Scalable 3D Semantic Segmentation for Gun Detection in CT Scans

Nicolas Wagner*† Christoph Reich*† Marius Memmel*†

Abstract

With the increased availability of 3D data, the need for solutions processing those also increased rapidly. However, adding an additional dimension to already existing 2D approaches leads to immense memory consumption and higher computational complexity. These issues cause current hardware to reach its limitations which most often can only be overcome by reducing the input resolution drastically. Our main contribution is a novel deep 3D semantic segmentation method for gun detection in baggage CT scans. With our solution, we enable fast training and low video memory consumption for high resolution voxelized volumes. We achieve this through a moving pyramid approach that utilizes multiple forward passes at inference time for segmenting an instance.

1. Introduction

In the past decade, deep convolutional neural networks (CNNs) have emerged as one of the most successful tools in computer vision [29, 9]. Tasks like object classification[21], object detection[1, 21], or semantic segmentation[24] are most often solved in the 2D image space. Transferring known 2D deep learning solutions to 3D volumes poses various issues. Most notably, the increased computational complexity to process 3D volumes, as well as their immense memory consumption, is not manageable by consumer or even state of the art graphic cards (GPUs) [21]. Another contributing factor to the interest in processing 3D data is the decreasing cost and technical improvement of 3D scanning. With improvements in techniques like LiDAR (Light Detection and Ranging), a larger amount of data is available.

Application areas reach from medical imaging over construction and planning to airport security. To improve the safety of the environment past the check-in, airport scanners are used to check passenger luggage before entering the area. An important part of this is detecting prohibited items at the airport security screening. To be more precise all sorts of firearms. For this purpose, they have created a dataset with 3D CT (computed tomography) scans of hand luggage all containing the aforementioned items. Processing those scans on current GPUs leads to excessive memory consumption culminating in a low batch size and long loading times.

In the following, we propose a novel 3D semantic segmentation method, the HiLo-Network, to overcome those challenges for this specific dataset. We assume that HiLo-Networks can be applied to a broader range of 3D datasets. The main goal is to reduce memory consumption while retaining a fast training process. A tradeoff is made allowing multiple forward passes at inference time to obtain a scalable approach that can run on most consumer level GPUs. As a result, our approach is particularly suitable for commercial detection devices effectively limiting their production costs. The implementations to reproduce our results and testing are available under this link.

2. Related Work

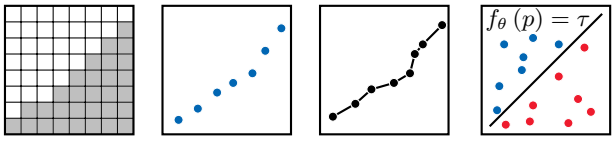
The methods and architectures used to perform 3D semantic segmentation depend on the representation of 3D data. In what follows, data types and corresponding architectures are introduced briefly.

A common representation of 3D data are point clouds (see figure 1b). These are generally created using LiDAR scanners. Points are recorded sparsely and distributed in space. This means that only points of shapes are stored and the background is left out. Effective architectures used to segment point clouds are PointNet[17] and its improved successor PointNet++[19]. It should be noted that due to the lack of neighborhood information no convolutional methods can be applied. Because of this, point clouds are often converted to voxelized volumes [23, 18].

Meshes use multiple polygons to represent shapes (see figure 1c). An often used mesh type is the triangular mesh named after the triangles used to represent shapes. More precisely, meshes can be seen as a graph that is spanned by vertices and edges. Vertices directly

*equal contribution, surname.name@stud.tu-darmstadt.de
†Technical University Darmstadt

correspond to points in point clouds. Due to this connection to point clouds, meshes suffer similar disadvantages. An advantage though is the additional neighborhood information between vertices. Meshes have been used in 3D semantic segmentation tasks in combination with CNNs or graph convolutions [25, 6, 14, 22].



(a) Voxel (b) Point (c) Mesh (d) Classifier Space

Figure 1. Various 3D representations (simplified 2D variants) [15].

Voxel representation can be described as a cubical volume that is divided into a regular grid (see figure 1a). Each sub-cube in the regular grid is called a voxel. In contrast to point clouds, this representation is of dense nature with the background also being part of the grid. Each voxel representing a shape takes a value that describes material information. Background voxels are all associated with a single value, too. An advantage of this type of representation is that CNNs with 3D convolutions can be applied, replacing a manual feature extraction by an automated one. One disadvantage over point clouds is the loss of detail that occurs depending on the resolution. With the background also being part of the representation, memory consumption also increases. To counter the described issues, attempts like reducing the input resolution or to extract features from different resolutions are made [3, 27].

Another approach combining the advantages of both voxels and point clouds is presented in [13]. The key idea is to limit the operations on a voxel representation when absolutely needed and to otherwise use a point cloud representation instead. This way a smaller memory footprint is maintained while also exploiting data locality and regularity.

Given that at this point, the research community did not find a universally applicable CNN approach for large 3D volumes, a step back to 2D can be made. This way the advantages of CNNs can be used up to an extent without running into memory problems like 3D CNNs would do. The multi-view representation splits up the 3D volume into 2D image slices [18, 5, 2]. As a result, CNNs can be applied but are restricted to 2D convolution. However, some global context is lost due to the split among the chosen axis.

Because all mentioned representations suffer from a tradeoff between high memory consumption, detail, and applicable architectures, attempts are made to optimize such. One approach is to use the octree[26] data structure to represent the shapes in a tree-based format. Another approach introduces the occupancy network (O-Net)[15] that utilizes the function space of a neural network to model shapes (see figure 1d). The basic idea of O-Nets is to represent a shape as a neural network classifier. For any point of interest the classifier can answer the question of whether or not the point is part of the shape. The O-Net is versatile when it comes to input data. Points clouds, single images from a multi-view representation as well as voxels can be transformed into a classifier representation.

A neural network architecture commonly used for 3D semantic segmentation is the U-Net[20] and its adaption to the 3D U-Net[4]. U-Net networks consist of an equal amount of down- and upsampling steps giving it the iconic U shape. The upsampling steps also take the output of the downsampling step at the same level as input. This residual connection enables the upsampling steps to take into account features determined by the downsampling steps. With the U-Net only being able to process 2D images, the 3D U-Net extends its functionality to a 3D input by replacing all 2D components by its 3D counterparts.

3. Method

3.1. Dataset 3D-CT

The dataset provided by a private security firm includes 2925 labeled CT scans of baggage containing firearms. For a better generalization, prohibited objects reach from ammo to pistols and even submachine guns. The CT scans are represented as voxelized volumes where each voxel contains material information. The further analysis involves semantic segmentation of the objects instead of a simple detection. For this purpose, the labels are given as a voxel representation, too. Each voxel indicates whether or not it is part of a firearm shape. Hence, the semantic segmentation task is binary. There is only one shape per instance.

The CT scans have a resolution of n x 416 x 616 where n varies among scans. To produce unified dimensions, n is capped or padded to a value of 640. The final dimensions of 640 x 416 x 616 translate to a memory consumption of roughly 650MB per scan. Each material value is further normalized to fit a range of 0 to 1. After the preprocessing, the dataset is split into a training set of 2600 samples, a validation set of 128 samples and a testing set of 196 samples. The transformed dataset is further referred to as 3D-CT.

3.2. Super Resolution Occupancy Network for Semantic Segmentation

The general idea of an O-Net is to define a 3D shape as a classifier for $\mathbb{R}^3$. More precisely, each 3D coordinate gets mapped onto a pseudo probability which is called occupancy value. Occupancy values give an estimation of whether or not a coordinate is part of the shape. Using a threshold value this decision can be influenced manually. Formally, the classification task can be described as

$$f_\theta : \mathbb{R}^3 \times \mathbf{L} \rightarrow [0, 1]\,, \tag{1}$$

where $l \in \mathbf{L}$ is a finite observation of $\mathbb{R}^3$ and θ are the parameters of f. O-Nets generate l by using an arbitrary neural network architecture to encode a shape. The mapping f itself is a fully connected decoder neural network. The input to the encoder depends on the chosen architecture. For voxelized volumes, a CNN encoder is used. The decoder parameters θ are trained by randomly sampling 3D coordinates from the bounding boxes of targeted shapes. Methods exist to transform the classifier representation of a shape back into a mesh representation[15]. Nonetheless, a significant speedup is possible if test and validation metrics are also calculated on sampled coordinates.
We use the O-Net to perform 3D semantic segmentation on voxelized volumes. In particular, we focus on solving the semantic segmentation of the 3D-CT dataset. Since the 3D-CT dataset results in a binary semantic segmentation task, this task is equal to defining 3D shapes. Hence, the O-Net architecture is suitable.

For the 3D-CT dataset, a single volume consumes about 650MB. Fortunately, O-Nets reduce the memory consumption for the decoder in comparison to U-Nets, as complete volumes do not have to be processed simultaneously. Nevertheless, super resolution is needed to manage the memory footprint of the inputted volumes that are processed by the CNN encoder. For the super resolution O-Net, volumes are downsampled using average pooling before they are encoded. The super resolution O-Net is still asked to classify high resolution coordinates. In other words, training is performed sampling high resolution coordinates with their corresponding high resolution labels while encoding low resolution volumes. For the inference of a complete volume, the Multiresolution IsoSurface Extraction (MISE) algorithm[15] is used but stopped before the Marching Cubes algorithm is conducted. Originally, [15] use a CPU version of MISE. We implement a GPU variant that is applicable to voxelized volumes.

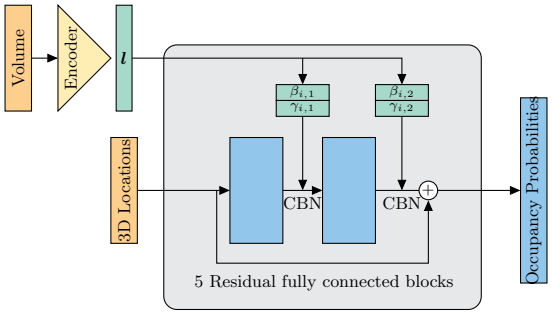


Figure 2. Super resolution O-Net architecture.

The originally proposed O-Net uses conditional batch normalization[16] to pass the output of the encoder to the decoder. In traditional batch normalization[8] a layer with d-dimensional input $\mathbf{X} = (X^{(1)}...X^{(d)})$ is normalized by

$$\gamma^{(c)} \frac{\mathbf{X}^{(c)} - \mathbb{E}\left(\mathbf{X}^{(c)}\right)}{\sqrt{\mathrm{Var}\left(\mathbf{X}^{(c)}\right)}} + \beta^{(c)} \tag{2}$$

where γ_c and β_c are learnable parameters per channel c. In conditional batch normalization, however, $\gamma(x)^{(c)}$ and $\beta(x)^{(c)}$ are learnable mappings of some input x. In O-Nets, $x = l$ holds for any conditional batch normalization layer. The previously mentioned mappings are typically implemented as fully connected layers. As a consequence, encodings are only inserted into an O-Net as normalization parameters (see figure 2). For this reason, we propose an alternative O-Net architecture (see figure 3). This architecture uses a concatenation of the coordinates with their corresponding (repeated) latent vectors as the decoder input.

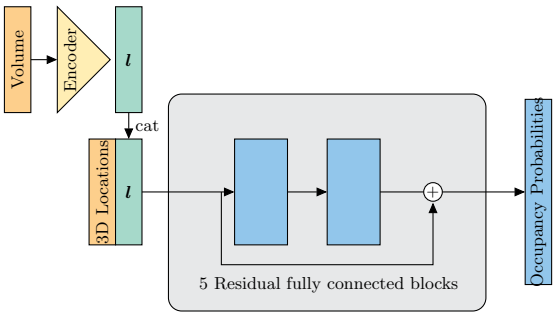


Figure 3. Alternative super resolution O-Net architecture.

3.3. Occupancy Network for Bounding Boxes

The output of an O-Net can also be used to build an axis-aligned bounding box (BB) around a gun. If a BB can be reliably constructed, it is sufficient to perform

semantic segmentation only within the BB. For this adoption, the basic O-Net architectures described in section 3.2 stay untouched. Nonetheless, in addition to using low resolution input volumes, only low resolution coordinates and labels are sampled. Low resolution labels are defined by max pooling associated high resolution labels. A BB is simply constructed by using the smallest and biggest predicted coordinates.
For this use case, an O-Net is not required to perform super resolution or to extract fine details. This task can be labeled as much easier than performing super resolution 3D semantic segmentation.

3.4. HiLo-Network

The challenge for a super resolution deep neural network is retrieving high resolution information from low resolution representations. In our approach, the HiLo-Network, we circumvent this complex generative task by again inputting dense high resolution information into a neural network. However, the main bottleneck, GPU-acceleration of gradient-based optimization, is improved by applying a divide and conquer procedure to semantic segmentation. During gradient descent, only small chunks of each instance within a batch need to be loaded into video RAM.

Independent of a particular architecture, the main concept of inference for HiLo-Networks is as follows. Instead of passing a complete volume into a network, only a small 3D chunk (window) of fixed-size w^3 is used. One can feed multiple windows into the network to conduct semantic segmentation on the full volume. Nonetheless, global relations between different windows are not taken into consideration yet. To overcome this problem, a common computer vision trick is applied. Centered around the first window, a seconder bigger window of size $(w * d)^3$ is constructed but downsampled by a factor d using average pooling. This small pyramid contains high resolution local information as well as low resolution more global information. If a window crosses the borders of a volume, zero-padding is applied. In our tests for small values of w, the pooling operation is conducted on the CPU without a significant increase in computation time. In case two windows are not sufficient, one can easily add more levels. The memory consumption of the fixed-size pyramid grows linear in terms of levels. In comparison, memory consumption grows nonlinear in the size of the volume dimensions. The downside of our approach is that multiple HiLo-Network passes are needed to segment a complete volume. However, in a likely real-time scenario in which only a few volumes per time step are processed, multiple pyramids per volume can be passed simultaneously.

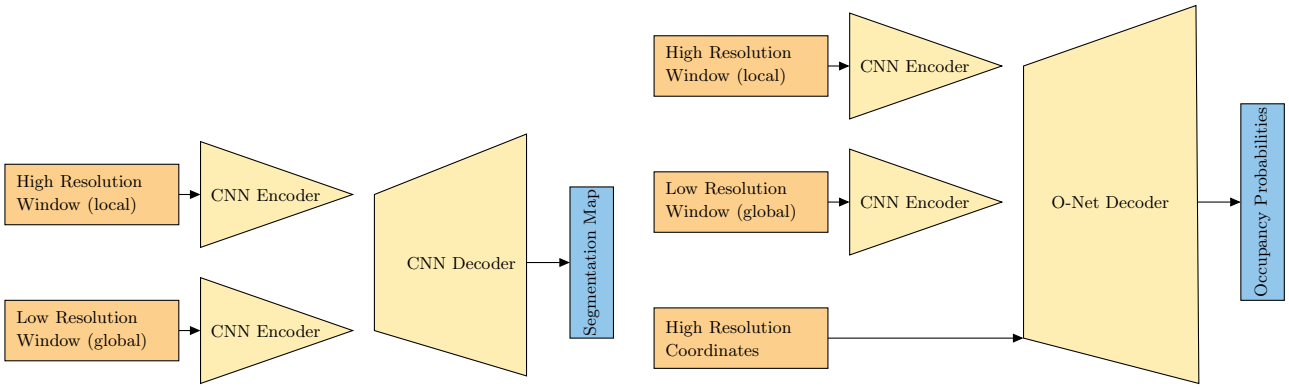
Coarse sketches of the utilized network architectures are displayed in figure 4a and 4b. We deploy two different architectures. The first one consists of a CNN encoder and an O-Net decoder[15] (O-Net-approach). Because small 3D windows can be handled by consumer GPUs, the second architecture includes a CNN encoder as well as a CNN decoder (CNN-approach). In both architectures, a separate encoder is trained per pyramid level. The resulting encodings are concatenated before they are processed by the decoder. For the O-Net-approach, a query of 3D coordinates has to be fed into the decoder, too.

Both architectures no longer need to be trained on complete instances but rather on pyramids constructed at randomly drawn locations within each instance. Any two pyramids with different locations but of the same instance can be independently processed. As a result, the sampling scheme of the pyramid locations can be biased. This comes in handy, as the considered CT scans mostly contain air. Detecting air is easy since it is related to only a single material value. Hence, regularly redrawing locations at which the surrounding pyramid does not contain a gun can reduce training time. More precisely, per training step a random location from each instance of a batch is drawn and, if needed, redrawn with a likelihood of 90%. Afterward, pyramids are built centered at those locations and processed by the HiLo-Network. For parameter validation a similar logic holds. Each time the validation set is evaluated, only one pyramid per instance is processed, too. Estimating validation losses leads to a significant decrease in training time. The estimation can be improved by considering a moving average over validation losses. Nevertheless, testing results can only be determined by evaluating non-overlapping[1] pyramids that fully cover a test instance.

3.5. Training Queues

Besides video RAM management during gradient descent optimization, data loading is a major concern. Although only pyramids with a comparably small memory footprint have to be loaded on the GPU, full instances have to be processed on the host. For the 3D-CT dataset, each file consumes about 650MB of memory. For quick access, files have to be stored in the host RAM. However, loading a dataset build of a few thousand of such large files into host RAM is clearly not feasible. Common deep learning libraries[2] provide multi-threading mechanisms to load files into

[1] Non-overlapping on the first pyramid level.
[2] https://pytorch.org/



(a) Architecture of the CNN-approach. (b) Architecture of the O-Net-approach.

Figure 4. HiLo-Network architectures.

host RAM parallel to performing gradient descent on the GPU. This allows covering slow hard disk accesses by computations. Nonetheless, multiple files have to be loaded to form a batch and, by construction, one thread/core can only load a single file into host RAM. Therefore, these mechanisms are only useful on expensive workstations that are not jointly used by other tasks.

We exploit that host RAM is usually bigger and cheaper than video RAM. For this, a training queue of a user-defined size is constructed in host RAM. Per training step, a batch is randomly drawn from the training queue rather than loaded from the hard drive. Before each batch formation, one new file is loaded from disk and added to the training queue. Likewise, one instance is deleted from the training queue if the user-defined queue size is reached. The training queue procedure reduces the number of files loaded from hard drive to one per processed batch. The loading happens parallel to gradient descent. If the queue size is sufficient, the data distribution can still be represented well enough.

We inspect three different training queues that vary in the implementation of sampling, adding, and deleting. First, a queue that uniformly draws instances (first in-first out). Furthermore, the instance which has been inserted first is replaced. Second, a queue that weighs an instance i according to its number of occurrences o_i (occurrence based). The more often an instance has already been drawn, the less its probability to get drawn again. Formally, the probability that i is drawn equals $e^{-o_i} / \sum_j e^{-o_j}$. The instance with the most occurrences is replaced. Third, a queue that weighs instances according to their hardness (hardness based). The worse the training loss h_i of an instance i, the bigger its probability to get drawn again. The instance with the best training loss is replaced. Formally, the probability that i is drawn equals $e^{h_i} / \sum_j e^{h_j}$.

4. Experiments

4.1. Super resolution Occupancy Network

Within this section, we evaluate and compare the performance of the proposed super resolution O-Net architectures for semantic segmentation on the 3D-CT dataset. At this, we describe in detail model architectures, training processes, and implementations.

The CNN encoders of the super resolution O-Nets are built from five standard convolutional residual blocks[7]. After the five residual blocks, a final fully connected layer is used to produce a scalar occupancy value per inputted coordinate. Volumes that are inputted to the encoder are downsampled by a factor of 8 using average pooling. We test a wide and a shallow CNN encoder. The wide one utilizes twice as many filters as the shallow one per layer. Super resolution O-Net decoders are implemented with five fully connected residual blocks.
For any but the last layer of a network, a leaky ReLU[28] is utilized as the activation function in combination with a normalization operation. Output layers apply a sigmoid activation function. Depending on the tested architecture, batch normalization or conditional batch normalization is deployed. If conditional batch normalization is used, the latent vector of the encoder is processed by two leaky ReLU activated fully connected layers to predict the normalization parameters. To optimize the O-Net parameters the Adam optimizer[10] is used. Adam is initialized with a learning rate of 0.001 as well as PyTorch default parameters and optimizes a binary cross entropy loss. The dataset is processed in batches of size 8. Each network is trained for 200 epochs, which takes

approximately 12h on a single GPU. During training, we save the model with the best validation loss for testing. Validation is conducted after every epoch.
At training time, 2^{14} coordinates are sampled per instance and training step. For the 3D-CT dataset, the class labels are highly unbalanced. We cope with this problem by using a biased sampling scheme. The applied scheme draws 60% of the coordinates uniformly from the targeted shapes. The remaining coordinates are sampled uniformly over the whole volume. A gun classification is assumed in case the output of a network is above 0.5.

Our tests differ in concatenation (Cat) and no concatenation, conditional batch normalization (CBN) and batch normalization as well as wide encoder (Wide) and shallow encoder. To compare the results of the different architectures, the intersection over union (IoU) is computed on a 3D-CT test set. During testing, we sample 2^{18} coordinates, uniformly drawn from each full volume, for a sophisticated approximation of the IoU.

Cat	CBN	Wide.	Param.	IoU
✓	✓	✓	3.4M	0.1744
✓	✗	✓	2.2M	0.1591
✗	✓	✓	3.3M	0.1689
✓	✓	✗	2.0M	**0.1943**
✓	✗	✗	0.7M	0.1612
✗	✓	✗	1.9M	0.1128

Table 1. Test results of different architectures for the super resolution O-Net.

From the low IoUs, shown in table 1, we conclude that no O-Net architecture is able to perform super resolution and 3D semantic segmentation at once. Due to this observation, we decide to evaluate the same models directly on low resolution labels to test the general 3D semantic segmentation capabilities of O-Nets. The low resolution labels are downsampled by a factor of 8 using max pooling. Again, we can conclude, from the

Cat	CBN	Wide.	Param.	IoU
✓	✓	✓	3.4M	0.1843
✓	✗	✓	2.2M	**0.1968**
✗	✓	✓	3.3M	0.1554
✓	✓	✗	2.0M	0.1836
✓	✗	✗	0.7M	0.1805
✗	✓	✗	1.9M	0.1643

Table 2. Test results of different O-Net architectures for a low resolution input volume and low resolution coordinates.

results shown in figure 2, that O-Nets in general are not able to segment firearms properly. We analyze the segmentation results visually to get an understanding of this failure. From these visualizations (see figures 5 and 7), we can observe that O-Nets can predict the rough position of an object accurately. However, the O-Net is not capable of predicting detailed shapes. The described behavior leads us to the idea of using O-Nets to predict a bounding box rather than a fine segmentation.
We can also observe that the whole training process tends to be noisy, since the validation metrics have a high fluctuation (see figures 9 and 10). This is most likely due to the random nature of O-Nets.

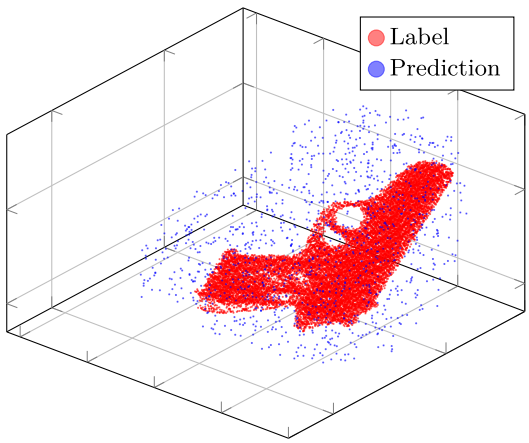


Figure 5. Exemplary super resolution O-Net semantic segmentation result.

4.2. Occupancy Network for Bounding Boxes

We test the same networks as in section 4.1 for BB construction. Before the construction of BBs, a value of 10 is subtracted from and added to the minimum and maximum predicted coordinates respectively. This procedure improves BB IoU results.

Cat	CBN	Wide	Param.	BB IoU
✓	✓	✓	3.4M	0.5093
✓	✗	✓	2.2M	0.5056
✗	✓	✓	3.3M	0.4162
✓	✓	✗	2.0M	**0.5314**
✓	✗	✗	0.7M	0.5039
✗	✓	✗	1.9M	0.3833

Table 3. Test results of different O-Net architectures (low resolution input volumes, high resolution coordinates) for predicting an axis-aligned bounding box.

The test results, shown in table 3 and 4, confirm the previously observed behavior that O-Nets are able to

Cat	CBN	Wide	Param.	BB IoU
✓	✓	✓	3.4M	0.5191
✓	✗	✓	2.2M	**0.5640**
✗	✓	✓	3.3M	0.4522
✓	✓	✗	2.0M	0.5118
✓	✗	✗	0.7M	0.5269
✗	✓	✗	1.9M	0.4710

Table 4. Test results of different O-Net architectures (low resolution input volumes, low resolution coordinates) for predicting an axis-aligned bounding box.

detect the coarse region around a shape. Additionally, a slight performance increase can be observed when training on a low resolution label rather than on a high resolution label. Furthermore, it can be derived from the results that conditional batch normalization performs worse than our alternative O-Net architecture. Since conditional batch normalization uses more parameters, too, it can be recommended to refrain from using it for BB construction. We also conclude from all O-Net experiments that rather the O-Net decoder is the main performance bottleneck. We justify this by the fact that the shallowness of the encoder has only a minor influence on the overall performance.

4.3. HiLo-Network

	$w = 16$	$w = 32$	$w = 48$
$d = 2$	0.6216	**0.6836**	**0.7059**
$d = 4$	0.6168	0.6700	0.6970
$d = 8$	**0.6488**	0.6640	0.6233

Table 5. Test IoUs for the CNN-approach. The results are broken down into window size w and downsampling factor d.

$w = 16$	$w = 32$	$w = 48$
1.1 GB	4.5 GB	12.3 GB

Table 6. Video RAM consumption during training with batch size 16 for the CNN-approach. The results are broken down into different window sizes w .

In this section, we evaluate HiLo-Networks on the 3D-CT dataset and give a detailed description of training procedures as well as architectures. In particular, we evaluate the two main hyperparameters of HiLo-Networks, the window size w and the downsampling factor d, in terms of test IoU and video RAM consumption. Furthermore, the CNN-approach is compared to the O-Net-approach.

	$w = 16$	$w = 32$	$w = 48$
$d = 2$	0.4248	0.5395	**0.5360**
$d = 4$	**0.4481**	**0.5962**	0.4992
$d = 8$	0.3963	0.4953	0.4400

Table 7. Test IoUs for the O-Net-approach. The results are broken down into window size w and downsampling factor d.

We implement all HiLo-Network encoders with 6 residual blocks[7]. Due to the small window sizes, only one average pooling operation is conducted after the first block in the CNN-approach. The decoder of the CNN-approach consists of 7 residual blocks with an upsampling operation after the first block. The decoder of the O-Net-approach uses 3 fully connected residual blocks. We use our alternative O-Net decoder architecture. In contrast to the CNN-approach, an average pooling operation is conducted after the first, second (for $w = 16, w = 32, w = 48$), third (for $w = 32, w = 48$), and fourth (for $w = 48$) encoding block. This prevents the first fully connected layer of the O-Net-approach decoder to grow untrainable large.

For any but the last layer of a network, SELU[11] is used as the non-linearity in combination with batch normalization[8]. Output layers deploy a sigmoid activation function. All networks are trained using the Adam optimizer[10]. The optimizer is initialized with a learning rate of 0.001 and PyTorch default parameters. The dataset is processed in batches of size 16. For the CNN-approach, the focal loss[12] whereas for the O-Net-approach the binary cross entropy is minimized. The focal loss manages unbalanced class occurrences automatically. In early testings, the focal loss appeared to be not useful for the O-Net-approach. Likely due to the random nature of O-Nets. The network parameters are validated every 128 training steps. The parameter set with the best validation IoU is used for testing. Each network is trained for 20 epochs. Training queues of the first type (first in-first out) and of size 512 are used. The O-Net-approach is trained and tested by sampling 2^{11} 3D coordinates per query. Since the considered windows are significantly smaller than the volumes from previous approaches, less sampled points are used. A gun classification is assumed in case the output of a HiLo-Network is above 0.5.

In sum, 18 networks are trained only to evaluate the HiLo-Network. On average, training of one HiLo-Network takes 3 days on a single GPU. In addition, many more networks were trained during the development of HiLo-Networks. Unfortunately, not only

memory consumption of volumes grows non-linear in terms of dimension sizes but also the inference runtime of HiLo-Networks. As we need to responsibly use computational resources, we make simplifying assumptions for training, validating, and testing. For training and validating, pyramid locations are only sampled from the axis-aligned BBs of firearms. Additionally, we only load BBs instead of complete files into host RAM. For testing, we follow a similar approach. However, to strengthen our results we extend the BBs by 50 voxels in each direction. This procedure is legitimized by the construction of the 3D-CT dataset. The dataset is artificially constructed with a limited number of objects. For each gun, there are multiple instances for which only the spatial arrangement of the objects varies. Hence, most object-gun combinations are covered by the extended BBs. Furthermore, huge parts of 3D-CT volumes represent air.

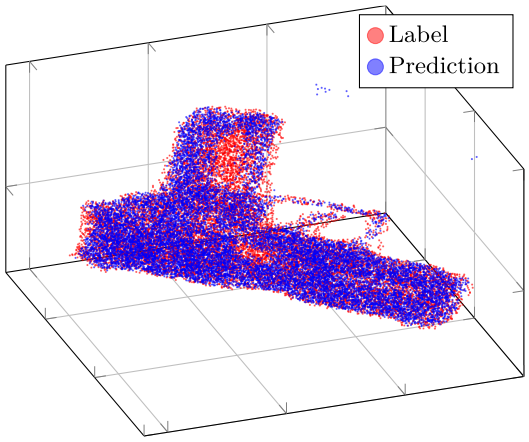


Figure 6. Exemplary HiLo-Network CNN-approach semantic segmentation result.

The test IoUs for the CNN-approach can be seen in table 5. For the small window size $w = 16$, the biggest tested downsampling factor of $d = 8$ results in a strong IoU improvement in comparison to $d = 2$ or $d = 4$. For $w = 32$ and $w = 48$ we observe a contrasting behavior. We account this to the specifications of the 3D-CT dataset. In particular, we assume that material properties are more important than shapes. As a consequence, a *huge* moving window instead of a moving pyramid seems to be already sufficient for 3D semantic segmentation. Surprisingly, the IoU seems to decrease if the downsampling factor is increased. We apply zero padding in case a pyramid level is partially out of range for a volume. We presume that this leads to worse results. As the downsampling factor is increased, this effect is amplified in theory and describes the observed behavior. In table 6 the video RAM consumption during training time is displayed. It can be seen that using window sizes of $w = 16$ or $w = 32$ allows to train HiLo-Networks even on cheap consumer cards while mostly preserving test IoU results. For visualizations of CNN-approach results see figures 6 and 8.

First in-first out	Hardness	Occurrence
0.6836	0.6615	0.6838

Table 8. Test IoUs for the CNN-approach. The results are divided into first in-first out, occurrence based, and hardness based training queues.

The results for the O-Net-approach, shown in table 7, are not discussed in detail. Overall, test IoUs are significantly worse than those of the CNN-approach and highly unstable. This indicates a general problem of O-Nets for 3D semantic segmentation, at least in combination with a sliding pyramid/window. Hence, variations due to varying values of w and d cannot be reasonably explained.

4.4. Training Queues

In table 8, test IoUs after 20 training epochs are shown separately for the three introduced training queues. For this, a CNN-approach HiLo-Network with $w = 32$ and $d = 2$ is trained. The size of each training queue is 512.

Besides the general speed advantages of training queues, we cannot add additional benefits through more complex batch sampling schemes. We identify two reasons for this. Generally, independent of a particular gun, the surrounding objects are the same for any instance of 3D-CT. Hence, favoring instances over others barely improves training. The second reason only holds for the hardness-based queue. Pyramids/windows only represent small fractions of an instance and probably do not give a good estimate of the actual hardness of a complete instance.

5. Conclusion

In this work, we propose two novel architectures for 3D semantic segmentation of voxelized volumes. We test our methods on a 3D CT scan dataset for firearm detection in luggage. First, we introduce a super resolution Occupancy Network. Unfortunately, the tested super resolution O-Net architectures are not able to achieve the desired results. Nonetheless, we demonstrate that O-Nets can be successfully deployed to extract bounding boxes of shapes. Second, we propose HiLo-Networks, a new scaleable neural network architecture to perform 3D semantic segmentation. Our ex-

periments show that HiLo-Networks can reliably segment guns within luggage. At this, they are memory efficient as well as scalable in terms of input resolution. In particular, HiLo-Networks can be trained on consumer level GPUs.

Future research should evaluate HiLo-Networks on other datasets as we expect them to work on a broader range of 3D datasets. Another aspect to be investigated is the inference time. In our implementation full volumes are segmented by using multiple forward passes. Efficiently extracting a bounding box before performing segmentation could be the key to reduce the inference time. For instance, our adaption of the O-Net for BB construction could be used.

References

[1] M. Bansal, M. Kumar, and M. Kumar. 2D Object Recognition Techniques: State-of-the-Art Work. *Archives of Computational Methods in Engineering*, 2020. 1

[2] A. Boulch, J. Guerry, B. Saux, and N. Audebert. Snapnet: 3d point cloud semantic labeling with 2d deep segmentation networks. *Computers & Graphics*, 71, 12 2017. 2

[3] H. Chen, Q. Dou, L. Yu, and P.-A. Heng. VoxResNet: Deep Voxelwise Residual Networks for Volumetric Brain Segmentation. pages 1–9, 2016. 2

[4] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3D U-net: Learning dense volumetric segmentation from sparse annotation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9901 LNCS:424–432, 2016. 2

[5] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao. GVCNN: Group-View Convolutional Neural Networks for 3D Shape Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 264–272, 2018. 2

[6] K. Guo, D. Zou, and X. Chen. 3D mesh labeling via deep convolutional neural networks. *ACM Transactions on Graphics*, 35(1), dec 2015. 2

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 5, 7

[8] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 3, 7

[9] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi. A Survey of the Recent Architectures of Deep Convolutional Neural Networks. pages 1–68, 2019. 1

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 5, 7

[11] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. *CoRR*, abs/1706.02515, 2017. 7

[12] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. 7

[13] Z. Liu, H. Tang, Y. Lin, and S. Han. Point-Voxel CNN for Efficient 3D Deep Learning. (NeurIPS), 2019. 2

[14] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst. Geodesic Convolutional Neural Networks on Riemannian Manifolds. *Proceedings of the IEEE International Conference on Computer Vision*, 2015-February:832–840, 2015. 2

[15] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3D reconstruction in function space. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:4455–4465, 2019. 2, 3, 4

[16] E. Perez, H. De Vries, F. Strub, V. Dumoulin, and A. Courville. Learning visual reasoning without strong priors. *arXiv preprint arXiv:1707.03017*, 2017. 3

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:77–85, 2017. 1

[18] C. R. Qi, H. Su, M. Niebner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view CNNs for object classification on 3D data. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:5648–5656, 2016. 1, 2

[19] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 2017-December:5100–5109, 2017. 1

[20] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9351:234–241, 2015. 2

[21] X. Shen. A survey of Object Classification and Detection based on 2D / 3D data. 1

[22] A. Taime, A. Saaidi, and K. Satori. A new semantic segmentation approach of 3D mesh using the stereoscopic image colors. *Multimedia Tools and Applications*, 77(20):27143–27162, 2018. 2

[23] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese. SEGCloud: Semantic segmentation of 3D point clouds. In *Proceedings - 2017 International Conference on 3D Vision, 3DV 2017*, pages 537–547, 2018. 1

[24] I. Ulku and E. Akagunduz. A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D images. 2019. 1

[25] P. Wang, Y. Gan, P. Shui, F. Yu, Y. Zhang, S. Chen, and Z. Sun. 3D shape segmentation via shape fully convolutional networks. *Computers and Graphics (Pergamon)*, 70:128–139, 2018. 2

[26] P. S. Wang, Y. Liu, Y. X. Guo, C. Y. Sun, and X. Tong. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Transactions on Graphics*, 36(4), 2017. 2

[27] Z. Wang and F. Lu. VoxSegNet: Volumetric CNNs for Semantic Part Segmentation of 3D Shapes. *IEEE Transactions on Visualization and Computer Graphics*, 14(8):1–1, 2019. 2

[28] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015. 5

[29] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi. Convolutional neural networks: an overview and application in radiology, 2018. 1

6. Appendix

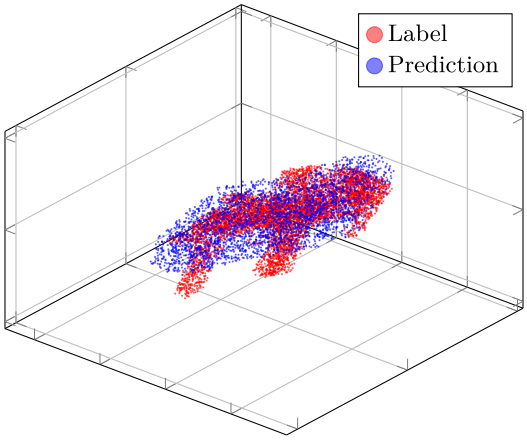


Figure 7. Exemplary super resolution O-Net semantic segmentation result.

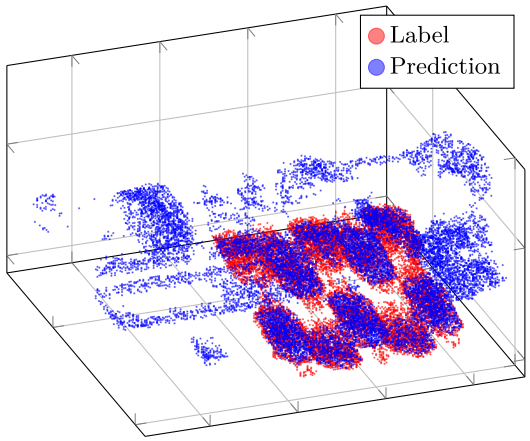


Figure 8. Exemplary HiLo-Network CNN-approach semantic segmentation result.

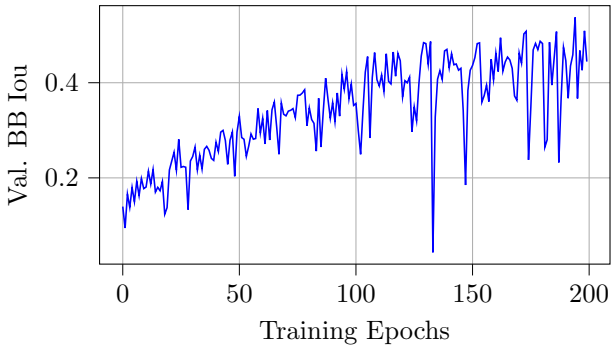


Figure 9. Curve of validation IoUs during training of our alternative O-Net architecture.

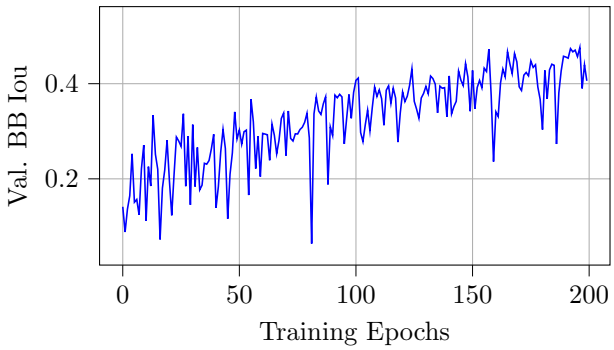


Figure 10. Curve of validation IoUs during training of the standard O-Net architecture.