MiniOverlapNet: Lightweight LiDAR Loop Closure Detection

Devin Ardeshna, *Stanford University*
Albert Chan, *Stanford University*

ABSTRACT

It is still essential for autonomous vehicles to localize within their environment even in GNSS-denied scenarios. However, since 3D LiDAR SLAM algorithms suffer from drift over time without an absolute reference, loop closure is used to create a constraint between matching scans which improves both the map and pose estimation. In this project, we propose a deep learning approach called MiniOverlapNet for lightweight 3D LiDAR loop closure detection. 3D scans are converted to 2D range images, which are fed into MiniOverlapNet. The network produces a prediction of overlap percentage between the scans and estimates relative orientation. We base our architecture off of OverlapNet which has good performance but heavy runtime cost which makes it unrealistic for real-time loop closure. By reducing the network size by 5x we increase inference capacity by 7x. Our approach maintains similar performance for overlap prediction but show slightly degraded performance for relative orientation estimation. The results are promising for using deep learning for real-time loop closure.

I. INTRODUCTION

In GNSS denied scenarios, it is essential for an autonomous vehicle to still be able to localize in its surroundings and navigate. Common approaches to this problem generally involve a fusion of sensor modalities such as camera, LiDAR, and odometry data. Regardless of the approach, without GNSS correction, the pose will increasingly drift away from the ground truth over time. Pose graph optimization and loop closure detection is essential to correct this drift without GNSS data. Pose graph optimization creates graph nodes as the autonomous vehicle moves along, with edges representing relative pose estimates between each graph node. When the robot revisits a previously visited location, the robot links the poses of the matched nodes, and by running least squares optimization to minimize the measurement residual, the algorithm ”closes the loop” on its pose, updating the locations of all nodes in the loop based on the confidence of each relative pose. Not only does the loop closure correct the pose of the vehicle, it also increases the accuracy of the built world map in the localization process. This paper will focus on the loop closure aspect of this problem, specifically the challenge of accurately detecting loop closures with LiDAR scans.

To illustrate the importance of accurate loop closure detection, consider a case where an autonomous vehicle has identified a false positive loop closure. This situation is illustrated in Figure 1. Notice that after the false loop closure is detected, there is an improper association between two poses, and the relative pose constraints between each node pull the remaining nodes out of shape and further away from the ground truth. In fact for this specific problem, it is most desirable to have a detection algorithm that generates no false positives, even at the cost of more false negatives.

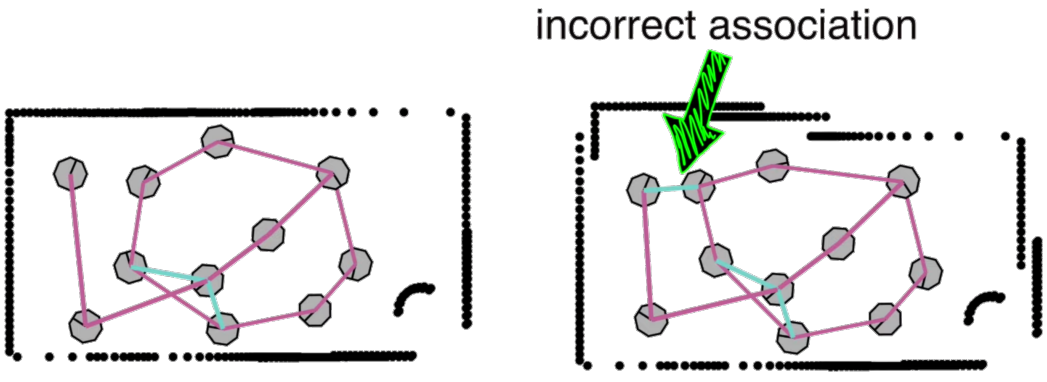

Figure 1: Effect of an incorrect loop closure on a 2D pose graph. Note how the border on the left resembles the rectangular perimeter of the room but is distorted once the the incorrect loop closure is introduced on the right.

II. RELATED WORK

SLAM algorithms can typically be divided into either visual based SLAM or LiDAR based SLAM approaches. Visual SLAM analyzes features in camera images to generate loop closures. Common approaches to Vision based loop closure include bag of words, keyframe, and even some learning based vision based method. LiDAR SLAM uses LiDAR point cloud data to perform scan matching for loop closure.

1. Geometric LiDAR Loop Closure

This paper will instead focus on 3D-LiDAR based loop closure. The most basic approach to matching a pair of LiDAR point clouds is Iterative Closest Point (ICP). ICP starts with two LiDAR scans that are assumed to have some point correspondences. Association is then made between points in each of the scans. There are different ways to choose what subset of the points are used. One option is to use all of the points in each scan, but this can be computationally expensive. Alternatives include randoms sampling, uniform sampling, or normal-based sampling where the number of points sampled from a region is chosen based on the rate of change of normals (e.g. curvature) in that region. Similarly, there are many different ways that association of points between the scans can be accomplished, giving rise to several different variants of ICP. The simplest method, also known as "Vanilla ICP" is proposed by Zhang (1994) using k-d trees to find the closest points between scans. Another alternative is closest compatible point, where compatibility constraints between points are taken into account. Finally, point-to-plane matching finds the nearest point on the normal surface defined by the second scan.

Once point associations are made, ICP computes the Euclidean error between each point correspondence and finds a rigid body transformation matrix that minimizes this error. Finally, the scans are re-positioned atop one another using the computed rigid body transformation, and the process repeats to convergence. ICP works particularly well for matching adjacent scans that are taken sequentially.

While ICP is a powerful method, it is not robust enough to be a general loop closure detector, capable of handling a wide range of rotations and transformations. Previous work has used ICP's residuals as a way to determine if two scans share similar features, but this has limitations Röhling et al. (2015). Furthermore, ICP converges slowly, and is sensitive to outliers, missing data, and partial overlaps. That is to say, ICP may detect the same car in two completely different locations, but because it is the same car, ICP may potentially latch on to it as a common geometric feature which minimizes the residuals for a large portion of the point cloud, generating a spurious minimum. Thus, a preprocessing step is needed to first generate a list of candidate scan pairs that are known to have some overlap. Then ICP can be used to fine tune the transformation between these pairs.

Another common technique for LiDAR scan matching is Normal Distributions Transform (NDT) matching. This method, initially developed for the 2D case by Biber and Straßer (2003), and expanded to the 3d case by Merten (2008), transforms the LiDAR point cloud into a probability distribution that gives the probability of sampling a point belonging to the cloud at a given spatial coordinate. A maximum likelihood estimator is then used to estimate the transformation between a pair of scans. Röhling et al. (2015), use a similar method, using 1D histogram similarity comparisons and the Wasserstein metric instead of probability distributions to create an lightweight loop closure detector that can be run in real time.

Significant prior work has been done on generating loop closure candidates that could be refined with ICP. Some other approaches to the loop closure problme include a work by Steder et al. (2010). They propose a new feature descriptor called normal aligned radial feature (NARF) to determine correspondences between 3D LiDAR scans. In contrast, He et al. (2016) presents M2DP, where scans are projected into multiple reference planes. The density distributions on each of these 2D planes are then used to generate feature descriptors for matching.

Overall, geometric methods show promise for loop detection but often show poor robustness to outliers with missing data or scenes with partial matches. In addition, in the case of algorithms like NARF, features are handcrafted with particular scenarios in mind, making it hard to generalize to new settings or scenarios.

2. Learning Based LiDAR Loop Closure

More recently, deep learning approaches for scan matching and loop closure have become more popular in the literature. PointNetVLAD, developed by Uy and Lee (2018) is an approach that combines PointNet and netVLAD. PointNet is able to directly learn features from 3D lidarscans, and netVLAD implements a bag of words type approach for scan matching. On the other hand, Barsan et al. (2020) approached the problem by formulating LiDAR localization as a Deep Recursive Bayesian Inference Network. Most recently OverlapNet Chen et al. (2021) is a learning based approach to the loop closure problem, with the novel idea of estimating percentage overlap between each LiDAR scan pair. OverlapNet will be the basis around which our algorithm is built due to its stellar performance and robustness.

Each 3D LiDAR scan is projected onto a 2D cylindrical map to generate depth, normal, and intensity feature embeddings. In addition input layers with semantic classification of objects in the map can be added. The machine learning model is a Siamese network with two legs with shared weights. Each leg contains a series of convolution layers to extract features, before passing

the intermediary feature vector into a delta head and a correlation head. The delta head consists of another series of convolution layers before a final dense layer that outputs the predicted overlap of the system. The correlation head performs cross-correlation between the feature vectors from each scan and estimates the relative orientation of the two scans.

As the authors describe, the main drawbacks of OverlapNet include its large network size and computational cost to for both training and inference. In its current form, OverlapNet shows good performance but cannot be used for real-time inference. The authors estimate that with desktop grade neural network training hardware, 200 pairwise comparisons can be made per second. If this were to run in a vehicle on a smaller onboard computer which also has to handle other critical vehicle proceses, this number would be drastically reduced. With roughly 10 new LiDAR scans per second it would be computationally infeasible to do more than a few pairwise comparisons for each incoming scan. Thus, to make OverlapNet feasible the network inference has to be sped up. As described in the subsequent sections, our approach uses a similar structure to OverlapNet, while reducing the size of the network. We believe this should drastically increase the inference capability of the system while maintaining comparable performance.

III. PROBLEM STATEMENT

Since ICP provides fine-tuning for scans with known correspondence, the problem reduces to finding loop closure candidates and providing initialization to ICP. The two scan candidates should have overlap in the physical space that they describe. One intuitive way to quantify overlap is to project points from one scan into the space of the second scan. Then, we can look at whether the projected points from the first scan are visible in the second scan. The overlap can be quantified by looking at the percentage of projected points that are not occluded.

More formally, using the notation of OverlapNet we construct the 2D range images $\mathcal{V}$ from each 3D LiDAR scan $\mathcal{P}$. Each 3D point $\mathbf{p}_i = (x, y, z)$ is mapped to a 2D image point $\mathbf{v}_i = (u, v)$ using the cylindrical projection $\mathit{\Pi} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ described below

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)/\pi\right] w \\ \left[1 - \left(\arcsin(zr^{-1}) + f_{\text{up}}\right)/f\right] h \end{pmatrix} \tag{1}$$

where $r = \|\mathbf{p}_i\|_2$ is the range, $f = f_{\text{up}} + f_{\text{down}}$ is the vertical field-of-view of the camera, and w, h are the width and height of the range image in pixels. If two 3D LiDAR points are mapped to the same image point, the projected point with the smaller range measurement r is chosen. If no LiDAR point is mapped to the image point, then we define that image point as invalid with $r = -1$.

For the pair of LiDAR scans $\mathcal{P}_1$ and $\mathcal{P}_2$ we define the corresponding range images $\mathcal{V}_1$ and $\mathcal{V}_2$. We also consider the camera-fixed coordinate frames for each scan C_1 and C_2. Given the poses of each coordinate frame $\mathbf{T}_{WC_1}$ and $\mathbf{T}_{WC_2}$ relative to the world frame W we can now define the range image of scan $\mathcal{P}_1$ projected into C_2 as $\mathcal{V}_1'$:

$$\mathcal{V}_1' = \mathit{\Pi}\left(\mathbf{T}_{WC_1}^{-1}\mathbf{T}_{WC_2}\mathcal{P}_1\right) \tag{2}$$

We can then compute the ground truth overlap between scans $\mathcal{P}_1$ and $\mathcal{P}_2$ by considering range images $\mathcal{V}_1'$ and $\mathcal{V}_2$ in C_2. We define overlap using the absolute difference of ranges in $\mathcal{V}_1'$ and $\mathcal{V}_2$:

$$O_{\mathcal{P}_1\mathcal{P}_2} = \frac{\sum_{(u,v)} \mathbb{I}\{|\mathcal{V}_1'(u, v) - \mathcal{V}_2(u, v)| \leq \epsilon\}}{\min(\text{valid}(\mathcal{V}_1'), \text{valid}(\mathcal{V}_2))} \tag{3}$$

where $\mathbb{I}$ is the indicator function and ϵ is a threshold for comparing ranges. $\text{valid}(\mathcal{V})$ denotes the number of pixels with $r \neq -1$, since some pixels may not have corresponding LiDAR points as described earlier. The summation is only conducted over pixels with valid ranges in both $\mathcal{V}_1'$ and $\mathcal{V}_2$.

Furthermore, we also consider the problem of predicting the relative transformation $\mathbf{T}_{C_1C_2}$. By predicting this transformation, ICP can be provided with a better initialization. To simplify the task, we only consider predicting the relative yaw (orientation) between C_1 and C_2 defined as γ_{12}.

IV. APPROACH

1. Proposed Algorithm

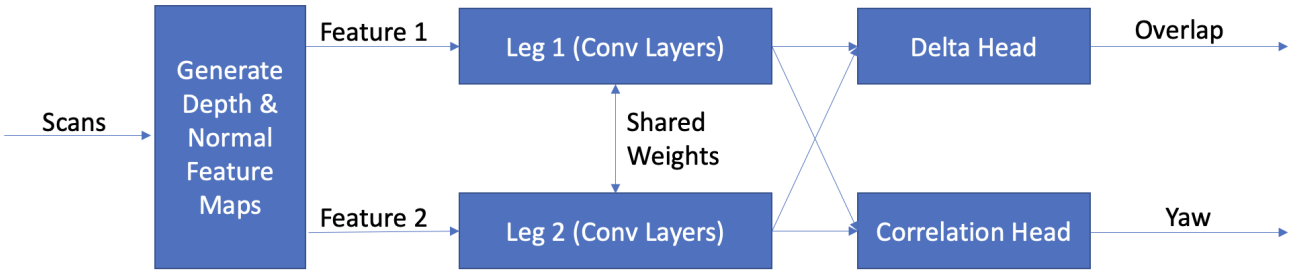

Figure 2: Diagram of our proposed architecture, MiniOverlapNet. We utilize a leg CNN that processes range and normal information from each scans. The encoder architecture and weights are shared for both scans. Two separate heads are used to estimate overlap percentage and relative orientation. This architecture is often referred to as a Siamese network in the literature.

Our proposed neural network, MiniOverlapNet, is based on the OverlapNet architecture. The network consists of a encoder (leg) which processes scans and provides a feature vector representing the scan. Then there are two separate heads that estimate overlap and relative orientation between two scans. We attempt to downscale the network in order to address some of the online runtime issues described in the original paper.

a) Encoder

Similar to OverlapNet, we process each scan from a 3D point cloud to a 2D representation. The input layer of the encoder consists of a $64 \times 450 \times 4$ tensor. The first layer of the tensor is the range image described in section III. The remaining layers contain normal information for each point in the range image. An example range and normal image are shown in Figure 3.

We calculate normals using finite differences of the range image. First, we construct vectors $\mathbf{u}$ and $\mathbf{v}$ on the face of the surface as follows:

$$\begin{aligned} \mathbf{u}(u,v) &= \mathit{\Pi}^{-1}(\mathcal{V}(u+1,v)) - \mathit{\Pi}^{-1}(\mathcal{V}(u,v)) \\ \mathbf{v}(u,v) &= \mathit{\Pi}^{-1}(\mathcal{V}(u,v+1)) - \mathit{\Pi}^{-1}(\mathcal{V}(u,v)) \end{aligned} \tag{4}$$

Then for each point (u, v) in the range image we compute the normalized cross product $\hat{\mathbf{w}}$:

$$\hat{\mathbf{w}}(u,v) = \frac{\mathbf{u}(u,v) \times \mathbf{v}(u,v)}{\|\mathbf{u}(u,v) \times \mathbf{v}(u,v)\|} \tag{5}$$

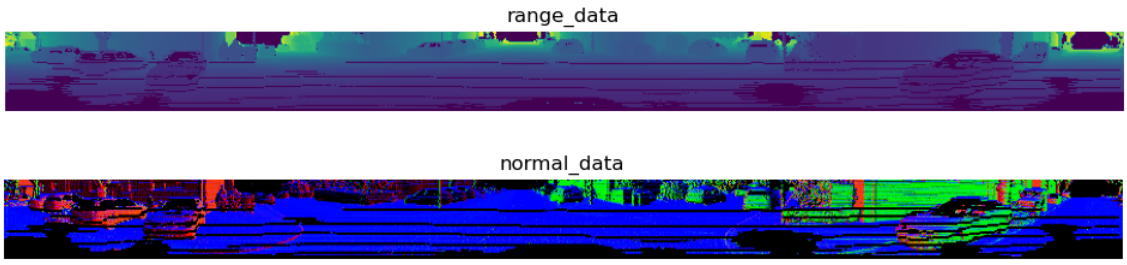

Figure 3: Example 2D range and normal images. The range image is single channel whereas the normal image has 3 channels that encode the normal direction.

The encoder network consists of a series of alternating Conv2D and MaxPool layers which progressively shrink the input to a horizontal feature vector of size $1 \times 180 \times 64$. The same encoder network with shared weights is used to encode both $\mathcal{P}_1$ and $\mathcal{P}_2$. This means that at inference time, feature vectors only need to be computed once using the encoder.

We note that because our range and normal images are cylindrical projections of the point cloud, they can be rolled left or right and content from one edge of the image will wrap around to the other side of the image. Due to the translation equivariance of the CNN architecture, this results in a similar shift in the horizontal feature vector. Thus, the width of this feature vector controls the resolution of the subsequent yaw estimation.

b) Delta Head

The purpose of the delta head is to estimate the overlap between scans using their feature vectors $\mathbf{L}_1$ and $\mathbf{L}_2$. The first layer in this head is what is referred to as a delta layer in the OverlapNet paper. The second feature vector $\mathbf{L}_2$ is transposed to $180 \times 1 \times 64$, and then the absolute difference $|\mathbf{L}_1 - \mathbf{L}_2|$ is computed, broadcasting accordingly. This produces an output tensor of shape $180 \times 180 \times 64$. By performing the delta operation, we encode the difference information for all possible orientations of the two scans. The subsequent Conv2D and fully-connected layers compress this information down to a single estimate of the overlap between the scans.

c) Correlation Head

The correlation head performs cross-correlation between the two feature vectors $\mathbf{L}_1$ and $\mathbf{L}_2$ to estimate the relative orientation. This operation is implemented as a Conv2D layer with $\mathbf{L}_1$ as the input and $\mathbf{L}_2$ as the kernel. To make sure the full correlation is computed, we created a padded version of $\mathbf{L}_1$ by horizontally stacking two copies of the feature vector. This ensures that $\mathbf{L}_2$ is slid across the entirety of $\mathbf{L}_1$.

After computing the cross-correlation between the two feature vectors, we take the argmax as the relative orientation between the two feature vectors. Because the alignment of these feature vectors matches the orientation of the input tensors which themselves are derived from the 3D scans, we use this value as a measurement of the relative orientation between $\mathcal{P}_1$ and $\mathcal{P}_2$.

d) Loss Functions for Training

The delta head of the network treated overlap as a regression problem, the loss is the absolute value of the error, passed through a sigmoid function. Since the overlap is always bounded between $0 - 1$, a sigmoid function is applied to the loss to bound the loss between $0 - 1$. The correlation head outputs a 1-D vector with 180 bins, resulting in a yaw resolution of 2°. Since argmax is non-differentiable, yaw estimation is treated as a classification problem and binary cross-entropy loss is used. The combined loss for the network is a linear combination of the overlap loss and the yaw angle loss defined below:

$$L_O = \sigma(a|\bar{Y}_O - Y_O| + b) \tag{6}$$

$$L_Y = \sum_{i=\{1,\dots,N\}} Y_Y^i \log(\hat{Y}_Y^i) + (1 - Y_Y^i) \log(1 - \hat{Y}_Y^i) \tag{7}$$

$$L = L_O + \alpha L_Y \tag{8}$$

2. Design Decisions

As described in previous sections, the main drawback of the original OverlapNet architecture is the large network that results in slow training and inference speeds. Accordingly, our design decisions revolve around making MiniOverlapNet more lightweight. To do so, we reduce the input size, feature vector size and number of layers in the network.

The OverlapNet architecture uses intensity and semantic embeddings in addition to the range and normal embeddings. They also use an input width of 900 pixels, resulting in an input tensor with size $64 \times 900 \times 20$. By removing the intensity and semantic embeddings and downsampling the input, MiniOverlapNet's input tensor is 90% smaller. It is worth noting that while the experiments in the OverlapNet paper use the full input tensor, their published code also omits the intensity and semantic emeddings. Thus we use this scaled OverlapNet with input shape $64 \times 900 \times 4$ for comparison.

Additionally we replace a number of Conv2D layers with MaxPool2D layers as described in Table 1. Maxpooling requires no weights and is cheaper to compute than convolutions. Additionally maxpooling is known to extract the sharpest features from an image which provides a better low-dimensional representation.

Finally, we reduce the size of the feature vector from $1 \times 360 \times 128$ to $1 \times 180 \times 64$. This is particularly important as it drastically reduces the size of the layers in the delta head. The delta head is the main bottleneck for inference, since a forward pass needs to be taken for each pairwise comparison. This is in contrast to the encoder, which only needs to be applied once to each scan.

Table 1: Comparison between OverlapNet and MiniOverlapNet architectures. MiniOverlapNet has 5x less parameters and replaces some of the Conv2D layers with MaxPool2D to reduce the number of computations.

OverlapNet

	Operator	Kernel	Stride	Shape	Parameters
Input	–	–	–	64 x 900 x 4	–
Leg	Conv2D + ReLU	5 x 15	(2, 2)	30 x 433 x 16	4,816
	Conv2D + ReLU	3 x 15	(2, 1)	14 x 429 x 32	23,072
	Conv2D + ReLU	3 x 15	(2, 1)	6 x 415 x 64	92,224
	Conv2D + ReLU	3 x 12	(2, 1)	2 x 404 x 64	147,520
	Conv2D + ReLU	2 x 9	(2, 1)	1 x 396 x 128	147,584
	Conv2D + ReLU	1 x 9	(1, 1)	1 x 388 x 128	147,584
	Conv2D + ReLU	1 x 9	(1, 1)	1 x 380 x 128	147,584
	Conv2D + ReLU	1 x 9	(1, 1)	1 x 372 x 128	147,584
	Conv2D + ReLU	1 x 7	(1, 1)	1 x 366 x 128	114,816
	Conv2D + ReLU	1 x 5	(1, 1)	1 x 362 x 128	82,048
	Conv2D + ReLU	1 x 3	(1, 1)	1 x 360 x 128	49,280
Overlap Head	Delta Layer	–	–	360 x 360 x 128	–
	Conv2D + Linear	1 x 15	(1, 15)	360 x 24 x 128	122,944
	Conv2D + ReLU	15 x 1	(15, 1)	24 x 24 x 128	123,008
	Conv2D + ReLU	3 x 3	(1, 1)	22 x 22 x 256	295,168
	Dense	–	–	1	123,905
Total					**1,769,137**

MiniOverlapNet

	Operator	Kernel	Stride	Shape	Parameters
Input	–	–	–	64 x 450 x 4	–
Leg	Conv2D + ReLU	5 x 9	(2, 2)	30 x 221 x 16	2,896
	Conv2D + ReLU	3 x 9	(2, 1)	14 x 213 x 32	13,856
	Conv2D + ReLU	3 x 8	(1, 1)	12 x 206 x 32	24,608
	MaxPool2D	–	(2, 1)	6 x 206 x 32	
	Conv2D + ReLU	3 x 7	(1, 1)	4 x 200 x 64	43,072
	MaxPool2D	–	(2, 1)	2 x 200 x 64	
	Conv2D + ReLU	1 x 7	(1, 1)	2 x 194 x 64	28,736
	MaxPool2D	–	(2, 1)	1 x 194 x 64	
	Conv2D + ReLU	1 x 7	(1, 1)	1 x 188 x 64	28,736
	Conv2D + ReLU	1 x 5	(1, 1)	1 x 184 x 64	20,544
	Conv2D + ReLU	1 x 5	(1, 1)	1 x 180 x 64	20,544
Overlap Head	Delta Layer	–	–	180 x 180 x 64	–
	Conv2D + Linear	1 x 7	(1, 7)	180 x 25 x 64	28,736
	Conv2D + ReLU	7 x 1	(7, 1)	25 x 25 x 64	28,736
	Conv2D + ReLU	3 x 3	(1, 1)	23 x 23 x 128	73,856
	Dense	–	–	1	67,713
Total					**382,033**

V. RESULTS

1. Experimental Procedure

In order to evaluate our approach, we train and test MiniOverlapNet on a portion of the KITTI odometry benchmark from Geiger et al. (2012). The dataset provides LiDAR scans recorded with a Velodyne HDL-64E around Karlsruhe, Germany. There are 11 sequences in the dataset (labeled 00–11) with ground truth poses. These sequences cover different urban areas, highways and country roads. We use sequences 05–09 for training as these recordings have overlap between scans due to loop closures. We use sequence 00 for testing as it also has a number of loop closures.

For each scan in the sequence, we precompute range and normal images using the procedures described in section III and section IV. This speeds up network training but we acknowledge that this would need to be done online if MiniOverlapNet were used for real-time loop closure detection. However for the purposes of this paper our comparisons will always assume range and normal images have zero computation time.

We then enumerate all possible scan pairs in each sequence. For each pair we generate ground truth overlap labels using Equation 3 and ground truth orientation labels using the provided ground truth poses. It is easy to see that the number of pairs grows with $O(n^2)$ as the sequence gets longer. This presents a few problems for training. First, the number of scan pairs to train on becomes too large for sequences with more than 1000 scans, which includes most of the KITTI sequences. Second, this results in class imbalance where the vast majority of scan pairs have no overlap. To combat these issues, after generating labels we prune the set of scan pairs to even out the overlap distribution. For pruning, we organize the scan pairs into bins of width 0.1 based on their overlap. Then we count the scan pairs in the bin $[0.4, 0.5)$ and ensure that all other bins have at most this number of scan pairs. This does not make the distribution entirely uniform but still leaves us with a sufficient number of pairs to train on. For example, sequence 07 is reduced from 607,000 scan pairs to 34,000. Finally, we split out 10% of the pruned training samples as a validation dataset to monitor during training.

We train the network on KITTI scenes 05–09 for 100 epochs using the Adagrad optimizer. We start with an initial learning rate of 0.001 and use a learning rate decay of 0.99 per epoch. All training and evaluation is done on a Google Cloud instance with 4 vCPUs, 26 GB of memory and a NVIDIA T4 GPU. Python 3.8 and Tensorflow 2.10 are used to implement both OverlapNet and MiniOverlapNet.

2. Baseline

As our baseline, we use a pretrained OverlapNet provided by the authors of the paper. This network also omits semantic and intensity embeddings from the input. However, it contains the full number of Conv2D layers proposed in their paper and also uses a feature vector of size $1 \times 360 \times 128$. The architecture of the OverlapNet used as our baseline is also described in Table 1.

3. Prediction Accuracy

After training MiniOverlapNet on KITTI sequences 05–09, we evaluated its performance on sequence 00, shown in Figure 4 and Table 2. The metrics we considered are absolute prediction error for both overlap and orientation. For evaluating orientation accuracy, we only included scan pairs with a true overlap over 0.7. This is because the orientation loss function described above also filters scan pairs in the same way.

As described in Table 2, the mean absolute error for overlap is 0.113 for MiniOverlapNet, while it is 0.091 for OverlapNet. Similarly, the RMS absolute error for overlap is 0.145 compared to 0.123. This indicates that for overlap prediction, MiniOverlapNet has similar performance despite a 5x reduction in parameters.

On the other hand, yaw prediction accuracy of MiniOverlapNet is about twice as worse as OverlapNet. However, we maintain that the yaw accuracy is of lesser importance since the intention of MiniOverlapNet is to be used prior to fine-tuning with ICP. As long as ICP has a rough initialization, it can still converge on the actual transform between the two scans but may just take more iterations.

4. Loop Closure

We also qualitatively tested MiniOverlapNet's ability to find loop closures with a simulation. The simulation uses ground truth pose but propagates a simulated pose covariance which is used to define a loose elliptical boundary in which loop closures may be present. We then used MiniOverlapNet to evaluate non-adjacent scan pairs in this boundary. In other words, pairs were only considered if they were separated in time and path distance by a minimum amount, to detect actual loops rather than sequential scans. The results on KITTI sequence 00 are shown in Figure 5. We can see that MiniOverlapNet correctly identifies an opportunity for loop closure.

5. Evaluation Time

Lastly we considered the inference time of MiniOverlapNet compared to OverlapNet. We evaluated this using all scan pairs from sequence 07, which is just over $607,000$ pairwise comparisons. This process took 42.3 minutes for OverlapNet while it took just 6.2 minutes for MiniOverlapNet. This is a 7x increase in inference capacity, from 240 pairs / second to over 1600 pairs / second. Assuming roughly 10 new LiDAR scans per second, this allows for upwards of 160 pairwise comparisons for each new scan, as opposed to just 24.

Table 2: Absolute overlap and orientation error on KITTI sequence 00. MiniOverlapNet shows comparable overlap accuracy but degraded orientation accuracy in comparison to the pretrained OverlapNet model.

	Overlap			Orientation (deg)		
	Mean	RMS	Max	Mean	RMS	Max
MiniOverlapNet	0.113	0.145	0.750	1.53	2.96	98.0
OverlapNet	0.091	0.123	0.801	0.88	1.45	29.0

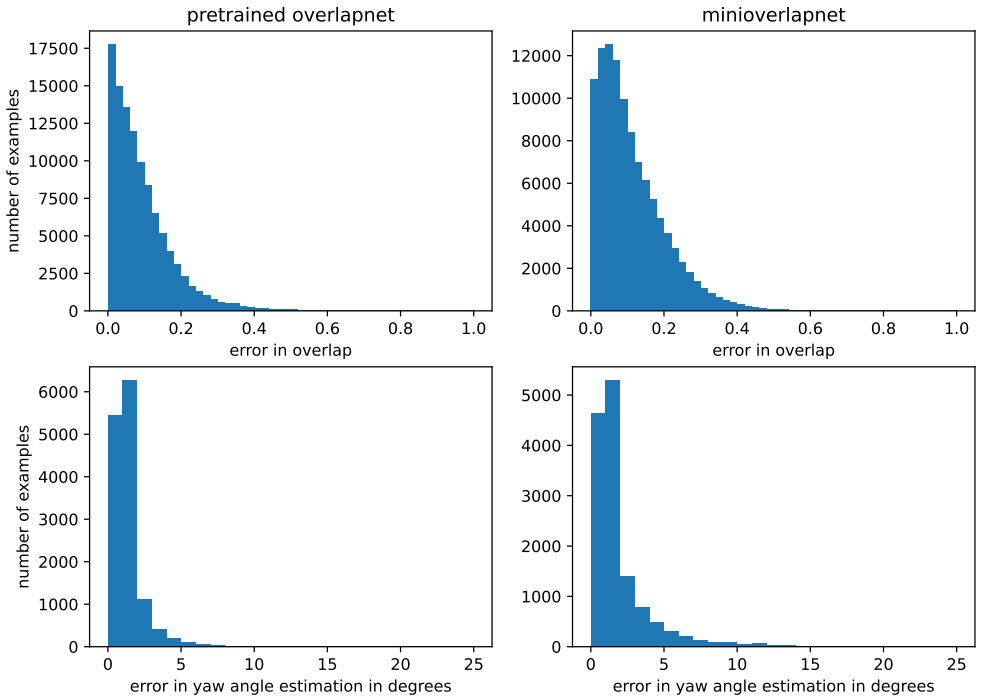

Figure 4: Histograms of absolute overlap and orientation error on KITTI sequence 00.

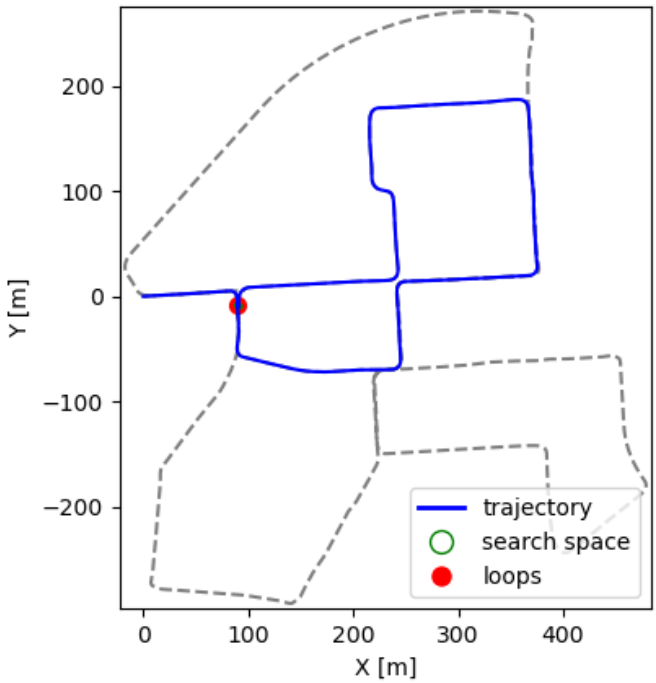

Figure 5: Loop closure simulation on KITTI sequence 00. Ground truth pose is used along with simulated covariance to define an elliptical boundary in which loop closures may be present. MiniOverlapNet correctly identifies a loop closure in the first overlapping segment.

VI. CONCLUSION

In this project we propose a deep learning architecture for 3D LiDAR loop closure detection called MiniOverlapNet. Our design choices revolve around simplifying the architecture of OverlapNet for faster inference time while maintaining similar performance. We are able to show that with a 5x reduction in parameters, MiniOverlapNet predicts overlap with similar accuracy to the full OverlapNet. However, orientation accuracy is about twice as inaccurate relative to OverlapNet. We believe that ICP fine-tuning will be able to bridge this gap in accuracy. Finally, we note that MiniOverlapNet provides a 7x reduction in inference time on our hardware. Overall the results are encouraging for real-time loop closure detection using deep learning.

VII. FUTURE DIRECTIONS

There are a few future directions for this work. The immediate next step is to integrate loop closure detection using MiniOverlapNet into a 3D LiDAR SLAM package such as hdl_graph_slam. We would use a threshold on the overlap prediction to determine if a loop closure should be added. We would then refine the relative pose between the scans via ICP, using the orientation estimate for initialization. Doing so would allow us to evaluate the mapping and odometry improvements associated with the loop closure detections from MiniOverlapNet. Additionally, transformers and attention have become popular in the deep learning literature over the past few years. We could investigate using these architectures to replace layers in MiniOverlapNet's encoder.

VIII. CONTRIBUTIONS

Both Albert and Devin contributed to literature review and worked together to develop the approach presented here. Albert developed the pipeline for data preprocessing, including range image, normal image and ground truth generation. Devin implemented the network architecture in Tensorflow and ran experiments on the Google Cloud instance. Both Albert and Devin worked together on writing the final presentation and report.

REFERENCES

Barsan, I. A., Wang, S., Pokrovsky, A., and Urtasun, R. (2020). Learning to localize using a lidar intensity map. *arXiv preprint arXiv:2012.10902*.

Biber, P. and Straßer, W. (2003). The normal distributions transform: A new approach to laser scan matching. 3:2743–2748.

Chen, X., Läbe, T., Milioto, A., Röhling, T., Vysotska, O., Haag, A., Behley, J., and Stachniss, C. (2021). Overlapnet: Loop closing for lidar-based slam. *arXiv preprint arXiv:2105.11344*.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Merten, H. (2008). The three-dimensional normal-distributions transform. *threshold*, 10:3.

Röhling, T., Mack, J., and Schulz, D. (2015). A fast histogram-based similarity measure for detecting loop closures in 3-d lidar data. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 736–741. IEEE.

Steder, B., Rusu, R. B., Konolige, K., and Burgard, W. (2010). Narf: 3d range image features for object recognition. In *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 44, page 2.

Uy, M. A. and Lee, G. H. (2018). Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4470–4479.

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152.

APPENDIX

1. Code

The code for this project can be found on GitHub at https://github.com/dardeshna/slam-loop-closure.