Explaining 3D Point Cloud Semantic Segmentation Models Through Adversarial Attacks

Supplementary Material

7. Algorithm

In our implementation, we address the restrictions of the CTA-seg algorithm mentioned in Sec. 3.1 as follows. We ensure that constraint 1 is achieved by passing only the initial probabilities for the target class to the loss function calculation. We then mask the obtained loss values before the backpropagation step by setting to zero the values for points not originally in the target class. This ensures that only such values are considered for the gradient computation. For constraint 2, we mask the obtained gradient values, setting to zero all gradient values that do not belong to the critical points and the target feature(s) currently being shifted. Algorithm 1 shows the in-detail algorithm for CTA-seg.

8. Visualizations

Fig. 8 and Fig. 9 show sample successful adversarial attacks for the spatial and color features, respectively, across all architectures for the $Chair$ class. In both cases, the degree of change needed for PointNet++ is so small as to be imperceptible.

Algorithm 1: CTA-seg

```
Data: Input point cloud P with shape N × D, deep learning model M, IG saliency map IG(P) with shape N × D,
      optimization rate α, distance-penalizing weight β, spatial distance function D_s, color distance function D_c,
      target_class, target_features list, local_max_iterations, global_max_iterations, iou_threshold below
      which we consider the attack successful
Result: Adversarial example P_adv with shape N × D
attributions ← flatten(sum of IG(P) across target_features);
attr_indexes ← indices of attributions elements sorted in descending order;
max_points ← count(attributions > 0);
global_iter_count ← 0;
first_it ← True;
for n_points from 1 to max_points with step = 50 do
    activation_track ← list() ;    /* Initializing activation track for local stopping */
    updated_point_cloud ← deepcopy(P);
    target_idxs ← attr_indexes[0 : n_points];
    for local_iteration from 0 to local_max_iterations do
        predictions ← softmax(M(updated_point_cloud));
        predicted_classes ← argmax(predictions, dim = 1);
        optimizer ← newly initialized Adam optimizer;
        if first_it then
            original_target_class_idxs ← flatten(argwhere(predicted_classes = target_class));
            first_it ← False;
        end
        updated_target_class_idxs ← flatten(argwhere(predicted_classes = target_class));
        local_iou ← IoU(original_target_class_idxs, updated_target_class_idxs);
        if local_iou < iou_threshold then
            return updated_point_cloud ;                              /* Successful attack */
        end
        activation ← mean(predictions[:, target_class]);
        activation_track.append(activation);
        loss ← α ∗ log(predictions[:
         , target_class]) + β ∗ [D_s(P, updated_point_cloud) + D_c(P, updated_point_cloud)] ;      /* We are
         only interested in decreasing the prediction of the target class */
        loss_mask ← zero-mask of the same shape as loss;
        loss_mask[original_target_class_idxs] ← 1 ; /* Mask the loss values to only consider
         the points belonging to the target class for the gradient computation */
        loss ← loss ∗ loss_mask;
        loss.backward();
        grad_mask ← zero-mask of the same shape as updated_point_cloud;
        grad_mask[target_idxs, target_features] ← 1 ; /* Mask the gradients to only optimize
         for the points to be shifted across the target feature(s) */
        updated_point_cloud.grad ← updated_point_cloud.grad ∗ grad_mask;
        optimizer.step() ;         /* Updating the adversarial example updated_point_cloud */
        if mean increase in activation_track then
            break ;                                                 /* Local stopping criterion */
        end
    end
    global_iter_count ← global_iter_count + 1;
    if global_iter_count > global_max_iterations then
        break ;                                                    /* Global stopping criterion */
    end
end
return Failed ;                                                        /* Unsuccessful attack */
```

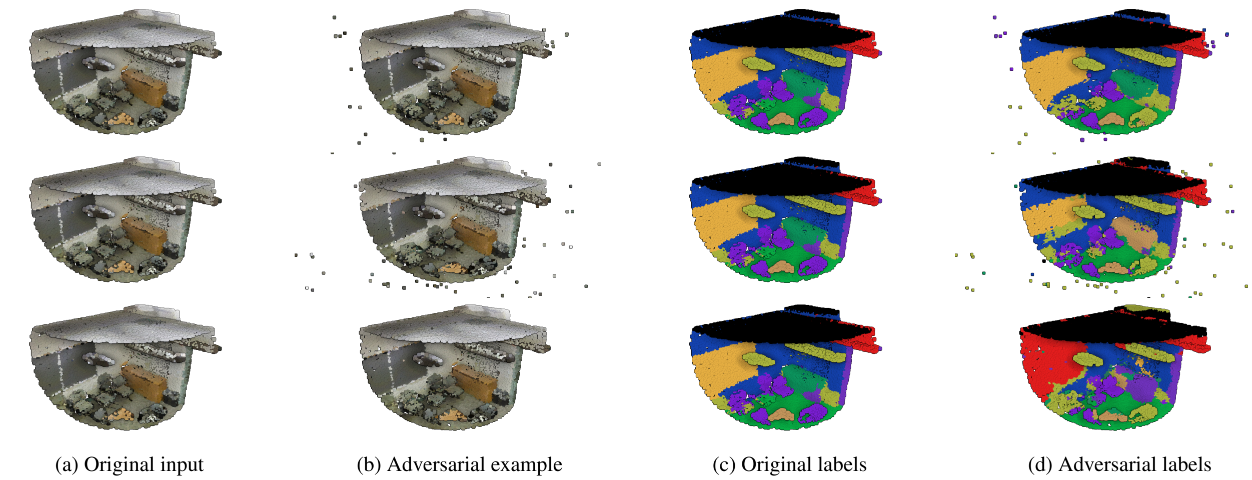

Figure 8. Sample CTA-seg outputs for attacks on spatial features for the $Chair$ class (purple labels). From top to bottom, PTv3, KP-FCNN, and PointNet++.

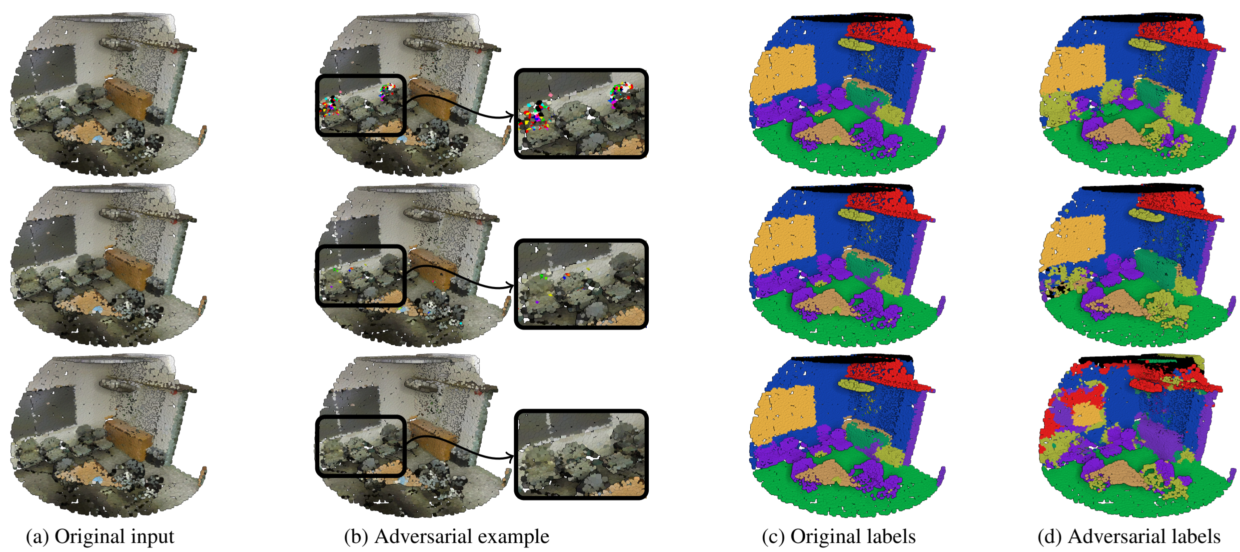

Figure 9. Sample CTA-seg outputs for attacks on color features for the $Chair$ class (purple labels). From top to bottom, PTv3, KP-FCNN, and PointNet++.