Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Querétaro

MANUAL TÉCNICO

Autores:

A01368818 Joel Sánchez Olvera

A01661090 Juan Pablo Cabrera Quiroga

A01704076 Adrián Galván Díaz

A01708634 Carlos Eduardo Velasco Elenes

A01709522 Arturo Cristián Díaz López

Fecha:

27 de noviembre del 2024

Introducción

Este manual tiene como objetivo proporcionar una guía detallada para la implementación del proyecto de detección de vacas en un ambiente productivo. El proyecto utiliza un modelo de aprendizaje profundo basado en YOLOv5 para detectar vacas en imágenes y predecir su postura, ya sea acostada o de pie.

El actual documento cubre todos los requerimientos y pasos necesarios para preparar el entorno, implementar el modelo, integrar los componentes del sistema y garantizar su correcto funcionamiento en un escenario real.

Requerimientos del sistema

Para garantizar el correcto y óptimo funcionamiento del sistema en un entorno productivo, es necesario cumplir con los siguientes requerimientos:

Hardware

1. Raspberry Pi
    a. Memoria RAM: 1 GB
    b. Almacenamiento: Tarjeta microSD de al menos 16 GB (recomendado 32 GB)
2. Periféricos (si se prefiere configuración por Raspberry Pi Desktop)
    a. Monitor, teclado y mouse
3. Cámara compatible con Raspberry Pi

Software

1. Sistema Operativo: Raspberry Pi OS (64-bit)
2. Dependencias del proyecto (descritas aquí)
3. Dependencias del sistema (descritas más adelante)

Conectividad

1. Acceso a internet para la descarga de dependencias y actualizaciones (necesario únicamente para la preparación del entorno, mas no para la ejecución)
2. Red local para pruebas o acceso remoto al sistema
3. SSH configurado para acceso remoto a la Raspberry Pi

Preparación del Entorno

Esta sección detalla los pasos necesarios para preparar el entorno en la Raspberry Pi y garantizar que todos los componentes estén correctamente configurados antes de desplegar el proyecto.

Instalación del Sistema Operativo

1. Descargue la herramienta Raspberry Pi Imager desde la página oficial.
2. Formatee la tarjeta microSD desde la herramienta Raspberry Pi Imager o desde el sistema de ficheros de su preferencia, al formato FAT32.
3. Instale el sistema operativo Raspberry Pi OS (64-bit) seleccionando su modelo de dispositivo y su tarjeta de almacenamiento microSD.
    a. Presione CTRL + SHIFT + X para entrar en el modo de configuración avanzada.
    b. Establezca un nombre de usuario y contraseña seguros.
    c. Habilite los ajustes regionales y seleccione su región.
    d. En la pestaña servicios, habilite SSH con autenticación por contraseña (la que definió anteriormente).
    e. Salga del modo de configuración avanzada y presione Siguiente para comenzar el flasheo del sistema operativo a su microSD.
4. Al finalizar, introduzca su tarjeta microSD en la Raspberry Pi y posteriormente conéctela a corriente. Tras encender, deberá ver un LED verde parpadeando, esto indica que el dispositivo está leyendo exitosamente la imagen de sistema operativo contenida en su microSD.

Conexión a la Raspberry Pi

Seleccione un método para establecer una conexión a la Raspberry Pi.

A. Conexión a través de Raspberry Pi Desktop:
    1. Conecte los periféricos necesarios (monitor, teclado y mouse) a la Raspberry Pi.
    2. Encienda la Raspberry Pi e inicie sesión con las credenciales configuradas previamente.
    3. Configure la red y el acceso SSH (opcional) desde las opciones de configuración del sistema.

B. Conexión Remota vía SSH
    1. Configure la conexión a internet en su Raspberry Pi
    2. Identifique la IP de su Raspberry Pi
    3. Desde una máquina conectada a la misma red, ejecute el comando:

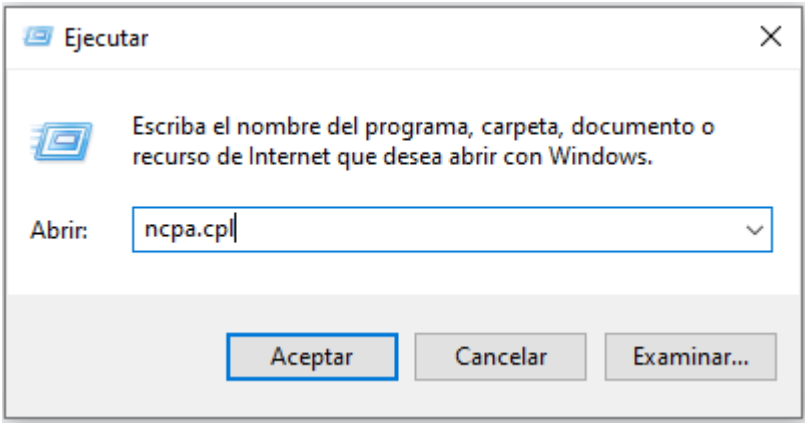
```
Unset
ssh <username>@<raspberry_pi_ip>
```

4. Introduzca la contraseña configurada en el setup del sistema operativo

C. Conexión a través de una red local
   1. Conecte la microSD nuevamente a su computador
   2. Abra el archivo de configuración 'cmdline.txt' usando el editor de texto Notepad++ (usar bloc de notas puede corromper el archivo de configuración).
   3. Añada la siguiente línea al final del archivo

```
Unset
ip=192.168.0.2::192.168.0.1:255.255.255.0::eth0:off
```

4. Cree un archivo vacío y sin extensión, llamado ssh dentro de su microSD.
5. Conecte su cable ethernet de la Raspberry Pi hacia su computadora y encienda el dispositivo.
6. Desde su computador, diríjase a la aplicación Ejecutar (Run). Introduzca 'ncpa.cpl' y presione Enter.



7. En la red asociada a su cable ethernet, seleccione: Clic derecho > Propiedades > Protocolo de Internet versión 4 (TCP/IPv4) > Propiedades
8. Configure lo siguiente:
   a. IP Address: 192.168.0.1

    b. Subnet Mask: 255.255.255.0
    c. Default Gateway: No es necesario llenar este campo
9. Verifique la conexión desde su computador haciendo un ping a su Raspberry Pi.

```
Unset
ping 192.168.0.2
```

10. Acceda a su Raspberry Pi usando el usuario y contraseña configurados en en el setup del sistema operativo:

```
Unset

ssh <username>@192.168.0.2
```

Instalación de dependencias locales

Una vez dentro de la Raspberry Pi, comenzaremos con la instalación del proyecto siguiendo los pasos descritos a continuación.

1. Ejecute el comando sudo apt update para actualizar los paquetes internos de la Raspberry Pi
2. Instale git usando el comando sudo apt install git -y
3. Instale python y python-pip para la ejecución del proyecto, sudo apt install python3 python3-pip
4. Instale paqueterías relacionadas al manejo de bases de datos, sudo apt install postgresql postgresql-contrib
5. Verifique el estatus del servicio de postgresql usando el comando sudo service postgresql status

Esto concluye la instalación de dependencias locales en el dispositivo Raspberry Pi.

Instalación del Proyecto

En esta sección, se detallan los pasos a seguir para llevar a cabo la instalación del proyecto desde el repositorio remoto de Github.

1. Ejecute el siguiente comando para clonar el repositorio:

```
Unset
git clone
https://github.com/JP-coder2000/cattle-segmentation.git
```

2. Acceda a la carpeta del repositorio clonado

```
Unset
cd cattle-segmentation/
```

3. Comience creando un entorno virtual de python para la instalación de dependencias internas

```
Unset
python -m venv env
```

4. Active el entorno virtual que acaba de crear

```
Unset
source env/bin/activate
```

5. Instale las dependencias necesarias para la ejecución principal del proyecto

```
Unset
pip install -r requirements.txt
```

Configuración de variables de entorno

Como siguiente paso, será necesario crear un archivo con la configuración correspondiente de variables de entorno que permitan establecer una conexión a la base de datos local.

En la carpeta cattle-segmentation (raíz del proyecto), cree el archivo .env usando el comando nano .env

Será necesario declarar las siguientes líneas en el archivo:

```
Unset
DB_NAME=cattle_segmentation
DB_USER=postgres
DB_PASSWORD=root
DB_HOST=localhost
DB_PORT=5432
DB_URL=postgresql://postgres:root@localhost:5432
DB_FULL_URL=postgresql://postgres:root@localhost:5432/cattle_segmentation
```

El usuario y contraseña por defecto para postgresql son postgres y root. Puede asegurarse de que las credenciales sean correctas accediendo a postgresql desde la consola. Ejecute el comando sudo -i -u postgres para acceder al usuario postgres. Posteriormente, use el comando psql para acceder a la consola de postgresql. Use el comando \du para ver la lista de usuarios disponibles. En caso de ser necesario, ajuste la contraseña del usuario postgres usando la siguiente query

```
Unset
ALTER USER postgres WITH PASSWORD 'root';
```

O usando la contraseña de su elección. La contraseña configurada para este usuario debe coincidir con la que usa en su archivo de configuración de variables de entorno. Al finalizar, puede utilizar el comando \q para salir de la consola de postgresql. Finalmente, utilice el comando exit para volver a la consola regular.

Inicialización de la base de datos

Para inicializar la base de datos, así como sus tablas y relaciones, simplemente acceda al directorio database usando el comando cd database/. Nuevamente, asegúrese de que su entorno virtual de python sigue encendido, deberá ver la leyenda (env) al inicio de su terminal. Ejecute el comando de inicialización de la base de datos.

```
Unset
python db_init.py
```

Deberá ver los mensajes

```
Unset
Creando la base de datos cattle_segmentation...
Activando la extensión pgcrypto...
Creando la tabla 'image_info'...
Creando la tabla 'cow_details'...
```

Puede asegurarse de que la ejecución haya sido exitosa entrando nuevamente a psql usando los comandos sudo -i -u postgres, psql y \l. Deberá ver el nombre cattle_segmentation correspondiente a la base de datos recién creada. Salga de la consola psql usando \q y posteriormente exit.

Ejecución del Proyecto

En este punto del manual, usted cuenta con el entorno configurado para comenzar a realizar predicciones y utilizar los modelos de deep learning desarrollados y contenidos en el repositorio. Diríjase a la carpeta source usando el comando cd source/ (será necesario subir un directorio si aún se encuentra en la carpeta database/, use el comando cd ..). Dentro del directorio, se encuentra el archivo main.py el cual se encuentra listo para simplemente ser ejecutado y retornar predicciones según las imágenes dadas.

Dicho archivo seguirá el siguiente flujo ante la ejecución:

1. Configuración del logger (guardado de logs en un archivo de texto, en la carpeta .logs/)
2. Validación de sistema operativo y configuración según resultado
3. Importación de librerías y módulos del proyecto
4. Carga de los modelos de detección y clasificación

5. Conexión a la base de datos
6. Procesamiento de imágenes
    a. Detección de objetos
    b. Clasificación de objetos
    c. Generación de registros en base de datos

La función principal del archivo, detect_objects(directory) toma como parámetro el path a un directorio y procesa todas las imágenes con extensión png, jpg y jpeg. Si desea modificar el directorio utilizado para realizar predicciones, simplemente modifique el siguiente bloque de código:

```
Unset
DIRECTORY = 'path/a/su/directorio'

results = detect_objects(DIRECTORY)
```

El directorio definido DIRECTORY = '../dataset/bounding/detect/' contiene un sample de imágenes que puede utilizar para validar que la configuración del entorno haya sido exitosa. Ejecute el script usando el comando python main.py. Deberá visualizar un mensaje como el siguiente:

```
Unset
INFO - Logger configurado correctamente.
INFO - Sistema operativo: Windows - Configuración de
pathlib.Path a WindowsPath
INFO - Modelo de detección cargado correctamente.
INFO - Modelo de clasificación cargado correctamente.
INFO - Conexión a base de datos inicializada.
INFO - Iniciando procesamiento de detección y clasificación en
imágenes del directorio ../dataset/bounding/detect/
INFO - Imagen procesada: val_0.jpg | Vacas detectadas: 2
INFO - Imagen procesada: val_1.jpg | Vacas detectadas: 1
INFO - Imagen procesada: val_2.jpg | Vacas detectadas: 2
INFO - Procesamiento de imágenes completado.
```

Como se mencionó anteriormente, cada vez que se ejecuta el script main.py se crea un nuevo archivo de texto que almacena los logs de ejecución y los guarda en la carpeta

./logs, esto es especialmente útil para realizar depuración de posibles errores que puedan surgir en la ejecución del archivo principal del proyecto.

Finalmente, puede observar los nuevos registros creados en su base de datos accediendo a la consola psql. Use los comandos:

```
Unset
sudo -i -u postgres
psql
\c cattle_segmentation
SELECT * FROM image_info;
SELECT * FROM cow_details;
```

Si puede observar los registros generados, significa que la ejecución e inserción de registros en la base de datos fue exitosa.

Estructura de la base de datos

El sistema utiliza una base de datos PostgreSQL para almacenar y gestionar la información generada por el modelo de detección de vacas. A continuación, se describe la estructura de la base de datos, diseñada para registrar tanto los metadatos de las imágenes procesadas como los detalles de cada vaca detectada.

La base de datos incluye dos tablas principales:

1. image_info: Contiene información general sobre las imágenes procesadas.
2. cow_details: Almacena información específica de cada vaca detectada en las imágenes.

A continuación se describe el diccionario de datos correspondiente a la base de datos creada.

Tabla 1: image_info
Registra los metadatos básicos de cada imagen procesada.

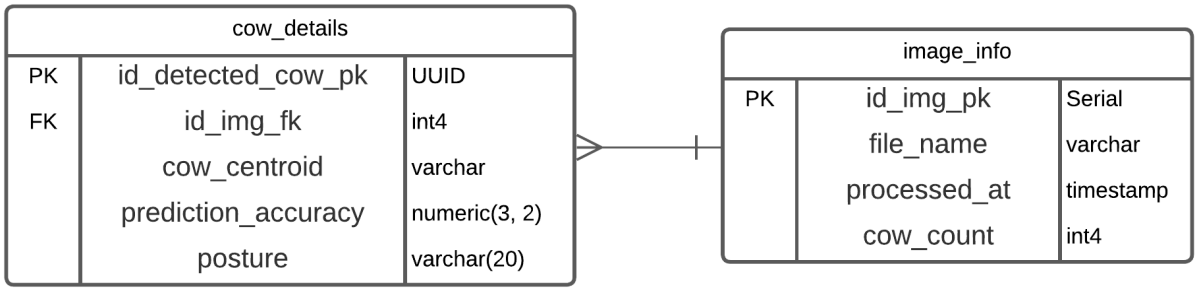
Campo	Tipo	Descripción
id_img_pk	SERIAL	Identificador único de la imagen (clave primaria). Incrementa en una unidad automáticamente por cada registro generado.
file_name	VARCHAR	Nombre del archivo procesado incluyendo su extensión.

processed_at	TIMESTAMP	Fecha y hora en que se procesó la imagen.
cow_count	INTEGER	Número de vacas detectadas en la imagen.

Tabla 2: cow_details
Registra la información específica de cada vaca detectada.

Campo	Tipo	Descripción
id_detected_cow_pk	UUID	Identificador único de la vaca detectada (clave primaria).
id_img_fk	INTEGER	Clave foránea que relaciona con la tabla image_info.
cow_centroid	VARCHAR	Coordenadas del centroide de la vaca detectada.
prediction_accuracy	NUMERIC(3, 2)	Número de vacas detectadas en la imagen.
posture	VARCHAR(20)	Postura de la vaca (acostada o de pie).

El siguiente diagrama muestra también la estructura de la base de datos.



Integración en Productivo

Para integrar el sistema en un ambiente productivo, se requiere configurar la ejecución automática del sistema, asegurar la disponibilidad de la base de datos y gestionar los posibles errores que puedan surgir. A continuación, se detallan los pasos necesarios para esta integración.

Automatización del proceso con systemd

El sistema puede ejecutarse automáticamente a intervalos regulares utilizando herramientas como un servicio de sistema en la Raspberry Pi.

1. Cree un archivo de servicio usando el comando

```
Unset
sudo nano /etc/systemd/system/cattle-segmentation.service
```

2. Defina la configuración del servicio

```
Unset
[Unit]
Description=Cattle Segmentation System
After=network.target

[Service]
User=<your_user>
WorkingDirectory=/home/<your_user>/cattle-segmentation/source
ExecStart=/home/<your_user>/project-env/bin/python main.py
Restart=always
Environment="DB_NAME=cattle_segmentation"
Environment="DB_USER=postgres"
Environment="DB_PASSWORD=root"
Environment="DB_HOST=localhost"
Environment="DB_PORT=5432"
```

3. Cree un archivo de timer en el mismo directorio

```
Unset
sudo nano /etc/systemd/system/cattle-segmentation.timer
```

4. Defina el siguiente contenido en este archivo

```
Unset
[Unit]
Description=Timer for Cattle Segmentation Service

[Timer]
OnBootSec=1min
OnUnitActiveSec=5min

[Install]
WantedBy=timers.target
```

OnBootSec=1min: Ejecuta el servicio por primera vez 1 minuto después del arranque del sistema.

OnUnitActiveSec=5min: Ejecuta el servicio cada 5 minutos.

5. Guarde y habilite el servicio y su timer

```
Unset
sudo systemctl daemon-reload
sudo systemctl enable cattle-segmentation.service
sudo systemctl start cattle-segmentation.service

sudo systemctl enable cattle-segmentation.timer
sudo systemctl start cattle-segmentation.timer
```

6. Verifique el estado del servicio

```
Unset
sudo systemctl status cattle-segmentation.service
sudo systemctl status cattle-segmentation.timer
```

De esta manera, el servicio se ejecutará cada 5 minutos y creará predicciones y registros para las nuevas imágenes encontradas en el directorio dado.

Interacción entre componentes

En el siguiente diagrama se puede apreciar la integración e interacción entre componentes del sistema.