3D Semantic Segmentation for Autonomous Cars

Yuntao Ma
Stanford University
yma42@stanford.edu

Albin Mosskull
Stanford University
mosskull@stanford.edu

Alana Xiang
Stanford University
zxiang@stanford.edu

1. Abstract

Fully autonomous vehicles can bring great benefit to society, and creating a good perception is a crucial part of making that happen. 3D semantic segmentation is perception the task of predicting the semantic class of each pixel in a Lidar image, which is what we explore as part of the 2022 Waymo Open Dataset Challenge. This dataset contain about 30 000 samples of Lidar and Camera data gathered from Waymo vehicles. The most common approaches to solve the problem is to either operate on point-clouds or project those point-clouds to a 2D representation, so that 2D semantic segmentation methods can be applied. We use the projection-based method SalsaNext trained on the Lidar images as our starting point, and try variations using naive late fusion with image data as well as changing the backbone to ConvNeXt. In the end, we achieve similar results with SalsaNext as with the variations, which was a mIoU of 0.56 on the test dataset, compared to the competition winner's 0.71.

2. Introduction

Solving the problem of making vehicles fully autonomous would bring great benefit to society. It could enable cheaper, faster and safer transportation of both people and goods, improving quality of life for all. In order to make autonomous vehicles truly safe, one of the most critical components is to have a good perception system. Without a good understanding of the environment through perception, it is impossible for an autonomous agent to make good decisions.

The perception systems that are used by autonomous vehicles today often use multiple sensors, to gather as much information as possible from the surroundings. Some of the most commonly used sensors are RGB cameras and Lidar. Using this sensory input, the perception system performs a task such as object detection, object tracking, and semantic segmentation. In this project we will explore semantic segmentation, which is the task of classifying each input (such as each pixel in an image), rather than determining a bounding box, as is the task in object detection. Semantic segmentation gives an unparalleled understanding of the sensory input, as it gives a clear picture of every single input to the perception system is. This is easy to interpret for a human and gives rich information that could be used for other parts of the autonomous vehicles algorithms.

Our project is more precisely about exploring improvements to 3D semantic segmentation, the task of classifying the semantic label of Lidar Range Images, as part of The Waymo 2022 Open Dataset Challenge. That is, our task will be to classify each of the Lidar returns to a semantic class. There are 23 different classes commonly seeing around autonomous vehicle setting in the Waymo Open Dataset [17], such as "Car", "Pedestrian" or "Building".

The input to our will be the data in Waymo's dataset, which contain Lidar and camera images for all of the vehicle's sensors at a given time frame. We then use a projection-based semantic segmentation network, a type of CNN, to produce a label for each pixel in the Lidar data. Our goal is to maximize the mean Intersection-Over-Union (mIoU) metric.

3. Related Work

In [7] He et al. presents a survey of deep learning based 3D segmentation techniques. These techniques were categorized by the "representation" of input data. The four categories presented in the He et al. paper were RGB-D, Voxel, Projected Images, and Point Based representations.

A leading RGB-D segmentation technique is SISNet [2], which was presented by Cai et al. at CVPR 2021. SISNet is able to tell apart similar objects, and is "capable of inferring fine-grained shape details." While many Lidar based techniques can struggle to tell apart objects with similar shapes, RGB-D techniques such as SISNet can use additional information such as color and texture. Unfortunately, depth reconstruction from an RGB image is simply not competitive with leveraging Lidar range data.

A state-of-the-art voxel-based segmentation technique is VMNet [8], which was presented in ICCV 2021. VMNet cleverly incorporates geodesic information, avoiding many pitfalls of conventional voxel-based pitfalls, such as ambiguating ”spatially close objects.” Unfortunately, voxel-based segmentation methods are not the most accurate for outdoor scenes.

The last two categories, Projected Images and Point Based representations, expect Lidar data as input.

The class of projection-based methods a point cloud onto one or more 2D surfaces, which then enables 2D segmentation methods such as 2D CNNs to be applied. Within these methods, some of the leading examples are RangeNet++, SqueezeNetv3 and SalsaNext. Since our dataset contains Lidar Range Images rather than point cloud data, we choose to use projection-based methods since they can be directly applied to the input data without performing transformations.

RangeNet++ [13] is a well known projection based method, which uses a Darknet backbone together with a kNN operation during the postprocessing to better recover the labels of the points in the point cloud. RangeNet++ was able to perform segmentation remarkably quickly, which is relevant to real-world applications. However, it does somewhat sacrifice accuracy to accommodate online training. Since we do not have access to point clouds and did not have stringent requires on prediction time, we didn’t find RangeNet++ to be a good fit for this competition.

SqueezeSegV3 [9] introduces a new feature called Spatially-Adaptive-Convolution (SAC) that adopts different filters for different locations in the input image. The filter can therefore learn to recognize shapes even if they are further away and have lesser intensity. They also implement this in a way that enables efficient calculations.

SalsaNext [4] further improved upon RangeNet++ by using ResNet-like backbone. It introduce Contextual Module in the beginning of the network to aggregate global context information. It also use dilated convolution to increase receptive field without greatly increasing the number of parameters. SalsaNext replaced the down sample method with Average Pooling and up sample method with Pixel Shuffle to further reduce the number of parameters without a decrease in performance. Finally, it also added Lovasz-Softmax loss [1], a convex Lovasz extension of mIoU function, which encourage the network to improve mIoU scores.

The current state-of-the-art techniques in Lidar 3D segmentation are generally Point Based. This is likely attributable to inherent flaws in projection. According to the Zhu et al. Cylinder 3D paper [21], projecting 3D data onto 2D surfaces distorts geometric information. At the time of its publication, Cylinder 3D outperformed all other models on the SemanticKITTI automotive scene dataset. It would later be out-performed by $(AF)^2$-S3Net [3], which fuses point based data with voxel data.

Another novel Point Based segmentation technique Lif-Seg [20]. Lif-Seg infuses the point cloud with a 3x3 pixel map, which enables the network to use the texture and colors in its 3D semantic segmentation to differentiate similarly shaped objects.

4. Dataset

4.1. Data

We use the Waymo Open Dataset [17] for our project. The dataset is made up of multiple contexts. Each of the contexts consists of roughly 20 seconds segments of real-world driving data recorded using with a Waymo vehicle. Each segment contains multiple frames recorded at around 30 Hz. Each frame can be considered as a single unit of data point in the dataset.

Each frame contains camera data in JPEG format, Lidar range images, a 2D-to-3D correspondence mapping between the two sensors and 3D segmentation label in range image format. There are 23 different classes. It also contains irrelevant data to our task, such as 2D object labels, 3D object labels and object key points label. We will now describe the different data that is in each frame in more detail.

The camera images are RGB images from five cameras associated with five different directions. The directions are front, front left, front right, side left, and side right. The front, front left, front right camera have a resolution of (1920 x 1280). The side left, and side right camera have a resolution of (1920 x 886). Due to the way the sensors are mounted onto the vehicle, they have different, partially overlapping field of views. The cameras covers a greater height, and also have a much higher resolution compared to the Lidar senors. The cameras do not cover the area behind the vehicle, which the Lidar does. An example of how the camera images can look like is shown in Fig. 1, for two of the five cameras.

Figure 1. Two camera images for a certain frame.

Each frame contains data from five Lidars - one mid-range Lidar (top) and four short-range Lidars (front, side left, side right, and rear). The mid-range Lidar is truncated to 75m, and the short-range Lidars is truncated to 25m. The top Lidar has a resolution of 64 x 2650. The strongest two intensity returns are provided for all five Lidars. The Lidar data is presented as a range image. A range image represents a Lidar point cloud in the spherical coordinate system, where each row corresponds to an elevation, and each column corresponds to an azimuth. Each Lidar range image has 3 channels, which contain range, Lidar intensity and Lidar elongation. The whole Lidar image has a much greater width than height which makes it difficult to show, but we include a snippet of a Lidar return in Fig. 2.

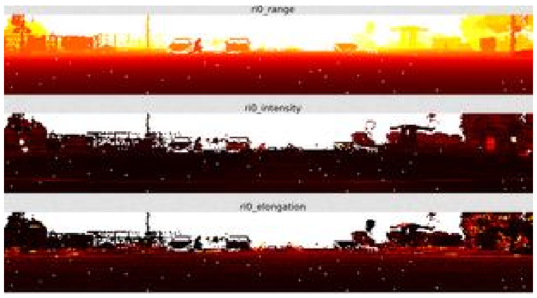

Figure 2. A snippet of the Lidar returns for a given frame, for the first return, plotted as a heat map. The channels are from top to bottom: Range, Intensity, Elongation.

The 2D-to-3D correspondence map has the same shape as the Lidar image which has 6 channels for Lidar to camera projection. These channels contain information about the camera id and locations of pixel in the image associated with that Lidar image point if any. The projection method used takes rolling shutter effect into account.

The 23 classes 3D segmentation label are only provided for the top Lidar sensor with 2Hz frequency. This means not all Lidar data has 3D segmentation label and as such we are only expected to provide a semantic segmentation label for the top Lidar.

4.2. Preprocessing

Waymo has done the majority of the preprocessing such as timestamp alignment, and projection to provide usable data out of box. The data is provided in TFRecord format. Since our choices of deep learning framework is PyTorch, and a portion of data is irrelevant to our task, we do some preprocessing to accelerate the training. We extract the relevant data and convert it to the HDF5 format. The relevant data for us are the frames that have a 3D segmentation label. We create data points consisting of the top Lidar range image, all 5 RGB images, the 3D segmentation labels, the 2D-to-3D correspondence, and spherical pixel projection using the 2D-to-3D correspondence. In total we have 23691 training samples, 5976 validation samples, and 2982 test samples. The test sample does not have any labels and will be used for challenge submission. We also computed class frequency across training data and validation data. This will be used for the weighted cross entropy later on.

For training, we truncate the range channel to a max value of 75, and intensity, elongation channel to max value of 1.5 using Lidar sensor specification to avoid outliers or faulty points skewing the data. Then all input data is normalized to [0,1] before it is input into the network. We also apply random flip along y-axis with a probability of 0.5.

5. Method

As previously mentioned, we chose to use projection-based methods for the semantic segmentation, based on our data format. Based on our findings during the literature review, SalsaNext is one of the best projection methods. We therefore chose to first implement it, and then improve upon it, all of which will be described in this section. We implement all of our methods using Pytorch [14] and Pytorch Lightning [5].

5.1. SalsaNext

5.1.1 Network Architecture

SalsaNext is built with a standard encoder-decoder architecture, as illustrated in Fig. 3. The encoder is consists of 3 contextual block and 5 downsample blocks that down sample the features by 16x. The decoder consists of 4 upsample blocks that up sample hidden feature to original image size and a 1x1 convolution classification head. Each block are design based on basic ResNet [6] block.

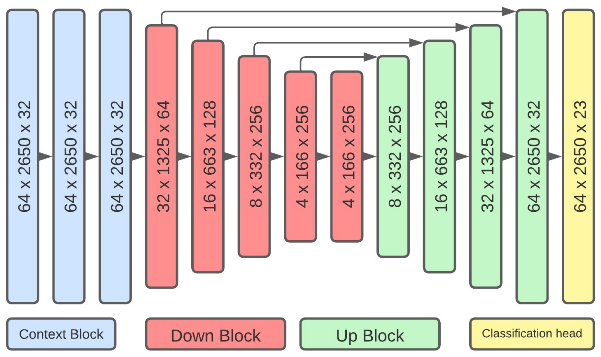

Figure 3. SalsaNext network architecture, the number shown are the output dimension of each layer

Each contextual block is made up of two dilated convolution layers with skip connections, without changing the dimension of the features. This aims to aggregate context information and gather global information, alongside the detailed spatial information.

Both the downsample block and the upsample block are made using one 1x1 convolution layer follow by three dilated convolution layers. Then, each dilated convolution output are concatenated and followed up by a 1x1 convolution with a residual connection in order to let the network gather information from different depth with various receptive fields. The 2x downsample is achieved using a average pool layer with 3x3 kernel and stride 2. The 2x up sample is achieved using a pixel shuffle layer.

5.1.2 Dilated Convolution

Having a large receptive field play crucial role in extracting good spatial features. A naive way to increase receptive field is increasing the size of the kernel. Another way to increase the effective receptive field is to increase the depth of convolution block. However, one of the challenge facing 3D semantic segmentation using projected Lidar image is the large input and output dimension, which would produce large intermediate memory during training. Using two method mentioned above will increase the number parameter and training memory drastically. Instead, we can use dilated convolution [19] to address this issue.

In traditional convolution, eq. 1, the sliding window and kernel has the same shape, which means the spatial area of image the kernel is visible to is the same to the spatial area of the kernel. Thus, the receptive field is the same as the kernel size.

$$(I * K)[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} I[i+u, j+v]K[u,v] \quad (1)$$

The dilated convolution, eq. 2, increase the size of sliding window by adding ”gap” between each element of the sliding window, effectively increase spatial area of image the kernel is visible to.

$$(I *_l K)[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} I[i+lu, j+lv]K[u,v] \quad (2)$$

Within each convolution block, the feature is produce by a layer with 3x3 kernel and 1 dilation, a layer with 3x3 kernel and 2 dilation, and a layer with 2x2 kernel and 2 dilation.

5.1.3 Pixel-Shuffle Layer

In a traditional semantic segmentation task, upsampling is achieved using transpose convolutions which contain large amount of parameters. We replaced the transpose convolutions with the pixel-shuffle layer [15]. Pixel shuffle operation can be 3

$$PS(I) : I[H \times W \times Cr^2] \rightarrow I[Hr \times Wr \times C] \quad (3)$$

where H, W, C, and r the height, width, channel number and upscaling ratio, respectively. Pixel shuffle operation can be considered as reshaping pixels from the channel dimension to spatial dimension.

In this implementation pixel shuffle does not work well when the dimension size is an odd number. Since the width of our input dimension (64 x 2650) is not divisible by 16 (4 2x downscale and upscale). We added adaptive padding to automatically adjust the intermediate spatial dimension.

5.1.4 Baseline

Due to the complexity of our problem, a simple baseline model was not really applicable to our problem. The first network that we tried was therefore SalsaNext, with the features described above, but using unweighted Cross-Entropy as the loss instead of the more sophisticated options we will describe later. We used certain code snippets from the original SalsaNext [4] implementation, although we re-implemented most of it ourselves to fit in our modular training pipeline.

5.2. ConvNeXt Encoder

The encoder block of the SalsaNext is based on ResNet. A natural extension to SalsaNext that we thought of was therefore of applying state of art convolution neural network design that improves upon the original ResNet. ConvNeXt [11] is a modernized version of the original ResNet. The authors took inspiration from recent development within vision transformers and employed features such as layer norm, depth wise convolution, and inverted bottle neck. For this work, we implemented an encoder block that was inspired by features adopted in ConvNext by replacing the dilated convolution layer with a depth-wise 7x7 convolution layer, a 1x1 convolution with 4x more channels, and a final 1x1 convolution.

Beside changing convolution block architecture, we also replaced the batch normalization with layer normalization, replaced Leaky ReLU activation with GELU, reducing number of normalization and activation, and adding stochastic depth using drop connect [18].

5.3. Fusion

5.3.1 Late Fusion

Another separate track we explore is to leverage the camera images for the segmentation. Our task is to do 3D semantic segmentation, which means that a major challenge with using camera data is how to do the conversion between camera and Lidar data, specifically how to utilize the sparse (from a camera point of view) Lidar labels.

One strategy that we considered for this was to train a 2D semantic segmentation network on the full camera images, and then combine that with output of a Lidar-only network in the end. The problem with this approach is that there labels available are very sparse, as we found that less than 1% of pixels in a camera image have a label if we project the Lidar labels onto the pixels. The reason for this is that the cameras simply have a much greater resolution. Because of this finding, we eventually abandoned the idea.

What we instead did was to try projecting camera pixels onto a the Lidar range image. That is, for each Lidar range pixel, find the closest matching image pixel and add it there. This was done using the mapping function Waymo provided. This yields a result such as what is shown in Fig. 4, which we call a projected pixel image.

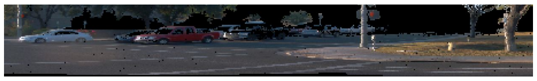

Figure 4. A snippet of the projected pixel image.

The benefit of using this approach is that we now have an input format on the exact same shape as our Lidar range image, and corresponding labels for each datapoint. This enables us to use the same network as we use for the Lidar range images, as the only difference will be the number of channels.

One important thing to note is that the cameras do not cover all of the Lidar points, and as such there will be regions that the projected pixel image does not cover.

Given that we have this projected-pixel based network, another issue to tackle will be how we can merge it with the Lidar-only model. For this we took inspiration by Simonyan and Zisserman [16] that use Two-Stream Convolutional Networks for Action Recognition in Videos. They find that naively fusing two sources of information yielded better result, and employing a SVM for the fusion only marginally improved the results. We therefore combine our results by

$$y_{comb} = K \cdot y_{Lidar} + (1 - K) \cdot y_{img} \tag{4}$$

and

$$y = I(y_{comb}) + I^{-1}(y_{lidar}), \tag{5}$$

where y is the final output, K is a coefficient for how much to trust the Lidar compared to the image models, and I is a map that is 1 where we have image data and 0 where we do not have image data (where we thus only use the Lidar output.

5.3.2 Early Fusion

The structure and information in a RGB pixel and a Lidar range image pixel is inherently different. Naively concatenating the projected pixel and range image in an early fusion step might not be ideal. However, this can be fixed by passing the projected pixel and the range image through separate contextual modules and concatenating the intermediate projected pixel features and range image features together before feeding into the downsample layers. In this way, the contextual layer can capture the information in both RGB space and range image space. In our implementation of this, the original single 32 channel contextual block are split into two 16 channel contextual block with one for range image input and one for projected pixel input.

5.4. Loss Function

The performance of the classification model with probability output can be measured using cross entropy. By maximizing the cross entropy, we maximize the log likelihood of the prediction with respect to the correct class. The cross entropy loss can be formulated as

$$\mathcal{L}_{ce}(y, \hat{y}) = -\sum_i p(y_i) \log(p(\hat{y}_i)). \tag{6}$$

Autonomous vehicle segmentation data often has imbalanced classes. For example, roads and buildings cover significantly larger surface area than other classes such as bicycles. To handle class imbalance, we can increase the loss of the minority class to encourage the network to learn to learn the representation equally across different classes. The weighted cross entropy loss can be formulated as

$$\mathcal{L}_{wce}(y, \hat{y}) = -\sum_i w_i p(y_i) \log(p(\hat{y}_i)) \tag{7}$$

where weight w_i is computed using class frequency f_i

$$w_i = \frac{1}{\sqrt{f_i}}. \tag{8}$$

In recent literature [10], it was found that by perturbing the first coefficient of the polynomial extension of the loss function could increase performance in classification, object detection and semantic segmentation. This is called the poly-1 loss with respect to the Cross Entropy loss and is defined as

$$\mathcal{L}_{poly1}(y, \hat{y}) = \mathcal{L}_{ce/wce} + \epsilon(1 - p(y_i)). \tag{9}$$

The intersection-over-union (IoU) score (see section 6.2) is the main evaluation metric of this challenge. However, the IoU function is an non-differentiable discrete function. Directly maximizing IoU score is therefore not possible. In [1], the author propose a convex surrogate of the IoU function by using Lovasz extension for submodular functions. The Lovasz-Softmax Loss is formulated as

$$\mathcal{L}_{ls} = \frac{1}{|C|} \sum_{c \in C} \overline{\Delta_{J_c}}(m(c)) \tag{10}$$

where

$$m_i(c) = \begin{cases} 1 - x_i(c) & \text{if} c = y_i(c) \\ x_i(c) & \text{otherwise} \end{cases} \tag{11}$$

where $|C|$ is the class number, $\overline{\Delta_{J_c}}$ is the Lovasz extension (convex surrogate) of the Jaccard index, $x_i(c) \in [0, 1]$ and $y_i(c) \in \{-1, 1\}$ hold the predicted probability and ground truth label of pixel i for class c, respectively. Optimizing this function will therefore most directly optimize the objective in the competition.

The final loss function will be a convex combination of the above mentioned loss function.

6. Results

6.1. Setting

We trained all models for 100 epochs using AdamW [12] with a learning rate of 4e-3. We used a 5-epoch linear warm up and a cosine decaying schedule afterward. The learning rate is updated every single batch. We use an effective batch size of 50 and a weight decay of 0.05. We also have a early stop with patience value of 15. To prevent overfitting, we also applying a random flip along y-axis with probability of 0.5. We used distributed data parallelism to trained the model with 10 RTX A4000 GPU.

We did not perform and cross-validation due to each training taking around 12 hours. The proper parameter were found using coarse parameter search, i.e. we started with conventional parameter values, then adjusted them to observe effect on final evaluation result. The parameters above were found using the training with the best evaluation result.

6.2. Metrics

Our primary metric is the mean Intersection over Union, mIoU. IoU is defined as

$$\text{IoU} = \frac{\text{true positive}}{\text{true positive} + \text{false negative} + \text{false positive}} \tag{12}$$

and can most easily be understood as that the intersection of predictions and ground truth divided by the union of predictions and ground truth. If these two overlap completely, the IoU is one. Furthermore, mean IoU is defined as

$$\text{mIoU} = \frac{1}{N} \sum_{k=1}^{N} \text{IoU}_k, \tag{13}$$

where N is the number of classes. If a class is absent, the IoU for that class was set as 0 for the validation set, but 1 in the test dataset. This was due to a missunderstanding in the definition of the metric.

6.3. Quantitative Result

The results for our main metric mIoU for the different models on the validation dataset are shown in Tab. 1. Note that the first model is the baseline model for the project. Since the competition we took part in used this as the most important metric, we did not check our result using any other metric.

Backbone	Sensor	Loss Function	mIoU
SalsaNext	Lidar	Unweighted CE	0.38
SalsaNext	Cam	Unweighted CE	0.28
SalsaNext	Cam	WCE + LS	0.37
SalsaNext	Lidar	Poly1 WCE + LS	0.45
ConvNeXt	Lidar	WCE + LS	0.479
SalsaNext	Lidar+Cam	WCE + LS	0.490
SalsaNext	**Lidar**	**WCE + LS**	**0.492**

Table 1. Results of models on the validation dataset.

In sec. 6.5 we also showcase further qualitative results on the test dataset.

6.4. Qualitative Result

In order to understand what our model is doing, we can visualize our predictions as a heatmap. In Fig. 5 we overlay our predictions on the projected pixels, for a snippet of a Lidar range image from the validation set. In the picture, we notice that the cars have been captured well (in blue) and the people as well (in red).

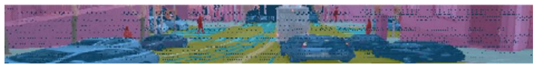
Figure 5. Predicted labels overlayed onto the projected pixels

In Fig. 6 we show which predictions we got correct for the same snippet as above. Plots of this kind has been one of our primary ways of finding where the model fails. Initially, we found that it often did bad on the minority classes, which caused us to change the loss accordingly. We have also noticed, as in Fig. 6, that errors often can arise on the edges of objects. It is possible this could be improved using some kind of post-processing, but we did not have time to try that. Also, we can notice that our model generally seems to perform quite well visually, compared to the quantitative metric result. This can be attributed to the fact that mIoU is quite an unforgiving metric, in the sense that scores get low even for slight missmatches in predictions and labels.

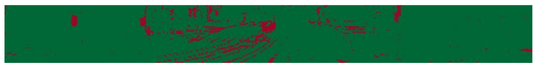
Figure 6. Correct and incorrect predictions, for a snippet of a model output. Correct in green, incorrect in red.

In order to further connect back to our problem setting, we can also project our semantic labels onto the 3D world, which is where the data was generated from in the first place. This gives the result shown in Fig. 7.

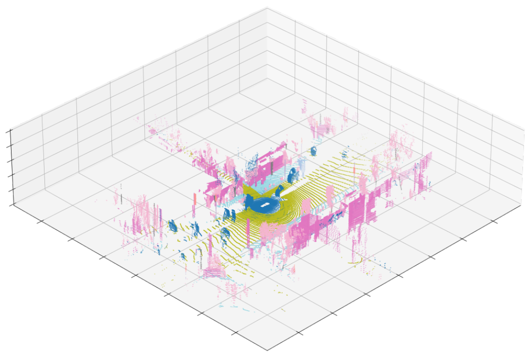
Figure 7. 3D Re-projection of the semantic segmentation prediction

We use multiple techniques in our model training to prevent overfitting. We apply a central dropout with drop out rate of 0.2 at output of each convolution block. For ConvNeXt encoder, we also use a stochastic depth with drop rate of 0.1. We also employed data augmentation, weight regularization, and early stopping. For all of our models, we did not observe any behavior of overfitting.

6.5. Competition results

The final submission model are trained jointly on both training and validation split. Since we do not observe any overfitting behavior in previous experiment, the model with highest training accuracy is selected. Our result for the different classes on the test dataset were as shown in Tab. 2. The classes we do worst on are ”Motorcyclist”, ”Other Vehicle” and ”Traffic Light”. Our best classes are ”Car”, ”Road” and ”Building”.

Class	IoU
Car	0.8705
Pole	0.6097
Other Vehicle	0.1918
Walkable	0.5954
Motorcycle	0.5624
Pedestrian	0.7275
Curb	0.6575
Other Ground	0.3961
Traffic Light	0.2292
Bus	0.6172
Bicycle	0.4836
Bicyclist	0.6275
Tree Trunk	0.5524
Sign	0.5276
Vegetation	0.7237
Truck	0.5336
Lane Marker	0.4045
Sidewalk	0.7982
Construction Cone	0.4138
Building	0.8938
Motorcyclist	0.0001
Road	0.8702

Table 2. IoU for different classes on the test dataset.

The Waymo Open Dataset Competition concluded on the 23rd of May. The winning entry used a Cylinder3D method and achieved a mIoU of 0.7118 compared to our result of 0.5585.

7. Conclusions

We found that the Lidar-only SalsaNext method worked the best. It performed a lot better when we used the Lovasz-Softmax + Class-weighted cross-entropy loss. This is not too surprising, as using a loss that directly optimizes the metric of interest is very sensible. Weighting the classes to promote the minority classes also seemed important based on our earlier findings, such as in our qualitative analysis when we saw that minority classes weren't noticed.

Doing the naive late fusion with the projected pixel image model did not improve the performance further. The reason for this is likely that the Lidar by itself already yielded a much better result and that the naive fusion was not able to improve for the areas where pixel information might have been able to help. It could be an interesting direction to further explore fusion, perhaps by using more pixels, by doing some smarter type of downsampling or by fusing the information in another way. It is also possible that late fusion can work if one uses a method more advanced than just naively combining the two streams. For example one would not want to use pixel values when it is dark outside, as they will likely only disturb the performance then, which a more advanced fusion method might be able to learn.

Based on the results in the competition, it also seems like point-based methods performed best, which we we did not initially think would be the case. For our part it would be interesting to explore those further as a future work, as they might be more suited overall for this type of challenge.

8. Contributions & Acknowledgements

8.1. Contributions

Yuntao Ma implemented the data processing pipeline, modular training framework with Pytorch and Pytorch Lightning, SalsaNext network, and explored using ConvNext as encoder.

Albin Mosskull wrote the naive late fusion code, explored approaches for using SqueezeSegv3 and the full-image based approach (though they were not used in the end) and wrote the majority of the midpoint and final report text.

Alana Xiang wrote exploratory code for using RangeNet++ on Lidar data, which we later decided to replace with a SalsaNext model. Alana also wrote the initial submission code.

8.2. Acknowledgements

We would like to thank Stanford Artificial Intelligence Laboratory (SAIL) for using their compute cluster to train our models.

We used code from Waymo for accessing the dataset [Github].

We used code from SalsaNext as a starting point for implementing their algorithm [Github].

References

[1] Maxim Berman and Matthew B. Blaschko. Optimization of the jaccard index for image segmentation with the lovász hinge. *CoRR*, abs/1705.08790, 2017. 2, 6

[2] Yingjie Cai, Xuesong Chen, Chao Zhang, Kwan-Yee Lin, Xiaogang Wang, and Hongsheng Li. Semantic scene completion via integrating instances and scene in-the-loop. 2021. 1

[3] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. (af)2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. 2021. 2

[4] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast semantic segmentation of lidar point clouds for autonomous driving. *CoRR*, abs/2003.03653, 2020. 2, 4

[5] William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. 3

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 3

[7] Yong He, Hongshan Yu, Xiaoyan Liu, Zhengeng Yang, Wei Sun, Yaonan Wang, Qiang Fu, Yanmei Zou, and Ajmal Mian. Deep learning based 3d segmentation: A survey, 2021. 1

[8] Zeyu Hu, Xuyang Bai, Jiaxiang Shang, Runze Zhang, Jiayu Dong, Xin Wang, Guangyuan Sun, Hongbo Fu, and Chiew-Lan Tai. Vmnet: Voxel-mesh network for geodesic-aware 3d semantic segmentation. 2021. 2

[9] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016. 2

[10] Zhaoqi Leng, Mingxing Tan, Chenxi Liu, Ekin Dogus Cubuk, Xiaojie Shi, Shuyang Cheng, and Dragomir Anguelov. Polyloss: A polynomial expansion perspective of classification loss functions. 2022. 6

[11] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 5

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. 2017. 6

[13] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet ++: Fast and accurate lidar semantic segmentation. pages 4213–4220, 11 2019. 2

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 3

[15] Wenzhe Shi, Jose Caballero, Ferenc Huszár, 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. *CoRR*, abs/1609.05158, 2016. 4

[16] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014. 5

[17] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2

[18] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. 28(3):1058–1066, 17–19 Jun 2013. 5

[19] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. pages 636–644, 2017. 4

[20] Lin Zhao, Hui Zhou, Xinge Zhu, Xiao Song, Hongsheng Li, and Wenbing Tao. Lif-seg: Lidar and camera image fusion for 3d lidar semantic segmentation. *CoRR*, abs/2108.07511, 2021. 2

[21] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. 2020. 2