3. Object Detection

The main problem solved in this section is how to use the dnn module in OpenCV to import a trained object detection network. However, there are requirements for the version of opencv.

At present, there are three main methods for object detection using deep learning:

- Faster R-CNNs
- You Only Look Once (YOLO)
- Single Shot Detectors (SSDs)

Faster R-CNNs is the most commonly heard neural network based on deep learning. However, this method is technically difficult to understand (especially for deep learning novices), difficult to implement, and difficult to train.

In addition, even if the "Faster" method is used to implement R-CNNs (here R stands for Region Proposal), the algorithm is still relatively slow, about 7FPS.

If we pursue speed, we can turn to YOLO, because it is very fast, reaching 40-90 FPS on TianXGPU, and the fastest version may reach 155 FPS. But the problem with YOLO is that its accuracy needs to be improved.

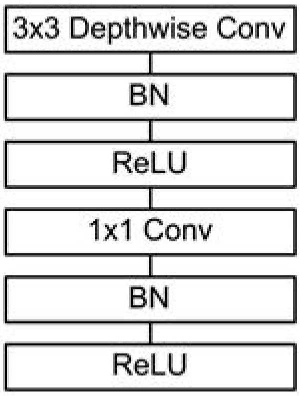
SSDs was originally developed by Google and can be said to be a balance between the above two. Compared with Faster R-CNNs, its algorithm is more direct. Compared with YOLO, it is more accurate.

3.1. Model structure

The main work of MobileNet is to replace the previous standard convolutions with depthwise sparable convolutions to solve the problems of computational efficiency and parameter quantity of convolutional networks. The MobileNets model is based on depthwise sparable convolutions, which can decompose standard convolutions into a depth convolution and a point convolution (1 × 1 convolution kernel). **Deep convolution applies each convolution kernel to each channel, while 1 × 1 convolution is used to combine the output of channel convolution.**

Batch Normalization (BN) is added to the basic components of MobileNet, that is, at each SGD (stochastic gradient descent), the standardization process is performed so that the mean of the result (each dimension of the output signal) is 0 and the variance is 1. Generally, when you encounter slow convergence or gradient explosion during neural network training, you can try to solve the problem. In addition, in general use, you can also add BN to speed up training and improve model accuracy.

In addition, the model also uses the ReLU activation function, so the basic structure of the depthwise separable convolution is shown in the figure below:



The MobileNets network is composed of many depthwise separable convolutions shown in the figure above. The specific network structure is shown in the figure below:

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s1	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

3.2、Code analysis

List of recognizable objects

```
[person, bicycle, car, motorcycle, airplane, bus, train,
truck, boat, traffic light, fire hydrant, street sign,
stop sign, parking meter, bench, bird, cat, dog, horse,
sheep, cow, elephant, bear, zebra, giraffe, hat, backpack,
umbrella, shoe, eye glasses, handbag, tie, suitcase,
frisbee, skis, snowboard, sports ball, kite, baseball bat,
```

```
baseball glove, skateboard, surfboard, tennis racket,
bottle, plate, wine glass, cup, fork, knife, spoon, bowl,
banana, apple, sandwich, orange, broccoli, carrot, hot dog,
pizza, donut, cake, chair, couch, potted plant, bed, mirror,
dining table, window, desk, toilet, door, tv, laptop, mouse,
remote, keyboard, cell phone, microwave, oven, toaster,
sink, refrigerator, blender, book, clock, vase, scissors,
teddy bear, hair drier, toothbrush]
```

Load category [object_detection_coco.txt], import model [frozen_inference_graph.pb], specify deep learning framework [TensorFlow]

```
# Load COCO class name
with
open('/home/jetson/jetcobot_ws/src/jetcobot_visual/config/object_detection_coco.t
xt', 'r') as f:
self.class_names = f.read().split('\n')
# Display different colors for different targets
self.COLORS = np.random.uniform(0, 255, size=(len(self.class_names), 3))
# Load DNN image model
self.model =
cv.dnn.readNet(model='/home/jetson/jetcobot_ws/src/jetcobot_visual/config/frozen_
inference_graph.pb',
config='home/jetson/jetcobot_ws/src/jetcobot_visual/config/ssd_mobilenet_v2_coco.
txt', framework='TensorFlow')
```

Import the image, extract the height and width, calculate the 300x300 pixel blob, and pass this blob into the neural network

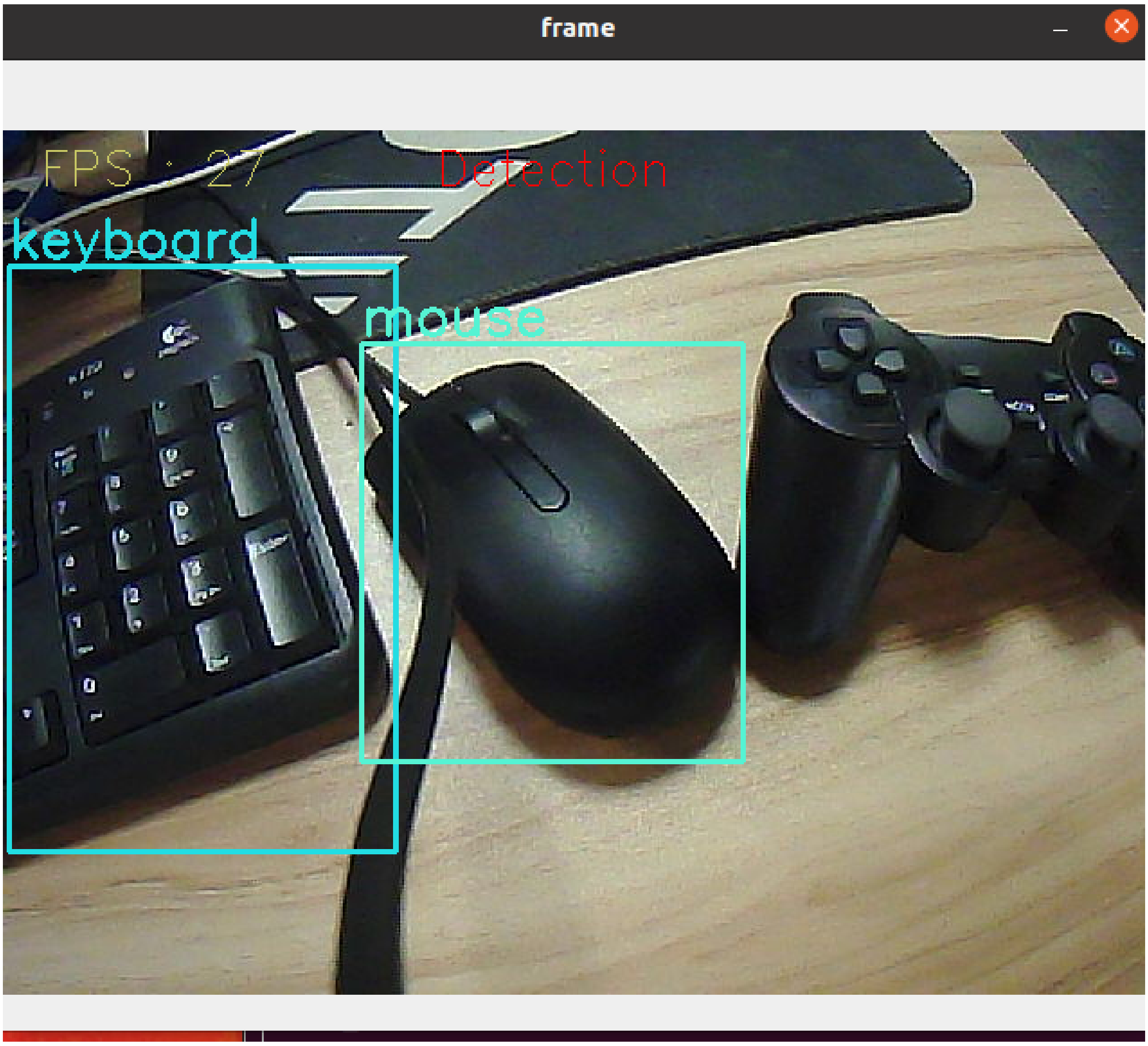
```
def Target_Detection(image):
image_height, image_width, _ = image.shape
# Create blob from image
blob = cv.dnn.blobFromImage(image=image, size=(300, 300), mean=(104, 117, 123),
swapRB=True)
model.setInput(blob)
output = model.forward()
# Loop through each detection
for detection in output[0, 0, :, :]:
# Extract the confidence of the detection
confidence = detection[2]
# Draw bounding box only if the detection confidence is above a certain
threshold, otherwise skip
if confidence > .4:
# Get the class ID
class_id = detection[1]
# Map class ids to classes
class_name = class_names[int(class_id) - 1]
color = COLORS[int(class_id)]
# Get bounding box coordinates
box_x = detection[3] * image_width
box_y = detection[4] * image_height
# Get the width and height of the bounding box
box_width = detection[5] * image_width
box_height = detection[6] * image_height
```

```
# Draw a rectangle around each detected object
cv.rectangle(image, (int(box_x), int(box_y)), (int(box_width), int(box_height)),
color, thickness=2)
# Write the class name text on the detected object
cv.putText(image, class_name, (int(box_x), int(box_y - 5)),
cv.FONT_HERSHEY_SIMPLEX, 1, color, 2)
return image
```

3.3 Startup

```
ros2 run jetcobot_visual detect_object
```

Camera display:


frame
FPS : 27
Detection
keyboard
mouse