



香港中文大學

The Chinese University of Hong Kong

OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds

Shaoshuai Shi

Ph.D. Student of Multimedia Laboratory

The Chinese University of Hong Kong

Nov. 30, 2020

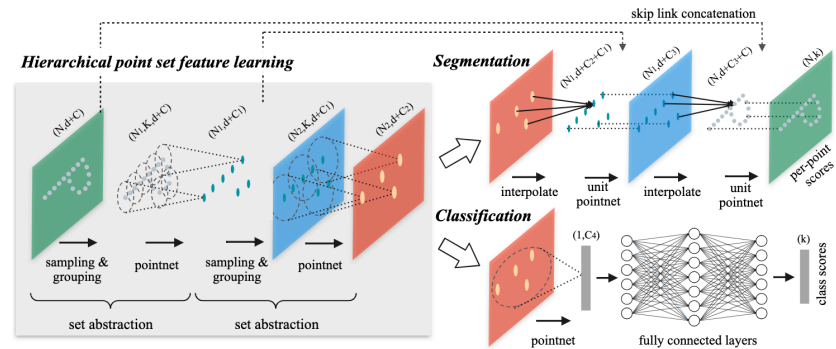
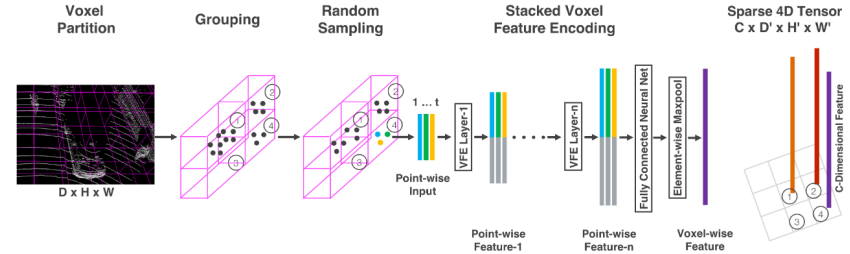
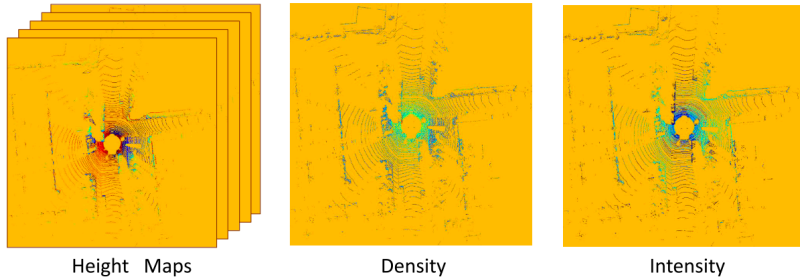
Outline

- Introduction to 3D object detection from point clouds
 - Feature learning from point clouds (voxel-based vs. point-based)
 - 3D box generation (anchor-based vs. anchor-free)
 - Two-stage detector with 3D box refinement
- OpenPCDet architecture and experiments

Feature learning from point clouds



- 2D CNN with bird's view representation
- 3D Voxel CNN (or sparse conv) with Voxel Feature Encoding (VFE)
- PointNet++ series
- Combination of the above strategies

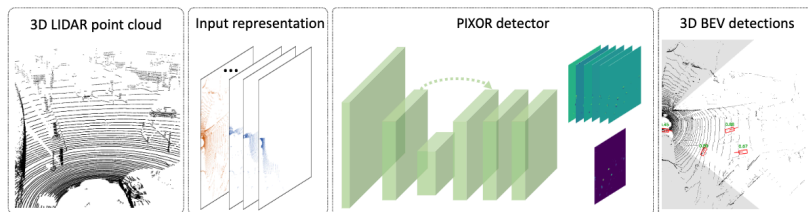


Reference

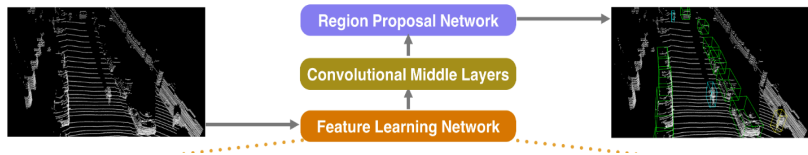
1. Xiaozhi Chen, et al., Multi-View 3D Object Detection Network for Autonomous Driving
2. Yin Zhou, et al., VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection
3. Charles Qi, et al., PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space

Anchor-based 3D box generation

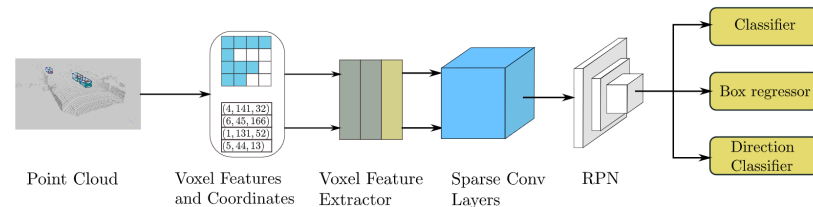
- Generate 3D boxes by selecting and tuning pre-defined 3D anchors on the bird-view 2D feature maps



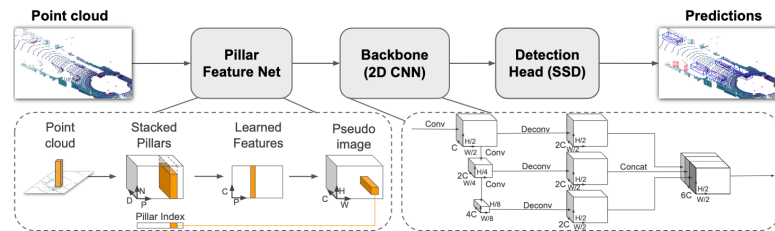
PIXOR



VoxelNet



SECOND



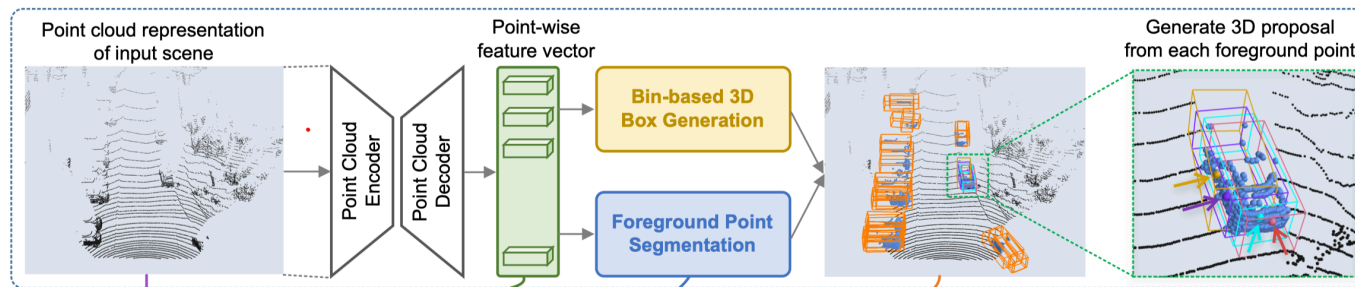
PointPillar

Reference

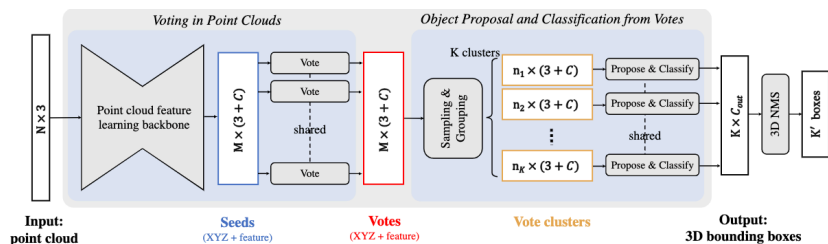
1. Yan Yan, et al., SECOND: Sparsely Embedded Convolutional Detection
2. Bin Yang, et al., PIXOR: Real-time 3D Object Detection from Point Clouds
3. Alex H. Lang, et al., PointPillars: Fast Encoders for Object Detection from Point Clouds
4. Yin Zhou, et al., VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection

Anchor-free 3D box generation

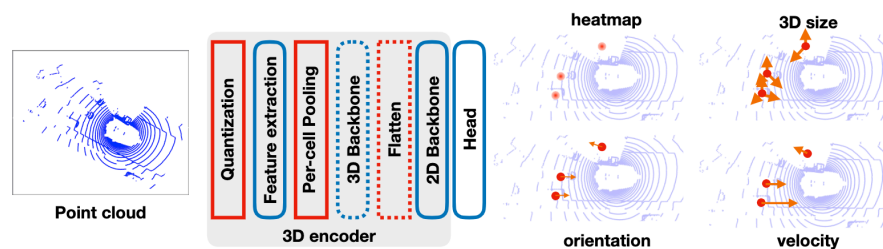
- Generating 3D box without predefined dense anchors



PointRCNN



VoteNet



CenterPoint

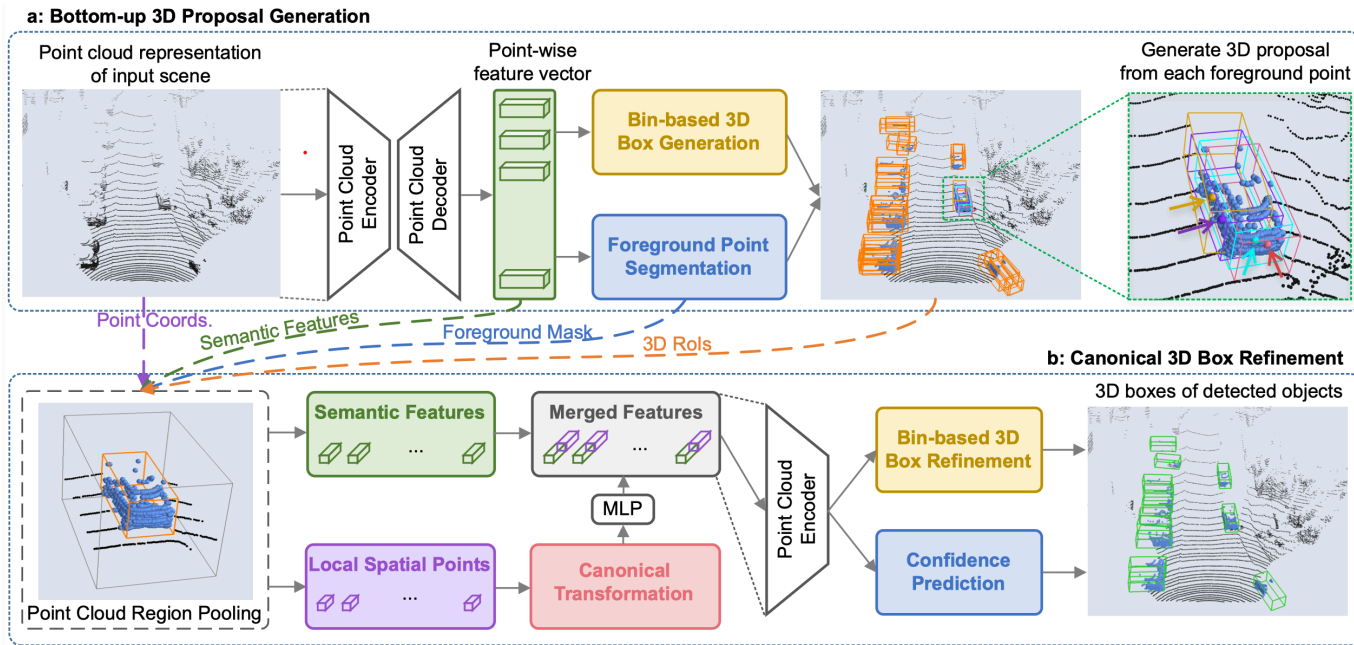
Reference

1. Shaoshuai Shi, et al., PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud
2. Charles Qi, et al., Deep Hough Voting for 3D Object Detection in Point Clouds
3. Tianwei Yin, et al., Center-based 3D Object Detection and Tracking

Two-stage detector with 3D box refinement



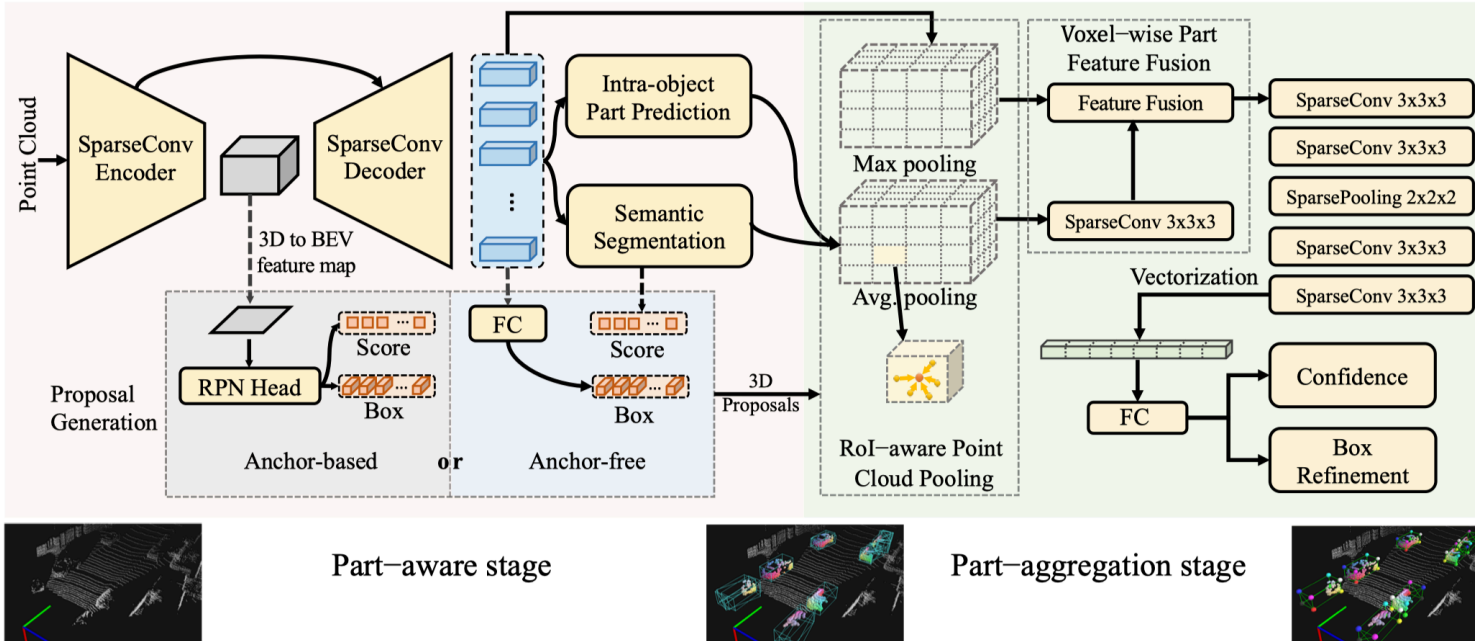
- PointRCNN: 3D box refinement with point cloud region pooling



Two-stage detector with 3D box refinement



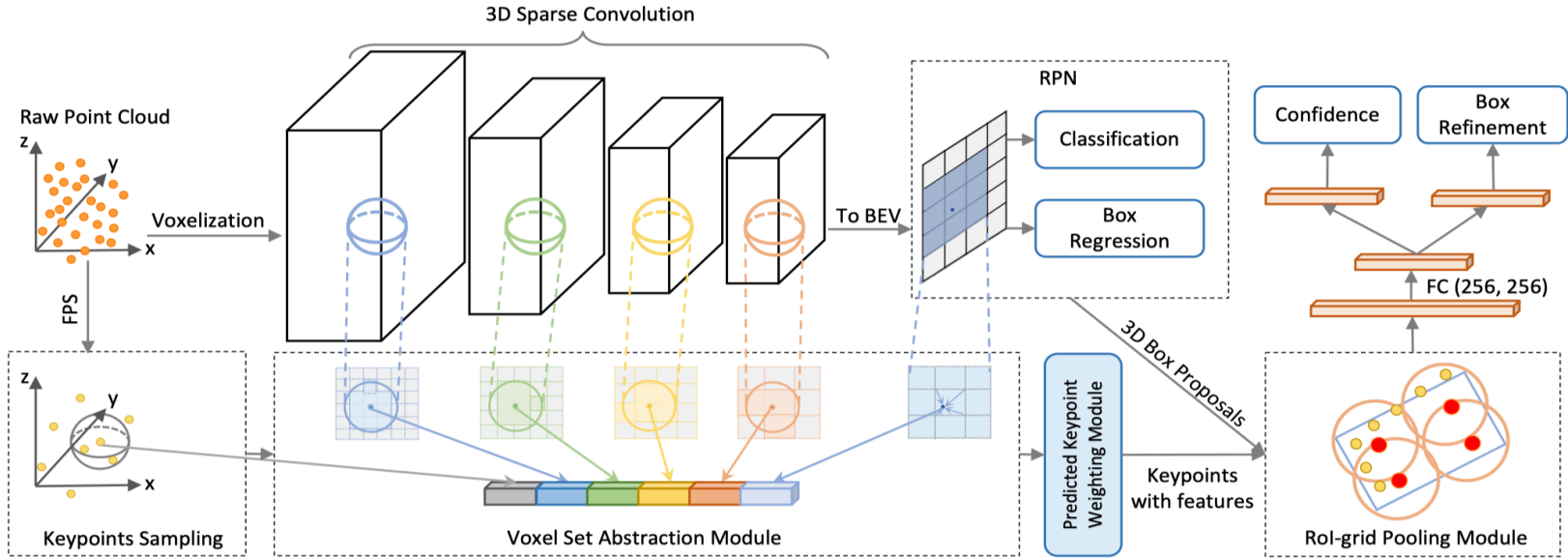
- Part-A2-Net: 3D box refinement with RoI-aware point cloud pooling



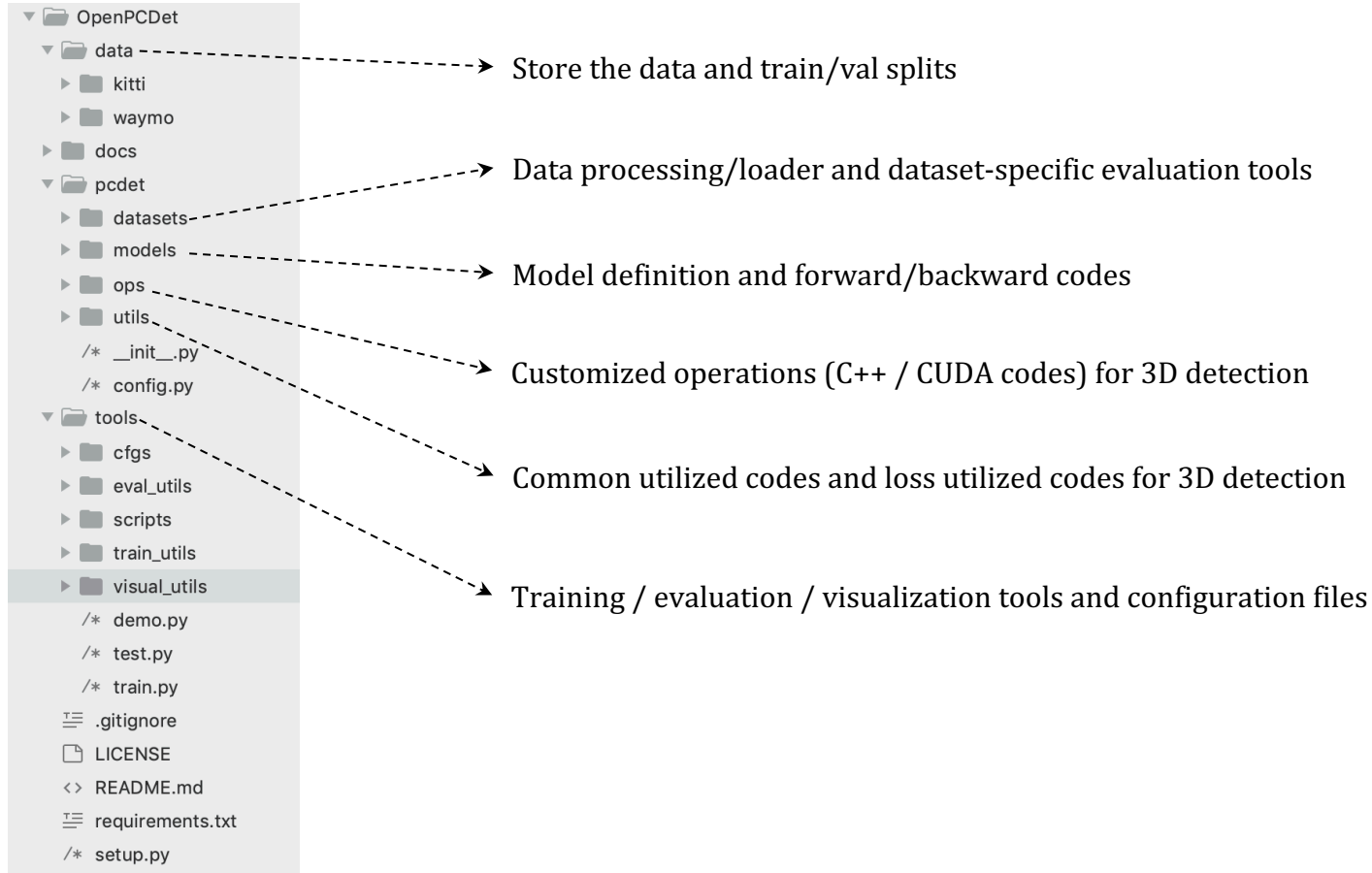
Two-stage detector with 3D box refinement



- PV-RCNN: 3D box refinement with RoI-grid pooling



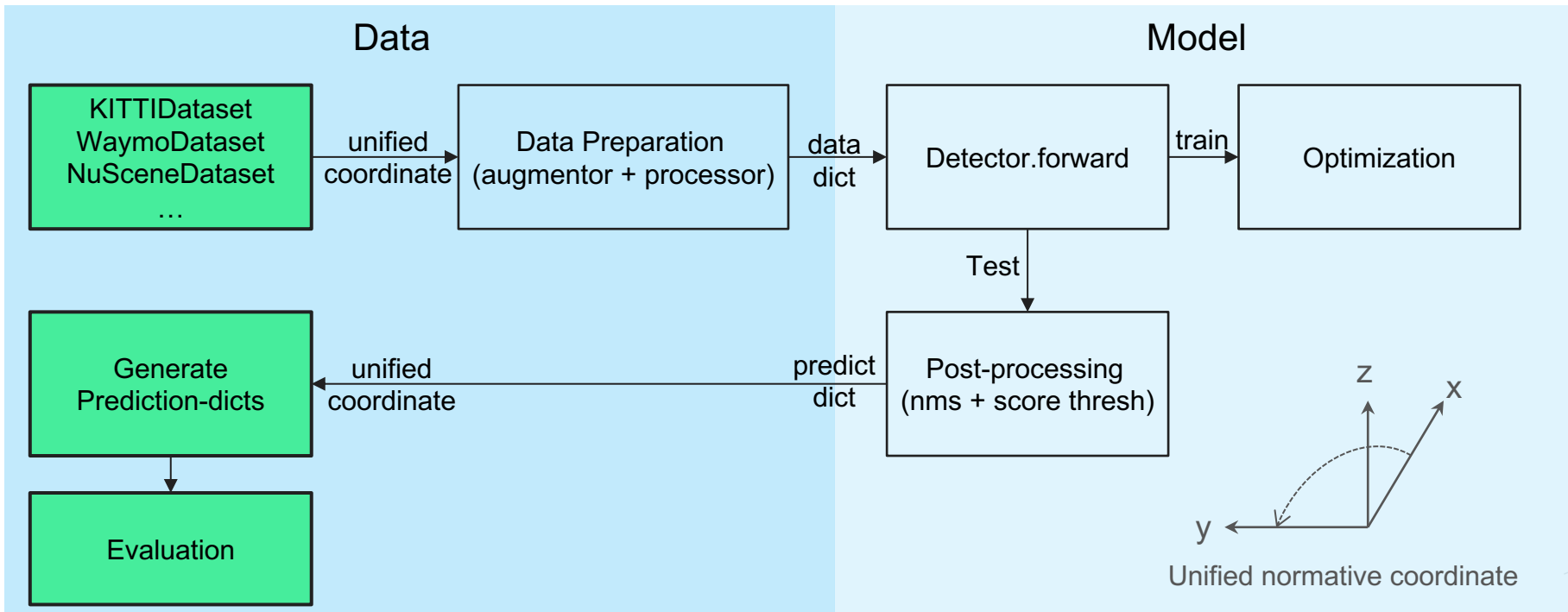
OpenPCDet: 3D Detection Toolbox



OpenPCDet: Architecture

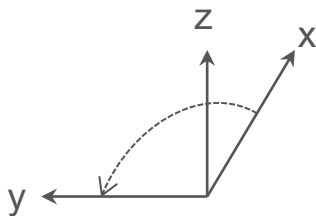
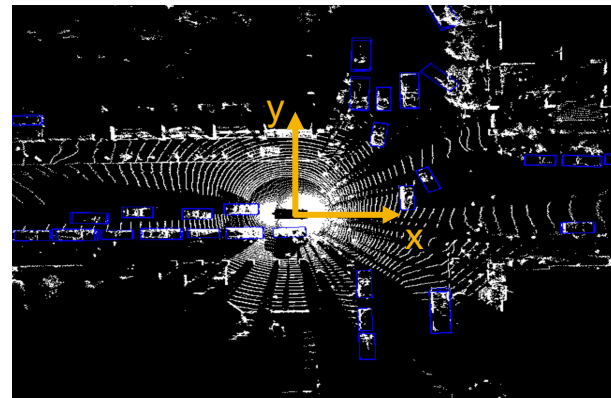


- Data-Model separation with unified coordinate and box definition across different datasets

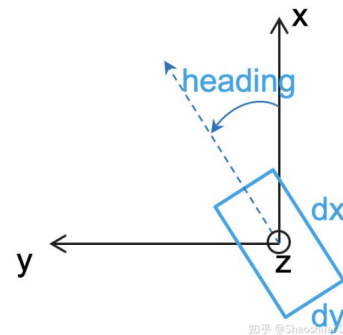


OpenPCDet: Unified coordinate and 3D box definition

- Unified coordinate:
 - x-axis points towards the front
 - y-axis points towards the left
 - z-axis points towards the up direction
- Unified 3D box definition:
 - $(cx, cy, cz, dx, dy, dz, \text{heading})$
 - Heading angle: the angle from x-axis to y-axis (heading = 0 for the direction of x-axis)

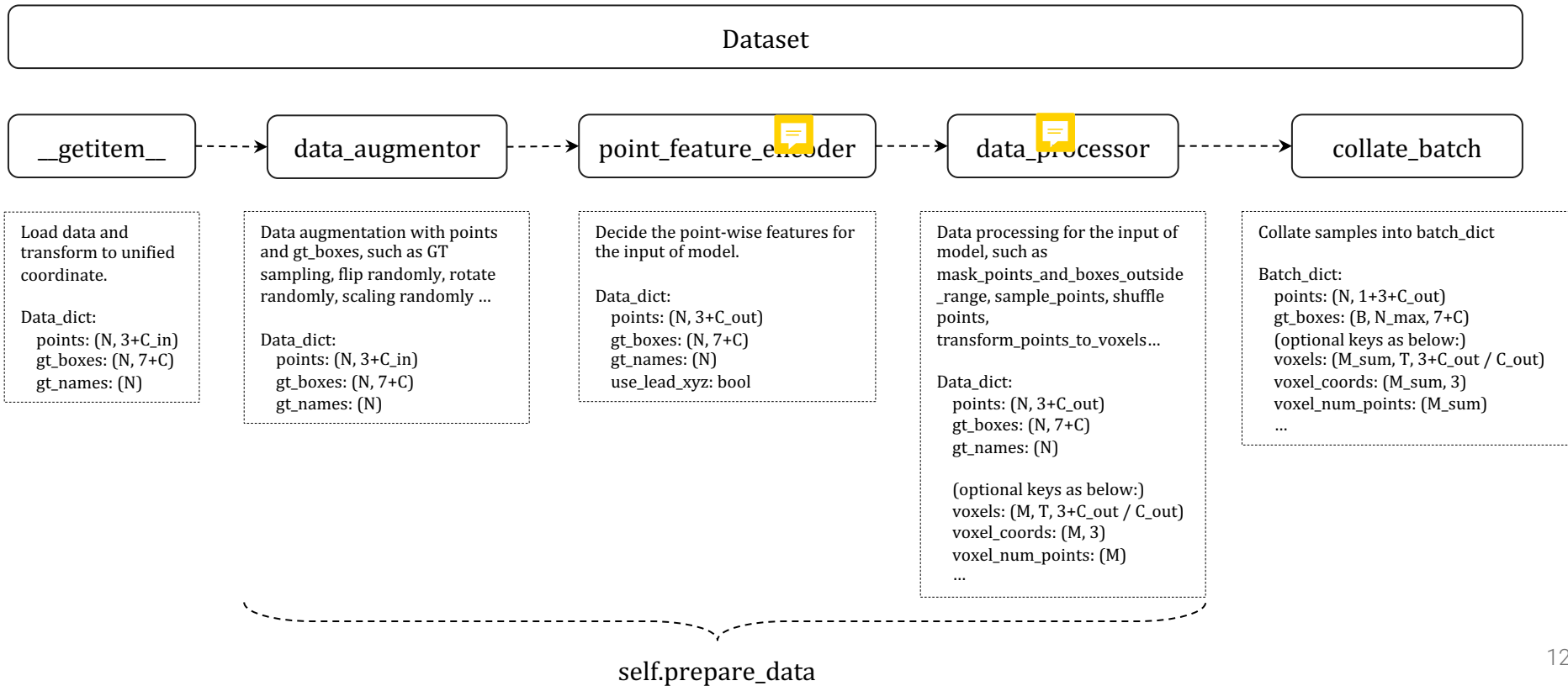


Unified normative coordinate



Unified 3D box definition

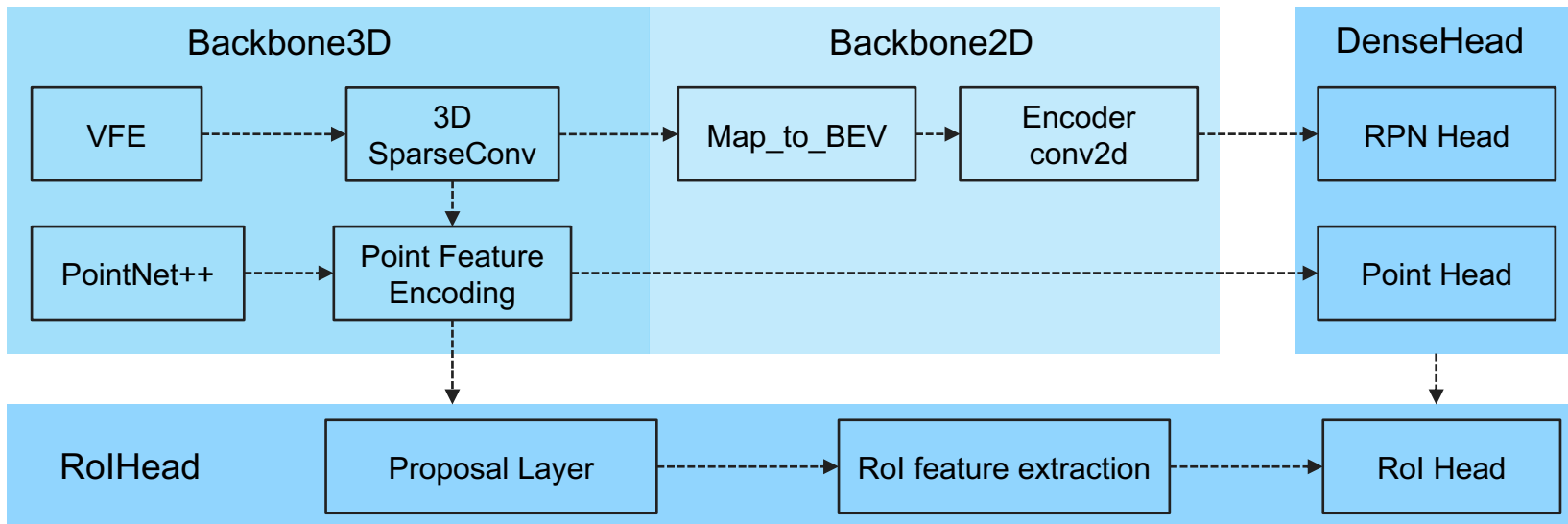
OpenPCDet: Data flow



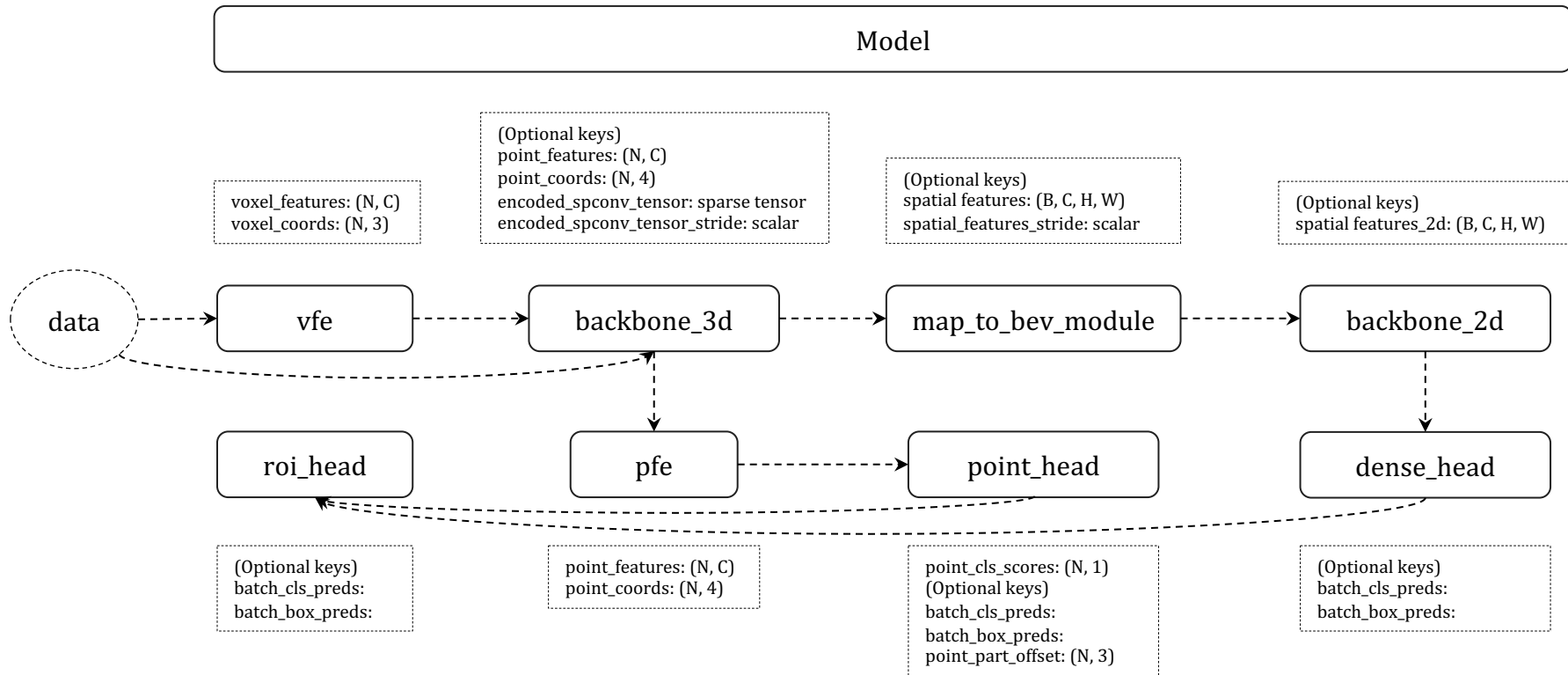
OpenPCDet: Modularization



- Modularization to support various 3D detectors within one framework
 - Backbone: 3D Voxel CNN / Sparse Conv / PointNet++ / 2D CNN
 - DenseHead: Anchor-based / Anchor-free / Point-based
 - RoIHead: proposal generation / RoI Pooling / proposal target assigner



OpenPCDet: Model topology



OpenPCDet: Build detector with configuration dict



- Build your detector with customized configs
 - Detector3DTemplate.build_networks()

```
self.module_topology = [  
    'vfe', 'backbone_3d', 'map_to_bev_module', 'pfe',  
    'backbone_2d', 'dense_head', 'point_head', 'roi_head'  
]
```

```
def build_networks(self):  
    model_info_dict = {  
        'module_list': [],  
        'num_rawpoint_features': self.dataset.point_feature_encoder.num_point_features,  
        'num_point_features': self.dataset.point_feature_encoder.num_point_features,  
        'grid_size': self.dataset.grid_size,  
        'point_cloud_range': self.dataset.point_cloud_range,  
        'voxel_size': self.dataset.voxel_size  
    }  
    for module_name in self.module_topology:  
        module, model_info_dict = getattr(self, 'build_%s' % module_name)(  
            model_info_dict=model_info_dict  
        )  
        self.add_module(module_name, module)  
    return model_info_dict['module_list']
```

OpenPCDet: Model forward & optimization



- Call each module sequentially with their topology
- Optimization (Calculate the losses within each head):
 - DETECTOR.get_training_loss()
 - HEAD.get_loss()

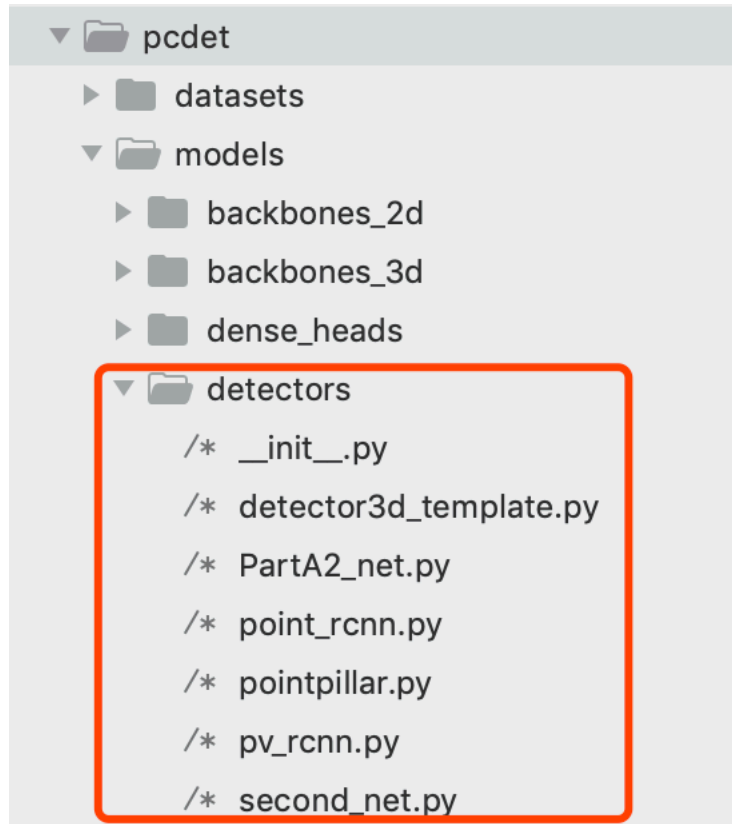
```
def forward(self, batch_dict):  
    batch_dict = self.vfe(batch_dict)  
    batch_dict = self.backbone_3d(batch_dict)  
    batch_dict = self.map_to_bev_module(batch_dict)  
    batch_dict = self.backbone_2d(batch_dict)  
    batch_dict = self.dense_head(batch_dict)
```

```
1 from .detector3d_template import Detector3DTemplate  
2  
3  
4 class PointRCNN(Detector3DTemplate):  
5     def __init__(self, model_cfg, num_class, dataset):  
6         super().__init__(model_cfg=model_cfg, num_class=num_class, dataset=dataset)  
7         self.module_list = self.build_networks()  
8  
9     def forward(self, batch_dict):  
10        for cur_module in self.module_list:  
11            batch_dict = cur_module(batch_dict)  
12  
13        if self.training:  
14            loss, tb_dict, disp_dict = self.get_training_loss()  
15  
16            ret_dict = {  
17                'loss': loss  
18            }  
19            return ret_dict, tb_dict, disp_dict  
20        else:  
21            pred_dicts, recall_dicts = self.post_processing(batch_dict)  
22            return pred_dicts, recall_dicts  
23  
24    def get_training_loss(self):  
25        disp_dict = {}  
26        loss_point, tb_dict = self.point_head.get_loss()  
27        loss_rcnn, tb_dict = self.roi_head.get_loss(tb_dict)  
28  
29        loss = loss_point + loss_rcnn  
30        return loss, tb_dict, disp_dict  
31
```


OpenPCDet: DetectorTemplate



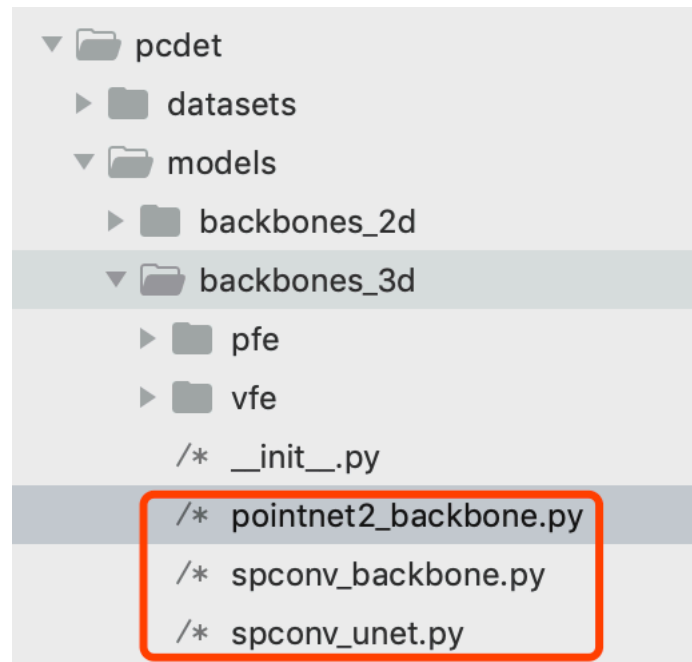
- Task: The top module for a detector
 - Build_networks
 - Forward
 - Loss calculation
 - Post_processing (NMS + score threshold)



OpenPCDet: 3D backbone networks



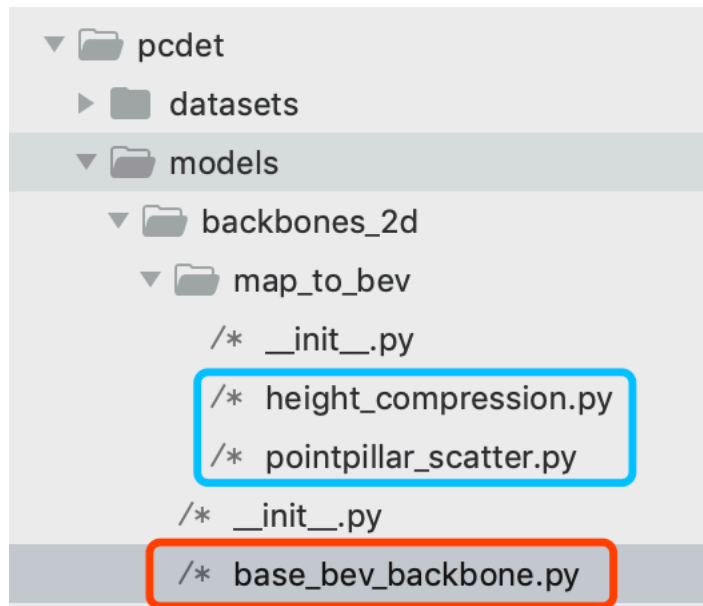
- Task: extract voxel-wise / point-wise features
- 3D encoder with sparse convolution (with VFE):
 - VoxelBackBone8x
 - VoxelResBackBone8x
- 3D UNet with sparse convolution (with VFE):
 - UNetV2
- Point-wise networks (PointNet++)
 - PointNet2MSG



OpenPCDet: 2D backbone networks



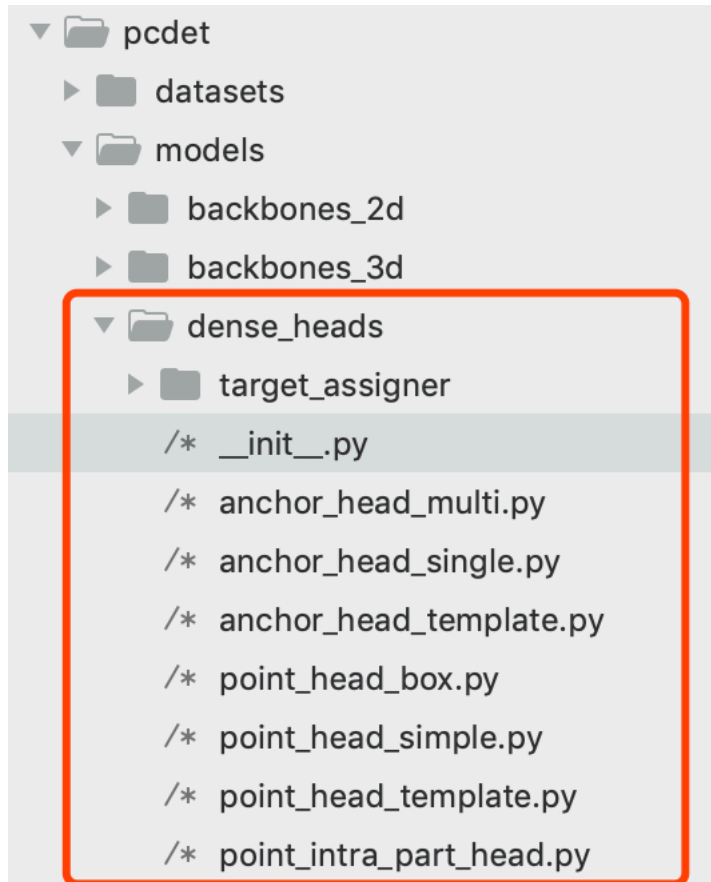
- Task: extract 2D feature maps
 - (with map_to_bev_module)
- Map_to_bev_module (map 3D features to 2D maps):
 - HeightCompression
 - PointPillarScatter
- 2D convolution encoder with FPN-like upsampling
 - BaseBEVBackbone



OpenPCDet: Denseheads



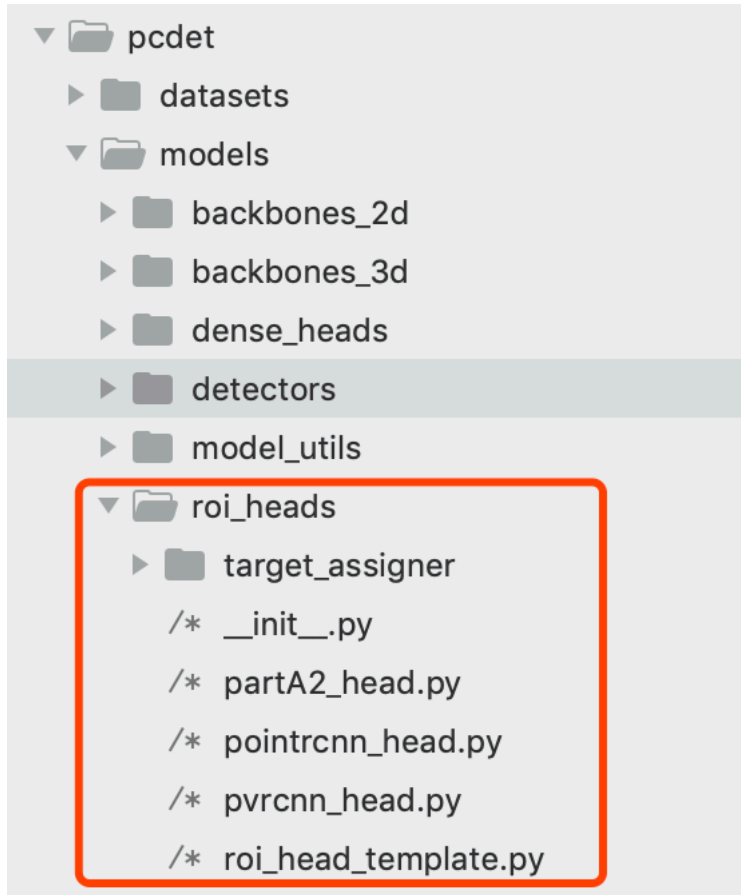
- Task: Generate dense 3D boxes
 - Target assigning
 - Head loss calculation
- Dense head with BEV features (AnchorHeadTemplate):
 - AnchorHeadSingle
 - AnchorHeadMulti
 - CenterHead (anchor-free)
- Dense head with point-wise features (PointHeadTemplate):
 - PointHeadSimple
 - PointHeadBox
 - PointIntraPartOffsetHead



OpenPCDet: RoIHeads



- Task: Refine 3D proposals with RoI-aligned features
 - Extract RoI-aligned features
 - proposal_layer
 - ProposalTargetLayer
 - Head loss calculation
- Stage-II RoI refinement (RoIHeadTemplate):
 - PointRCNNHead
 - PartA2FCHead
 - PVRCNNHead



OpenPCDet: Configuration files



- Hierarchical configuration with YAML file
- Each model takes its own config

```
1 CLASS_NAMES: ['Car', 'Pedestrian', 'Cyclist']
2
3 DATA_CONFIG:
4   _BASE_CONFIG_: cfgs/dataset_configs/kitti_dataset.yaml
5
6
7 MODEL:
8   NAME: SECONDNet
9
10  VFE:
11    NAME: MeanVFE
12
13  BACKBONE_3D:
14    NAME: VoxelBackBone8x
15
16  MAP_TO_BEV: ...
17
18
19  BACKBONE_2D: ...
20
21
22  DENSE_HEAD: ...
23
24
25  POST_PROCESSING: ...
26
27
28
29
30
31
32 OPTIMIZATION: ...
```

```
class DatasetTemplate(torch_data.Dataset):
    def __init__(self, dataset_cfg=None, class_names=None, training=True, root_path=None, logger=None):
        super().__init__()
```

```
class Detector3DTemplate(nn.Module):
    def __init__(self, model_cfg, num_class, dataset):
```

```
class AnchorHeadTemplate(nn.Module):
    def __init__(self, model_cfg, num_class, class_names, grid_size,
```

OpenPCDet example: Build one-stage detector



- SECOND / PointPillar

```
1 CLASS_NAMES: ['Car', 'Pedestrian', 'Cyclist']
2
3 DATA_CONFIG:
4   _BASE_CONFIG_: cfgs/dataset_configs/kitti_dataset.yaml
5
6
7 MODEL:
8   NAME: SECONDNet
9
10  VFE:
11    NAME: MeanVFE
12
13  BACKBONE_3D:
14    NAME: VoxelBackBone8x
15
16  MAP_TO_BEV: {}
17
18
19  BACKBONE_2D: {}
20
21
22  DENSE_HEAD: {}
23
24
25  POST_PROCESSING: {}
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

SECOND

```
1 CLASS_NAMES: ['Car', 'Pedestrian', 'Cyclist']
2
3 DATA_CONFIG: {}
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50 MODEL:
51   NAME: PointPillar
52
53   VFE: {}
54
55
56
57
58
59
60   MAP_TO_BEV: {}
61
62
63
64   BACKBONE_2D: {}
65
66
67
68
69
70
71
72   DENSE_HEAD: {}
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130   POST_PROCESSING: {}
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145   OPTIMIZATION: {}
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

PointPillar

OpenPCDet example: Build two-stage detector



- PointRCNN / PV-RCNN

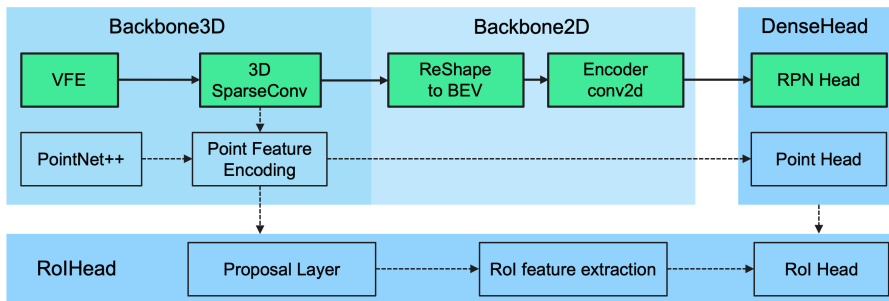
```
1 CLASS_NAMES: ['Car', 'Pedestrian', 'Cyclist']
2
3 DATA_CONFIG: {}
21
22 MODEL:
23   NAME: PointRCNN
24
25   BACKBONE_3D: {}
36
37   POINT_HEAD: {}
62
63   ROI_HEAD: {}
125
126   POST_PROCESSING: {}
139
140
141 OPTIMIZATION: {}
161
```

PointRCNN

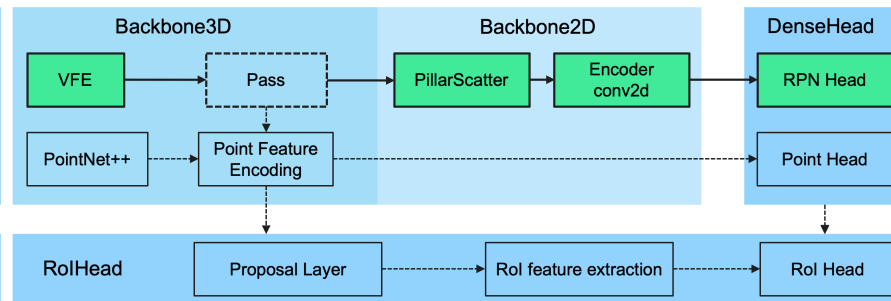
```
1 CLASS_NAMES: ['Car', 'Pedestrian', 'Cyclist']
2
3 DATA_CONFIG: {}
31
32 MODEL:
33   NAME: PVRCNN
34
35   VFE:
36     NAME: MeanVFE
37
38   BACKBONE_3D:
39     NAME: VoxelBackBone8x
40
41   MAP_TO_BEV: {}
44
45   BACKBONE_2D: {}
53
54   DENSE_HEAD: {}
111
112   PFE: {}
145
146   POINT_HEAD: {}
158
159   ROI_HEAD: {}
214
215   POST_PROCESSING: {}
228
229
230 OPTIMIZATION: {}
250
```

PV-RCNN

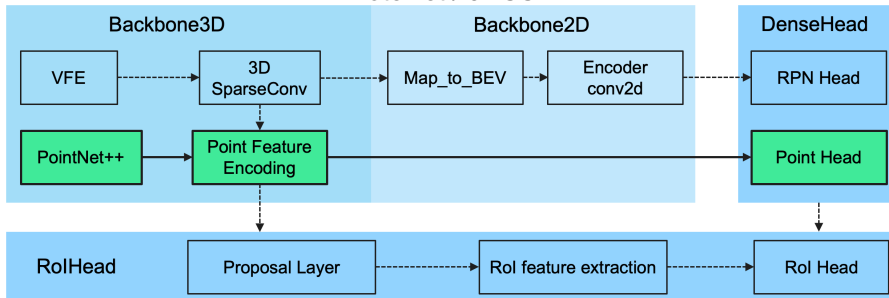
VoxelNet / SECOND / VoxelFPN



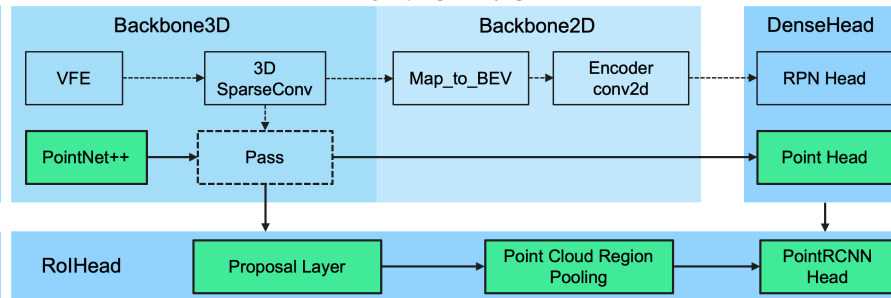
PointPillar / PIXOR / CenterPoint



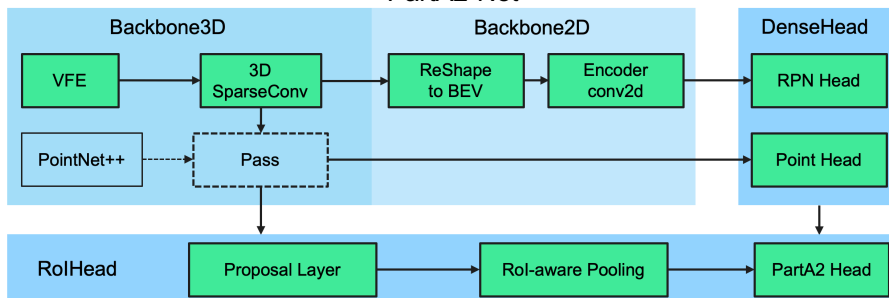
VoteNet / 3DSSD



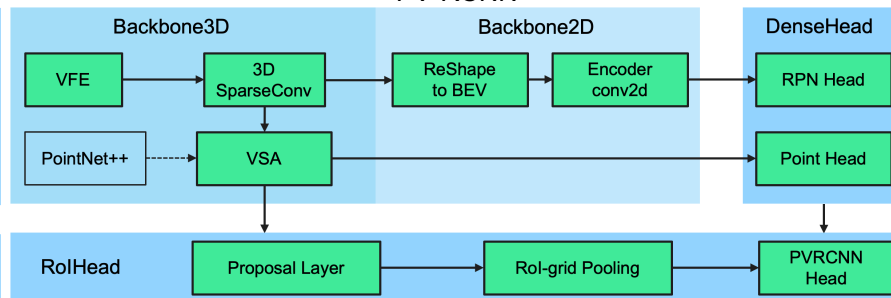
PointRCNN / STD



PartA2-Net



PV-RCNN





- How to add customized dataset ?
 1. Write your own DatasetModule that inherited from DatasetTemplate
 2. Overload the self.__getitem__() function to load point clouds / gt_boxes and transform them to the unified coordinate and box definition of OpenPCDet
 3. Call self.prepare_data() to process the data
 4. Overload the self.generate_prediction_dicts() function to transform the predicted results to the format what you like.
 5. Overload the self.evaluation() function to evaluate the results with your own metric.

OpenPCDet: 3D Detection Toolbox



- How to support more models ?
 1. Write your own detector that inherited from DetectorTemplate
 2. Write your own configuration files
 3. Write your own modules to specific directories if we do not provide it
 4. Overload the forward() function
 5. Overload the get_training_loss() function

OpenPCDet: Tips with OpenPCDet



- Group the configuration files into different directories
- Multi-gpu training and multi-gpu testing
- View the training process in the tensorboard
- Start a separate evaluation program for fast training
- The following command line parameters are useful:
 - `--extra_tag`
 - `--pretrained_model`
 - `--set`
 - `--eval_tag`
 - `--eval_all`

Training script:

```
bash scripts/dist_train.sh 8 --cfg_file cfgs/kitti_models/pv_rcnn.yaml --extra_tag ex1_tag --set OPTIMIZATION.LR 0.03
```

Testing script:

```
sh scripts/dist_test.sh 8 --cfg_file cfgs/kitti_models/pv_rcnn.yaml --extra_tag ex1_tag --eval_all --eval_tag nms02 --set MODEL.POST_PROCESSING.NMS_CONFIG.NMS_THRESH 0.2
```

Q & A

Welcome to Star / Fork / PR to
OpenPCDet (<https://github.com/open-mmlab/OpenPCDet>)

Shaoshuai Shi
Ph.D. Student of Multimedia Laboratory
The Chinese University of Hong Kong



智東西 公开课

专注讲解新兴技术创新与应用