Tecnológico de Monterrey

Campus Querétaro

Manual de Usuario Técnico

Gamaliel Marines Olvera	A01708746
Uri Jared Gopar Morales	A01709413
José Antonio Miranda Baños	A01611795
María Fernanda Moreno Gómez	A01708653
Oskar Adolfo Villa López	A01275287
Luis Ángel Cruz García	A01736345

Inteligencia artificial avanzada para la ciencia de datos II
Grupo 501

Introducción

En este documento se incluyen los pasos de configuración, instalación y uso del modelo de identificación de uso de camas de vacas. Las pruebas de arquitectura se realizaron en una Raspberry Pi 3 de 2015, por lo que el manual se centra en el uso en este dispositivo. Sin embargo, se puede utilizar el modelo en dispositivos capaces de ejecutar Python 3 de 64 bits.

El proceso para utilizar el modelo a grandes rasgos es el siguiente:

1. Seleccionar las camas en la aplicación de escritorio.
2. Subir el archivo de los pesos del modelo y de la selección de las camas a la Raspberry.
3. Instalar la librería de Python en la Raspberry.
4. Incluir las funciones de la librería en el código principal de Python, utilizando el archivo de los pesos del modelo y de la selección de las camas.

Configuración

La configuración debe hacerse sola una vez cuando se inicia el modelo, y se debe realizar cada vez que se quiera mover la cámara a otra posición.

Conocimientos previos sugeridos

- Manejo de Python 3.
- Manejo de Raspberry PI con Raspbian o sistema operativo basado en linux.
- Manejo de conexiones por SSH o protocolo alternativo.

Requisitos previos

- Archivo de los pesos del modelo. Los pesos son el resultado del entrenamiento del modelo, y se incluyen en los entregables del proyecto. Este archivo se llama *model_weights.pth*
- Archivo ejecutable de la aplicación de selección de camas. Este archivo se encuentra entre los entregables como “aplicacion_seleccion_camas_<sistema operativo>”. Por ejemplo: “aplicacion_seleccion_camas_windows”.
- Muestra de imagen de camas . Esta imagen debe ser igual a las que se van a analizar cuando el modelo esté en funcionamiento.
- Raspberry Pi 3 2015 o superior. El dispositivo debe estar configurado con Raspbian o algún sistema operativo basado en Linux y contar con Python 3 o superior. Guía de instalación de Python. Guía de uso de Raspberry
- Una computadora, ya sea PC o laptop, con Windows 10 o posterior, MacOS o Linux (Basado en Debian x64).

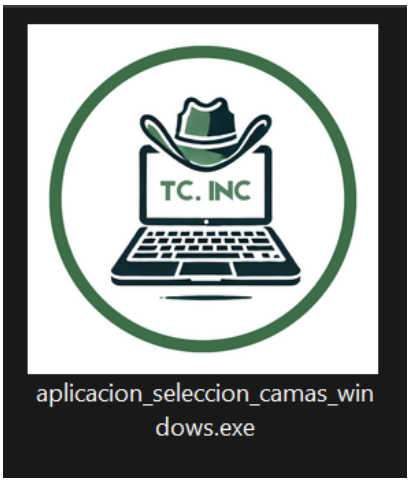
Instalación de interfaz de selección de camas

Antes de poder utilizar el modelo se deben configurar las camas que el modelo va a analizar. Para facilitar este proceso, se provee una aplicación que cuenta con una interfaz de usuario en la cual se pueden seleccionar las camas de manera gráfica. Esta

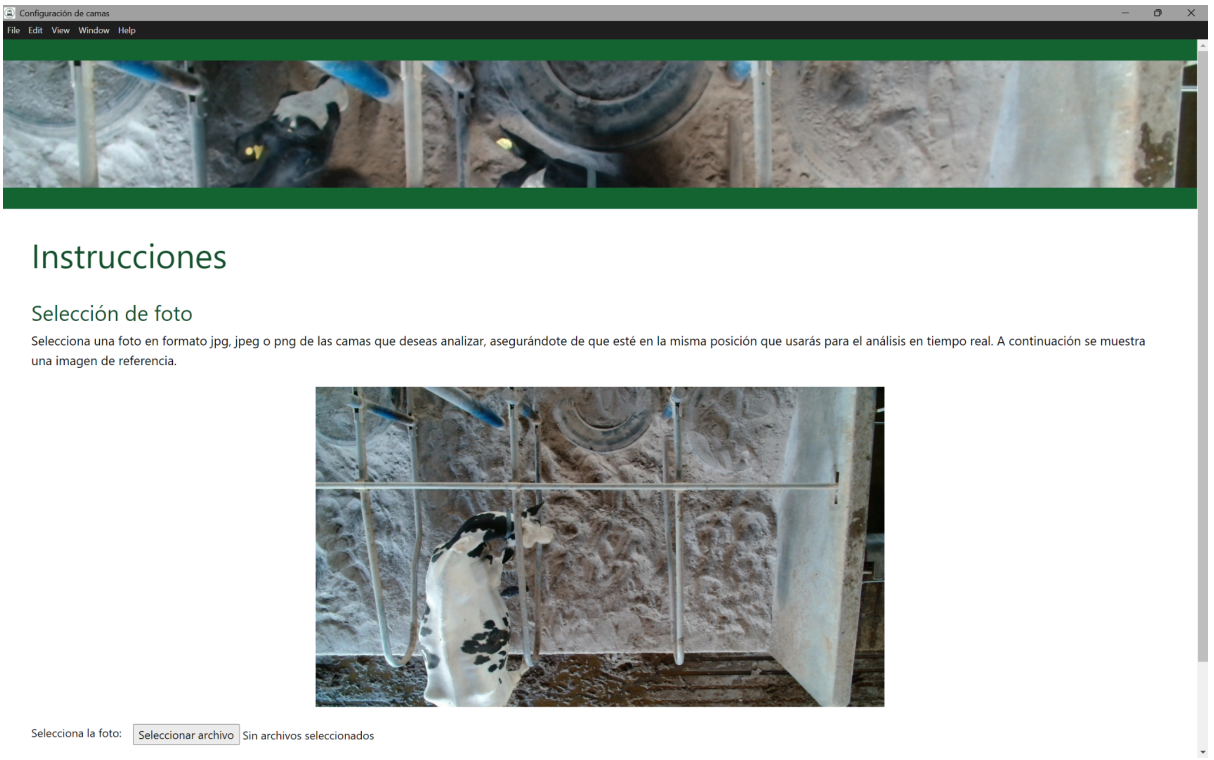
configuración se debe realizar en una computadora diferente a la Raspberry Pi, ya que esta no soporta la ejecución de la aplicación de selección.Para instalar la aplicación, sigue los pasos de acuerdo al sistema operativo:

Windows

1. Da click en el archivo *aplicacion_seleccion_camas_windows.exe*.



2. Se abrirá la aplicación directamente. No requiere instalación.



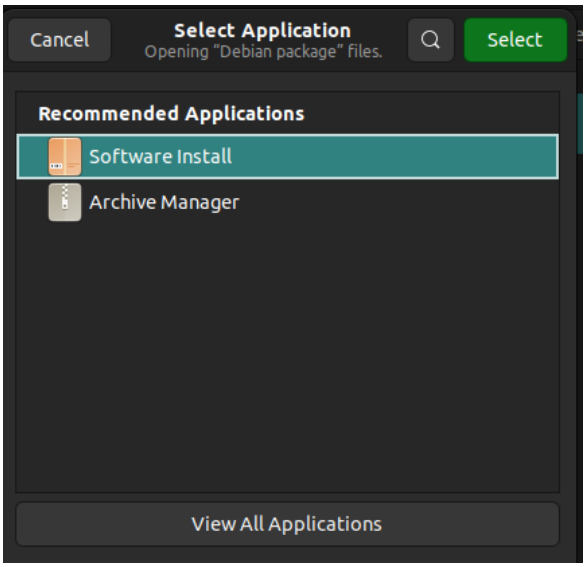
MacOS

1. Da doble click en el archivo *aplicacion-seleccion-camas-macos.zip*.

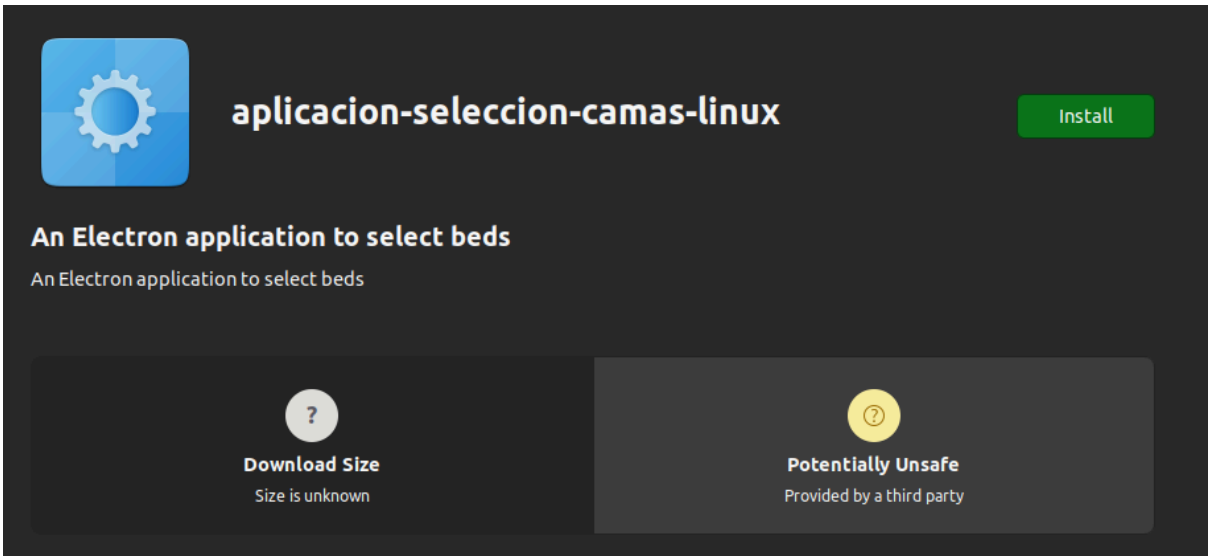
2. Se mostrará la aplicación instalada.

Linux

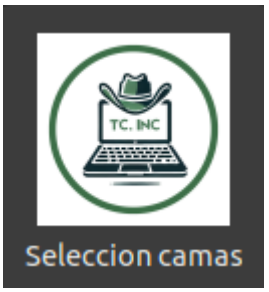
1. Da click derecho en el archivo *aplicacion-seleccion-camas-linux.deb* y selecciona la opción “Open with another application”.
2. Selecciona “Software Install”.



3. Se abrirá una ventana de Software Install. Da click en “Install”.



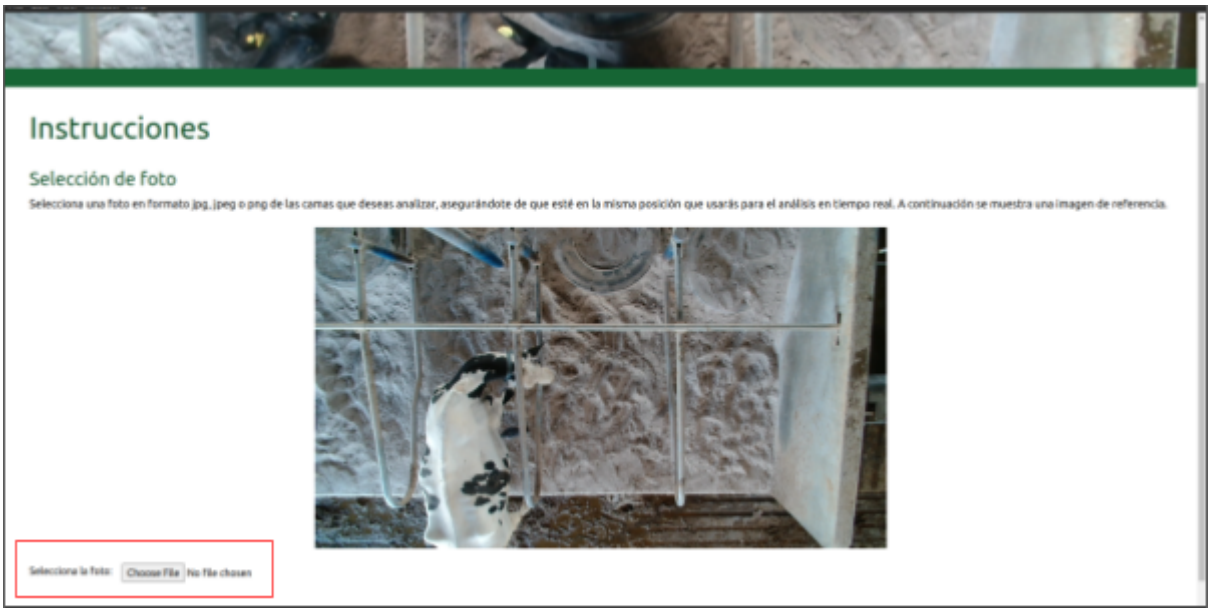
4. Se instalará la aplicación con el nombre *seleccion-camas.*



Selección de camas

Una vez que se abrió la aplicación, los pasos para la selección son los siguientes:

1. Abrir la aplicación en el sistema operativo correspondiente.
2. Seleccionar una imagen de muestra. Es necesario contar con una imagen que sea igual a las imágenes que se van a analizar con el modelo. Con esta imagen se realiza la selección de las camas.



3. Seguir los pasos de la aplicación para seleccionar cada una de las camas.
4. Haz clic en guardar

Una vez que se completen estos pasos, se tendrá el archivo con la selección. Este archivo se utilizará para la siguiente sección de la configuración. El archivo se guardará en la carpeta *Documentos* con el nombre *coordinates.json*

Subir archivos a Raspberry

Una vez que se tienen el archivo de los pesos del modelo y el archivo de la selección de las camas, se deben transferir a la Raspberry. Esto puede realizarse mediante SSH o algún protocolo de conexión similar, o mediante una conexión física. La ubicación de los archivos no necesita seguir ninguna regla, pero se recomienda que se encuentren ambos en la

misma ubicación para facilitar su acceso. En la siguiente liga se encuentra un tutorial para realizar la transferencia por medio de SSH.

Instalar librería de Python

Una vez que se tienen los archivos necesarios en la Raspberry, el resto de la configuración se debe realizar directamente en la Raspberry, ya sea conectando un teclado y monitor o utilizando algún protocolo para acceder remotamente a la terminal de comando de la Raspberry.

Las funciones necesarias para utilizar el modelo se empaquetaron en una librería de Python para facilitar su uso. La librería se encuentra en el repositorio público PyPi, bajo el nombre bed-classifier. Para instalar la librería, se puede utilizar el comando en la terminal de la Raspberry:

```
Unset
pip install bed-classifier
```

La librería tiene dependencias a otras librerías, las cuales se instalarán automáticamente al ejecutar el comando. Este paso puede tardar algunos minutos.

Una vez teniendo la librería instalada, se han completado todos los pasos necesarios de configuración.

Uso

Importar funciones

Para utilizar las funciones del modelo, se debe importar la librería en un script de Python.

Para ello, se utiliza la función *import.* Se pueden importar las funciones individualmente o en conjunto. Para importar las funciones individualmente, se utiliza la siguiente línea:

```
Python
from bed-classifier.functions import <nombre de la funcion>
```

Para importar todas las funciones, se utiliza la siguiente línea:

```
Python
from bed-classifier.functions import *
```

Una vez que se importan las funciones, se pueden utilizar escribiendo su nombre seguido de los parámetros necesarios dentro de paréntesis. Por ejemplo:

```python
load_model(path_to_model)
```

Documentación de funciones

A continuación se listan las funciones disponibles en la librería. Las funciones se dividen en principales y helpers. Las funciones principales son las que se necesitan para el uso común. Las funciones helpers son utilizadas dentro de las funciones principales y no se necesitan usar individualmente.

Funciones principales:

```python
load_model(model_path, device="cpu")
```

Regresa un objeto que contiene el modelo cargado. Este modelo se pasa como parámetro a otras funciones para hacer las predicciones.
Parámetros:

- model_path: Un string que contiene la ubicación del archivo de los pesos del modelo.
- device: Un string que indica el dispositivo en el cual se ejecutan las predicciones. Es útil para cuando se cuenta con GPU y se quiere utilizar en lugar de la CPU. El valor default es “cpu”. Para ver la lista de posibles valores, consulta la documentación de PyTorch.

Returns:

- Un objeto de tipo Module de PyTorch.

```python
predict_and_save(model, image_path, coordinates_path, output_path):
```

Realiza una predicción de todas las camas contenidas en una imagen. La predicción se guarda en un archivo CSV. Si el archivo existe, guarda una entrada más. Si no existe, se crea el archivo.

Parámetros:

- model: El modelo de PyTorch. Se debe cargar previamente con la función *load_model.*
- image_path: La ubicación de la imagen a analizar. Para poder guardar la fecha de la imagen, ésta debe estar nombrada con el formato “AAAA-MM-DD-HH-MM-SS” correspondiente a año, mes, día, hora, minuto y segundo, en ese orden. La imagen debe estar en formato JPG.
- coordinates_path: La ubicación del archivo con coordenadas de cada cama. Este archivo se genera con la aplicación de selección. El archivo es de formato JSON.
- output_path: La ubicación del archivo CSV en donde se almacenan los resultados. Si el archivo no existe, se creará al ejecutar la función.

Funciones helpers:

```
Python
predict_from_image(model, image)
```

Regresa la predicción del modelo de una imagen de tipo Image de la librería Pillow.
Parámetros:

- model: El modelo de PyTorch. Se debe cargar previamente con la función *load_model.*
- image: La imagen de tipo Image de Pillow.

Returns:

- Un string con la predicción del modelo: "cama_vacia", "vaca_acostada" o "vaca_parada".

```
Python
crop_images_from_json(image_path, json_path)
```

Recibe la ubicación de una imagen y la recorta de acuerdo a un archivo con las coordenadas de cada cama.
Parámetros:

- image_path: image_path: La ubicación de la imagen a recortar. Debe estar en formato JPG.
- json_path: La ubicación del archivo con coordenadas de cada cama. Este archivo se genera con la aplicación de selección. El archivo es de formato JSON.

Returns:

- Una lista que contiene las imágenes recortadas almacenadas en un objeto de tipo Image de Pillow.

Manejo de errores

La librería incluye mensajes de error para los errores comunes, como errores en las ubicaciones de los archivos o un formato incorrecto en el nombre de las imágenes. Todos los errores que ocurran durante la ejecución se almacenan en un archivo llamado *error_log.log*. Se incluye el mensaje de error, además de la fecha y hora.

Si ocurre un error, se detendrá la ejecución del programa. Para evitar que la ejecución se detenga si hay algún error, se puede utilizar el manejo de excepciones de Python. Sin embargo, esto no es recomendable porque los resultados se seguirán guardando y pueden ser incorrectos.

Ejemplo de uso

```python
from bed_classifier.functions import predict_and_save, load_model

def main():
	path_to_model_weights = "path/to/model_weights.pth"
	image_path = "path/to/image.jpg"
	coordinates_path = "path/to/coordinates.json"
	path_to_database = (
		"path/to/database.csv"  # This will be created if it does not
exist
	)

	model = load_model(path_to_model_weights)
	predict_and_save(
		model,
		image_path,
		coordinates_path,
		path_to_database,
	)

if __name__ == "__main__":
	main()
```