
You Only Look Once: Unified, Real-Time Object Detection

000
001
002
003
004
005
006
007
008
009 **Anonymous Author(s)**
010 Affiliation
011 Address
012 email
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
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
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911<br

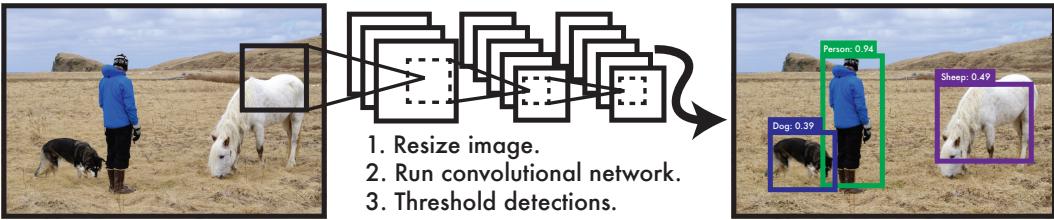


Figure 1: The YOLO Detection System. Our system (1) resizes the input image to 448×448 , (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model’s confidence. For all examples in this paper we use a confidence threshold of 0.2 unless otherwise noted.

Finally, these detection pipelines rely on independent techniques at every stage that cannot be optimized jointly. A typical pipeline uses Selective Search for region proposals, a convolutional network for feature extraction, a collection of one-versus-all SVMs for classification, non-maximal suppression to reduce duplicates, and a linear model to adjust the final bounding box coordinates. Selective Search tries to maximize recall while the SVMs optimize for single class accuracy and the linear model learns from localization error.

Our system is refreshingly simple. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. Our system enforces spatial diversity in the proposed bounding boxes without relying on non-maximal suppression to eliminate duplicates. We train our network on full images to directly optimize detection performance. At test time, a single network evaluation produces complete detections of multiple objects in multiple categories without any pre or post-processing.

2 Unified Detection

We unify the separate components of object detection into a single neural network. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are. Our network uses features from the entire image while predicting each bounding box. It also predicts every bounding box for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real-time speeds while maintaining high average precision.

2.1 Design

Our system divides the image into a 7×7 grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts a bounding box and class probabilities associated with that bounding box.

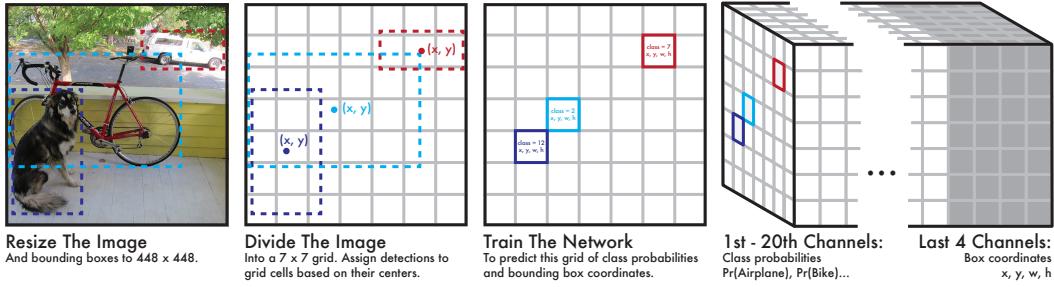


Figure 2: The Model. Our system models detection as a regression problem to a $7 \times 7 \times 24$ tensor. This tensor encodes bounding boxes and class probabilities for objects in the image.

We implement this model as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset [2]. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.

Our network architecture is inspired by the GoogLeNet model for image classification [9]. The network has 24 convolutional layers followed by 2 fully connected layers. However, instead of the inception modules used by GoogLeNet we simply use 1×1 reduction layers followed by 3×3 convolutional layers. We also replace maxpooling layers with strided convolutions.

The final output of our network is a 7×7 grid of predictions. Each grid cell predicts 20 conditional class probabilities, and 4 bounding box coordinates.

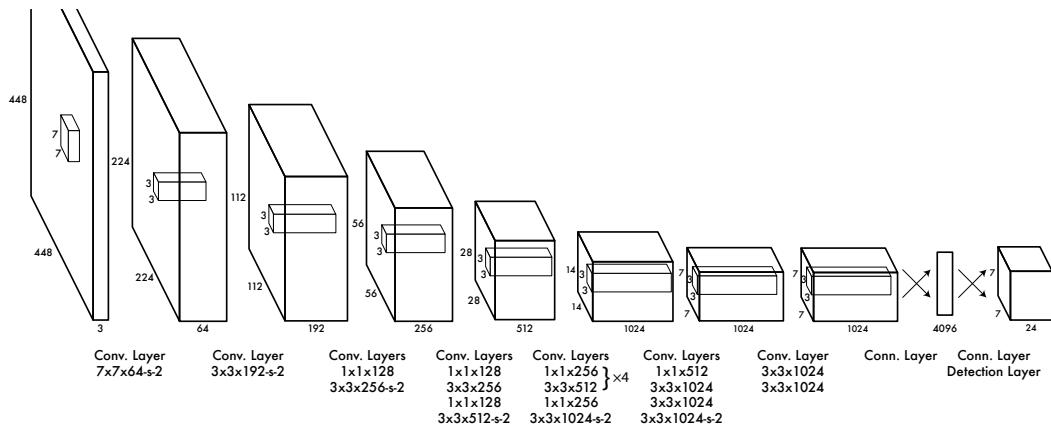


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. The network uses strided convolutional layers to downsample the feature space instead of maxpooling layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

2.2 Training

We pretrain our convolutional layers on the ImageNet 1000-class competition dataset [7]. For pre-training we use the first 20 convolutional layers from figure 3 followed by a maxpooling layer and two fully connected layers. We train this network for approximately a week and achieve a top-5 accuracy of 86% on the ImageNet 2012 validation set.

We then adapt the model to perform detection. We add four convolutional layers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual information so we increase the input resolution of the network from 224×224 to 448×448 .

Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parameterize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1. We use a logistic activation function to reflect these constraints on the final layer. All other layers use the following leaky rectified linear activation:

$$\phi(x) = \begin{cases} 1.1x, & \text{if } x > 0 \\ .1x, & \text{otherwise} \end{cases} \quad (1)$$

We optimize for sum-squared error in the output of our model. We use sum-squared error because it is easy to optimize however it does not perfectly align with our goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal. To remedy this, we use a scaling factor to adjust weight given to error from coordinate predictions versus error from class probabilities. In our final model we use a scaling factor of 4.

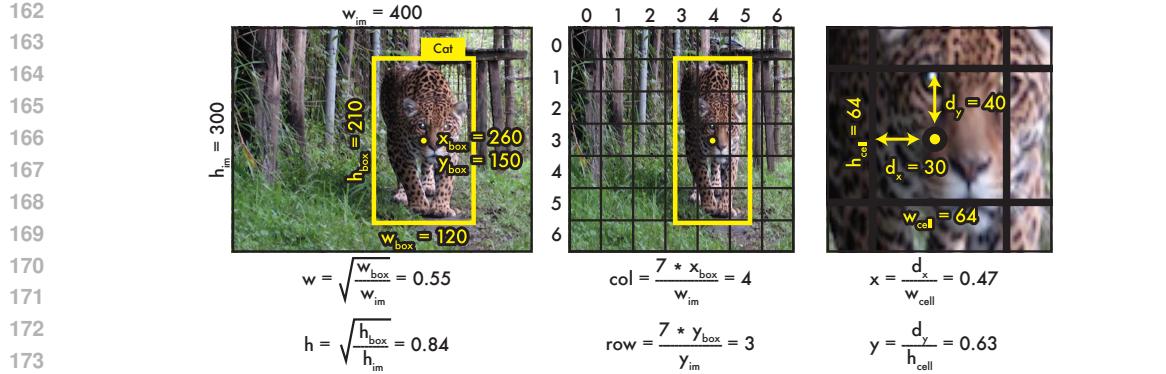


Figure 4: The Coordinate System. This image shows an example transformation from bounding box coordinates to the coordinates used by our output layer.

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.

If cell i predicts class probabilities $\hat{p}_i(\text{aeroplane}), \hat{p}_i(\text{bicycle})\dots$ and the bounding box $\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i$ then our full loss function for an example is:

$$\sum_{i=0}^{49} \left(\mathbb{1}_{\text{obj}}((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2) + \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \right) \quad (2)$$

Note that if there is no object in a cell we do not consider any loss from the bounding box coordinates predicted by that cell. In this case, there is no ground truth bounding box so we only penalize the associated probabilities with that region.

We train the network for about 120 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012 as well as the test set from 2007, a total of 21k images. Throughout training we use a batch size of 64, a momentum of 0.9 and a decay of 0.0005. We use two learning rates during training: 10^{-2} and 10^{-3} . Training diverges if we use the higher learning rate, 10^{-2} , from the start. We use the lower rate, 10^{-3} , for one epoch so that the randomly initialized weights in the final layers can settle to reasonable values. Then we train with the following learning rate schedule: 10^{-2} for 80 epochs, and 10^{-3} for 40 epochs.

To avoid overfitting we use dropout and extensive data augmentation. A dropout layer with a probability $\text{Pr} = .5$ after the first connected layer prevents co-adaptation between layers. For data augmentation we introduce random scaling and translations of up to 10% of the original image size. We also randomly adjust the exposure and saturation of the image by up to a factor of 2.

2.2.1 Nuisance Variables

Each grid cell predicts class probabilities for that area of the image. There are 49 cells with a possible 20 classes each yielding 980 predicted probabilities per image. Most of these probabilities will be zero since only a few objects appear in any given image. Left unchecked, this imbalance pushes all of the probabilities to zero, leading to divergence during training.

To overcome this, we add an extra variable to each grid location, the probability that any object exists in that location regardless of class. Thus instead of 20 class probabilities we have 1 "objectness" probability, $\text{Pr}(\text{Object})$, and 20 conditional probabilities: $\text{Pr}(\text{Airplane}|\text{Object})$, $\text{Pr}(\text{Bicycle}|\text{Object})$, etc.

216 To get the unconditional probability for an object class at a given location we simply multiply the
217 "objectness" probability by the conditional class probability:
218

$$\text{Pr}(\text{Cat}) = \text{Pr}(\text{Object}) * \text{Pr}(\text{Cat}|\text{Object}) \quad (3)$$

221 We can optimize these probabilities independently or jointly using a novel "detection layer" in our
222 convolutional network. During the initial stages of training we optimize them independently to
223 improve model stability. We update the "objectness" probabilities at every location however we
224 only update the conditional probabilities at locations that actually contain an object. This means
225 there are far fewer probabilities getting pushed towards zero.

226 During later stages of training we optimize the unconditioned probabilities by performing the re-
227 quired multiplications in the network and calculating error based on the result.
228

229 2.2.2 Predicting IOU

231 Like most detection systems, our network has trouble precisely localizing small objects. While it
232 may correctly predict that an object is present in an area of the image, if it does not predict a precise
233 enough bounding box the detection is counted as a false positive.

234 We want YOLO to have some notion of uncertainty in its probability predictions. Instead of predict-
235 ing 1-0 probabilities we can scale the target class probabilities by the IOU of the predicted bounding
236 box with the ground truth box for a region. When YOLO predicts good bounding boxes it is also
237 encouraged to predict high class probabilities. For poor bounding box predictions it learns to predict
238 lower confidence probabilities.

239 We do not train to predict IOU from the beginning, only during the second stage of training. It is not
240 necessary for good performance but it does boost our mean average precision by 3-4%.

242 2.3 Inference

244 Just like in training, predicting detections for a test image only requires one network evaluation.
245 The network predicts 49 bounding boxes per image and class probabilities for each box. YOLO is
246 extremely fast at test time since it only requires a single network evaluation unlike classifier-based
247 methods.

248 Non-Maximal Suppression

249 The grid design enforces spatial diversity in the bounding box predictions. Often it is clear which
250 grid cell an object falls in to and the network only predicts one box for each object. However, some
251 large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-
252 maximal suppression can be used to fix these multiple detections. While not critical to performance
253 as it is for R-CNN or DPM, non-maximal suppression adds 2-3% in mAP.

255 3 Comparison to Other Detection Systems

257 **Deformable parts models.** Deformable parts models (DPM) use a sliding window approach to
258 object detection [3]. DPM uses a disjoint pipeline to extract static features, classify regions, predict
259 bounding boxes for high scoring regions, etc. Our system replaces all of these disparate parts with
260 a single convolutional neural network. The network performs feature extraction, bounding box
261 prediction, non-maximal suppression, and contextual reasoning all concurrently. Instead of static
262 features, the network trains the features in-line and optimizes them for the detection task. Our
263 unified architecture leads to a faster, more accurate model than DPM.

264 **R-CNN.** Another class of detection algorithms uses region proposals instead of sliding windows
265 to find objects in images. These systems use region proposal methods like Selective Search [10]
266 to generate potential bounding boxes in an image. Instead of scanning through every region in
267 the window, now the classifier only has to score a small subset of potential regions in an image.
268 Good region proposal methods maintain high recall despite greatly limiting the search space. This
269 performance comes at a cost. Selective Search, even in "fast mode" takes about 2 seconds to propose
regions for an image.

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR.CNN.S.CNN	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
DEEP.LENS.COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	68.8	75.9	71.4
Fast R-CNN + YOLO	70.0	83.0	78.4	73.4	55.7	42.5	78.2	72.7	89.5	48.2	74.0	56.4	87.2	80.8	80.7	74.4	41.1	70.0	67.1	81.2	66.0
NoC	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
NUS.NIN.C2000	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
R-CNN VGG BB	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
NUS.NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
Feature Edit	56.3	74.0	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
YOLO	53.6	71.6	62.2	55.2	35.9	33.2	62.4	53.7	78.1	34.0	52.9	38.7	72.2	67.3	66.3	62.1	25.5	50.0	46.9	67.4	46.3
R-CNN BB	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

Table 1: PASCAL VOC 2012 Leaderboard. YOLO compared with the full comp4 (outside data allowed) public leaderboard as of June 5th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the top detection method that is not based on the R-CNN detection framework. Fast R-CNN + YOLO is one of the top methods overall, with a 1.6% boost over Fast R-CNN and the highest average precision in 6 out of 20 classes.

R-CNN shares many design aspects with DPM. After region proposal, R-CNN uses the same multi-stage pipeline of feature extraction (using CNNs instead of HOG), SVM scoring, non-maximal suppression, and bounding box prediction using a linear model [4].

YOLO shares some similarities with R-CNN. Each grid cell proposes a potential bounding box and then scores that bounding box using convolutional features. However, our system puts spatial constraints on the grid cell proposals which helps mitigate multiple detections of the same object. Our system also proposes far fewer bounding boxes, only 49 per image compared to about 2000. Finally, our system combines these individual components into a single, jointly optimized model.

Deep Multibox. Most R-CNN variants use Selective Search to generate region proposals. Szegedy et al. instead train a convolutional neural network to predict regions of interest [1]. MultiBox can also perform single object detection by replacing the confidence prediction with a single class prediction. However, MultiBox cannot perform general object detection and is still just a piece in a larger detection pipeline.

Both YOLO and MultiBox use a convolutional network to predict bounding boxes and confidence scores. However, YOLO additionally predicts class-specific probabilities making it able to perform general object detection. Instead of predicting bounding boxes based on clusters, YOLO uses the grid approach to enforce spatial diversity in predicted clusters.

Overfeat. Sermanet et al. train a convolutional neural network to perform localization and adapt that localizer to perform detection [8]. Overfeat efficiently performs sliding window detection but it is still a disjoint system. Overfeat optimizes for localization, not detection performance. Like DPM, the localizer only sees local information when making a prediction. Overfeat cannot reason about global context and thus requires significant post-processing to produce coherent detections.

4 Experiments

We present detection results for the PASCAL VOC 2012 dataset and compare our mean average precision (mAP) and runtime to other top detection methods. We also perform error analysis on the VOC 2007 dataset. We compare our results to Fast R-CNN, one of the highest performing versions of R-CNN [5]. We use publicly available runs of Fast R-CNN available on GitHub. Finally we show that a combination of our method with Fast R-CNN gives a significant performance boost.

4.1 VOC 2012 Results

On the VOC 2012 test set we achieve 53.6 mAP. This is lower than the current state of the art, closer to R-CNN based methods that use AlexNet. Our system struggles with small objects compared to its closest competitors. On categories like bottle, sheep, and tv/monitor YOLO scores 8-10 percentage points lower than R-CNN or Feature Edit. However, on other categories like cat

	mAP	Prediction Time	Images/Sec	Run Time
R-CNN (VGG-16)	66.9	48.2 hr	0.02 fps	1500x
Fast R-CNN (VGG-16)	59.2	3.1 hr	0.45 fps	100x
R-CNN (Small VGG)	60.2	14.4 hr	0.09 fps	500x
Fast R-CNN (Small VGG)	59.2	2.9 hr	0.48 fps	93x
R-CNN (CaffeNet)	58.5	12.2 hr	0.11 fps	409x
Fast R-CNN (CaffeNet)	57.1	2.8 hr	0.48 fps	93x
YOLO	58.8	110 sec	45 fps	-

Table 2: Prediction Timing. mAP and timing information for R-CNN, Fast R-CNN, and YOLO on the VOC 2007 test set. Timing information is given both as frames per second and the time each method takes to process the full 4952 image set. The final column shows the relative speed of YOLO compared to that method.

and horse YOLO achieves significantly higher performance. We further investigate the source of these performance disparities in section 4.3.

4.2 Speed

At test time YOLO processes images at 45 frames per second on an Nvidia Titan X. It is considerably faster than classifier-based methods with similar mAP. Normal R-CNN using AlexNet or the small VGG network take 400-500x longer to process images. The recently proposed Fast R-CNN shares convolutional features between the bounding boxes but still relies on selective search for bounding box proposals which accounts for the bulk of their processing time. YOLO is still around 100x faster than Fast R-CNN.

4.3 VOC 2007 Error Analysis

Using the methodology and tools of Hoiem et al. [6] we analyze our performance on the VOC 2007 test set. We compare YOLO to Fast R-CNN using VGG-16, one of the highest performing object detectors.

Figure 5 compares frequency of localization and background errors between Fast R-CNN and YOLO. A detection is considered a localization error if it overlaps a true positive in the image but by less than the required 50% IOU. A detection is a background error if the box does not overlap any objects of any class in the image.

Fast R-CNN makes around the same number of localization errors and background errors. Over all 20 classes in the top N detections 13.6% are localization errors and 8.6% are background errors.

YOLO makes far more localization errors but relatively few background errors. Averaged across all classes, of its top N detections 24.7% are localization errors and a mere 4.3% are background errors. This is about twice the number of localization errors but half the number of background detections.

YOLO uses global context to evaluate detections while R-CNN only sees small portions of the image. Many of the background detections made by R-CNN are obviously not objects when shown in context.

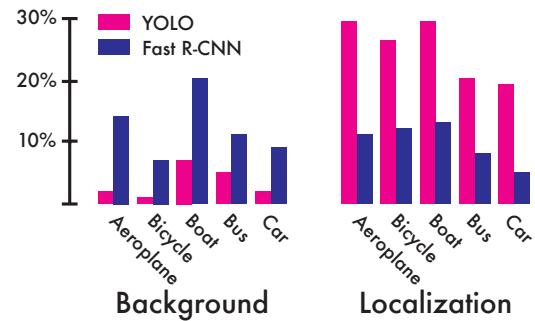


Figure 5: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

	mAP	Combined mAP	% Pt Increase
Fast R-CNN, No Augmentation	-	71.8	-
Fast R-CNN (2007 data)	66.9	72.4	.6
Fast R-CNN (Small VGG)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	59.1	74.7	2.9

Table 3: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. The model’s base mAP is listed as well as its mAP when combined with the top model on VOC 2007. Other versions of Fast R-CNN provides only a small marginal benefit while combining with YOLO results in a significant performance boost.

4.4 Combining Fast R-CNN and YOLO

YOLO makes far fewer background mistakes than Fast R-CNN. By using YOLO to eliminate background detections from Fast R-CNN we get a significant boost in performance. For every bounding box that R-CNN predicts we check to see if YOLO predicts a similar box. If it does, we give that prediction a boost based on the probability predicted by YOLO and the overlap between the two boxes. This reorders the detections to favor those predicted by both systems. Since we still use Fast R-CNN’s bounding box predictions we do not introduce any localization error. Thus we take advantage of the best aspects of both systems.

The best Fast R-CNN model achieves a mAP of 71.8% on the VOC 2007 test set. When combined with YOLO, its mAP increases by 2.9% to 74.7%. We also tried combining the top Fast R-CNN model with several other versions of Fast R-CNN. Those ensembles produced small increases in mAP between .3 and .6%, see table 3 for details. Thus, the benefit from combining Fast R-CNN with YOLO is unique, not a general property of combining models in this way.

Using this combination strategy we achieve a significant boost on the VOC 2012 and 2010 test sets as well, around 2%. The combined Fast R-CNN + YOLO model is the second highest performing model on the VOC 2012 leaderboard.

5 Conclusion

We introduce YOLO, a unified pipeline for object detection. Our model is simple to construct and can be trained directly on full images. Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and every piece of the pipeline can be trained jointly.

The network reasons about the entire image during inference which makes it less likely to predict background false positives than sliding window or proposal region detectors. Moreover, it predicts detections with only a single network evaluation, making it extremely fast.

YOLO achieves impressive performance on standard benchmarks considering it is 2-3 orders of magnitude faster than existing detection methods. It can also be used to rescore the bounding boxes produced by state-of-the-art methods like R-CNN for a significant boost in performance.

References

- [1] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2155–2162. IEEE, 2014.
- [2] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015.
- [3] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

- 432 [4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection
433 and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference*
434 *on*, pages 580–587. IEEE, 2014.
- 435 [5] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.
- 436 [6] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *Computer Vision-*
437 *ECCV 2012*, pages 340–353. Springer, 2012.
- 438 [7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla,
439 M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Internation-*
440 *al Journal of Computer Vision (IJCV)*, 2015.
- 441 [8] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition,
442 localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.
- 443 [9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabi-
444 novich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
- 445 [10] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recogni-
446 *tion*. *International journal of computer vision*, 104(2):154–171, 2013.

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