Visualizing YOLOv8 Fish Detection Results and Creating Annotated Videos

Karthik M Dani

2024-08-23

Importing Libraries

In this cell, we import the necessary libraries for data handling, image processing, visualization, and file management.

Explanation:

- `PIL` and `PIL.Image`: For image processing tasks.
- `IPython.display.display`: For displaying images within the notebook.
- `glob`: For file pattern matching.

```
import pandas as pd
import numpy as np
import PIL
from PIL import Image
from IPython.display import display
import matplotlib.pyplot as plt
from glob import glob
import random
import cv2
import warnings
warnings.simplefilter('ignore')
```

Visualizing a Random Sample of Images

This cell randomly sample a specified number of images from the validation dataset and display them in a grid for visual inspection.

Explanation:

- `root_path`: Path pattern to load images from the validation directory.
- `num_samples`: Number of random images to display.
- `images_data`: List of all image file paths sorted alphabetically.
- `random_images`: Randomly sampled image paths.

- `num_rows`: Calculated number of rows needed to display the images.
- `plt.figure()`: Initializes the figure with a specific size.
- `plt.subplot()`: Defines the subplot layout for each image.
- `plt.imshow()`: Displays the image.
- `plt.axis('off')`: Hides the axis for a cleaner display.

```
root_path = 'cleaned_data/validation/images/*'
num_samples = 200
images_data = sorted(glob(root_path))
random_images = random.sample(images_data, num_samples)

num_rows = num_samples // 2
plt.figure(figsize=(16, 8 * num_rows))

print("Num rows:", num_rows)
for i in range(num_samples):
    plt.subplot(num_rows, 2, i + 1)
    plt.imshow(cv2.imread(random_images[i]))
    plt.axis('off')
plt.show()
```

Loading a YOLOv8 Model

In this cell, we load a pre-trained YOLOv8 model from a specified weights file.

Explanation:

- `from ultralytics import YOLO`: Imports the YOLO class from the Ultralytics library.
- `model = YOLO('best.pt')`: Initializes a YOLOv8 model with pre-trained weights specified in `'best.pt'`.

```
from ultralytics import YOLO
model = YOLO('best.pt')
```

Predicting Objects with YOLOv8 and Displaying Results

In this cell, we use the YOLOv8 model to predict objects in randomly sampled images, process the detection outputs, and print details about detected objects.

Explanation:

- `yolo_outputs = model.predict(random_images[i])`: Runs the YOLOv8 model on the image to get predictions.
- `output = yolo_outputs[0]`: Extracts the first output (in case there are multiple outputs).
- `box = output.boxes`: Retrieves bounding box information.
- `names = output.names`: Gets the class names for detected objects.

- `for key, value in names.items()`: Updates class names to "FISH".
- `for j in range(len(box))`: Iterates over each detected object.
  - `labels = names[box.cls[j].item()]`: Gets the label for the detected class.
  - `coordinates = box.xyxy[j].tolist()`: Gets the bounding box coordinates.
  - `confidence = np.round(box.conf[j].item(), 2)`: Gets the confidence score.
- `images.append(output.plot()[:, :, ::-1])`: Stores the image with detected bounding boxes in the `images` list.

```
images = []

for i in range(num_samples):
    yolo_outputs = model.predict(random_images[i])
    output = yolo_outputs[0]
    box = output.boxes
    names = output.names

    for key, value in names.items():
        names[key] = "FISH"

    for j in range(len(box)):
        labels = names[box.cls[j].item()]
        coordinates = box.xyxy[j].tolist()
        confidence = np.round(box.conf[j].item(), 2)

        print(f'In this image {len(box)} {labels} has been detected.')
        print(f'Coordinates are: {coordinates}')
        print(f'Confidence is: {confidence}')
        print('-------')

    # Store the image in the 'images' list
    images.append(output.plot()[:, :, ::-1])
```

Annotating Images with YOLOv8 Predictions and Displaying Results

In this cell, we use the YOLOv8 model to predict objects in images, draw bounding boxes and labels on the images, and then display these annotated images.

Explanation:

- `yolo_outputs = model.predict(cv2.imread(random_images[i]))`: Loads and processes the image using YOLOv8.
- `img = cv2.imread(random_images[i])`: Reloads the image to draw bounding boxes and labels.

- `for j in range(len(box))`: Iterates through each detected object.
  - `cv2.rectangle()`: Draws the bounding box on the image.
  - `cv2.putText()`: Adds a label and confidence score to the image.
- `images.append(img)`: Adds the annotated image to the `images` list.
- `plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))`: Displays the annotated image using Matplotlib, converting from BGR to RGB color space.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

images = []

for i in range(num_samples):
    yolo_outputs = model.predict(cv2.imread(random_images[i]))  # Load image using cv2.imrea
    output = yolo_outputs[0]
    box = output.boxes
    names = output.names

    for key, value in names.items():
        names[key] = "FISH"

    img = cv2.imread(random_images[i])  # Load the image again to draw on it

    for j in range(len(box)):
        labels = names[box.cls[j].item()]
        coordinates = box.xyxy[j].tolist()
        confidence = np.round(box.conf[j].item(), 2)

        # Draw bounding box and label on the image
        pt1 = (int(coordinates[0]), int(coordinates[1]))
        pt2 = (int(coordinates[2]), int(coordinates[3]))
        cv2.rectangle(img, pt1, pt2, (0, 255, 0), 2)
        cv2.putText(img, f'{labels} {confidence}', (pt1[0], pt1[1] - 5), cv2.FONT_HERSHEY_SI

    # Store the annotated image in the 'images' list
    images.append(img)

# Now 'images' list contains images with bounding boxes and labels drawn on them
# Display each image in the images list
for img in images:
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for matplotlib
    plt.axis('off')  # Turn off axis
    plt.show()
```

Creating a Video from Annotated Images

In this cell, we generate a video file by combining a sequence of annotated images.

Explanation:

- `output_video = cv2.VideoWriter('output_video.avi', cv2.VideoWriter_fourcc(*'MJPG'), 1, (img.shape[1], img.shape[0]))`: Initializes the video writer with:
  - `'output_video.avi'`: Output video file name.
  - `cv2.VideoWriter_fourcc(*'MJPG')`: Codec for video encoding (MJPEG).
  - `1`: Frame rate (1 frame per second).
  - `(img.shape[1], img.shape[0])`: Frame size (width x height of the images).
- `output_video.write(img)`: Writes each image to the video file.
- `output_video.release()`: Finalizes and closes the video file.

```
import cv2

# Define the video writer
output_video = cv2.VideoWriter('output_video.avi', cv2.VideoWriter_fourcc(*'MJPG'), 1, (img.

# Write each image in the images list to the video
for img in images:
    output_video.write(img)

# Release the video writer
output_video.release()
```

Thank you!