

Semana 3: Integración y despliegue de redes neuronales en el RoboMaster

1 Introducción

El objetivo durante esta última semana será la integración en hardware de los modelos de detección de RoboMasters entrenados en la semana anterior. Habitualmente, la integración conlleva todo un proceso de selección y montaje de hardware, adaptación de las redes neuronales a los requisitos del aparato, integración y puesta a punto del modelo para aplicar inferencia, evaluación del funcionamiento en entornos controlados, optimización en tiempo real, pruebas en el mundo real, mantenimiento y actualización. Para simplificar, en esta asignatura nos dedicaremos únicamente a la integración y evaluación en entornos controlados.

Para llevar a cabo esta tarea, haremos uso de la SDK abierta de Python disponible en los RoboMasters. Esta SDK nos permite establecer una conexión directa a través de una red Wi-Fi generada por el propio RoboMaster o a través de un router externo, lo que facilita la comunicación y el control desde nuestro ordenador. Dado que los ordenadores del laboratorio no tienen antena Wi-Fi, optaremos por conectarnos a través del router del laboratorio. Una vez conectados, podremos diseñar un circuito de ejemplo en el que nuestro RoboMaster deberá llevar a cabo la búsqueda y detección de otros RoboMasters para dispararles y dejarlos fuera de combate. Esto implica tener que indicar al RoboMaster cómo debe moverse, cómo debe rotar sobre sí mismo, cómo debe girar el cañón y hacia dónde debe apuntarlo, cuándo debe realizar la detección, y cómo debe disparar a los RoboMasters detectados. Además, los estudiantes podrán incluir funcionalidades adicionales como reproducir audios, jugar con las luces LEDs de los RoboMasters o diseñar rutas más complejas que mejoren la demo final a presentar.

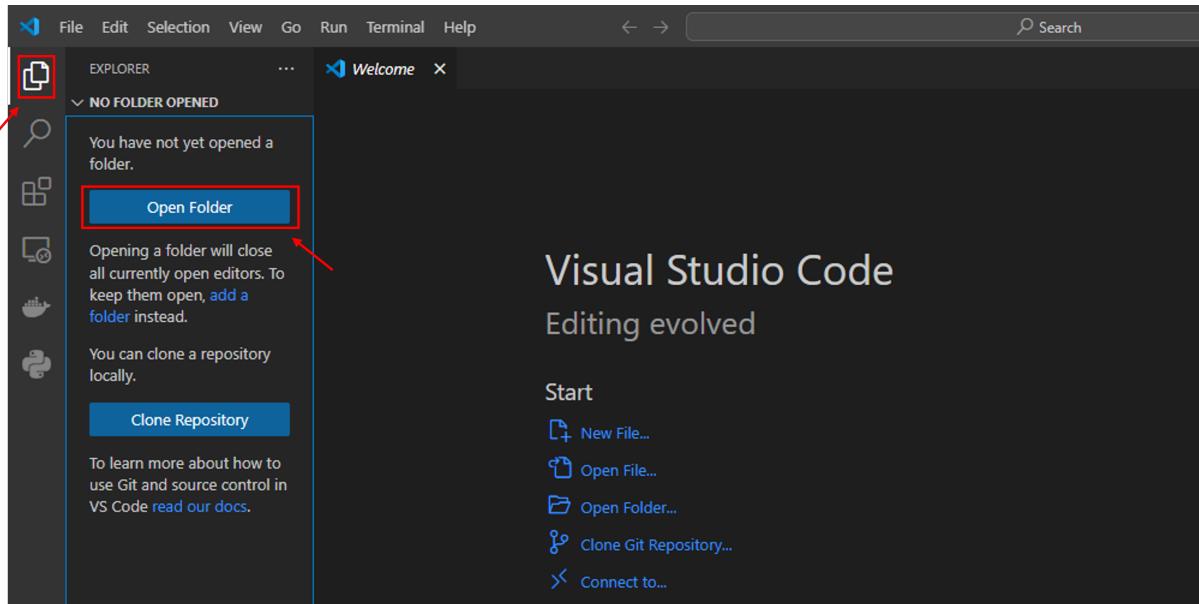
2 VSCode

Dado que no es posible conectarse a los RoboMasters a través de JupyterHub, vamos a utilizar VSCode. Abra la aplicación y, tal y como se muestra en la Figura 1a, seleccione el proyecto ubicado en la carpeta `/home/PIDS/RoboMaster`. Debería ver un fichero `connect.py` para conectarnos al robot, un fichero `demo.py` sobre el que programaremos la demo, y varios ficheros de ejemplo para aprender a usar el robot. Asegúrese (Figura 1b, esquina inferior derecha) de que el entorno virtual que va a utilizar VSCode es el que se ha preparado para esta práctica (carpeta `venv` con Python3.8). Si no lo es, haga click en el intérprete que aparezca para seleccionar el correcto o use Ctrl+Shift+P y escriba “Select Interpreter”. El intérprete está en la ruta `venv/bin/python`.

Para ejecutar un script, ábralo y pulse Ctrl+F5 y seleccione “Python Debugger” o pulse en el desplegable junto al botón triangular (“Play”) de la esquina superior derecha (ver figura 1b) y elija la opción “Run Python File” (esto ejecuta el script la primera vez, el resto de veces puede pulsar en el botón triangular “Play”).

3 Uso del RoboMaster

Los RoboMasters tienen 2 métodos de conexión: “networking” y “Wi-Fi”. El modo Wi-Fi es el que empleamos en la semana 1 para conectarnos a los robots y consiste en que el robot emite su propia



(a) Directorio de trabajo

```

neural_network.py
lab > notebooks > week3 > neural_network.py > auto_aim
1 import cv2
2 import numpy as np
3 from robomaster import robot
4
5 def auto_aim(image, best_pred) -> float:
6     """
7         Once a robot is found, get the angle to move
8         :param image: image containing bounding box.
9         :param best_pred: best box prediction.
10        :return:
11            alpha: angle between image center and best bounding found.
12    """
13
14
15    # Get image shape
16    h, w = image.shape[:2]
17
18    # Get camera center
19    cam_center = np.array([w / 2, h / 2])
20
21    # Get (best) bounding box center
22    box_center = np.array([(best_pred[2] + best_pred[0]) / 2, (best_pred[3] + best_pred[1]) / 2])
23
24    # Get distance between camera center and box center
25    box_cam_dist = box_center[0] - cam_center[0]
26
27    # Check if it's on the right or on the left of the image
28    right_left = np.sign(box_cam_dist)
29
30    # Get distance to robot. Camera has 120° of field of view (FOV), half of that is 60° => 2 * pi / 3
31    rob_cam_dist = w / (2 * np.tan(np.pi / 3))
32
33    # Calculate angle
34    box_cam_dist = np.abs(box_cam_dist)
35    alpha = right_left * np.rad2deg(np.arctan(box_cam_dist / rob_cam_dist)) * 90 / 120 # Seems FOV != 120° -> FOV = 90°
36    return alpha
37
38
39 def detect_and_shoot(model):
40
41    # Start video stream
42    ep_camera.start_video_stream(display=False)
43
44    # Receive image
45    img = ep_camera.read_cv2_image()
46

```

(b) Entorno virtual

Figure 1: Configuración de VSCode.

red Wi-Fi. En esta semana, dado que los ordenadores del laboratorio no tienen un adaptador de Wi-Fi, vamos a utilizar el método networking. Este método consiste en generar un código QR con información acerca de la red Wi-Fi del laboratorio y escanearlo con el robot, de forma que este pueda conectarse a la red del laboratorio.

Para ello, abra el script *connect.py* (Figura 2a), escriba la SSID y la contraseña de la red Wi-Fi del laboratorio en las variables correspondientes y añada el número de serie de su RoboMaster (cuadro rojo superior en la Figura 2b). El número de serie no es necesario para conectar el robot

```

import time
from robomaster import conn
from MyQR import myqr
from PIL import Image

QR_CODE_NAME = "qrcode.png"
WIFI_SSID = "<wifi-ssid>"
WIFI_PASSWD = "<wifi-password>"
SERIAL_NUMBER = "<robot-serial-number>"

if __name__ == '__main__':
    # Manually indicate the IP of your computer if it is not found automatically
    # robomaster.config.LOCAL_IP_STR = "XXX.XXX.XXX.XXX"

    # Write serial number to txt
    with open('serial_number.txt', 'w') as f:
        f.write(SERIAL_NUMBER)

    # Create QR to connect the robot to your Wi-Fi
    helper = conn.ConnectionHelper()
    info = helper.build_qrcode_string(ssid=WIFI_SSID, password=WIFI_PASSWD)
    myqr.run(words=info)

    # Show the QR
    img = Image.open(QR_CODE_NAME)
    img.show()

    # Wait for connection (scan the qr with the robot)
    print("Waiting for connections...")
    time.sleep(5)
    if helper.wait_for_connection():
        print("Connected!")
    else:
        print("Connect failed!")

```

(a) Entorno virtual



(b) Activar el modo ”networking”.

Figure 2: Connectar el RoboMaster a la red Wi-Fi del laboratorio.

a la red del laboratorio, pero es vital para el resto de scripts, ya que es la única forma en que su ordenador puede discernir entre su propio robot y el robot de algún otro grupo que esté conectado en la misma red (de lo contrario, podría lanzar su script sobre el robot de otro grupo y provocar algún accidente). El número se serie se guardará en un archivo llamado *serial_number.txt*.

Tras escribir la SSID y la contraseña, lance el script, agarre a su robot y apunte con cuidado la cabeza en dirección al QR emergente (si no emerge el QR, abra el archivo *qrcode.png*). Si todo va

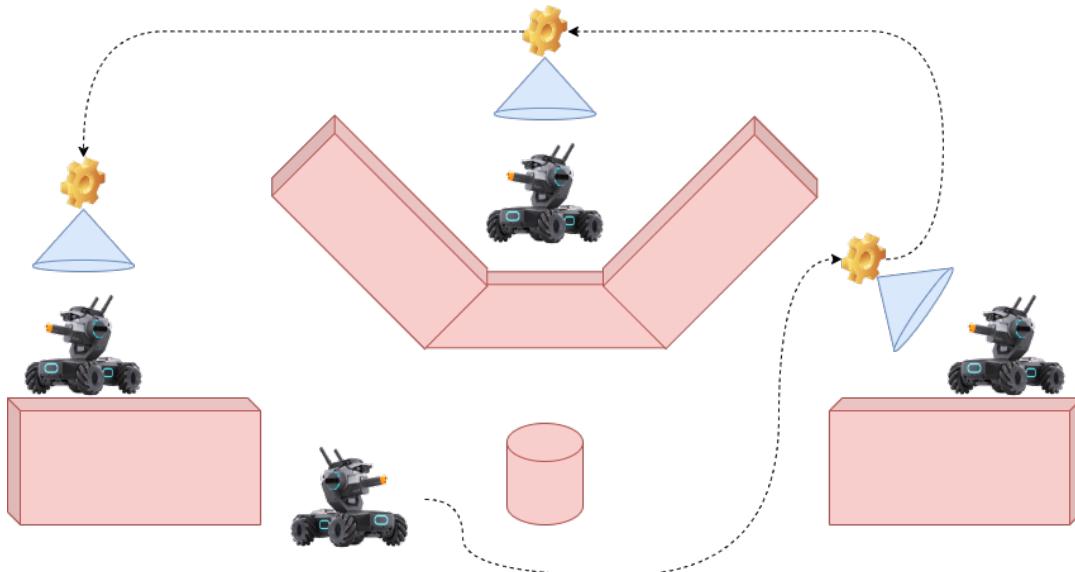


Figure 3: Circuito de ejemplo.

bien, el robot le dirá que se ha conectado con éxito y ya podrá probar el resto de scripts. Juegue con todos los scripts de ejemplo y analicelos detenidamente. Le servirán para poder completar el script *demo.py* sobre el que programará el circuito final.

4 Circuito

La evaluación final del proyecto se llevará a cabo en un circuito donde se pondrán a prueba los distintos componentes que se han desarrollado a lo largo del proyecto. El circuito estará compuesto por una serie de obstáculos, que cada grupo deberá sortear de la forma más eficiente posible, como se muestra en el ejemplo de la Figura 3. A lo largo del circuito estableceremos puntos estratégicos donde llevar a cabo la detección de RoboMasters adversarios mediante el empleo de la red neuronal entrenada durante la semana 2 del proyecto. La tarea de los alumnos será desarrollar una estrategia efectiva para el recorrido del circuito, así como una estrategia de detección eficiente. Los objetivos detectados serán disparados e inutilizados (utilizando el módulo "blaster").

El diseño de una estrategia de recorrido y detección es fundamental para garantizar que el RoboMaster pueda cumplir con éxito su tarea de detectar RoboMasters adversarios mientras recorre un espacio de manera estratégica. La detección de adversarios es una tarea computacionalmente costosa, por lo que será esencial plantear una estrategia eficiente para lograr una detección efectiva.

Una estrategia comúnmente utilizada en esta práctica implica que el RoboMaster se desplace de manera planificada a puntos estratégicos predefinidos en el circuito para realizar detecciones. Para implementar esta estrategia, los alumnos deberán programar la lógica que guiará al RoboMaster a través del circuito y definir cuándo y dónde se realizarán las detecciones. A continuación, se presenta un ejemplo de estrategia que los alumnos pueden considerar:

1. Desplazamiento a Puntos Estratégicos: El RoboMaster se mueve siguiendo una ruta predefinida que incluye paradas en puntos estratégicos en el circuito. Estos puntos estratégicos se eligen en función de la visibilidad y las posibles ubicaciones de adversarios.
2. Captura de Imágenes: En cada punto estratégico, el RoboMaster detiene su desplazamiento y captura una imagen utilizando su cámara. Esta imagen servirá como base para la detección de adversarios.

3. Detección en Ángulos Estratégicos: El RoboMaster lleva a cabo una detección sobre la imagen capturada en ese punto estratégico. Si no se detecta ningún RoboMaster adversario en esa imagen, el RoboMaster rota su torreta en un ángulo predeterminado (por ejemplo, 90 grados), captura una nueva imagen y realiza una nueva detección. Este proceso se repite hasta que se haya completado un barrido de 360 grados.
4. Apuntado y disparo: Se evalúan los resultados de la detección, se considera(n) solo aquella(s) detección(es) con un valor de confianza superior a un cierto umbral, se utiliza la función de autoapuntado (disponible en el script *demo.py*) y se dispara con el blaster.

Es importante destacar que esta estrategia es solo un ejemplo y que existen diversas variantes y enfoques posibles. Los alumnos tendrán la oportunidad de diseñar su propia estrategia de circuito, adaptándola a las necesidades específicas de la tarea y aprovechando al máximo los recursos computacionales disponibles.

NOTA: Quizás eché en falta algunas funciones, como aquellas para cargar el modelo y puede que alguna otra. Revise los scripts de las semanas anteriores y utilice las herramientas que necesite para llevar a cabo la demo.

5 Objetivos

Aunque se permite una cierta versatilidad a la hora de diseñar el circuito, los alumnos deben obligatoriamente cumplir los siguientes puntos:

- Dado un punto inicial y una serie de puntos de parada, los alumnos deben diseñar un recorrido que pase por todos ellos **sin colisionar** con ningún obstáculo.
- Idealmente, se debe inutilizar a todos los RoboMasters que haya en el circuito (cada RoboMaster debe recibir 12 disparos para ser eliminado). Se entiende que los modelos de detección pueden fallar (principalmente, debido a la escasez de muestras de entrenamiento) y equivocarse en algún punto.
- Piense cómo hacer una buena estrategia de detección cada vez que se detenga en un punto de parada. ¿Debe disparar solo a la mejor detección en la imagen (mayor valor de confianza predicho por la red) o debe disparar a todo lo que se detecte? ¿Debe seguir rotando el gimbal (parando cada X grados a realizar la detección) hasta que detecte algo o hasta que llegue a los 360 grados?
- Cada imagen sobre la que se ha aplicado la detección con el modelo debe guardarse en una carpeta dedicada. Las bounding boxes predichas (y su valor de confianza) deben pintarse en la imagen.
- Los alumnos deben usar, obligatoriamente, todos los módulos del robot: cámara, audio, blaster, leds, chassis y gimbal.
- Se invita a los alumnos a ser originales con sus circuitos. Bailes, circuitos complicados, obstáculos extra, etc. son bienvenidos.

- Describa en la memoria cómo ha empleado cada módulo del robot, cómo ha diseñado su circuito, qué funcionalidades extra ha incluido y qué estrategia (detallada paso por paso) ha seguido para girar el gimbal, realizar la detección y elegir a qué “bounding boxes” disparar.
- Presente un informe que contenga las observaciones y conclusiones extraídas. Este informe será incremental (se irá expandiendo a lo largo del proyecto). Se requiere incluir una introducción al problema a resolver, una explicación de cómo ha empleado cada módulo del robot y cómo ha diseñado su circuito (haga un esquema o un dibujo), qué funcionalidades extra ha incluido y qué estrategia (detallada paso por paso) ha seguido para girar el gimbal, realizar la detección y elegir a qué bounding boxes disparar. Se valorará la limpieza y presentación del documento.

NOTA: Sobra decirlo, pero trate a los robots con sumo cuidado. Está **PROHIBIDO** aumentar la velocidad del robot. Intente no equivocarse entre los ejes “x” e “y”, ya que es fácil confundirse (deje espacio en todas direcciones cuando haga pruebas). Cuando lance un script para mover el robot, asegúrese de que algún compañero lo sigue de cerca para evitar colisiones. Detenga de inmediato el script que esté ejecutando (Ctrl+C) si hay algún peligro o potencial accidente. Trate también con mucho cuidado el gimbal, que es bastante sensible. No agarre al robot por la cabeza o el gimbal, agárrelo siempre desde la base. Por último, no apunte a los ojos a nadie si va a disparar con el blaster, ya que puede ser molesto a la vista.