Project 4

September 26, 2021

Copyright 2020 Google LLC.

```
[ ]: # Licensed under the Apache License, Version 2.0 (the "License");
     # you may not use this file except in compliance with the License.
     # You may obtain a copy of the License at
     #
     # https://www.apache.org/licenses/LICENSE-2.0
     #
     # Unless required by applicable law or agreed to in writing, software
     # distributed under the License is distributed on an "AS IS" BASIS,
     # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
     # See the License for the specific language governing permissions and
     # limitations under the License.
```

1 Video Classification with Pre-Trained Models Project

In this project we will import a pre-existing model that recognizes objects and use the model to identify those objects in a video. We'll edit the video to draw boxes around the identified object, and then we'll reassemble the video so the boxes are shown around objects in the video.

##Team Members

1. Jose Martinez
2. Wren Priest
3. Maria Quintero

2 Exercises

2.1 Exercise 1: Coding

You will process a video frame by frame, identify objects in each frame, and draw a bounding box with a label around each car in the video.

Use the SSD MobileNet V1 Coco (*ssd_mobilenet_v1_coco*) model. The video you'll process can be found on Pixabay. The 640x360 version of the video is smallest and easiest to handle, though any size should work since you must scale down the images for processing.

Your program should:

- Read in a video file (use the one in this colab if you want)
- Load the TensorFlow model linked above

- Loop over each frame of the video
- Scale the frame down to a size the model expects
- Feed the frame to the model
- Loop over detections made by the model
- If the detection score is above some threshold, draw a bounding box onto the frame and put a label in or near the box
- Write the frame back to a new video

Some tips:

- Processing an entire video is slow, so consider truncating the video or skipping over frames during development. Skipping frames will make the video choppy. But you'll be able to see a wider variety of images than you would with a truncated video with all of the original frames in the clip.
- The model expects a 300x300 image. You'll likely have to scale your frames to fit the model. When you get a bounding box, that box is relative to the scaled image. You'll need to scale the bounding box out to the original image size.
- Don't start by trying to process the video. Instead, capture one frame and work with it until you are happy with your object detection, bounding boxes, and labels. Once you get those done, use the same logic on the other frames of the video.
- The Coco labels file can be used to identify classified objects.

2.1.1 Student Solution

Initial Importing and Setup

[ ]:
```
import urllib.request
import os

base_url = 'http://download.tensorflow.org/models/object_detection/'
file_name = 'ssd_mobilenet_v1_coco_2018_01_28.tar.gz'

url = base_url + file_name

urllib.request.urlretrieve(url, file_name)

import tarfile
import shutil

dir_name = file_name[0:-len('.tar.gz')]

if os.path.exists(dir_name):
  shutil.rmtree(dir_name)

tarfile.open(file_name, 'r:gz').extractall('./')

os.listdir(dir_name)

import tensorflow as tf
```

```
frozen_graph = os.path.join(dir_name, 'frozen_inference_graph.pb')

with tf.io.gfile.GFile(frozen_graph, "rb") as f:
  graph_def = tf.compat.v1.GraphDef()
  loaded = graph_def.ParseFromString(f.read())

outputs = (
  'num_detections:0',
  'detection_classes:0',
  'detection_scores:0',
  'detection_boxes:0',
)

def wrap_graph(graph_def, inputs, outputs, print_graph=False):
  wrapped = tf.compat.v1.wrap_function(
    lambda: tf.compat.v1.import_graph_def(graph_def, name=""), [])

  return wrapped.prune(
    tf.nest.map_structure(wrapped.graph.as_graph_element, inputs),
    tf.nest.map_structure(wrapped.graph.as_graph_element, outputs))

model = wrap_graph(graph_def=graph_def,
                   inputs=["image_tensor:0"],
                   outputs=outputs)
```

[ ]:
```
# Object detection dictionary
labels = {
    0:"background",
    1:"person",
    2:"bicycle",
    3:"car",
    4:"motorcycle",
    5:"airplane",
    6:"bus",
    7:"train",
    8:"truck",
    9:"boat",
    10:"trafficlight",
    11:"firehydrant",
    12:"unknown",
    13:"stopsign",
    14:"parkingmeter",
    15:"bench",
    16:"bird",
    17:"cat",
    18:"dog",
    19:"horse",
```

```
20:"sheep",
21:"cow",
22:"elephant",
23:"bear",
24:"zebra",
25:"giraffe",
26:"unknown",
27:"backpack",
28:"umbrella",
29:"unknown",
30:"unknown",
31:"handbag",
32:"tie",
33:"suitcase",
34:"frisbee",
35:"skis",
36:"snowboard",
37:"sportsball",
38:"kite",
39:"baseballbat",
40:"baseballglove",
41:"skateboard",
42:"surfboard",
43:"tennisracket",
44:"bottle",
45:"unknown",
46:"wineglass",
47:"cup",
48:"fork",
49:"knife",
50:"spoon",
51:"bowl",
52:"banana",
53:"apple",
54:"sandwich",
55:"orange",
56:"broccoli",
57:"carrot",
58:"hotdog",
59:"pizza",
60:"donut",
61:"cake",
62:"chair",
63:"couch",
64:"pottedplant",
65:"bed",
66:"unknown",
```

```
    67:"diningtable",
    68:"unknown",
    69:"unknown",
    70:"toilet",
    71:"unknown",
    72:"tv",
    73:"laptop",
    74:"mouse",
    75:"remote",
    76:"keyboard",
    77:"cellphone",
    78:"microwave",
    79:"oven",
    80:"toaster",
    81:"sink",
    82:"refrigerator",
    83:"unknown",
    84:"book",
    85:"clock",
    86:"vase",
    87:"scissors",
    88:"teddybear",
    89:"hairdrier",
    90:"toothbrush"
}
```

Data Preprocessing

[ ]:
```
import cv2

# import video
cars_video = cv2.VideoCapture("cars.mp4")
# get video properties and store as variables
height = int(cars_video.get(cv2.CAP_PROP_FRAME_HEIGHT))
width = int(cars_video.get(cv2.CAP_PROP_FRAME_WIDTH))
fps = cars_video.get(cv2.CAP_PROP_FPS)
total_frames = int(cars_video.get(cv2.CAP_PROP_FRAME_COUNT))

# print video properties
print(f'height: {height}')
print(f'width: {width}')
print(f'frames per second: {fps}')
print(f'total frames: {total_frames}')
print(f'video length (seconds): {total_frames / fps}')

# determine how much padding is needed to get to 300 by 300
left_pad, right_pad, top_pad, bottom_pad = 0, 0, 0, 0
if height > width:
```

```
  left_pad = int((height-width) / 2)
  right_pad = height-width-left_pad
elif width > height:
  top_pad = int((width-height) / 2)
  bottom_pad = width-height-top_pad

# pads cars.mp4 and outputs padded.mp4
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
input_video = cv2.VideoCapture("cars.mp4")
output_video = cv2.VideoWriter('padded.mp4', fourcc, fps, (width + left_pad +␣
↪right_pad, height + top_pad + bottom_pad))
for i in range(0, int(total_frames)):
  # progress ticker, will count from 0 to 1501
  print(i, end=" ")
  input_video.set(cv2.CAP_PROP_POS_FRAMES, i)
  ret, frame = input_video.read()
  frame_square = cv2.copyMakeBorder(
        frame,
        top_pad,
        bottom_pad,
        left_pad,
        right_pad,
        cv2.BORDER_CONSTANT,
        value=(255,255,255))
  if not ret:
    raise Exception("Problem reading frame", i, " from video")
  output_video.write(frame_square)

input_video.release()
output_video.release()

# scales padded.mp4 and outputs scaled.mp4
cars_video = cv2.VideoCapture("padded.mp4")
output = cv2.VideoWriter('scaled.mp4', fourcc, 25.0, (300, 300))
while True:
  ret, frame = cars_video.read()
  if ret == True:
    b = cv2.resize(frame,(300,300),fx=0,fy=0)
    output.write(b)
  else:
    break

cars_video.release()
output.release()
```

```
height: 360
width: 640
frames per second: 25.0
```

```
total frames: 1501
video length (seconds): 60.04
0 1 2 3 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 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 1001 1002 1003 1004 1005
1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021
1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037
1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053
1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069
1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085
1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101
1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117
1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133
1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149
1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165
1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181
1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197
1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213
1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229
1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245
1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261
1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277
1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293
1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325
1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341
1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357
1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373
1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389
1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405
1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421
1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437
1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453
1454 1455 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469
1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485
1486 1487 1488 1489 1490 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500
```

Model Input and Output

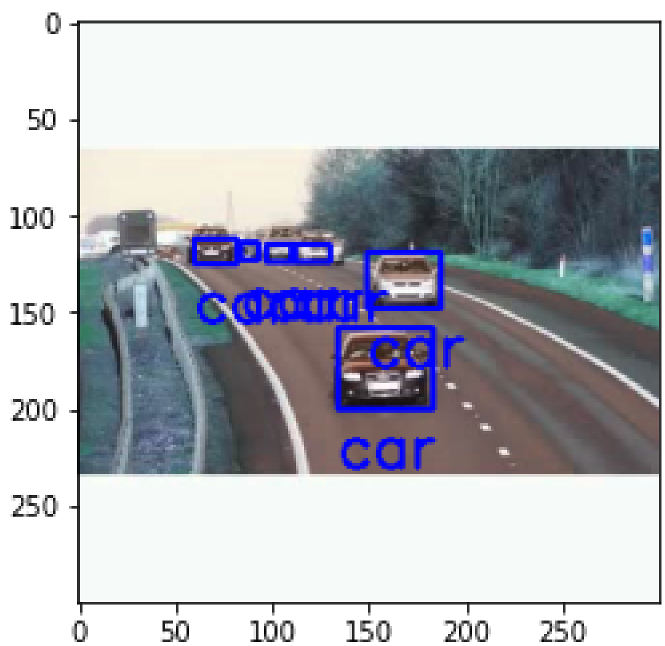
[ ]:
```python
import matplotlib.pyplot as plt

# set video output parameters
output_video = cv2.VideoWriter('classed.mp4', fourcc, 25.0, (300, 300))

# loop over frames
for i in range(0, total_frames):
  # progress ticker, will count from 0 to 1501
  print(i, end=" ")
  # reading in video frame
  input_video = cv2.VideoCapture('scaled.mp4')
```

```
  input_video.set(cv2.CAP_PROP_POS_FRAMES, i)
  ret, frame = input_video.read()
  input_video.release()
  if not ret:
    raise Exception(f"Problem reading frame {i} from video")
  # input_image will be edited while frame is used as a static variable
  input_image = frame
  # convert to tensor for model
  tensor = tf.convert_to_tensor([frame], dtype=tf.uint8)
  # run tensor frame through model
  detections = model(tensor)

  # extract information from model detections
  num_detect = int(detections[0].numpy()[0])
  classes = detections[1].numpy()[0, :num_detect]
  scores = detections[2].numpy()[0, :num_detect]
  boxes = detections[3].numpy()[0, :num_detect]

  # bounding boxes and text labels
  H, W, _ = input_image.shape
  for x in range(num_detect):
      box = boxes[x]
      y1, x1, y2, x2 = box
      x1 *= W
      x2 *= W
      y1 *= H
      y2 *= H
      # uses dictionary to map detection number to class name
      label = labels[classes[x]]
      # draw boxes and labels
      cv2.rectangle(input_image, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 
 ↪255), 2)
      cv2.putText(input_image, label, (int(x1), int(y2+30)),
                  cv2.FONT_HERSHEY_SIMPLEX, 1, [0, 0, 255], 2)

  # test to be sure model was seeing and classifying frames
  plt.imshow(input_image)

  # write each annotated image into 'classed.mp4'
  output_video.write(input_image)

output_video.release()
```

0 1 2 3 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 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 1001 1002 1003 1004 1005
1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021
1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037
1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053
1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069

```
1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085
1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101
1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117
1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133
1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149
1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165
1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181
1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197
1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213
1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229
1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245
1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261
1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277
1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293
1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325
1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341
1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357
1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373
1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389
1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405
1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421
1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437
1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453
1454 1455 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469
1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485
1486 1487 1488 1489 1490 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500
```



2.2 Exercise 2: Ethical Implications

Even the most basic models have the potential to affect segments of the population in different ways. It is important to consider how your model might positively and negatively affect different types of users.

In this section of the project, you will reflect on the positive and negative implications of your model. Frame the context of your model creation using this narrative:

> The city of Seattle is attempting to reduce traffic congestion in its downtown area. As part of this project, they plan to allow each local driver one free trip to downtown Seattle per week. After that, the driver will have to pay a $50 toll for each extra day per week driven. As an early proof of concept for this project, your team is tasked with using machine learning to correctly identify automobiles on the road. The next phase of the project will involve detecting license plate numbers and then cross-referencing that data with RFID chips that should be mounted in all local drivers' cars.

2.2.1 Student Solution

Positive Impact

Your model is trying to solve a problem. Think about who will benefit from that problem being solved and write a brief narrative about how the model will help.

> The model will benefit the city because there will be less traffic, less car accidents, and less pollution. Citizens of the city will avoid driving downtown with a toll. The less traffic flow will also be better for emergency vehicles such as ambulances and fire trucks. Lastly, less traffic will also be better for the environment.

Negative Impact

Models rarely benefit everyone equally. Think about who might be negatively impacted by the predictions your model is making. This person(s) might not be directly using the model, but they might be impacted indirectly.

> Citizens that work in the downtown area will be negatively impacted because the toll will be a financial obstacle for them to get to work. For example Taxis would be especially negatively impacted since they made need to travel in and out of the city constantly.

Bias

Models can be biased for many reasons. The bias can come from the data used to build the model (e.g., sampling, data collection methods, available sources) and/or from the interpretation of the predictions generated by the model.

Think of at least two ways bias might have been introduced to your model and explain both below.

> One source of bias in the model could be data collection bias since not all cars look the same. Some cars such as trucks, trailers, buses, or semi trucks transporting multiple

> cars can be mistaken in the system. In the case of a car transporting multiple cars, the data would be skewed.
>
> Another source of bias could be reporting bias. An example would be vehicles such as buses considered cars. Since buses are public transportation and encourage people to leave their cars at home, the model should not classify busses as a car.

Changing the Dataset to Mitigate Bias

Having bias in your dataset is one of the primary ways in which bias is introduced to a machine learning model. Look back at the input data you fed to your model. Think about how you might change something about the data to reduce bias in your model.

What change or changes could you make to reduce the bias in your dataset? Consider the data you have, how and where it was collected, and what other sources of data might be used to reduce bias.

Write a summary of changes that could be made to your input data.

> To reduce the bias in the dataset, the source of data for the model could be collected from multiple car dealership to reduce the amount of mistake between a car and vehicles such as trucks or trailers. In summary the input data would collect a larger variety of vehicles.

Changing the Model to Mitigate Bias

Is there any way to reduce bias by changing the model itself? This could include modifying algorithmic choices, tweaking hyperparameters, etc.

Write a brief summary of changes you could make to help reduce bias in your model.

> To reduce bias in the model itself, the model could be trained to classify exceptions in the system such as buses or vehicles transporting multiple cars.

Mitigating Bias Downstream

Models make predictions. Downstream processes make decisions. What processes and/or rules should be in place for people and systems interpreting and acting on the results of your model to reduce bias? Describe these rules and/or processes below.

> Since the predictions have potential bias reporting, we can adjust the process of the system to recognize exception adn classify them according to most similar features to another vehicle.