employed, guaranteeing a one-to-one mapping between input and output pixels. This approach preserved the dimensions of the images during convolution, facilitating a direct correspondence between the extracted features and the corresponding 3D points in the real environment. The methodology rooted in the VGG16 architecture proved to be effective in extracting meaningful information and bridging the gap between the two-dimensional world of images and the three-dimensional real world.
It was observed that the utilization of multimodal data sources, such as RGB and 3D point clouds, not only elevates the precision of the segmentation process within the source dataset but also extends these gains to the target dataset. Moreover, an evaluation of the model's performance in scenarios where there is no labeled data highlights the potential of Unsupervised Domain Adaptation (UDA) in adapting segmentation models to new domains.

To sum up, the results obtained, although limited, demonstrate that the proposed preprocessing and feature extraction technique serves as a good starting point to mitigate the domain shift between data with different distributions and containing distinct information (LIDAR vs Depth and Synthetic vs Real). This lays a strong foundation for the development of more advanced domain adaptation techniques, particularly in the context of test time adaptation.

References

[1] A. Anand, H. S. Koppula, T. Joachims, and A. Saxena, "Contextually guided semantic labeling and search for three-dimensional point clouds," *The International Journal of Robotics Research*, vol. 32, no. 1, pp. 19–34, 2013.

[2] D. Wolf, J. Prankl, and M. Vincze, "Fast semantic segmentation of 3d point clouds using a dense crf with learned parameters," in *2015 IEEE International conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 4867–4873.

[3] A. Golovinskiy, V. G. Kim, and T. Funkhouser, "Shape-based recognition of 3d point clouds in urban environments," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 2154–2161.

[4] M. Weinmann, B. Jutzi, S. Hinz, and C. Mallet, "Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 286–304, 2015.

[5] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert, "Contextual classification with functional max-margin markov networks," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 975–982.

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[7] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[8] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3075–3084.

[9] Z. Zeng, Y. Xu, Z. Xie, J. Wan, W. Wu, and W. Dai, "Rg-gcn: A random graph based on graph convolution network for point cloud semantic segmentation," *Remote Sensing*, vol. 14, no. 16, p. 4055, 2022.

[10] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 108–11 117.

[11] X. Yan, J. Gao, C. Zheng, C. Zheng, R. Zhang, S. Cui, and Z. Li, "2dpass: 2d priors assisted semantic segmentation on lidar point clouds," in *European Conference on Computer Vision*. Springer, 2022, pp. 677–695.

[12] J. Li, H. Dai, H. Han, and Y. Ding, "Mseg3d: Multi-modal 3d semantic segmentation for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 694–21 704.

[13] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 681–687.

[14] Q.-H. Pham, T. Nguyen, B.-S. Hua, G. Roig, and S.-K. Yeung, "Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[15] Q. Ha, K. Watanabe, T. Karasawa, Y. Ushiku, and T. Harada, "Mfnet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5108–5115.

[16] J. Zhang, R. Liu, H. Shi, K. Yang, S. Reiß, K. Peng, H. Fu, K. Wang, and R. Stiefelhagen, "Delivering arbitrary-modal semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 1136–1147.

[17] F. Barbato, G. Rizzoli, and P. Zanuttigh, "Depthformer: Multimodal positional encodings and cross-input attention for transformer-based segmentation networks," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[18] G. Rizzoli, F. Barbato, and P. Zanuttigh, "Multimodal semantic segmentation in autonomous driving: A review of current approaches and future perspectives," *Technologies*, vol. 10, no. 4, p. 90, 2022.

[19] H. Luo, K. Khoshelham, L. Fang, and C. Chen, "Unsupervised scene adaptation for semantic segmentation of urban mobile laser scanning point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 169, pp. 253–267, 2020.

[20] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.

[21] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.

[22] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to adapt structured output space for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7472–7481.

[23] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 4376–4382.

[24] M. Jaritz, T.-H. Vu, R. d. Charette, E. Wirbel, and P. Pérez, "xmuda: Cross-modal unsupervised domain adaptation for 3d semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 12 605–12 614.

[25] B. Zhang, Z. Wang, Y. Ling, Y. Guan, S. Zhang, and W. Li, "Mx2m: Masked cross-modality modeling in domain adaptation for 3d semantic segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 3, 2023, pp. 3401–3409.

[26] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6

[27] G. Rizzoli, D. Shenaj, and P. Zanuttigh, "Source-free domain adaptation for rgb-d semantic segmentation with vision transformers," *arXiv preprint arXiv:2305.14269*, 2023.

[28] B. Xing, X. Ying, R. Wang, J. Yang, and T. Chen, "Cross-modal contrastive learning for domain adaptation in 3d semantic segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 3, 2023, pp. 2974–2982.

[29] M. Toldo, A. Maracani, U. Michieli, and P. Zanuttigh, "Unsupervised domain adaptation in semantic segmentation: a review," *Technologies*, vol. 8, no. 2, p. 35, 2020.

[30] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," *arXiv preprint arXiv:2006.10726*, 2020.

[31] J. Song, J. Lee, I. S. Kweon, and S. Choi, "Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 11 920–11 929.

[32] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," *Advances in neural information processing systems*, vol. 29, 2016.

[33] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[34] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3723–3732.

[35] K. Zhou, Y. Zhang, Y. Zang, J. Yang, C. C. Loy, and Z. Liu, "On-device domain generalization," *arXiv preprint arXiv:2209.07521*, 2022.

[36] Y. Zhang, Y. Wei, Z. Ji, J. Bai, W. Zuo *et al.*, "Towards diverse and faithful one-shot adaption of generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 37 297–37 308, 2022.

[37] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, "Semi-supervised domain adaptation via minimax entropy," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 8050–8058.

[38] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[39] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3234–3243.

[40] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "A dataset for semantic segmentation of point cloud sequences," *arXiv preprint arXiv:1904.01416*, vol. 2, no. 3, p. 12, 2019.

[41] B. Graham, M. Engelcke, and L. Van Der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9224–9232.

[42] X.-X. Yin, L. Sun, Y. Fu, R. Lu, Y. Zhang *et al.*, "U-net-based medical image segmentation," *Journal of Healthcare Engineering*, vol. 2022, 2022.

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[44] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

Acknowledgments

Italiano:
Vorrei ringraziare il mio relatore, Professore Pietro Zanuttigh, e i corrrelatori, Dot. Francesco Barbato e Dot.ssa Giulia Rizzoli, per avermi seguito nella realizzazione di questa tesi, fornendomi il supporto per permettermi di apprendere le conoscenze teoriche e tecniche necessarie.

Vorrei ringraziare tutta la mia famiglia, per il grande supporto che in tutti questi anni sono riusciti a fornirmi, aiuto che mi ha permesso di superare tutte le difficoltà nonostante a volte l'ansia mi portava a credere di non farcela.

Vorrei ringraziare anche la mia ragazza, Serena, per essermi stata vicina sempre nonostante i momenti difficili. Non è facile immaginare come sarebbe stato senza di lei ma, sicuramente, tutto questo percorso sarebbe stato molto più difficile.

Come promesso ringrazio la mia amica Veronica, che nell'ultimo anno mi ha sempre fatto compagnia durante i viaggi in treno e mi ha sempre dimostrato il suo grandissimo incorragiamento.

Ringrazio tutti i miei amici/colleghi dell'università (impossibile nominarli tutti) per aver reso tutte le giornate di studio più allegre e leggere grazie alla loro presenza. Tra tutti vorrei ringraziare Riccardo, che mi supporta (e sopporta) ormai dal nostro primo anno e nonostante la distanza trova sempre il tempo per passare del tempo con me.

Infine ringrazio me stesso, per essere riuscito ad arrivare fino a qui, orgoglioso dei risultati ottenuti ed essere riuscito a dimostrare a me stesso di poter raggiungere quello in cui credo.

English:
I would like to thank my thesis advisor, Professor Pietro Zanuttigh, and my co-advisors, Dr. Francesco Barbato and Dr. Giulia Rizzoli, for guiding me in the completion of this thesis and providing me with support to acquire the necessary theoretical and technical knowledge.

I want to express my gratitude to my entire family for the immense support they have provided me over the years, which helped me overcome all the difficulties, even when anxiety sometimes made me doubt my abilities.

I also want to thank my girlfriend, Serena, for always being by my side, even during challenging times. It's hard to imagine how it would have been without her, but undoubtedly, this journey would have been much more difficult.

As promised, I thank my friend Veronica, who kept me company during train journeys in the past year and showed unwavering encouragement.

I appreciate all my university friends and colleagues (impossible to name them all) for making all the study days more cheerful and lighter with their presence. Among them, I would like to thank Riccardo, who has been supporting (and putting up with) me since our first year, and despite the distance, always finds time to spend with me.

Lastly, I thank myself, for having managed to come this far, proud of the results achieved, and for having proven to myself that I can reach what I believe in.