# Supplementary Material for SliceNets — A Scalable Approach for Object Detection in 3D CT Scans

Anqi Yang[1], Feng Pan[2], Vishwanath Saragadam[1], Duy Dao[2],
Zhuo Hui[1], Jen-Hao Rick Chang[1], Aswin C. Sankaranarayanan[1]
[1] Carnegie Mellon University, [2] IDSS Corporation

## 1. More on Slice-and-fuse

Figure 1 illustrates the proposed slicing operation. The lower row demonstrates a set of XY slices generated from the input density volume. Compared to projecting the whole volume at one time, the proposed slice operation alleviates the heavy occlusion from the background clutter objects and can potentially expose the target object directly to the detectors. For example, there is a target gun exposed in the middle image of the lower row.

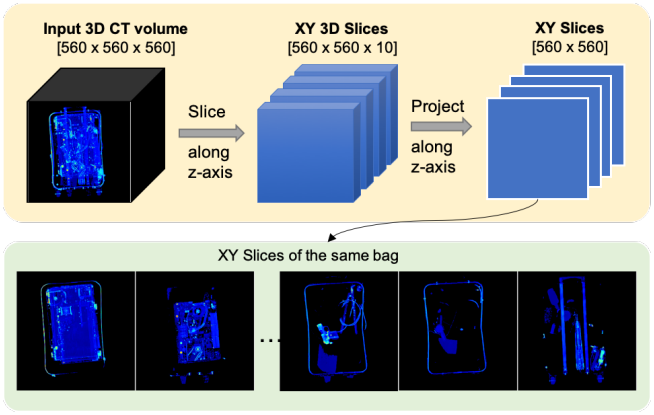


Figure 1. Illustration of slice-operation along z-axis.

## 2. Retinal-SliceNet implementation details

Fig 2 shows the architecture of Retinal-SliceNet. The only component that needs to be trained is RetinaNet which is applied to each individual 2D slice. The RetinaNet has two key ingredients, that we briefly summarize. First, it uses the feature pyramid network [1] as backbone to extract features from the input image and construct a feature pyramid. For each pyramid level, they attach two FCNs to it - a classification subnet that predicts the confidence score of each object class, and a box regression subnet that regresses the bounding box locations. Second, given that the number of pixels occupied by the target is usually a small fraction of the entire slice, there is often a severe class imbalance.

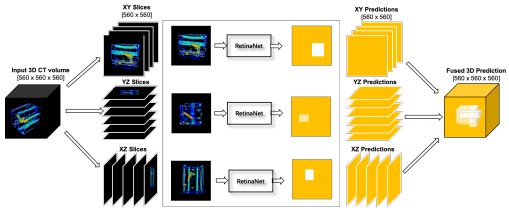


Figure 2. Retinal-SliceNet architecture.

To address this, a focal loss is adopted at training which under-penalizes small errors in the predicted score while over-penalizing larger errors; the net effect is that a large portion of the training data that is easily classified does not significantly affect the model thereby providing robustness to class imbalance.

The training of Retinal-SliceNet involves only the training of [2]. We use all 118,790 positive slices as well as 11,879 negative slices randomly selected from all negative training samples. For negative slices, we set a target bounding box to the upper left pixel and all other positions as background clutter. One Nvidia TITAN Xp GPU is used during training. The input images are of size $560 \times 560$. The model is trained with a mini-batch size of 8 images for 200 epochs. We use a pretrained FPN [1] as backbone network. We adopt SGD for optimization with a learning rate of 0.0001, a weight decay of 0.0001 and momentum of 0.9. For the focal loss function, we use $\gamma = 2$ and $\alpha = 0.25$.

After obtaining the bounding boxes locations and scores for each slice, we generate a prediction slice, whose pixels in the bounding boxes are filled with corresponding scores and background pixels with value 0. All predicted slices are fed to the fusion stage to generate a 3D prediction. We further threshold the 3D prediction to keep only regions with high possibility to be target objects and give a bounding box to each connected regions.

## 3. U-SliceNet implementation details

As is shown in Figure 3, U-SliceNet has three key components — voxel-wise labeling network, region proposal,

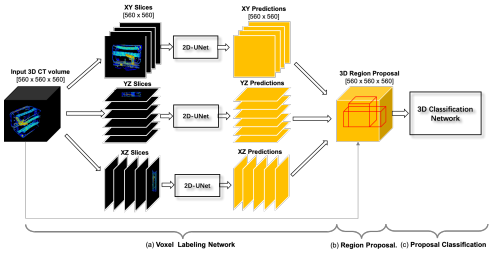


Figure 3. U-SliceNet architecture.

and proposal classification. We will describe the training and implementation details of each component in this section.

**Voxel-wise labeling network.** Given a volumetric input, the voxel-labeling network generates a 3D prediction using slice-and-fuse strategy. The only component we need to train is a 2D UNet that is applied to each slice. The 2D UNet has the following architecture. It has eight fully convolutional layers `conv1-conv8`, followed by eight deconvolution layers `deconv1-deconv8`, with a skip connection connecting each pair of them. In the downsampling pathway, each `convi` (`i`=1,...,8) layer outputs a feature map with a spatial resolution of $2^i$ lower than the input image and $32 * 2^i$ feature channels. Each conv layer is followed by a LeakyReLU ($\alpha = 0.2$) activation and a batch normalization. The bottleneck feature has a spatial resolution of $1 \times 1$ with 8192 channels. In the upsampling pathway, each `deconvi` (`i`=8,...,1) layer outputs a feature map with a spatial resolution of $2^{(i-1)}$ lower than the output image, and the final has the same number of channels as the number of object classes. Each `deconvi` (`i`=8,...,2) layer is followed by an ReLU activation and a batch normalization layer.

During the training of 2D-UNet [3], each input image is resized to $256 \times 256$. We use all 118,790 positive slices as well as 11,879 negative slices randomly selected from all negative training samples. We use one GPU with a mini-batch of 16 images. We adopt Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ and train the network for 200 epochs. The initial learning rate is 0.002 and is updated every 50 epochs according to lambda update rule. We adopt focal loss as training criterion, with $\gamma = 5$ and $\alpha = 0.5$.

**Region proposal.** After obtaining the voxel-wise prediction, we use a threshold $r_1$ to filter out regions that have small probabilities to be target objects. Among the kept regions, we select anchor points using a spatial interval of 15 voxels, and propose a fixed set of 155 anchors at each location.

**Proposal classification.** We train two small 3D classification networks on the region proposals generated from training baggage — one for gun class and one for sharp class. Each network has three convolutional blocks followed by two fully connected layers. Each convolutional block is composed of four 3D convolution layers, each followed by an ReLU activation. conv1 of each block outputs a feature map with spatial resolution $2^i$ lower than the input 3D feature map and $64 \times 2^i$ number of feature channels. And `conv2-4` of each block keep the spatial resolution and number of channels the same.

When training gun classifier, we use all training bags with guns in them, as well as $10\%$ of clear bags and bags containig knives. Similarly for training the sharp classifier. During each training iteration, the network takes in a mini-batch of 5 bags and randomly selects upto 5000 negative proposals and positive proposals. The training labels are determined according to the Intersection-over-Union (IoU) ratios with the ground-truth bounding box as in [4]. The proposals that have IoU greater than 0.4 are assigned as positive samples, and those have IoU less than 0.1 are assigned as negative samples. Four GPUs are used in training. We use an Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate of 0.0001 and momentum of 0.9 and train the network for 500 epochs. A focal loss with $\alpha = 0.25$ and $\gamma = 2$ is used as training criterion.

## Acnowledgement

The authors thank Ms. Bijia Chen, Mr. Mark Caron, Dr. James Connelly, and Dr. Omar AlKofahi. The authors also thank Profs. Vijaya Kumar Bhagavatula, Srinivasa Narasimhan, and David Held for insightful comments and suggestions.

This work is supported by Department of Homeland Security (DHS) Science and Technology Directorate (S&T) under Contract Number HSHQDC-17-C-B0020. Any opinions, findings, conclusions or recommendations expressed in this thesis do not reflect the view of DHS S&T or IDSS Corporation. All of the weapons images are from a DETECT 1000 that is not in a deployed system configuration.

## References

[1] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[2] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.