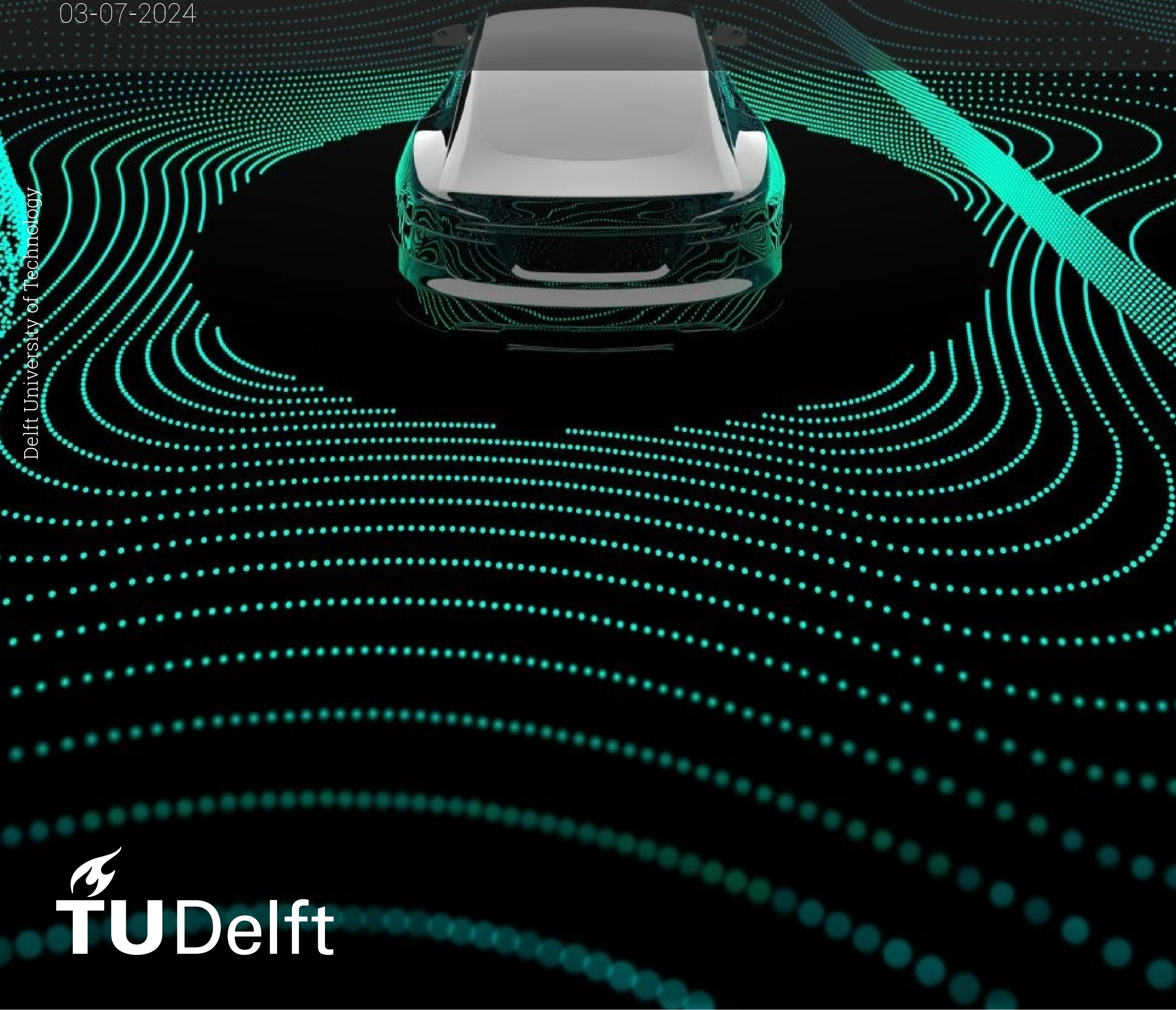
MultiViT: 2D to 3D transfer learning using a jointly optimized Vision Transformer without the need for image labels

Tibbe Lukkassen

Supervisors: Yancong Lin, Holger Caesar

03-07-2024



Delft University of Technology

TU Delft

MultiViT: 2D to 3D transfer learning using a jointly optimized Vision Transformer without the need for image labels

by

Tibbe Lukkassen
TU Delft

Tibbe Lukkassen 4662210

Supervisor:	Holger Caesar
Daily supervisor:	Yancong Lin
Faculty:	Mechanical Engineering, Delft University of Technology
MSc programme:	Robotics

Cover: https://www.darpa.mil/ddm_gallery/assured-autonomy.png

TU Delft

MultiViT: 2D to 3D transfer learning using a jointly optimized Vision Transformer without the need for image labels

Tibbe Lukkassen
TU Delft

Abstract

2D to 3D transfer learning has proven to be an effective method to transfer strong 2D representations into a 3D network. Recent advancements show that casting semantic segmentation of outdoor LiDAR point clouds as a 2D problem, such as through range projection, enables the processing of 3D point clouds with a Vision Transformer (ViT). For autonomous vehicles, collecting camera data alongside point cloud data is inexpensive. In this work, we investigate how we can best use these available 2D images, by exploring the benefit of camera pretraining to LiDAR semantic segmentation by using a shared ViT backbone. We propose a label generation method that generates pixel-wise pseudo labels for the camera images from the LiDAR annotations. We show that we can effectively use these for pretraining before finetuning on point cloud semantic segmentation. Besides, we compare jointly optimizing a shared ViT encoder for image and point cloud semantic segmentation simultaneously, with finetuning on point cloud semantic segmentation after pretraining on the camera domain. We show that joint optimization can lead to improved performance compared to pretraining when training from scratch. By using a shared encoder for the camera and LiDAR domain we can investigate the joint Camera-LiDAR feature space and find it is possible to create a shared feature space where LiDAR and camera features from the same class are mapped to the same location in the feature space. However, this does not contribute to a better performance on LiDAR semantic segmentation. These experiments provide valuable insight into 2D to 3D transfer learning and the creation of a shared Camera and LiDAR feature space.

1. Introduction

In recent years, self-driving cars have garnered significant attention in research. The concept of vehicles operating autonomously brings about several advantages, including enhanced road safety, reduced pollution, improved accessibility for individuals with mobility challenges, and a decrease in driving-related stress [36]. The presence of self-driving cars on our roads is becoming more common, yet there remains a continual pursuit of enhancements in this domain. Autonomous vehicles are often equipped with LiDAR and camera sensors [24]. Rich spatial information received from a LiDAR sensor, and the strong semantic information, obtained from a camera sensor, are both important components for an autonomous vehicle to understand its surroundings [15]. Because of their importance both camera and LiDAR semantic segmentation are extensively researched.

Vision Transformers (ViT) [8] have proven to be very effective for image semantic segmentation [5, 28]. Recent works [1, 13, 22, 33] have shown that it is also possible to achieve competitive performance on the point cloud semantic segmentation task using this ViT architecture. This has made it possible to transfer 2D knowledge from the rich semantic information captured by a camera sensor to a network used for 3D point cloud perception. For example, Pix4point [22] and Simple3D-Former [33] use a transformer pretrained on image data for indoor object detection. Pix4point [22] uses the k-NN algorithm with a graph convolution on the aggregated neighborhoods to extract input tokens for the ViT encoder such that it can use the original ViT architecture. Simple3D-Former [33] edits the tokenizer, position embedding, and head to be able to use the ViT for 3D tasks. Recently, RangeViT [1] shows the same strategy also applies to autonomous driving datasets where the point clouds are significantly sparser and noisier. Specifically, RangeViT first projects the point cloud to the range view and then uses a convolutional stem to steer the range projection to the right dimensions for the ViT. Pretraining the ViT on an image dataset like ImageNet21K [23] and Cityscapes [6] has been proven beneficial for point cloud semantic segmentation.

Building on the RangeViT [1] framework, we explore the knowledge transfer from camera sensor to LiDAR sensor using a shared ViT backbone for LiDAR semantic segmentation in autonomous driving. RangeViT employs pretrained vision transformers on ImageNet21K [23] and CityScapes [6] as the backbone. We first question if performance can be further enhanced by pretraining directly on the same dataset, i.e., both camera and LiDAR data are from the same scene, thereby addressing potential domain gaps between the datasets. However, the adopted nuScenes dataset only provides point-wise semantic labels, while pixel-wise labels are absent. We leverage foundation models (SAM) and weak bounding box annotations to generate pixel-wise pseudo labels, enabling supervised pixel-wise semantic segmentation. Besides we question if fine-tuning the weights after pretraining on camera data for LiDAR semantic segmentation is the best strategy to leverage the knowledge gained from the camera modality. Inspired by works on multitask learning, we test joint training from scratch instead of pretraining on the camera modality and fine-tuning on specific downstream tasks where the general knowledge across sensors might be lost. To conclude we aim to answer two critical questions:

- Does pretraining on image datasets from the same domain improve performance more than pretraining on different domain image datasets for 2D to 3D transfer learning?
- Can joint training preserve 2D knowledge more effectively within the network, resulting in improved performance, compared to first pretraining on camera data and then finetuning on the LiDAR data?

To answer these questions, we conduct experiments from the following aspects:

Generating pseudo labels. Since the nuScenes benchmark adopted in RangeViT does not provide pixel-wise semantic labels, We create labels using the SAM model [12] and 3D bounding box annotation labels from LiDAR data. We evaluate this method on the nuImages mini dataset with their provided pixel-wise semantic labels. Besides we demonstrate that the pseudo labels can be effectively used to train a camera encoder.

Pretraining with pseudo labels. We compare the effectiveness of pretraining a vision transformer on different sets of image data before finetuning on 3D semantic segmentation. To be specific, we compare image pertaining on a different domain with accurate ground truth labels (e.g., CitySpace [6]) and on the same domain using generated pseudo labels. We find that using the pseudo labels helps to improve performance for the task of 3D semantic segmentation, but that labels from a different domain with accurate ground truth labels improve performance more.

Comparing joint optimization and pretraining. Inspired by multitask learning, we investigate whether jointly optimizing a network for both LiDAR and camera semantic segmentation is a superior strategy compared to pertaining and finetuning. We show that joint optimization improves performance when the model is trained from scratch.

Creating a multimodal shared feature space. Lastly, we evaluate the benefit of having a shared feature space for the LiDAR and camera sensors. We find that it is *not* beneficial for downstream performance to enforce a shared latent space where LiDAR and camera features from the same class are mapped to the same location in the feature space.

2. Related work

2.1. 2D to 3D transfer learning

Recent works have explored different methods for knowledge transfer from 2D to 3D [1, 17, 18, 20, 22, 25, 33, 34].

Contrastive learning. [17, 18, 20, 25] reach this goal by using a self-supervised pretraining method. They use contrastive learning to transfer strong 2D representations to a 3D network. PPKT [18] is one of the first works to make use of this 2D-to-3D representation distillation paradigm. It uses a contrastive loss between features from a frozen and well-trained camera encoder and a trainable LiDAR encoder. SLidR [25] builds upon this method by contrasting patches instead of pixels by generating superpixels. Both ST-SLidR [20] and SEAL [17] build on top of SLidR [25]. ST-SLidR [20] adds a semantically tolerant loss to improve performance. SEAL [17] improves this method by leveraging vision foundation models to create patches instead of using superpixels.

fusion methods. Differing from 2D to 3D transfer learning, fusion methods [2, 15, 31, 32] show performance improvement when using multiple modalities during inference. Both UniBEV [32] and Transfusion [2] show that fused training can improve performance on 3D object detection with LiDAR data when compared to training on LiDAR only. UniBEV [32] uses a uniform deformable attention-based architecture for both its camera and LiDAR branch to build the LiDAR and camera BEV features to create better aligned features across modalities. The key idea of Transfusion [2] is to reposition the focus of the fusion process, from hard association to soft association, leading to the robustness against degenerated image quality and sensor misalignment. In contrast to these works, other fusion methods like PointPainting [31] show they are dependent on both modalities during inference and degrade in performance when only the LiDAR modality is used. This makes them ineffective for 2D to 3D transfer learning.

Transfer learning. A different method deployed, is to copy the weights of a network pretrained on the camera modality into a network for the LiDAR modality [1, 22, 33, 34]. Image2point [34] is able to do this by converting a 2D CNN to a 3D CNN by inflating the 2D convolution into 3D convolutions. Pix4point [22] and Simple3D-Former [33] use a transformer pretrained on image data for indoor object detection. Pix4point [22] uses the k-NN algorithm with a graph convolution on the aggregated neighborhoods to extract input tokens for the ViT encoder such that it is able to use the original ViT architecture. Simple3D-Former [33] uses a teacher branch to maintain the 2D knowledge in the network. RangeViT [1] proposes a LiDAR network that enables them to use a Vision Transformer as its core encoder by transforming the point cloud into a range image and is the first method to use this for outdoor point cloud semantic segmentation. They use pretrained weights for this Vision Transformer, trained on RGB image data on datasets like CityScapes [6] and ImageNet21K [23]. They show improved performance when training from these pretrained weights. Our method builds upon RangeViT [1] and extends this work into a method where the ViT can be jointly optimized for both camera and LiDAR semantic segmentation simultaneously.

2.2. Training with pseudo labels

The field of weakly supervised semantic segmentation (WSSS) focuses on reducing the label cost of fully supervised semantic segmentation [21]. Instead of labeling all the pixels manually, it resorts to weaker yet cheaper alternatives. Examples include labeling sparse points, using bounding box labels, scribbles in an image, or image-level classification results as the label to create the semantic label. BBAM [14] and [27] both use bounding boxes to generate semantic and instance segmentation labels. Another line of works [29] makes use of the Segment Anything model (SAM) [12] to create semantic labels. [29] creates bounding boxes using text prompts for the image. These bounding boxes are then processed by SAM [12] to create a semantic mask using the bounding box as the input. It shows promising performance with only very little performance degradation. In this work, we also use bounding boxes as the input to create the pseudo label and use SAM [12] to create the pseudo labels within this bounding box.

2.3. Multi task learning

Multi-task learning seeks to combine several tasks within one network, training them simultaneously in an end-to-end manner. LiDARMultiNet [35] proposes to unify the three major LiDAR perception tasks in a single network that exploits the synergy between these tasks. LidarMLT [9] proposes joint training with a single network for object detection and road understanding, and achieves on-par single-task perception performance, regardless of the combination of multiple tasks. LiDARFormer [37] shows it is possible to develop a unified transformer-based multi-task LiDAR perception network with the ability to learn global context information. These works show that multiple losses can serve as an auxiliary loss and help exploit the synergy between the different tasks which can lead to improved performance. In contrast to these methods, there are also works that investigate a shared network with shared weights across multiple modalities to solve multiple tasks. UniT [10] builds a unified transformer that takes both image and text as input. After joint training, it is able to accomplish multiple tasks ranging from visual perception and natural understanding to joint vision and language learning. Our work draws inspiration from multitask learning and uses a shared backbone for 2D to 3D transfer learning. To the best of our knowledge, there are currently no works that investigate joint training for cross-modality tasks in LiDAR-camera perception.

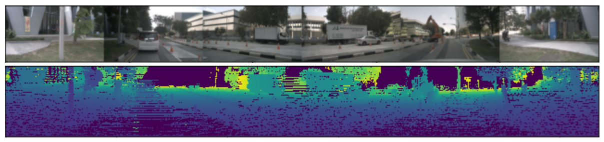


Figure 1. A visualization of range view compared with the camera image.

3. Methodology

We first describe the generation of pixel-wise semantic labels using SAM due to the lack of annotation on the nuScenes benchmark. Second, we detail the fine-tuning after pretraining and joint learning strategies adopted in the comparison. Notably, the backbone model is ViT which can be trained on either a single modality or jointly optimized for both modalities simultaneously. Fig. 2 shows an overview of the proposed pipeline.

3.1. Pixel-wise semantic label generation

To create the pixel-wise semantic labels, we leverage the 3D bounding box annotation from the LiDAR data. First, we map the 3D bounding boxes to the 2D images using the camera intrinsic matrix. Second, we use SAM [12] (the ViT-B model) to create a mask within this bounding box. The created mask shares the same semantic class as the bounding box annotation. Objects that are completely occluded by other objects in the camera image can still be annotated with a 3D bounding box, but they should not be assigned a pixel-wise semantic label. Therefore bounding boxes that are entirely occluded by different bounding boxes are excluded. Besides, for boxes of which only a small part falls within the image, we create a mask using SAM but add them to the ignored class. The mask created for these cases is often not precise, but since it is created within a bounding box this part of the image does certainly contain a foreground object. All pixels not part of a foreground class or the ignored class are labeled as background.

It is worth mentioning that this method is only able to create labels for foreground objects that have a bounding box label. Different static background objects that are not labeled with a bounding box annotation are grouped into a single background class. Examples are shown in Fig. 4.

3.2. Multi modal pertaining, fine-tuning and joint learning

We adopt the same design as RangeViT [1], consisting of a convolutional stem, a ViT encoder, a Decoder, and a 3D refiner module. Given LiDAR data as input, RangeViT first conducts the range projection [13], which converts the Li-

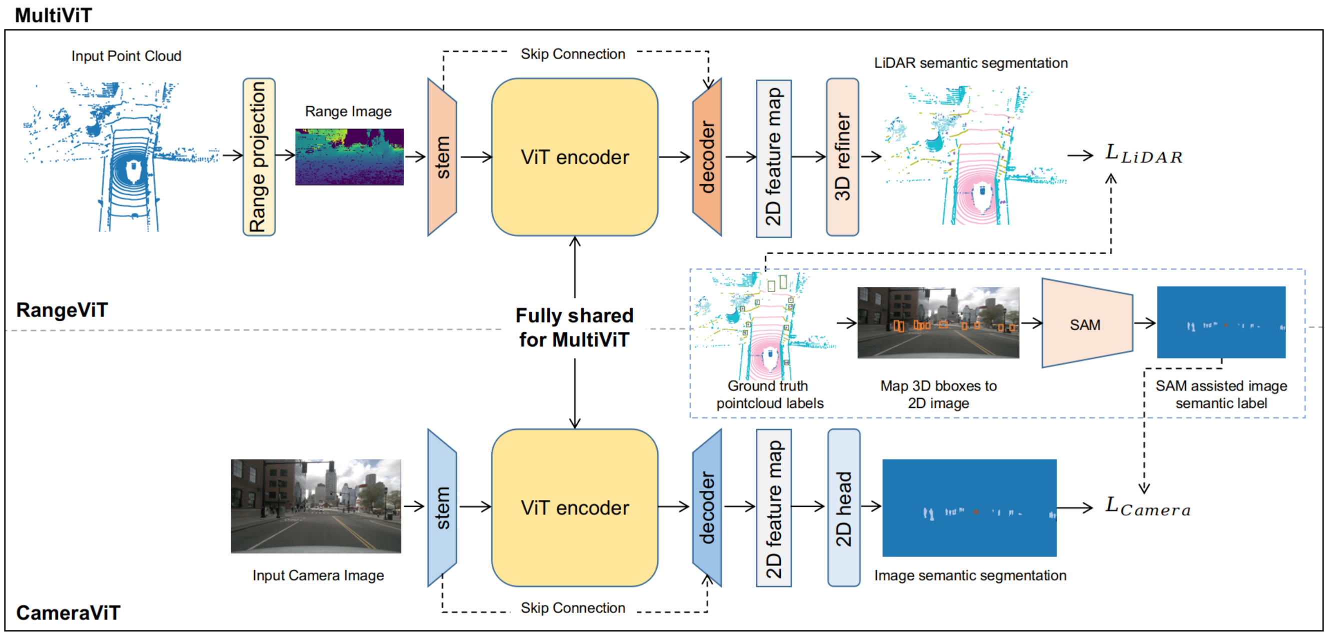


Figure 2. Overview of the MultiViT framework. We research the knowledge transfer from 2D to 3D via a shared backbone design, for LiDAR semantic segmentation in autonomous driving. We first generate pixel-wise semantic labels for camera data using SAM [12], enabling supervised training on camera data. Later, we compare pretraining on camera data followed by finetuning on LiDAR data to joint leaning from camera and LiDAR data.

DAR data as grid representation as images. We visualize the range projection in Fig. 1. We adapt RangeViT such that it can also be used for camera semantic segmentation. We refer to this model as CameraViT. Additionally, we propose MultiViT by sharing the ViT encoder between RangeViT and CameraViT such that the ViT encoder can be jointly optimized with camera and LiDAR data. MultiViT can eventually be used for both LiDAR and Camera semantic segmentation. We name the LiDAR branch used for LiDAR semantic segmentation MultiViT_L and name the camera branch used for camera semantic segmentation MultiViT_C, as shown in Fig. 2.

3.2.1 Model Architecture

RangeViT. RangeViT first conducts range projection to convert LiDAR point cloud into grid data for sequential processing. The LiDAR data is rasterized into a 2D cylindrical project of size $H \times W$ [13]. Where H and W are the height and width, respectively. The rasterization for each point p_n can be formulated as follows:

$$\begin{pmatrix} u_n \\ v_n \end{pmatrix} = \begin{pmatrix} \frac{1}{2}[1 - \arctan(p_n^y, p_n^x)\pi^{-1}]W \\ [1 - (\arcsin(p_n^z, p_n^d)^{-1}) + \phi^{down})\xi^{-1}]H \end{pmatrix}$$

, where (u_n, v_n) denotes the grid coordinate of point p_n in range image $R(u, v)$. The range in this range view is calculated by $p_n^d = \sqrt{(p_n^x)^2 + (p_n^7)^2 + (p_n^z)^2}$, which indicates the distance between the point and the LiDAR sensor. $\xi = |\phi^{up}| + |\phi^{down}|$ is the vertical field-of-views of the sensor and ϕ^{up} and ϕ^{down} are the inclination angles at the upward and downward directions, respectively. Fig. 1 shows an example of a point cloud projected in range view. The range, intensity, x, y, and z values are concatenated for each rasterized point.

After range projection, the image-like input data is fed into the convolutional stem. This module uses four residual blocks of SalsaNext [7] and produces pixel-wise features with D_h channels. To steer these pixel-wise features into tokens suitable for the ViT, we first apply an average pooling layer to reduce the spatial dimension of the features from $H \times W$ to $(H/P_H) \times (W/P_W)$, where P_W and P_H are the patch sizes. For RangeViT we use a patch size of 2×8 in all experiments. We then apply a 1×1 convolutional layer with D output channels to form the final visual tokens to match the input dimension and number of tokens of a traditional ViT.

The output of the convolutional stem is fed into a ViT [8]. The vision transformer produces a feature of size 384 for each patch. Afterward, the decoder turns the per patch feature into a feature map of $D_h \times P_H \times P_W$ with a 1×1 convolution layer, followed by a Pixel Shuffle layer [26]. These features are concatenated with the stem features produced by the convolutional stem. These concatenated features are passed through the last two convolutional layers of the decoder with Leaky ReLU and batch normalization. The decoder output results in the 2D refined feature map with a feature of size 256 for each 'pixel' in the range image. To project these features from this 2D feature map back into a feature for each point cloud point, a 3D refiner module is used. This module makes use of a KPConv layer [30] to do this. After this KPConv [30] layer the logits are obtained by applying a BatchNorm, a ReLU, and a final points-wise linear layer on these point features.

CameraViT. CameraViT has several key changes compared to RangeViT, as it does not need the range projection and 3D refiner module. The convolutional stem, the ViT, and the decoder are kept exactly the same as for RangeViT. The patch size used in the convolutional stem for CameraViT has size 8×8. The 3D refiner is replaced by a single 2D convolution with a BatchNorm, a Relu, and a linear layer to produce a logit for each pixel.

MultiViT. MultiViT combines CameraViT and RangeViT without changing anything to their design. The model is trained with both Camera and LiDAR data. Both modalities are processed by the same ViT encoder with the same weights but the data is not fused. The model consists of two modality-specific stems and two modality-specific decoders. This allows the model to separately process camera and LiDAR data and process them with the same ViT encoder.

3.2.2 Implementation details

Training loss. We use the same loss functions for training RangeViT, CameraViT, and MultiViT, including a multi-class focal loss [16] and a Lovasz-softmax loss [3]. The multi-class focal loss is a scaled version of the cross-entropy loss and adapts the loss values accordingly, making it a better option for datasets with a high class imbalance. The Lovasz-softmax loss allows for direct optimization of the mean intersection-over-union. When training on Camera data, there is a large class imbalance in the dataset. Almost 90% of all the image pixels are labeled as background while the other 10% is divided over the other 10 foreground classes. Because of this large class imbalance, we use a correction factor λ to balance the Lovasz and focal loss, where the total loss for RangeViT and CameraViT can be formulated as $L_{LiDAR} = L_{Lovasz} + \lambda_l L_{focal}$ and $L_{camera} = L_{Lovasz} + \lambda_c L_{focal}$ respectively. When training CameraViT we set $\lambda_c = 3$. When training RangeViT we set $\lambda_l = 1$. When training MultiViT, the total loss is $L_{total} = L_{LiDAR} + L_{camera}$.

Inference. As in [1] we use a sliding-window method during inference for RangeViT. At inference, the range image is divided into overlapping crops of the same size as those used during training. In comparison, the network never sees the entire range image during training but uses crops extracted from the full range image.

4. Experiments

4.1. Experimental setup

Dataset and metrics. We validate our approach for LiDAR semantic segmentation on the nuScenes [4] dataset. This is a dataset consisting of 1,000 scenes of 20 seconds in Boston and Singapore. It consists of various urban scenarios and lighting and weather conditions. The LiDAR data has been annotated with 16 semantic classes and 3D bounding boxes. From the 16 semantic classes, 10 classes are classified as foreground classes and 6 classes as static background classes. These 6 background classes do not have bounding box annotations. The dataset is split into 28,130 training and 6,019 validation LiDAR scans. Each LiDAR scan is associated with 6 corresponding camera images which form a full 360-degree field of view. The dataset set therefore consists of 168,780 camera images for training and 36,114 for validation. We use the mean Intersection over Union (mIoU) for performance evaluation.

Augmentations. When training on LiDAR data, we use the same augmentations as in [1], including (a) flips over the y-axis (the vertical axis on the range image), (b) random translations, and (c) random rotations between ±5○ (using the roll, pitch, and yaw angles). All augmentations are applied randomly with a probability of 0.5. We randomly sample a crop of size 32 × 384 from the full range image of size 32 × 2048. When training on camera data, we use a combination of augmentation techniques, including random flipping, random scaling, and random rotation. Images from all 6 views are of size 256 × 704. For MultiViT training, we use both LiDAR and camera data. We divide the range image into 6 even crops of size 32 × 384, each of which aligns with a camera view. We sample only aligned range image crops and their camera images. We train CameraViT 6 times fewer epochs than RangeViT as each LiDAR scan corresponds to 6 camera views.

Model configuration. For all experiments (RangeViT, CameraViT, and MultiViT), we use the ViT-S backbone [8]. It has 12 layers, 6 attention heads, and a feature size of 384. The convolutional stem, decoder, and 3D refiner use a feature of size 256.

Training setup. We use the AdamW optimizer [19] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay 0.01 as in [1]. The batch size during training is set to 32 for RangeViT, and to 12 for MultiViT and CameraViT. Regarding the learning rate schedule, we employ a linear warm-up from 0 to its peak value over a specified number of warm-up epochs, followed by a cosine annealing schedule that decreases the learning rate to 0 over the remaining epochs. Specifically, when training from scratch, we initialize the learning rate to $lr = 0.008$. Conversely, when initializing the Vision Transformer (ViT) from pre-trained weights on the ImageNet21K dataset [23], we adjust the learning rate to $lr = 0.004$.

	Method	camera pretraining	barrier	bicycle	bus	car	construction	motorcycle	pedestrian	traffic cone	trailer	truck	driveable	other flat	sidewalk	terrain	manmade	vegetation	**mIoU (%)**
Random initialization																			
1	RangeViT	-	**74.8**	25.6	82.1	87.0	39.0	66.7	74.9	58.2	**59.3**	72.6	**96.1**	70.3	**73.1**	**73.8**	87.8	85.9	70.5
2	RangeViT	nuScenes	74.7	**26.4**	**87.5**	**88.1**	**40.1**	**72.1**	**75.2**	**58.8**	58.9	**73.6**	96.0	**70.6**	72.2	73.3	**88.0**	**86.2**	**71.4**
In21k Initialization																			
3	RangeViT	In21k	75.8	35.3	87.0	89.0	46.9	76.3	**77.9**	64.7	64.7	76.1	**96.5**	71.8	73.6	73.7	88.8	87.2	74.1
4	RangeViT	In21k + CS	73.7	**37.9**	**91.3**	**90.1**	**49.2**	**76.4**	77.6	**65.2**	**66.2**	**79.4**	96.4	**71.9**	**74.1**	**74.1**	**89.0**	**87.3**	**75.0**
5	RangeViT	In21K + nuScenes	**76.2**	35.9	88.9	88.2	43.8	74.0	77.2	64.6	63.0	77.5	96.3	71.5	73.8	73.7	88.8	87.0	73.8

Table 1. nuScenes [4] validation set comparison for different starting weights for the ViT. All models have been trained on the nuScenes [4] dataset for LiDAR semantic segmentation. The camera pretraining column indicates what camera data has been used for initiating the weights of the ViT. Best results are for each training setting are in bold. (-) = trained from scratch, (In21k) = pretrained on ImageNet21k, (In21k + CS) = pretrained on ImageNet21K and CitScapes, (nuScenes) = pretrained on nuScenes images using the generated pixel-wise semantic labels, (In21K + nuScenes) = pretrained on ImageNet21K and nuScenes images using the generated pixel-wise semantic labels.

Similar to [1], we train RangeViT for 150 epochs with 10 warm-up epochs. We train CameraViT and MultiViT for 25 epochs with 2 warm-up epochs.

Baselines. As our baseline we compare with RangeViT [1]. We reproduce their results and mention these results in our work. RangeViT [1] employs pretrained vision transformers on ImageNet21K [23] and CityScapes [6] trained with [28].

Optimization. During training, occasionally, a large sudden dip in model performance occurred, after which the model often was not able to recover to its original state. The occurrence of this phenomenon has been random and several attempts have been made to better understand it. More details on this can be found in appendix 6. In the occurrence of such a dip, the training will be restarted from the most recent training checkpoint saved before the dip. In almost all cases the dip did not reoccur after restarting.

4.2. Evaluating pixel-wise pseudo label generation

The quality of pseudo labels is a key to supervised learning. To evaluate the effectiveness of our pseudo label generation pipeline, we conduct an evaluation on the manually-annotated nuImages dataset [4]. This dataset has a minimal domain gap compared to nuScenes [4], as both datasets were collected using the same sensor setup and in similar environments. The nuImages dataset provides ground truth image semantic labels, allowing us to assess the accuracy of the SAM-assisted semantic label generation pipeline. A notable difference between these datasets is that the nuScenes dataset contains only 3D bounding boxes, while nuImages contains 2D bounding boxes. Our method generates labels from the 3D bounding boxes, which, when mapped to the 2D image, often do not tightly fit around the objects. To ensure a fair comparison, we apply random inflation to the bounding boxes, increasing their size by a random percentage between 0 and 20 percent of the original size. We present the mean Intersection over Union (mIoU) of the pixel-wise pseudo labels using the inflated 2D bounding boxes on the nuImages-mini dataset in Tab. 2.

Class	IoU
barrier (69)	87.7
bicycle (14)	73.8
bus (5)	94.1
car (117)	90.3
construction (5)	78.3
motorcycle (13)	89.0
pedestrian (193)	73.1
traffic cone (44)	70.8
trailer (2)	85.1
truck (28)	81.7
mIoU	**82.4**

Table 2. Evaluation of the generated pixel-wise semantic labels on the nuImages [4] mini dataset. To assess if the labels are accurate enough to train CameraViT, we calculate the IoU score for each class and report the mIoU. The numbers between the brackets are the number of occurrences of the objects from each class in the dataset.

Tab. 2 shows a quantitative result. The bicycle, pedestrian, and traffic cone classes show a relatively lower accuracy compared to the other classes. Evaluating the qualitative results from Fig. 4 most of the mistakes are found at the object borders. The labels look promising and show potential to be used to effectively distill camera information to the LiDAR encoder.

4.3. How does pertaining on camera data benefit LiDAR semantic segmentation?

We aim to study how pertaining on camera data benefits LiDAR semantic segmentation. To do this we first evaluate

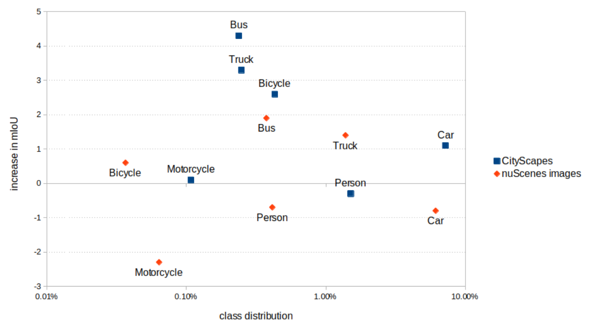


Figure 3. This figure analyses the difference between pretraining on the CityScapes dataset and on the nuScenes images. When the ViT is initialized with ImageNet pretrained weights, pretraining on CityScapes finds performance improvements while pretraining on the nuScenes images does not. Since both datasets consist of different class distributions this figure shows the correlation between the class distribution and the increase in mIoU for each class compared to training directly from ImageNet pretrained weights. The figure shows no clear correlation between improvements in mIoU. (e.g. the truck class is less common in the CityScapes dataset but still a higher increase in mIoU is found.)

Model	**pretraining**	**Contrastive loss**	**mIoU**
CameraViT	-	-	57.0
MultiViT_C	-	-	60.2
CameraViT	In21k	-	63.4
MultiViT_C	In21k	-	61.8
MultiViT_C	In21k	pull	61.8
MultiViT_C	In21k	pull + push	58.1

Table 3. The evaluation scores for different training methods evaluated on image semantic segmentation. The models are evaluated on the images from the nuScenes validation set using the generated pixel-wise semantic labels. CameraViT means only the Camera branch is trained. MultiViT_C means MultiViT is used in a joint training objective and the Camera branch is used for evaluation.

training CameraViT on the generated pixel-wise semantic labels. Thereafter we evaluate different camera pretraining settings for LiDAR semantic segmentation.

The results presented in Tab. 3 demonstrate that the CameraViT model achieves satisfactory mean Intersection over Union (mIoU) when evaluated on the nuScenes images using the generated pixel-wise semantic labels. Qualitative analysis in Fig. 5 indicates that distant objects are often challenging to distinguish. However, the model generally exhibits a robust capability to segment different classes in complex images. Based on the quantitative data in Tab. 3 combined with the qualitative assessment from Fig. 5 we conclude that the CameraViT model can be effectively trained using the SAM-assisted generated labels.

Tab. 1 shows the results on the nuScenes [4] validation set for LiDAR semantic segmentation by fine-tuning various pretrained models on camera data. The comparison between Experiment 1 and Experiment 2 shows that pretraining on the nuScenes camera data, even though with imperfect pseudo label, improves performance on LiDAR semantic segmentation by 0.9 in terms of mIoU.

However, the comparison between Experiment 3 and Experiment 5 tells a different story. When initialized by ImageNet pre-trained weights, extra camera pretraining on nuScenes decreases the performance by 0.3. Meanwhile, Experiment 4 shows that extra pertaining on CitySpaces [6] can increase the mIoU from 74.1 to 75.0. The difference between Experiment 4 and Experiment 5 indicates the pertaining dataset plays an important role in improving LiDAR semantic segmentation. Since both CityScapes and the nuScenes images consist of different scenes and thereby have different class distributions, Fig. 3 shows the correlation between the increase in mIoU and the class distributions for the common classes between CityScapes and the nuScenes images. The figure shows no clear proof that the increased performance for the different classes derives from the difference in class distributions between the datasets. Besides CityScapes consists of much fewer images compared to the nuScenes dataset.

Our primary speculation is that, despite coming from the same scene, the pixel-wise semantic label on nuScenes is not as precise as the manually annotated CitySpace dataset, thus leading to performance decrease.

4.4. Pretraining+finetuning *v.s.* joint training

One common strategy in knowledge transfer is pertaining on camera data and then fine-tuning on LiDAR data. Another strategy, inspired by multi-task learning, is joint learning on both camera and LiDAR data, which has the potential to maintain decent performance on both camera and LiDAR data. In this section, we compare these two strategies numerically.

Tab. 4 presents the results on the nuScenes [4] validation set for LiDAR semantic segmentation. The comparison between Experiment 6 and Experiment 7 shows that joint training increases the mIoU by 0.9 over finetuning after pretraining and indicates that performance is enhanced by joint optimization. When comparing Experiment 8 and Experiment 9, where the ViT is initialized with ImageNet pretrained weights, finetuning after pretraining can outperform joint training by 0.6. ImageNet initialization has, again, become a crucial factor, without which joint training exhibits advantage, while with this initialization, finetuning excels.

Notably, the performance of joint training on *bus*, *construction* classes is noticeably lower than finetuning when using ImageNet initialization. Besides, Tab. 3 shows that using ImageNet initialization does significantly improve performance on image semantic segmentation which might

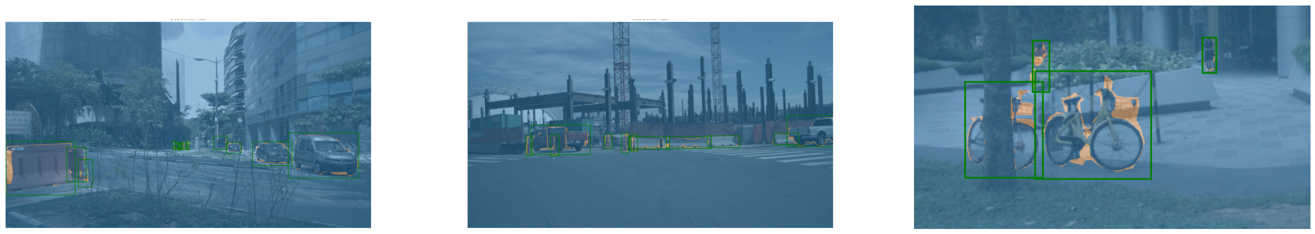


Figure 4. Three different images from the nuImages dataset [4] with the correctly labeled pixels using the generated pixel-wise semantic label. The images show the bounding box used as the input for SAM [12]. The blue pixels are the correctly labeled pixels while the yellow pixels are wrongly labeled. Mostly the object borders show wrongly labeled pixels.

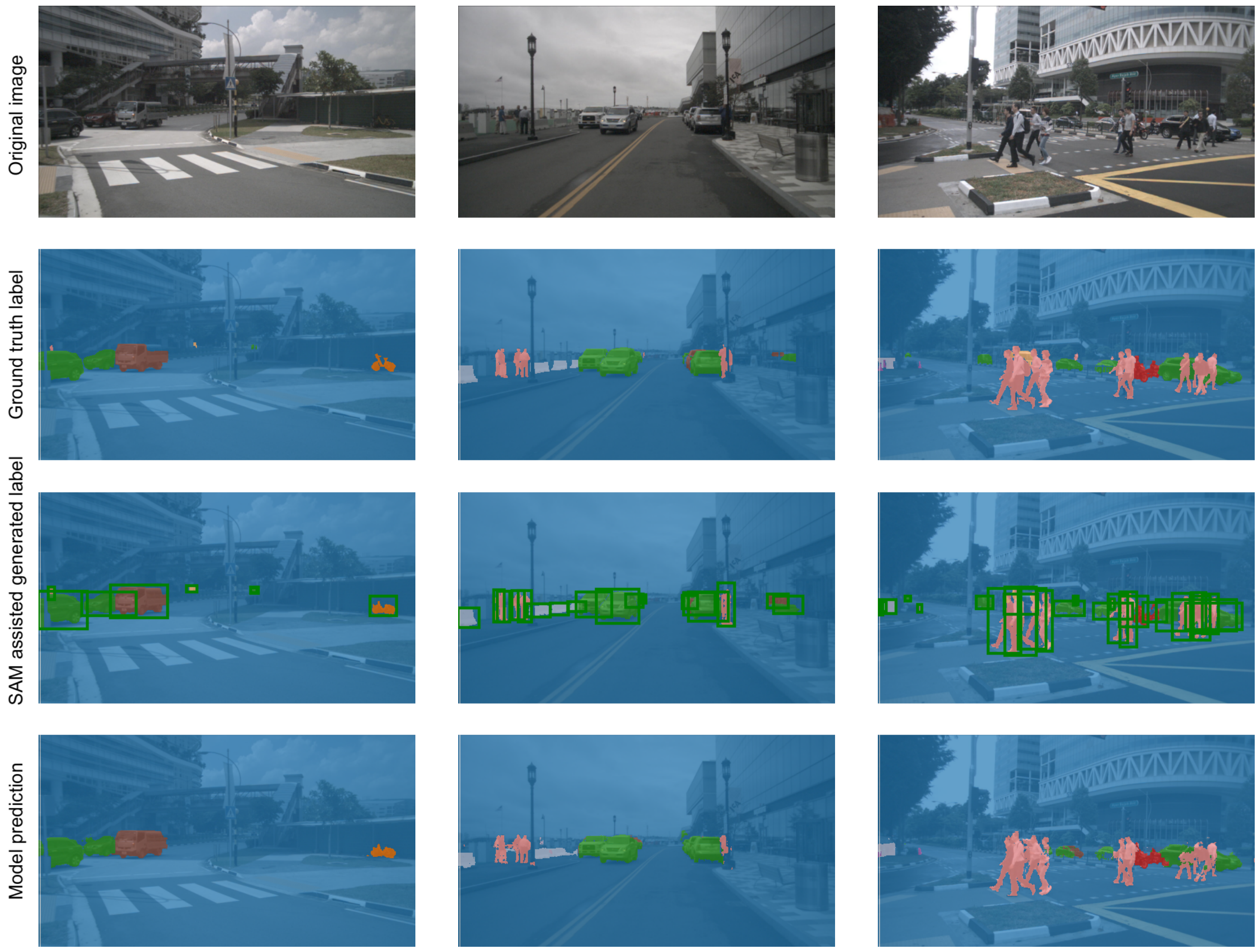


Figure 5. Three different scenarios from the nuImages dataset [4] are used to evaluate the label generation method and the CameraViT model trained from scratch. From top to bottom, we show the original image, the ground label, the generated pixel-wise semantic label with the bounding boxes used as the input, and finally the model prediction from CameraViT (-).

help for finetuning with this ViT. However, it remains unclear why such dramatic changes happen when employing ImageNet pretrained weights to initialize ViT.

To conclude, these results suggest that a well-initialized ViT encoder contributes more to task-specific finetuning than joint training.

	Method	camera pretraining	barrier	bicycle	bus	car	construction	motorcycle	pedestrian	traffic cone	trailer	truck	driveable	other flat	sidewalk	terrain	manmade	vegetation	**mIoU (%)**
Random initialization																			
6	RangeViT	nuScenes	74.7	26.4	87.5	88.1	40.1	**72.1**	75.2	**58.8**	58.9	73.6	96.0	70.6	72.2	73.3	88.0	86.2	71.4
7	MultiViT_L	-	**75.2**	**28.9**	**87.7**	**91.4**	**41.4**	69.7	**75.7**	57.4	**60.7**	**77.5**	**96.6**	**71.3**	**73.8**	**73.7**	**88.7**	**87.0**	**72.3**
In21k initialization																			
8	RangeViT	nuScenes	**76.2**	**35.9**	**88.9**	88.2	**43.8**	74.0	**77.2**	**64.6**	63.0	**77.5**	96.3	**71.5**	73.8	73.7	88.8	87.0	**73.8**
9	MultiViT_L	-	75.4	35.1	83.0	**90.8**	40.9	**75.7**	77.0	62.4	**64.1**	75.4	**96.7**	70.9	**74.0**	**73.8**	**88.9**	**87.1**	73.2

Table 4. Results on nuScenes [4] validation set. The RangeViT models are first pre-trained with camera data and then finetuned with LiDAR data, while the MultiViT_L models are jointly trained with nuScenes camera and LiDAR data. The pretraining column indicates what data has been used for pretraining. Best results for each setting are in bold.

	Seperate train	**Joint train**	**Joint train, pull loss**	**Joint train, pull + push loss**
barrier	-	-	-	-
bicycle	0.30	0.04	0.34	**0.47**
bus	0.19	0.18	0.16	**0.59**
car	-0.06	-0.03	0.06	**0.73**
construction	-	-	-	-
motorcycle	0.26	0.23	0.19	**0.29**
pedestrian	0.05	-0.02	0.24	**0.84**
traffic cone	0.37	0.1	0.13	**0.47**
trailer	-	-	-	-
truck	-0.17	0.16	0.21	**0.70**
Mean	0.13	0.09	<u>0.19</u>	**0.58**

Table 5. Cosine similarity between the average camera feature and average LiDAR feature for each class at the ViT encoder output on the nuScenes mini dataset. All the models used are initiated with ImageNet pretrained weights. The average feature for each class is calculated using features from correctly predicted points and pixels. This ensures a representative average feature for each class. The highest similarity for each class is in bold and the second highest mean similarity is underlined. This shows that joint training does not converge towards a solution where camera and LiDAR features have a high cosine similarity. Adding a contrastive loss creates a feature space where the LiDAR and camera branches show a high cosine similarity for each class.

4.5. Creating a shared feature space

Joint training with a shared ViT encoder (MultiViT) creates a shared feature space for the camera and LiDAR data. We analyze the learned features per class within the shared feature space to better understand joint training.

We first use the features at the ViT encoder output to calculate the average feature for each class on the nuScenes mini dataset. Next, we compute the per-class cosine similarity between the averaged camera and LiDAR features. Tab. 5 shows the similarity between the average camera and LiDAR features. We observe that neither the joint training nor the finetuning (after pretraining) reaches a solution where the ViT encoder features from the camera and LiDAR branches for the same class exhibit high cosine similarity.

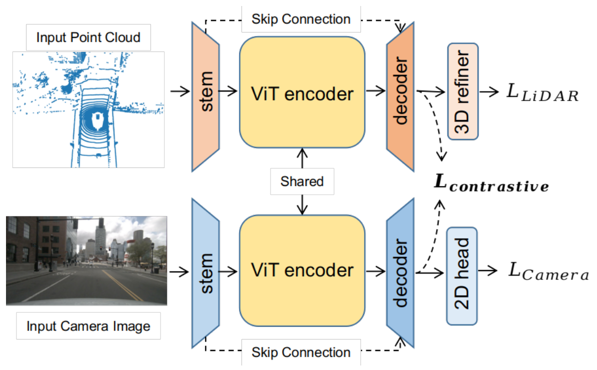
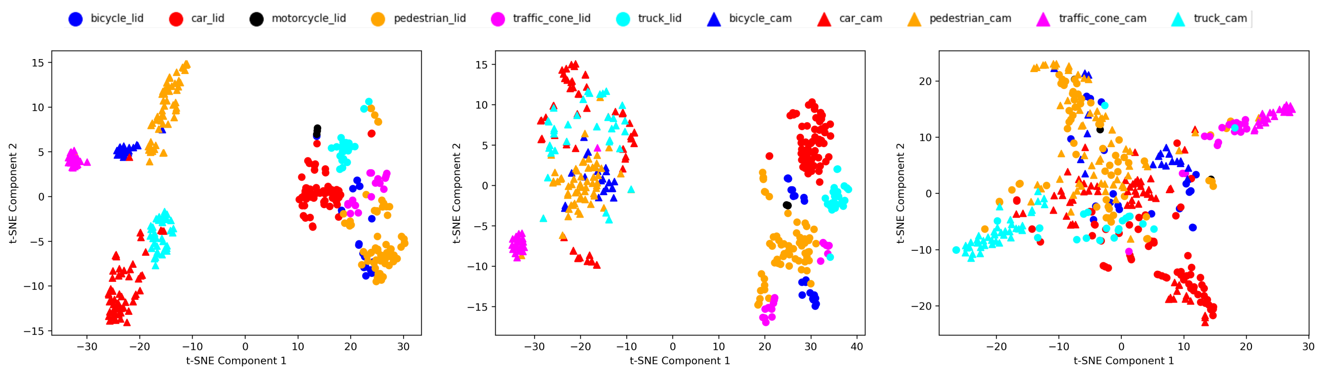


Figure 6. A simplification of the MultiViT framework which shows where the contrastive loss has been applied to create a shared LiDAR and camera feature space.

Since MultiViT does not inherently converge towards a solution where camera features and LiDAR features from the same class are mapped to the same location in the latent space, it is likely that enforcing a shared feature space is not beneficial for model performance. To validate this statement, we conduct an additional experiment by incorporating a contrastive loss on top of the Lovasz and Focal losses to better align the camera and LiDAR features. We utilize a supervised contrastive loss [11], employing labels to identify camera and LiDAR features from the same class as positive pairs. To compute this loss, we first calculate the average feature for each camera and LiDAR class within each mini-batch. LiDAR and camera features from matching classes are treated as positive pairs, while features from different classes serve as negative pairs. We apply the contrastive loss on top of the decoder features, as it requires point and pixel-specific embeddings to calculate the aver-

	Method	Contrastive loss type	barrier	bicycle	bus	car	construction	motorcycle	pedestrian	traffic cone	trailer	truck	driveable	other flat	sidewalk	terrain	manmade	vegetation	**mIoU (%)**
10	MultiViT_L	-	**75.4**	**35.1**	83.0	90.8	40.9	**75.7**	**77.0**	**62.4**	**64.1**	75.4	**96.7**	70.9	**74.0**	**73.8**	**88.9**	**87.1**	73.2
11	MultiViT_L	Pull only	75.3	**35.1**	**87.4**	**91.0**	**41.9**	73.5	75.6	61.8	62.3	**77.5**	96.6	**71.6**	73.8	73.7	88.7	86.9	**73.3**
12	MultiViT_L	Pull + Push	72.2	24.9	86.9	89.8	39.9	67.9	73.0	48.8	55.9	75.0	96.2	69.7	72.9	73.0	87.0	85.3	69.9

Table 6. Results on nuScenes [4] validation set for the task of pointcloud semantic segmentation. All models are jointly trained on Image and LiDAR semantic segmentation using MultiViT and at experiment 11 and 12 a contrastive loss is added to align the camera and LiDAR features. The ViT for each experiment is initialized with In21k weights. Best results are in bold. (-) = joint training only, (pull only) = joint trianing with pull loss, (pull + push) = joint training with the pull and push loss added



(a) Joint training only (b) Joint training with pull loss (c) Joint training with pull and push loss

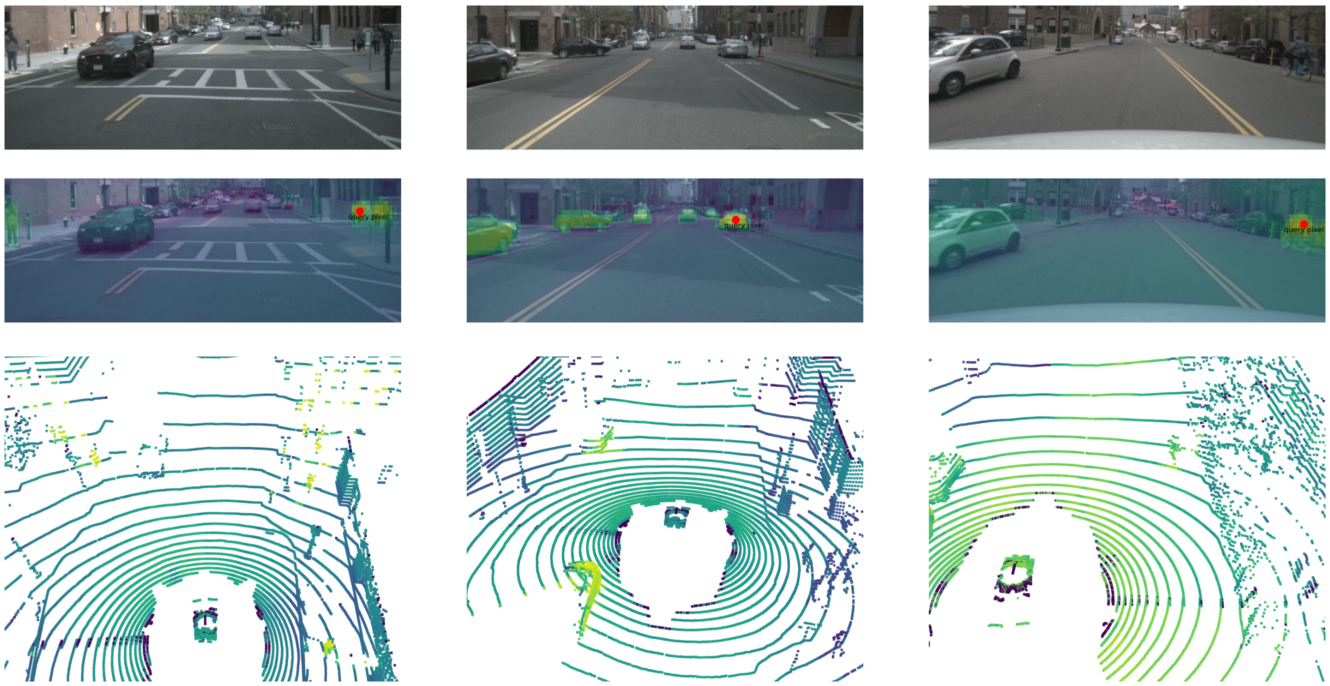
Figure 7. t-SNE plots with camera- and LiDAR features from MultiViT (initiated with ImageNet pre-trained weights). The three plots show the result using the different contrastive losses. Point and pixel features at the decoder output are used. The points and pixels are randomly sampled from the nuScenes mini dataset and for each plot, the same points and pixels are used. Dots are features from the LiDAR branch while triangles are features from the camera branch. We obtain the best mIoU result in (a) where no contrastive loss is added. In comparison, adding contrastive loss (both pull and push losses) decreases the performance despite higher correlation between LiDAR and camera features.

aged per-class feature, as shown in Fig. 6.

We conduct two separate experiments, each utilizing a different type of contrastive loss. In the first experiment, a supervised pull loss is added, only using the positive pairs to pull together LiDAR and camera features from the same class. In the second experiment, a pull and push loss is used which also uses the negative pairs to push away dissimilar class features. We first validate whether a shared feature space has been achieved by assessing the cosine similarity between the average features for each class at the encoder output. The results, presented in Tab. 5, indicate that using the pull loss results in higher cosine similarity between camera and LiDAR features compared to joint training. However, the cosine similarity remains low. Conversely, when using both pull and push losses, the camera and LiDAR features exhibit significantly higher cosine similarity. As illustrated in Fig. 8, a pixel feature from a pedestrian has high cosine similarity to both pixel and point features originating from other pedestrians.

Tab. 6 presents the results on LiDAR semantic segmentation using the MultiViT models optimized with the various contrastive losses. The comparison between Experiments 10 and 12 indicates that the cosine similarity between LiDAR and camera features correlates negatively to the performance. Adding the pull and push loss increases feature similarity between modalities from 0.09 to 0.58 but decreases the performance substantially by 3.3 (from 73.2 to 69.9). Fig. 7 visualizes the latent feature space using t-SNE plots, where joint training without enforcing contrastive loss shows a clear separation between LiDAR and camera features. In contrast, Fig. 7c shows that adding the pull and push loss better aligns the LiDAR and camera features from the same class. This aligns with the findings in Tab. 5. Besides, Fig. 7c shows more faint class boundaries compared to the other t-SNE plots. However, it remains unclear why this results in a performance decrease.

We conclude that enforcing a shared latent space, where LiDAR and camera features are mapped to the same location in the feature space does not improve performance.



(a) Pedestrian pixel as query feature (b) Car pixel as query feature (c) Bicycle pixel as query feature

Figure 8. This figure shows the cosine similarity between a query feature from the image compared with the other pixel features of the image and the pointcloud features from the same scene. The features are generated using MultiViT trained with the pull and push loss. The features at the output of the encoder are used. The top image shows the original image. The middle image shows the cosine similarity of the image features compared with the query pixel. The bottom image shows the cosine similarity of the pointcloud features compared with the query pixel. In all images yellow means a high cosine similarity and blue means a low cosine similarity. The red dot indicates the query pixel. The scenes show that there is a clear similarity in the pedestrian and car classes between the LiDAR and camera features. There is less similarity in the bicycle class between the LiDAR and camera features.

5. Conclusion and Discussion

We use LiDAR annotations to create pixel-wise semantic labels for image semantic segmentation and use this to study the benefit of camera pertaining to Lidar semantic segmentation by using a shared ViT backbone. Further, we compare finetuning against joint training in multi-modal perception. Finally, because our model MultiViT is able to process both camera and LiDAR data with the same encoder, we are able to investigate the shared feature space between the camera and LiDAR and analyze how this shared feature space affects performance.

We find that pretraining the ViT backbone with camera data generally enhances the performance of LiDAR semantic segmentation and show that when the model is randomly initialized the generated pixel-wise semantic labels improve performance. We also observe that having precise pixel-wise annotation is the key to the performance, no matter whether the camera data comes from the same domain or not. The model gains the most performance improvement on nuScenes LiDAR semantic segmentation by utilizing the ImageNet21K and CityScapes pretrained weights, despite the domain gap between ImageNet, CityScapes and nuScenes.

Besides, we conclude that joint training does improve performance compared to pretraining when training a randomly initialized model from scratch. When starting from a well-initialized ViT, e.g., initialized by ImageNet21K weights, finetuning leads to the most performance gain. We find that our joint training approach does not lead to a feature space where camera and LiDAR features are well aligned. Additionally, we show that by adding a contrastive loss, it is possible to create a shared feature space where the camera and LiDAR features are well aligned. However, this does not contribute to a better performance on LiDAR semantic segmentation.

Discussion. Although our pseudo label generation pipeline using SAM is able to generate semantic masks for the foreground classes, it is unable to provide masks for background classes. In comparison, CityScapes provides both foreground and background labels. The lack of background labels results in extensive class imbalance on the nuScenes images since more then 90% of the pixels are attributed to the background class. This may decrease the value of camera pertaining. Besides, using a larger SAM model, like the ViT-H model, might generate more precise pixel-wise semantic labels which can increase the value of camera pretraining.

To ensure a fair comparison between RangeViT and MultiViT_L, hyper parameters such as the number of epochs and the selected optimizer were kept the same, with only a different learning rate used for training MultiViT from ImageNet21K weights. There might be hyper parameters which are more suitable for joint training which can improve performance.

Future work. As mentioned, there are multiple design choices in terms of model optimization, data sampling, and hyperparameter tuning, which we leave for future works. Besides, it remains unclear why contradicting results arise when employing ImageNet pretrained weights to initialize ViT. Further analysis could provide more clarity. Also, the contrastive loss is currently used in a supervised setting. Although we demonstrate the contrastive loss does not improve performance in a supervised setting, it could be beneficial in a self-supervised setup where the Lovasz and Focal loss are removed, similar to contrastive learning approaches like [17, 18, 20, 25]. Using MultiViT offers the benefit of a shared main encoder for both the camera and LiDAR branches. This approach might eliminate the need for a frozen, well-trained 2D encoder as required in [17, 18, 20, 25].

References

[1] Angelika Ando, Spyros Gidaris, Andrei Bursuc, Gilles Puy, Alexandre Boulch, and Renaud Marlet. Rangevit: Towards vision transformers for 3d semantic segmentation in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5240–5250, 2023. 1, 2, 3, 5, 6

[2] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1090–1099, 2022. 2

[3] Maxim Berman, Amal Rannen Triki, and Matthew B. Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 5, 6, 7, 8, 9, 10

[5] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. In *The Eleventh International Conference on Learning Representations*, 2023. 1

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 3, 6, 7

[7] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In *Advances in Visual Computing - 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5-7, 2020, Proceedings, Part II*, pages 207–222. Springer, 2020. 4

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 1, 4, 5

[9] Di Feng, Yiyang Zhou, Chenfeng Xu, Masayoshi Tomizuka, and Wei Zhan. A simple and efficient multi-task network for 3d object detection and road understanding, 2021. 3

[10] Ronghang Hu and Amanpreet Singh. Unit: Multimodal multitask learning with a unified transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1439–1449, 2021. 3

[11] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, pages 18661–18673. Curran Associates, Inc., 2020. 9

[12] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. 2, 3, 4, 8

[13] Lingdong Kong, Youquan Liu, Runnan Chen, Yuexin Ma, Xinge Zhu, Yikang Li, Yuenan Hou, Yu Qiao, and Ziwei Liu. Rethinking range view representation for lidar segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 228–240, 2023. 1, 3, 4

[14] Jungbeom Lee, Jihun Yi, Chaehun Shin, and Sungroh Yoon. Bbam: Bounding box attribution map for weakly supervised semantic and instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2643–2652, 2021. 3

[15] Tingting Liang, Hongwei Xie, Kaicheng Yu, Zhongyu Xia, Zhiwei Lin, Yongtao Wang, Tao Tang, Bing Wang, and Zhi Tang. Bevfusion: A simple and robust lidar-camera fusion framework. In *Advances in Neural Information Processing Systems*, pages 10421–10434. Curran Associates, Inc., 2022. 1, 2

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 5

[17] Youquan Liu, Lingdong Kong, Jun Cen, Runnan Chen, Wenwei Zhang, Liang Pan, Kai Chen, and Ziwei Liu. Segment any point cloud sequences by distilling vision foundation models. *arXiv preprint arXiv:2306.09347*, 2023. 2, 12

[18] Yueh-Cheng Liu, Yu-Kai Huang, Hung-Yueh Chiang, Hung-Ting Su, Zhe-Yu Liu, Chin-Tang Chen, Ching-Yu Tseng, and Winston H. Hsu. Learning from 2d: Contrastive pixel-to-point knowledge transfer for 3d pretraining, 2021. 2, 12

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 5

[20] Anas Mahmoud, Jordan S. K. Hu, Tianshu Kuai, Ali Harakeh, Liam Paull, and Steven L. Waslander. Self-supervised image-to-point distillation via semantically tolerant contrastive loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7102–7110, 2023. 2, 12

[21] Deepak Pathak, Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional multi-class multiple instance learning, 2015. 3

[22] Guocheng Qian, Abdullah Hamdi, Xingdi Zhang, and Bernard Ghanem. Pix4point: Image pretrained standard transformers for 3d point cloud understanding, 2024. 1, 2

[23] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 1, 3, 5, 6

[24] Francisca Rosique, Pedro J. Navarro, Carlos Fernández, and Antonio Padilla. A systematic review of perception system and simulators for autonomous vehicles research. *Sensors*, 19(3), 2019. 1

[25] Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, and Renaud Marlet. Image-to-lidar self-supervised distillation for autonomous driving data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9891–9901, 2022. 2, 12

[26] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[27] Chunfeng Song, Yan Huang, Wanli Ouyang, and Liang Wang. Box-driven class-wise region masking and filling rate guided loss for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[28] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7262–7272, 2021. 1, 6

[29] Weixuan Sun, Zheyuan Liu, Yanhao Zhang, Yiran Zhong, and Nick Barnes. An alternative to wsss? an empirical study of the segment anything model (sam) on weakly-supervised semantic segmentation problems, 2023. 3

[30] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 5

[31] Sourabh Vora, Alex H. Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[32] Shiming Wang, Holger Caesar, Liangliang Nan, and Julian F. P. Kooij. Unibev: Multi-modal 3d object detection with uniform bev encoders for robustness against missing sensor modalities, 2024. 2

[33] Yi Wang, Zhiwen Fan, Tianlong Chen, Hehe Fan, and Zhangyang Wang. Can we solve 3d vision tasks starting from a 2d vision transformer?, 2022. 1, 2

[34] Chenfeng Xu, Shijia Yang, Tomer Galanti, Bichen Wu, Xiangyu Yue, Bohan Zhai, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Image2point: 3d point-cloud understanding with 2d image pretrained models, 2022. 2

[35] Dongqiangzi Ye, Weijia Chen, Zixiang Zhou, Yufei Xie, Yu Wang, Panqu Wang, and Hassan Foroosh. Lidarmultinet: Unifying lidar semantic segmentation, 3d object detection, and panoptic segmentation in a single multi-task network, 2022. 3

[36] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8: 58443–58469, 2020. 1

[37] Zixiang Zhou, Dongqiangzi Ye, Weijia Chen, Yufei Xie, Yu Wang, Panqu Wang, and Hassan Foroosh. Lidarformer: A unified transformer-based multi-task network for lidar perception, 2024. 3

MultiViT: 2D to 3D transfer learning using a jointly optimized Vision Transformer without the need for image labels

Supplementary Material

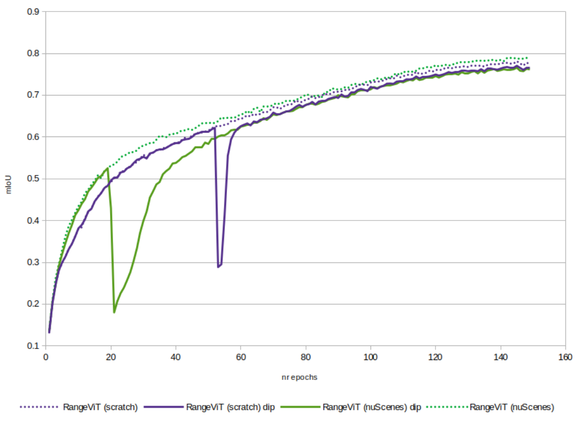


Figure 9. The training development of RangeViT from scratch and RangeViT pretrained on the nuScenes images with CameraViT where the dotted line shows the training development without a dip and the continuous line shows the training development with dip. Important to note is that when a dip has occurred the model is not able to fully recover to it's original state.

6. Training dip

During training of the different models occasionally a large sudden dip in model performance occurred, after which the model often was not able to recover to it's original state. This chapter will explain the occurrence of this phenomenon.

The occurrence of this phenomenon has been random and it has occurred for the training of MultiViT, CameraViT and RangeViT. The problem is visualized in Fig. 9. This figure shows the performance of the model on the train set when training RangeViT. The dotted line shows the training development of the model without the dip while the continuous lines show the model performance with the dip occurring. When the dip occurs the model is often able to recover to a decent performance, but the model will always under perform compared to when the dip does not occur. This is why it is important to address this.

Several attempts have been made to find the cause of this dip. An extensive analyses of the code has been done to investigate if this dip occurs due to a bug in the code. In this analyses no bug has been found. Besides, an analyses has been done which investigates if a faulty sample can be the cause of this dip. A faulty sample which causes an extremely high loss and therefore a wrong, but large update step could cause the model to degrade. Tab. 7 shows this analyses. From this table we can conclude that at the occurrence of the first high loss the model has already poor performance compared to earlier iterations. This shows that the loss grows over several steps when such a dip occurs and there is not a single mini batch which causes an extremely high loss. Lastly the optimization hyper parameters such as the learning rate have been adjusted to test it's effect. The dip still occurred while adjusting to a lower learning rate.

Iteration	**Loss**	**mIoU**	**Note**
139	0.61	52.0	Model performs well
633	1.21	30.7	First occurrence of high loss
634	1.61	20.9	Model degrades
635	1.53	17.8	Model degrades
636	1.49	18.2	Models starts recovery

Table 7. This table shows the results of an experiments to better understand the occurence of the dip. During the training of the model, several checkpoints have been saved at the moment a high loss occured. These checkpoints at different iterations have been used for the same minibatch. The loss and mIoU for this mini batch are reported in the table.

These analyses show several hints on the cause of the dip during the model training, but no clear reason for the occurrence has been found nor have the analyses convincingly excluded any of the earlier mentioned reasons. Restarting the training from the latest training checkpoint before a dip occurs has been an effective and trustworthy solution. In almost all cases a dip did not reoccur when the training was restarted from such a checkpoint which made it possible to receive all the results without a dip in the training curves.