A Supplementary materials for "Improving 3D Object Detection through Progressive Population Based Augmentation"

Table 7: List of point cloud transformations in the search space for point cloud 3D object detection

Operation Name	Description
GroundTruthAugmentor [31]	Augment the bounding boxes from a ground truth data base ($<$ 25 boxes per scene)
RandomFlip [33]	Randomly flip all points along the Y axis.
WorldScaling [37]	Apply global scaling to all ground truth boxes and all points.
RandomRotation [37]	Apply random rotation to all ground truth boxes and all points.
GlobalTranslateNoise	Apply global translating to all ground truth boxes and all points along x/y/z axis.
FrustumDropout	All points are first converted to spherical coordinates, and then a point is randomly selected. All points in the frustum around that point within a given phi, theta angle width and distance to the original greater than a given value are dropped randomly.
FrustumNoise	Randomly add noise to points within a frustum in a converted spherical coordinates.
RandomDropout	Randomly dropout all points.

Table 8: The range of augmentation parameters that can be searched by Progressive Population Based Augmentation algorithm for each operation

Operation Name	Parameter Name	Range
GroundTruthAugmentor	vehicle sampling probability	[0, 1]
	pedestrian sampling probability	[0, 1]
	cyclist sampling probability	[0, 1]
	other categories sampling probability	[0, 1]
RandomFlip	flip probability	[0, 1]
WorldScaling	scaling range	[0.5, 1.5]
RandomRotation	maximum rotation angle	[0, $\pi/4$]
GlobalTranslateNoise	standard deviation of noise on x axis	[0, 0.3]
	standard deviation of noise on y axis	[0, 0.3]
	standard deviation of noise on z axis	[0, 0.3]
FrustumDropout	theta angle width of the selected frustum	[0, 0.4]
	phi angle width of the selected frustum	[0, 1.3]
	distance to the selected point	[0, 50]
	the probability of dropping a point	[0, 1]
	drop type[6]	{'union', 'intersection'}
FrustumNoise	theta angle width of the selected frustum	[0, 0.4]
	phi angle width of the selected frustum	[0, 1.3]
	distance to the selected point	[0, 50]
	maximum noise level	[0, 1]
	noise type[7]	{'union', 'intersection'}
RandomDropout	dropout probability	[0, 1]

[6] Drop points in either the union or intersection of phi width and theta width.
[7] Add noise to either the union or intersection of phi width and theta width.

Algorithm 1 Progressive Population Based Augmentation

Input: data and label pairs $(\mathcal{X}, \mathcal{Y})$
Search Space: $\mathcal{S} = \{op_i : params_i\}_{i=1}^{n}$
Set $t = 0$, $num\_ops = 2$, population $\mathcal{P} = \{\}$, best params and metrics for each operation $historical\_op\_params = \{\}$
while $t \neq \mathcal{N}$ **do**
  for θ_i^t in $\{\theta_1^t, \theta_2^t, ..., \theta_{\mathcal{M}}^t\}$ (asynchronously in parallel) **do**
    # Initialize models and augmentation parameters in current iteration
    if $t == 0$ **then**
      $op\_params_i^t$ = Random.sample($\mathcal{S}$, num_ops)
      Initialize θ_i^t, λ_i^t, $params$ of $op\_params_i^t$
      Update λ_i^t with $op\_params_i^t$
    else
      Initialize θ_i^t with the weights of $winner_i^{t-1}$
      Update λ_i^t with λ_i^{t-1} and $op\_params_i^t$
    end if
    # Train and evaluate models, and update the population
    Update θ_i^t according to formular (2)
    Compute metric $\Omega_i^t = \Omega(\theta_i^t)$
    Update $historical\_op\_params$ with $op\_params_i^t$ and Ω_i^t
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{\theta_i^t\}$
    # Replace inferior augmentation parameters with better ones
    $winner_i^t \leftarrow$ Compete(θ_i^t, Random.sample($\mathcal{P}$))
    if $winner_i^t \neq \theta_i^t$ **then**
      $op\_params_i^{t+1} \leftarrow$ Mutate($winner_i^t$'s op_params, $historical\_op\_params$)
    else
      $op\_params_i^{t+1} \leftarrow op\_params_i^t$
    end if
  end for
  $t \leftarrow t + 1$
end while

Algorithm 2 Exploration Based on Historical Data

Input: $op\_params = \{op_i : params_i\}_{i=1}^{num\_ops}$, best params and metric for each operation $historical\_op\_params$
Search Space: $\mathcal{S} = \{(op_i, params_i)\}_{i=1}^{n}$
Set $exploration\_rate = 0.8$, $selected\_ops = []$, $new\_op\_params = \{\}$
if Random(0, 1) $<$ $exploration\_rate$ **then**
  $selected\_ops = op\_params$.Keys()
else
  $selected\_ops$ = Random.sample($\mathcal{S}$.Key(), num_ops)
end if
for i in Range(num_ops) **do**
  # Choose augmentation parameters, which successors will mutate
  # to generate new parameters
  if $selected\_ops[i]$ in op_params.Keys() **then**
    $parent\_params = op\_params[selected\_ops[i]]$
  else if $selected\_ops[i]$ in $historical\_op\_params$.Keys() **then**
    $parent\_params = historical\_op\_params[selected\_ops[i]]$
  else
    Initialize $parent\_params$ randomly
  end if
  $new\_op\_params[selected\_ops[i]]$ = MutateParams($parent\_params$)
end for

Acknowledgements

We would like to thank Peisheng Li, Chen Wu, Ming Ji, Weiyue Wang, Zhinan Xu, James Guo, Shirley Chung, Yukai Liu, Pei Sun of Waymo and Ang Li of DeepMind for helpful feedback and discussions. We also thank the larger Google Brain team including Matthieu Devin, Zhifeng Chen, Wei Han and Brandon Yang for their support and comments.