YOLO Assignment

MTech in Applied AI
Deep Learning Techniques

by

Sanjeev Kumar Pandey*
Faculty: Dr Anamika Gupta

Submitted May 21, 2025

* Student ID: 31050, Enrolment No:MT23AAI001, Email: pandey.sanjeev@yahoo.com

Table of Contents

1 Programming Assignment

1.1 Question 1

Define functions to convert between these two representations: corner_to_center converts from the twocorner representation to the center-width-height presentation, and center_to_corner vice versa.

1.1.1 Solution: Python code

Here are the two functions to convert bounding boxes between the two common YOLO representations:

1. `corner_to_center`: Converts from the two-corner format

$$(x_{\min}, y_{\min}, x_{\max}, y_{\max})$$

to the center-width-height format

$$(x_{\text{center}}, y_{\text{center}}, \text{width}, \text{height}).$$

2. `center_to_corner`: Converts from the center-width-height format back to the two-corner format.

```
1
2  def corner_to_center(x_min, y_min, x_max, y_max):
3      """
4      Convert bounding box from corner format (x_min, y_min, x_max, y_max)
5      to center format (x_center, y_center, width, height).
6      """
7      width = x_max - x_min
8      height = y_max - y_min
9      x_center = x_min + width / 2
10     y_center = y_min + height / 2
11     return x_center, y_center, width, height
12
13 def center_to_corner(x_center, y_center, width, height):
14     """
15     Convert bounding box from center format (x_center, y_center, width, height)
16     to corner format (x_min, y_min, x_max, y_max).
17     """
18     x_min = x_center - width / 2
19     y_min = y_center - height / 2
20     x_max = x_center + width / 2
21     y_max = y_center + height / 2
22     return x_min, y_min, x_max, y_max
23
24
25 corner_box = (100, 150, 200, 250)
26 center_box = corner_to_center(*corner_box)
27 corner_box_converted_back = center_to_corner(*center_box)
28 print(f'Corner to center : {center_box}')
29 print(f'Center to corner : {corner_box_converted_back}')
```

Listing 1: Code

1.1.2 Output Question 1

```
C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\subjects\com\anamika_singh\YOLO1_Assignment>python programming_assignment_q1.py
Corner to center : (150.0, 200.0, 100, 100)
Center to corner : (100.0, 150.0, 200.0, 250.0)
```

Figure 1: Output: Question 1

1.2 Question 2

Find an image and try to label a bounding box that contains the object.

1.2.1 Input Image

Figure 2: Input Image: Question 2

```
Microsoft Windows [Version 10.0.26100.3915]
(c) Microsoft Corporation. All rights reserved.

C:\Users\urssa>pip install opencv-python
Collecting opencv-python
  Downloading opencv_python-4.11.0.86-cp37-abi3-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: numpy>=1.21.2 in c:\users\urssa\appdata\local\programs\python\python311\lib\site-packages (from ope
ncv-python) (1.24.3)
Downloading opencv_python-4.11.0.86-cp37-abi3-win_amd64.whl (39.5 MB)
   ---------------------------------------- 39.5/39.5 MB 4.5 MB/s eta 0:00:00
Installing collected packages: opencv-python
Successfully installed opencv-python-4.11.0.86

C:\Users\urssa>
```

Figure 3: Depedency Installation

1.2.2 Solution: Python code

```python
# Find an image and try to label a bounding box that contains the object.



import cv2

# Load the image
image = cv2.imread('black_apples.jpeg')

# Define the bounding box for the center apple (x_min, y_min, x_max, y_max)
# These values are estimated for the center apple in your image

y_min = 40
y_max = 310

x_min = 10
x_width =320

for i in range(1, 4):
    if i>1:
```

```
        x_min = x_min + x_width + 20
    x_max = x_min + x_width -25

    # Draw the bounding box
    cv2.rectangle(image, (x_min, y_min), (x_max, y_max), color=(0, 255, 0), thickness=2)

    # Add the label above the bounding box
    label = "black apple"
    cv2.putText(image, label, (x_min, y_min - 10), cv2.FONT_HERSHEY_SIMPLEX,
            fontScale=0.8, color=(0, 255, 0), thickness=2)

# Save or display the result
cv2.imwrite('labeled_black_apple.jpg', image)
# To display the image (optional):
# cv2.imshow('Labeled Image', image)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
```

Listing 2: Code

1.2.3 Output Image

Figure 4: Input Image: Question 2

1.3 Question 3

Perform the following tasks in Python

1. **Setup and Load Model**
   (a) Install the required libraries.
   (b) Load a pre-trained YOLOv_ model.
2. **Inference on Images**
   (a) Run the model on sample images.
   (b) Visualize the bounding boxes with labels and confidence scores.
3. **Custom Dataset Inference**
   (a) Use a small custom dataset (e.g., traffic signs, animals, faces).
   (b) Run inference and evaluate the model's performance (precision, recall, mAP).

```
C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\subjects\com\anamika_singh\YOLO1_Assignment>pip
Collecting ultralytics
  Downloading ultralytics-8.3.139-py3-none-any.whl.metadata (37 kB)
Requirement already satisfied: numpy>=1.23.0 in c:\users\urssa\appdata\local\programs\
Requirement already satisfied: matplotlib>=3.3.0 in c:\users\urssa\appdata\local\progra
Requirement already satisfied: opencv-python>=4.6.0 in c:\users\urssa\appdata\local\pr
Requirement already satisfied: pillow>=7.1.2 in c:\users\urssa\appdata\local\programs\
Requirement already satisfied: pyyaml>=5.3.1 in c:\users\urssa\appdata\local\programs\
Requirement already satisfied: requests>=2.23.0 in c:\users\urssa\appdata\local\progra
Requirement already satisfied: scipy>=1.4.1 in c:\users\urssa\appdata\local\programs\p
Requirement already satisfied: torch>=1.8.0 in c:\users\urssa\appdata\local\programs\p
Collecting torchvision>=0.9.0 (from ultralytics)
  Downloading torchvision-0.22.0-cp311-cp311-win_amd64.whl.metadata (6.3 kB)
Requirement already satisfied: tqdm>=4.64.0 in c:\users\urssa\appdata\local\programs\p
Collecting psutil (from ultralytics)
  Downloading psutil-7.0.0-cp37-abi3-win_amd64.whl.metadata (23 kB)
Collecting py-cpuinfo (from ultralytics)
  Downloading py_cpuinfo-9.0.0-py3-none-any.whl.metadata (794 bytes)
Requirement already satisfied: pandas>=1.1.4 in c:\users\urssa\appdata\local\programs\
Collecting ultralytics-thop>=2.0.0 (from ultralytics)
  Downloading ultralytics_thop-2.0.14-py3-none-any.whl.metadata (9.4 kB)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\urssa\appdata\local\progra
Requirement already satisfied: cycler>=0.10 in c:\users\urssa\appdata\local\programs\p
Requirement already satisfied: fonttools>=4.22.0 in c:\users\urssa\appdata\local\progr
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\urssa\appdata\local\progr
Requirement already satisfied: packaging>=20.0 in c:\users\urssa\appdata\local\program
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\urssa\appdata\local\p
Requirement already satisfied: python-dateutil>=2.7 in c:\users\urssa\appdata\local\pr
Requirement already satisfied: pytz>=2020.1 in c:\users\urssa\appdata\local\programs\p
Requirement already satisfied: tzdata>=2022.1 in c:\users\urssa\appdata\local\programs
Requirement already satisfied: six>=1.5 in c:\users\urssa\appdata\local\programs\pytho
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\urssa\appdata\loca
Requirement already satisfied: idna<4,>=2.5 in c:\users\urssa\appdata\local\programs\p
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\urssa\appdata\local\prog
Requirement already satisfied: certifi>=2017.4.17 in c:\users\urssa\appdata\local\prog
Requirement already satisfied: filelock in c:\users\urssa\appdata\local\programs\pytho
Requirement already satisfied: typing-extensions>=4.10.0 in c:\users\urssa\appdata\loc
Requirement already satisfied: sympy>=1.13.3 in c:\users\urssa\appdata\local\programs\
Requirement already satisfied: networkx in c:\users\urssa\appdata\local\programs\pytho
Requirement already satisfied: jinja2 in c:\users\urssa\appdata\local\programs\python\
Requirement already satisfied: fsspec in c:\users\urssa\appdata\local\programs\python\
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\users\urssa\appdata\local\prog
Requirement already satisfied: colorama in c:\users\urssa\appdata\local\programs\pytho
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\urssa\appdata\local\program
Downloading ultralytics-8.3.139-py3-none-any.whl (1.0 MB)

Downloading torchvision-0.22.0-cp311-cp311-win_amd64.whl (1.7 MB)

Downloading ultralytics_thop-2.0.14-py3-none-any.whl (26 kB)
Downloading psutil-7.0.0-cp37-abi3-win_amd64.whl (244 kB)
Downloading py_cpuinfo-9.0.0-py3-none-any.whl (22 kB)
Installing collected packages: py-cpuinfo, psutil, ultralytics-thop, torchvision, ultr
```

```
Successfully installed psutil-7.0.0 py-cpuinfo-9.0.0 torchvision-0.22.0 ultralytics-8.
```

Listing 3: Install the required libraries for YOLOv8 Model

1. **Inference on Images**
   (a) Run the model on sample images.
   (b) Visualize the bounding boxes with labels and confidence scores.

1.3.1 Input Image

Figure 5: Input Image: Question 3 a

Figure 6: Input Image: Question 3 b

```python
from ultralytics import YOLO
import glob
import pandas as pd
# Load the pre-trained YOLOv8 model
model = YOLO('yolov8n.pt')

# List of image paths
image_paths = glob.glob('q3_2_*.jpeg')

# Run inference and save results to results/yolo_inference/
results = model.predict(
    image_paths,
    save=True,                  # Save images with bounding boxes
    project='results',           # Top-level results folder
    name='yolo_inference',      # Subfolder for this run
    exist_ok=True               # Overwrite if the folder exists
)

for result in results:
    # Show the results in a window (requires GUI support)
    result.show()


# To print detected boxes, labels, and confidence scores:
for result in results:
    boxes = result.boxes
    for box in boxes:
        xyxy = box.xyxy.cpu().numpy()[0]  # bounding box coordinates (x_min, y_min, x_max, y_max)
        conf = box.conf.cpu().numpy()[0]  # confidence score
        cls = int(box.cls.cpu().numpy()[0])  # class index
        label = model.names[cls]  # class label
        print(f"Label: {label}, Confidence: {conf:.2f}, Box: {xyxy}")
```

Listing 4: Install the required libraries for YOLOv8 Model

1.3.2 Output Image

Figure 7: Output Image: Question 3 a

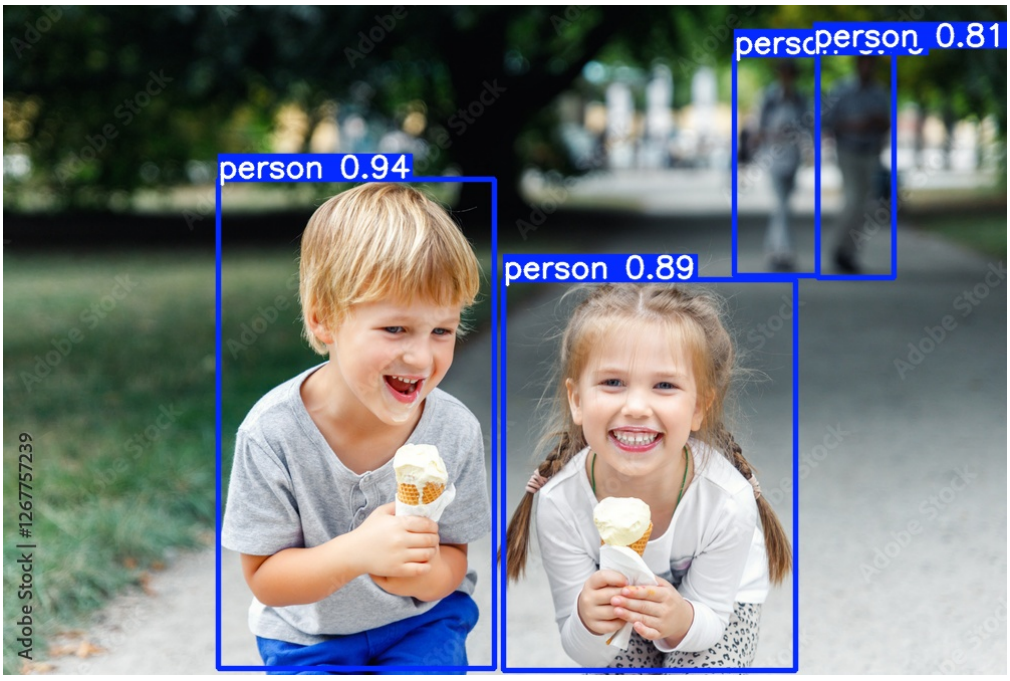

Figure 8: Output Image: Question 3 b

3. Custom Dataset Inference

1. Use a small custom dataset (e.g., traffic signs, animals, faces).
2. Run inference and evaluate the model's performance (precision, recall, mAP).

Automate YOLO Compatible Label

This script automates the creation of YOLO-compatible label files from images by leveraging a pretrained YOLOv8 model to detect objects and generate bounding box annotations. These label files are essential for training custom YOLO models, as they provide the ground truth bounding boxes and class labels in the required format.

Key Points in the Script

Bounding Box Conversion

The bounding box coordinates (xyxy) are converted to YOLO format by calculating the normalized center coordinates, width, and height as follows:

$$x_{\text{center}} = \frac{x_{\min} + x_{\max}}{2 \times \text{image width}}, \quad y_{\text{center}} = \frac{y_{\min} + y_{\max}}{2 \times \text{image height}}$$

$$\text{width} = \frac{x_{\max} - x_{\min}}{\text{image width}}, \quad \text{height} = \frac{y_{\max} - y_{\min}}{\text{image height}}$$

Label File Format

Each line in the label file corresponds to one detected object and includes the class ID and normalized bounding box parameters in the format:

`<class_id> <x_center> <y_center> <width> <height>`

Handling No Detections

Even if no objects are detected, the script creates an empty label file to maintain consistency in the dataset structure.

This approach is commonly used in preparing datasets for YOLO training, especially when you want to generate labels automatically from an existing model before fine-tuning or retraining on your custom data.

```python
from ultralytics import YOLO
import glob
import os

# Load the pre-trained YOLOv8 model
model = YOLO('yolov8n.pt')

# Define your custom dataset folder
image_dir = '/Sanjeev/VNIT_CLASSES/VNIT-AAI-SEM3/custom_dataset_yolo/train_1'
image_paths = glob.glob(os.path.join(image_dir, '*'))

# Run inference
results = model.predict(
    image_paths,
    save=False,  # Don't save images with bounding boxes
    verbose=False  # Suppress verbose output
)

# Process each result
for i, result in enumerate(results):
    image_path = image_paths[i]  # Get the corresponding image path
    label_file_name = os.path.splitext(os.path.basename(image_path))[0] + '.txt'
```

```
    label_file_path = os.path.join(image_dir, label_file_name)

    with open(label_file_path, 'w') as f:
        has_objects = False
        for box in result.boxes:
            xyxy = box.xyxy.cpu().numpy()[0]
            conf = box.conf.cpu().numpy()[0]
            cls = int(box.cls.cpu().numpy()[0])
            label = model.names[cls]

            # Convert bounding box to YOLO format
            x_center = (xyxy[0] + xyxy[2]) / (2 * result.orig_shape[1])
            y_center = (xyxy[1] + xyxy[3]) / (2 * result.orig_shape[0])
            width = (xyxy[2] - xyxy[0]) / result.orig_shape[1]
            height = (xyxy[3] - xyxy[1]) / result.orig_shape[0]

            # Write to label file
            f.write(f"{cls} {x_center:.6f} {y_center:.6f} {width:.6f} {height:.6f}\n")
            has_objects = True

        if not has_objects:
            print(f"No objects detected in {image_path}, created empty label file.")
        else:
            print(f"Label file created for {image_path}")
```

Listing 5: Write YOLO Label and create bounding box txt file

1.3.3 custom_data.yaml

This YAML file is a configuration file used by YOLO (You Only Look Once) object detection models (such as YOLOv5, YOLOv8, etc.) to specify the dataset details for training and validation. It tells the model where to find images and labels, how many classes to expect, and what the class names are.

```
train: /Sanjeev/VNIT_CLASSES/VNIT-AAI-SEM3/custom_dataset_yolo/train_1
val: /Sanjeev/VNIT_CLASSES/VNIT-AAI-SEM3/custom_dataset_yolo/test_1

nc: 20  # number of classes
names: ['bear', 'bird', 'bougainvillea', 'cat', 'cow', 'daisie', 'deer', 'dog', 'dolphin', 'elephant', 'ros
```

Listing 6: Write YOLO Label and create bounding box txt file

1.3.4 programming_assignment_q3_3.py

Purpose This script uses the Ultralytics YOLOv8 model to:

Run object detection inference on a set of images.

Save the inference results (images with bounding boxes).

Print detected object labels, bounding box coordinates, and confidence scores.

Evaluate the model on your custom dataset using the custom_data.yaml configuration and print key performance metrics (precision, recall, mAP).

```
from ultralytics import YOLO
import glob

# 1. Load the pre-trained YOLOv8 model
model = YOLO('yolov8n.pt')

# 2. Train the model on your custom dataset for 50 epochs
model.train(data='custom_data.yaml', epochs=50)

```

```python
# 3. List of image paths for inference
image_paths = glob.glob('custom_dataset/*')

# 4. Run inference and save results to results/3_inference/
results = model.predict(
    image_paths,
    save=True,                # Save images with bounding boxes
    project='results',        # Top-level results folder
    name='3_inference',       # Subfolder for this run
    exist_ok=True             # Overwrite if the folder exists
)

# 5. Print detected boxes, labels, and confidence scores
for result in results:
    for box in result.boxes:
        xyxy = box.xyxy.cpu().numpy()[0]      # bounding box coordinates
        conf = box.conf.cpu().numpy()[0]      # confidence score
        cls = int(box.cls.cpu().numpy()[0])   # class index
        label = model.names[cls]              # class label
        print(f"Label: {label}, Confidence: {conf:.2f}, Box: {xyxy}")

# 6. Run validation on your dataset YAML file
metrics = model.val(data='custom_data.yaml')

# 7. Print evaluation metrics safely
try:
    print(f"2. Recall: {metrics.box.mr():.4f}")
    print(f"2. mAP@0.5 (map50): {metrics.box.map50():.4f}")
    print(f"4. mAP@0.5:0.95 (map): {metrics.box.map():.4f}")
    print(f"1. Precision: {metrics.box.mp():.4f}")
except Exception as e:
    print("Metrics could not be calculated. Check if your validation set has labels.")
    print(f"Error: {e}")
```

Listing 7: Object detection inference

```
C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\subjects\com\anamika_singh\YOLO1_Assignment>python programming_assign
New https://pypi.org/project/ultralytics/8.3.141 available  Update with 'pip install -U ultralytics'
Ultralytics 8.3.139  Python-3.11.9 torch-2.7.0+cpu CPU (Intel Core(TM) i7-8650U 1.90GHz)
engine\trainer: agnostic_nms=False, amp=True, augment=False, auto_augment=randaugment, batch=16, bgr=0.0, b
Overriding model.yaml nc=80 with nc=20

                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
```

```
 22        [15, 18, 21]  1    755212  ultralytics.nn.modules.head.Detect           [20, [64, 128, 256]]
Model summary: 129 layers, 3,014,748 parameters, 3,014,732 gradients, 8.2 GFLOPs

Transferred 319/355 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
train: Fast image access  (ping: 0.40.0 ms, read: 3.23.8 MB/s, size: 38.4 KB)
train: Scanning C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1... 98 images, 16 backgrou
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Bear_17.jpeg: ignoring corrupt ima
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Bear_21_1.jpg: ignoring corrupt im
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Bear_3_2.jpg: ignoring corrupt ima
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Bear_6.jpg: ignoring corrupt image
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Bird_4_3.jpg: ignoring corrupt ima
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Bird_7_2.jpg: ignoring corrupt ima
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Elephant_23_4.jpg: ignoring corrup
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Elephant_28_2.jpg: ignoring corrup
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Giraffe_21.jpeg: ignoring corrupt
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Giraffe_21_1.jpg: ignoring corrupt
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Giraffe_28_4.jpg: ignoring corrupt
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Giraffe_6_2.jpg: ignoring corrupt
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Lion_6.jpg: ignoring corrupt image
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Panda_18.jpeg: ignoring corrupt im
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Panda_20_1.jpg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Panda_23.jpeg: ignoring corrupt im
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Panda_25_2.jpg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Panda_3_2.jpg: ignoring corrupt im
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Tiger_27_4.jpg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Tiger_8.jpg: ignoring corrupt imag
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Zebra_28_1.jpg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Zebra_30_3.jpg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\Zebra_3_1.jpg: ignoring corrupt im
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\bear_1.jpg: ignoring corrupt image
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\bougainvillea_00004.jpg: ignoring
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\cat_189.jpg: ignoring corrupt imag
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\daisies_00022.jpg: ignoring corrup
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\dog_109.jpg: ignoring corrupt imag
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\dog_260.jpg: ignoring corrupt imag
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\gardenias_00026.jpg: ignoring corr
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\gardenias_00051.jpg: ignoring corr
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\hydrangeas_00013.jpg: ignoring cor
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\hydrangeas_00026.jpg: ignoring cor
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\img_p1_11.jpeg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\img_p1_111.jpeg: ignoring corrupt
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\img_p1_20appapp.jpeg: ignoring cor
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\img_p1_72.jpeg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\img_p1_92.jpeg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\img_p2_100.jpeg: ignoring corrupt
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\img_p2_69.jpeg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\img_p3_46.jpeg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\img_p3_58.jpeg: ignoring corrupt i
train: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1\img_p3_70appapp.jpeg: ignoring cor
train: New cache created: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\train_1.cache
C:\Users\urssa\AppData\Local\Programs\Python\Python311\Lib\site-packages\torch\utils\data\dataloader.py:665
  warnings.warn(warn_msg)
val: Fast image access  (ping: 0.40.0 ms, read: 7.36.3 MB/s, size: 71.8 KB)
val: Scanning C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1.cache... 28 images, 9 backgr
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\Bear_10.jpg: ignoring corrupt image/l
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\Bird_15.jpeg: ignoring corrupt image/
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\Zebra_20.jpeg: ignoring corrupt image
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\dog_442.jpg: ignoring corrupt image/l
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\hydrangeas_00070.jpg: ignoring corrup
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\img_p1_25.jpeg: ignoring corrupt imag
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\img_p3_124.jpeg: ignoring corrupt ima
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\img_p3_50.jpeg: ignoring corrupt imag
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\lilies_00069.jpg: ignoring corrupt im
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\peonies_00006.jpg: ignoring corrupt i
```

```
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\peonies_00077.jpg: ignoring corrupt i
C:\Users\urssa\AppData\Local\Programs\Python\Python311\Lib\site-packages\torch\utils\data\dataloader.py:665
  warnings.warn(warn_msg)
Plotting labels to runs\detect\train\labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer
optimizer: AdamW(lr=0.000417, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs\detect\train
Starting training for 50 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/50         0G     0.7892      4.191      1.341         11        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9          0          0          0          0

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/50         0G     0.7977      4.271      1.364         16        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9          0          0          0          0

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       3/50         0G     0.7394      4.112      1.286         20        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9          0          0          0          0

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/50         0G     0.6739      3.966      1.267         19        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9          0          0          0          0

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       5/50         0G     0.6346      3.782      1.219         15        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9          0          0          0          0

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       6/50         0G     0.6584       3.85      1.238         16        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9          0          0          0          0

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       7/50         0G     0.6517      3.715      1.281         16        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9        0.1     0.0667        0.1     0.0801

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/50         0G     0.6487      3.555      1.215         19        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.111      0.233      0.209      0.148

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       9/50         0G     0.6167      3.436      1.181         22        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9     0.0461      0.233      0.202       0.15

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/50         0G     0.6602      3.328      1.129         16        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9     0.0565        0.5      0.236      0.177

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      11/50         0G     0.6446      3.091       1.19         18        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9       0.05        0.5      0.186      0.147
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      12/50         0G     0.5791      2.995      1.151         16        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.124        0.8      0.309      0.229

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      13/50         0G      0.682      2.936      1.266         17        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9     0.0938          1      0.392       0.29

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      14/50         0G     0.7101      2.981      1.256          8        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9     0.0833          1       0.45       0.35

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      15/50         0G     0.6377      2.794      1.261         17        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9     0.0518          1       0.54      0.417

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      16/50         0G     0.6553      2.485      1.209         20        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.509      0.433      0.522      0.409

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      17/50         0G     0.6873      2.591      1.261         24        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.638      0.297      0.472      0.366

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      18/50         0G     0.6339      2.472      1.178         24        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.639      0.277      0.465      0.353

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/50         0G     0.6937      2.544      1.276         19        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.861      0.267      0.427      0.308

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/50         0G     0.6271      2.389      1.198         17        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.878      0.267      0.428      0.322

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      21/50         0G     0.6518      2.245      1.186         12        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.973      0.323      0.487      0.351

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      22/50         0G     0.6789      2.044      1.264         16        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.961      0.333       0.49      0.359

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      23/50         0G      0.655      2.201      1.206         14        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.934      0.333       0.47      0.343

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      24/50         0G     0.6205       2.23      1.185         16        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.915      0.333      0.464      0.328
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      25/50         0G     0.5582      2.058      1.185         17        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.656      0.333      0.483      0.348

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      26/50         0G     0.6237       1.98       1.16         14        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.649       0.38       0.53      0.367

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      27/50         0G     0.5437      1.979      1.124         19        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.791      0.433      0.607      0.402

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      28/50         0G     0.5789      1.829      1.127         17        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.832      0.433      0.604      0.426

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      29/50         0G     0.6148      1.904      1.174         21        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.769      0.433      0.598      0.421

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      30/50         0G     0.5341      1.741      1.112         22        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.658      0.433      0.693      0.458

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      31/50         0G     0.7069      2.187      1.247         12        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.576      0.433      0.599      0.413

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      32/50         0G      0.571      1.905      1.149         15        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.632      0.433      0.596      0.409

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      33/50         0G     0.5401      1.713      1.118         17        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.733      0.433      0.696      0.475

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      34/50         0G     0.5495      1.796      1.091         19        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.847      0.432      0.695      0.467

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      35/50         0G     0.5194      1.648      1.098         17        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.867      0.421      0.509      0.369

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      36/50         0G     0.5542      1.734       1.15         25        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.875      0.411       0.49      0.362

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      37/50         0G     0.5231       1.67      1.084         15        640: 100%|
                  Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                    all         17          9      0.803      0.367      0.449       0.35
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      38/50         0G     0.5477      1.696      1.103         15        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.803      0.367      0.446       0.35

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      39/50         0G     0.5714      1.886      1.161         12        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.803      0.367      0.444      0.356

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      40/50         0G     0.5699      1.708      1.193         11        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.802      0.367      0.502      0.383
Closing dataloader mosaic
C:\Users\urssa\AppData\Local\Programs\Python\Python311\Lib\site-packages\torch\utils\data\dataloader.py:665
  warnings.warn(warn_msg)

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      41/50         0G     0.4338      2.588      1.125          3        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.808      0.367      0.593      0.427

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      42/50         0G     0.4441      2.404       1.01          9        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9       0.79      0.367      0.461      0.362

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      43/50         0G      0.473      2.291      1.136         11        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.768      0.367      0.476       0.37

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      44/50         0G     0.3864      2.104      1.001          9        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.747      0.367      0.505      0.385

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      45/50         0G     0.4293      2.288     0.9662          8        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.734      0.366      0.502       0.38

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      46/50         0G     0.4818      2.316      1.158          7        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.726      0.359      0.504      0.386

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      47/50         0G     0.4392      2.486      1.087          6        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.721      0.353      0.498      0.381

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      48/50         0G     0.4401      2.375      1.169          6        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.735       0.36      0.504      0.388

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      49/50         0G     0.4808      2.273      1.126          6        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.749      0.362      0.511      0.391

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
      50/50         0G     0.4778      2.336      1.149          7        640: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.758      0.363      0.478      0.373

50 epochs completed in 0.594 hours.
Optimizer stripped from runs\detect\train\weights\last.pt, 6.3MB
Optimizer stripped from runs\detect\train\weights\best.pt, 6.3MB

Validating runs\detect\train\weights\best.pt...
Ultralytics 8.3.139  Python-3.11.9 torch-2.7.0+cpu CPU (Intel Core(TM) i7-8650U 1.90GHz)
Model summary (fused): 72 layers, 3,009,548 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.736      0.433      0.696      0.475
                  bear          1          1          1          0      0.995      0.497
                 apple          2          2          1          0       0.19      0.145
                tomato          2          3      0.726      0.667       0.72      0.579
              kangaroo          2          2      0.375        0.5      0.578      0.357
                  lion          1          1       0.58          1      0.995      0.796
Speed: 4.8ms preprocess, 189.9ms inference, 0.0ms loss, 2.8ms postprocess per image
Results saved to runs\detect\train

0: 640x640 (no detections), 143.2ms
1: 640x640 (no detections), 143.2ms
2: 640x640 (no detections), 143.2ms
3: 640x640 (no detections), 143.2ms
4: 640x640 (no detections), 143.2ms
5: 640x640 (no detections), 143.2ms
6: 640x640 2 tomatos, 143.2ms
7: 640x640 (no detections), 143.2ms
Speed: 6.6ms preprocess, 143.2ms inference, 2.8ms postprocess per image at shape (1, 3, 640, 640)
Results saved to results\3_inference
Label: tomato, Confidence: 0.87, Box: [     5.0869      21.313      1482.3        1192]
Label: tomato, Confidence: 0.34, Box: [     46.278      25.329        1064      1179.2]
Ultralytics 8.3.139  Python-3.11.9 torch-2.7.0+cpu CPU (Intel Core(TM) i7-8650U 1.90GHz)
val: Fast image access  (ping: 0.30.0 ms, read: 63.365.1 MB/s, size: 68.3 KB)
val: Scanning C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1.cache... 28 images, 9 backgr
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\Bear_10.jpg: ignoring corrupt image/l
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\Bird_15.jpeg: ignoring corrupt image/
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\Zebra_20.jpeg: ignoring corrupt image
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\dog_442.jpg: ignoring corrupt image/l
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\hydrangeas_00070.jpg: ignoring corrup
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\img_p1_25.jpeg: ignoring corrupt imag
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\img_p3_124.jpeg: ignoring corrupt ima
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\img_p3_50.jpeg: ignoring corrupt imag
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\lilies_00069.jpg: ignoring corrupt im
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\peonies_00006.jpg: ignoring corrupt i
val: C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\custom_dataset_yolo\test_1\peonies_00077.jpg: ignoring corrupt i
C:\Users\urssa\AppData\Local\Programs\Python\Python311\Lib\site-packages\torch\utils\data\dataloader.py:665
  warnings.warn(warn_msg)
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all         17          9      0.736      0.433      0.696      0.475
                  bear          1          1          1          0      0.995      0.497
                 apple          2          2          1          0       0.19      0.145
                tomato          2          3      0.726      0.667       0.72      0.579
              kangaroo          2          2      0.375        0.5      0.578      0.357
                  lion          1          1       0.58          1      0.995      0.796
Speed: 8.3ms preprocess, 336.5ms inference, 0.0ms loss, 2.6ms postprocess per image
Results saved to runs\detect\train2
Metrics could not be calculated. Check if your validation set has labels.
Error: 'numpy.float64' object is not callable
```

Listing 8: Object detection inference execution output

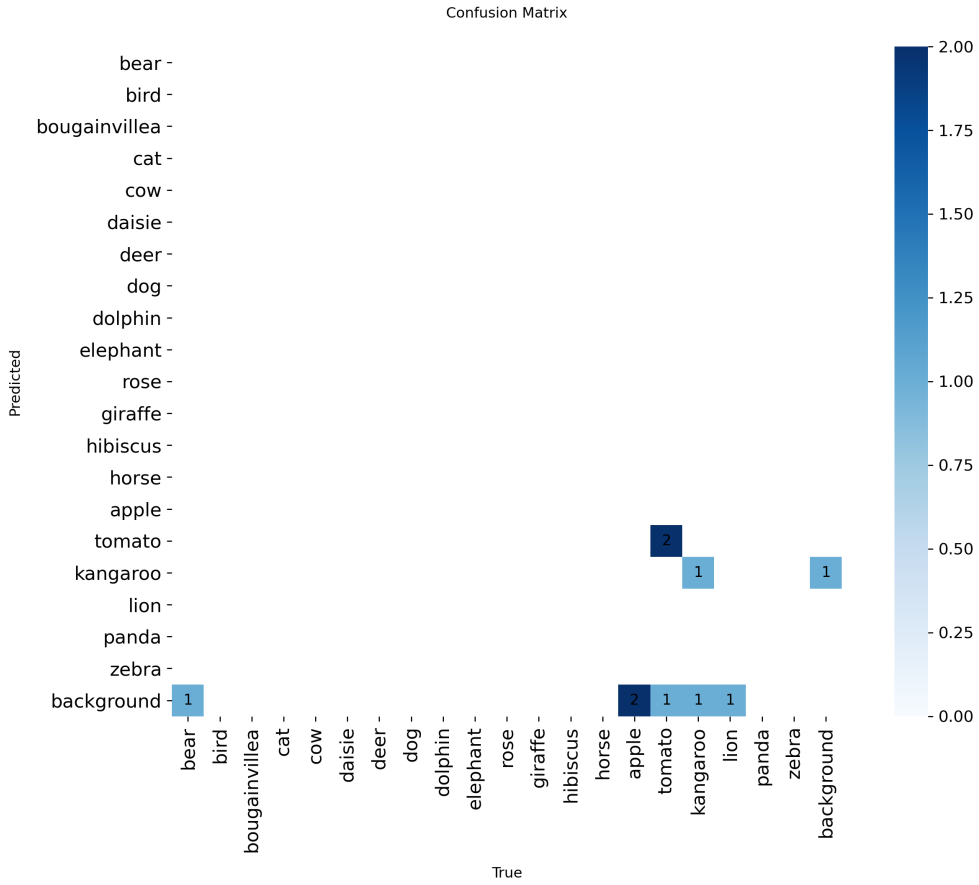

Figure 9: Confusion Matrix

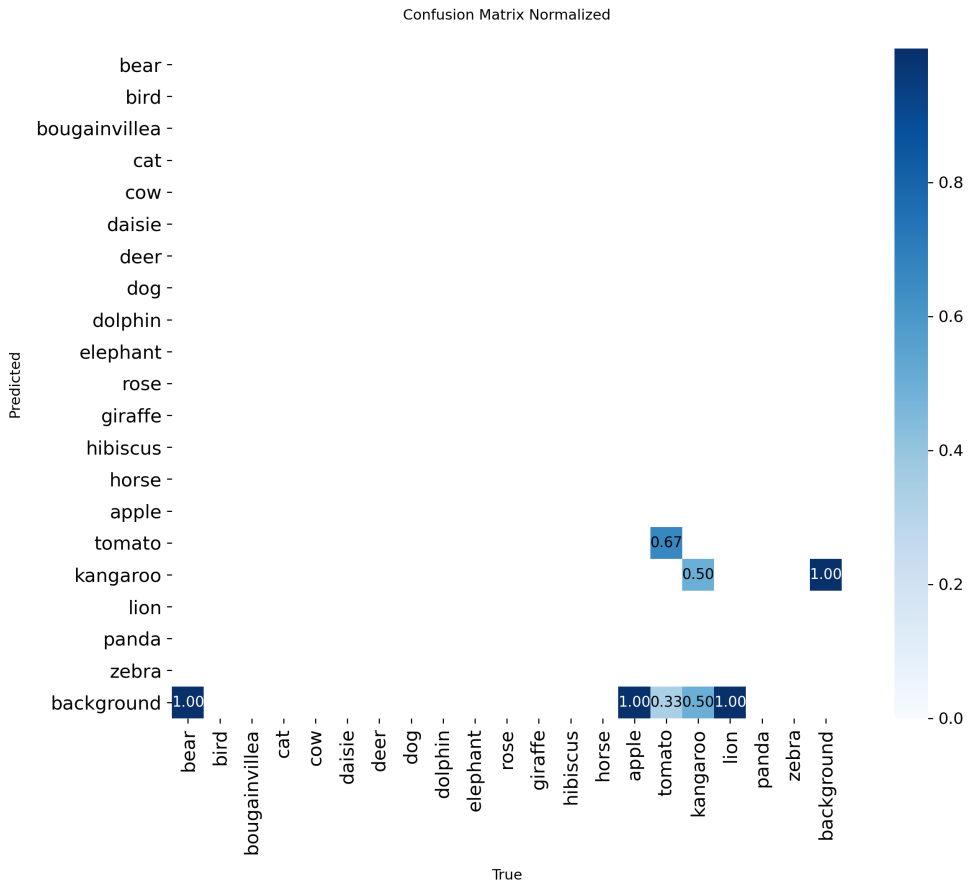

Figure 10: Normalized Confusion Matrix

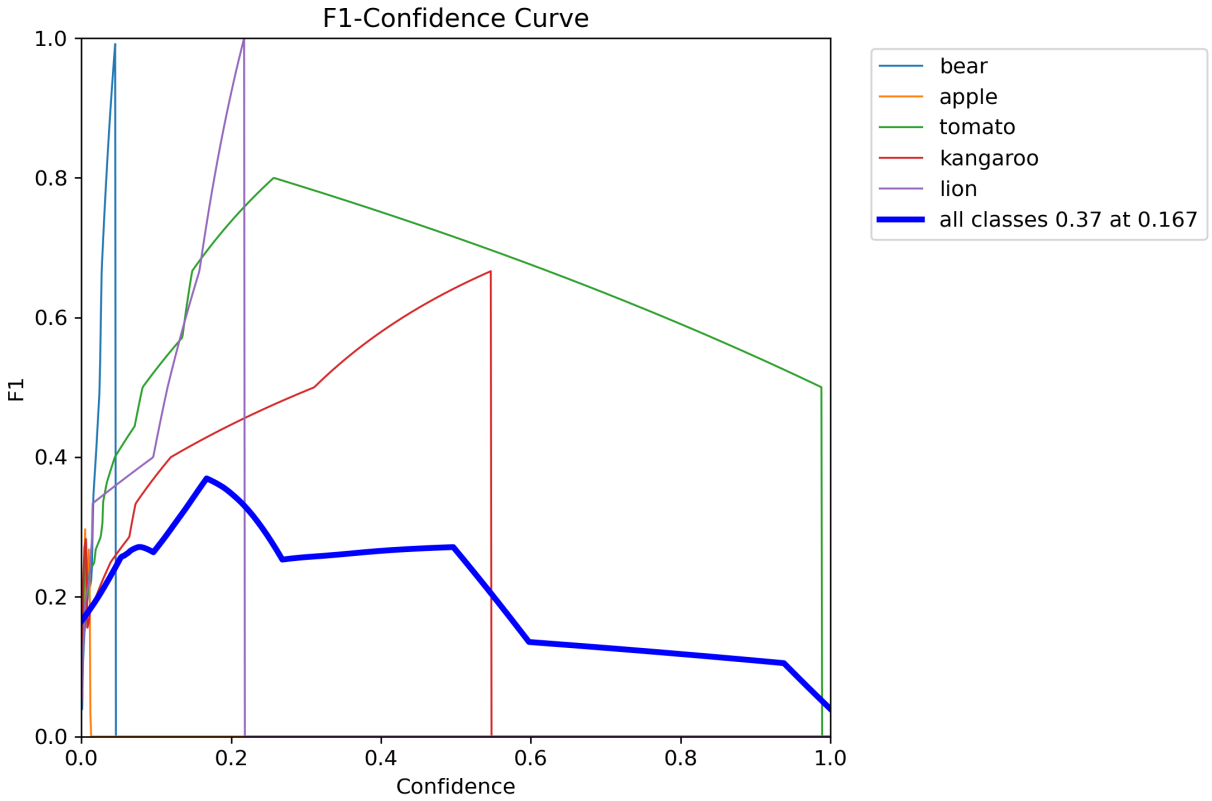

Figure 11: F1 Score Curve

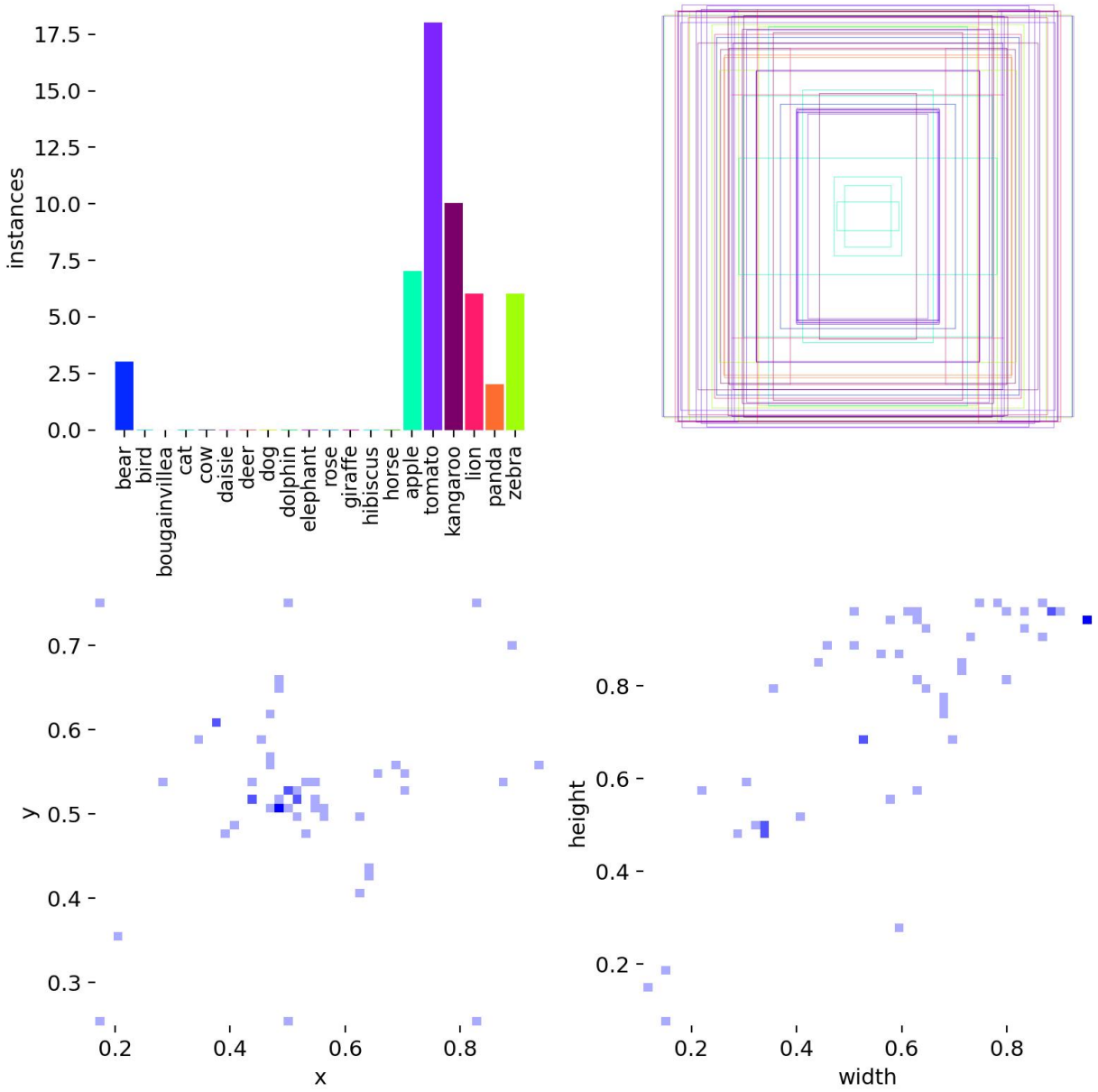

Figure 12: Labels Image

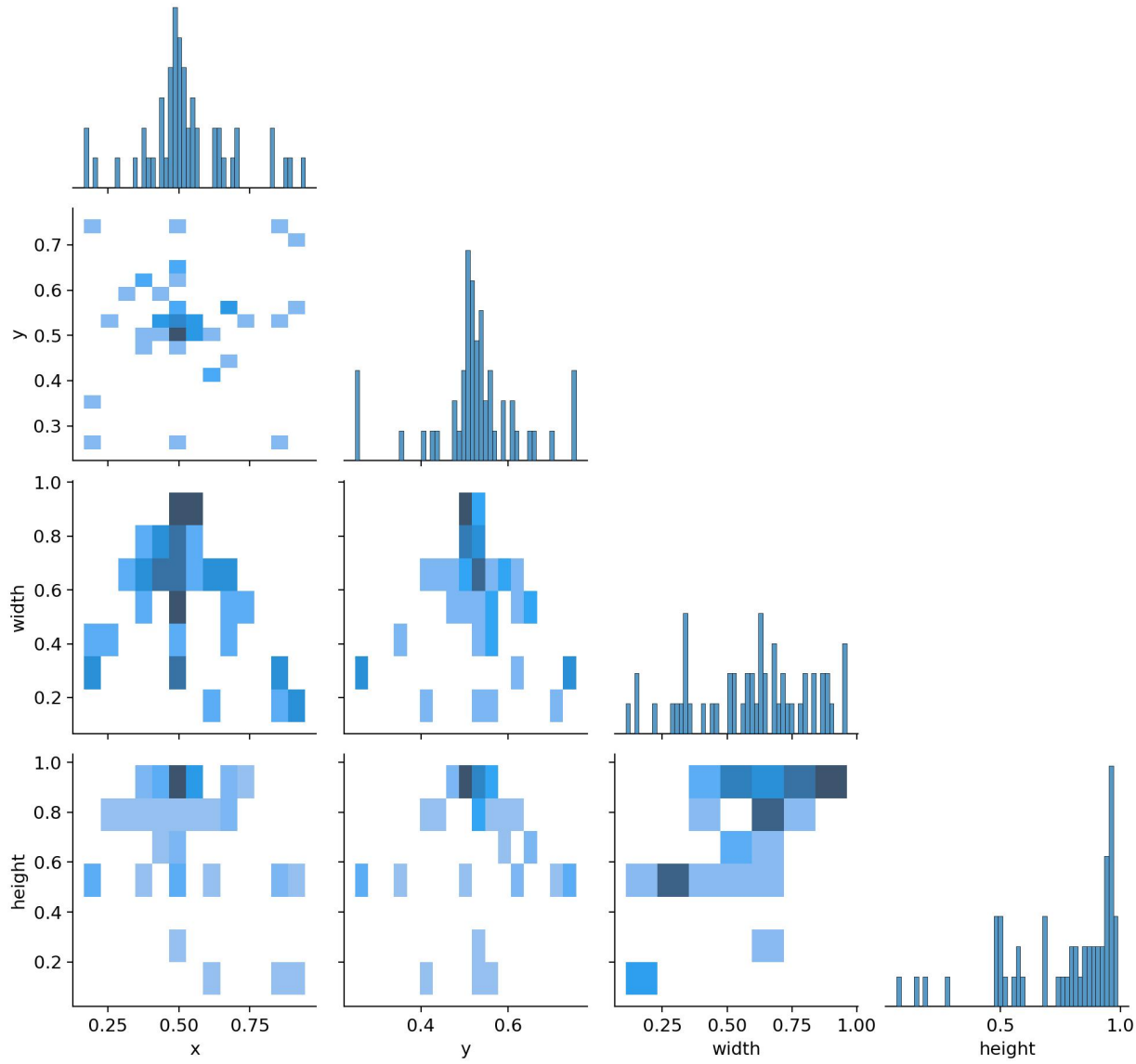

Figure 13: Labels Correlogram

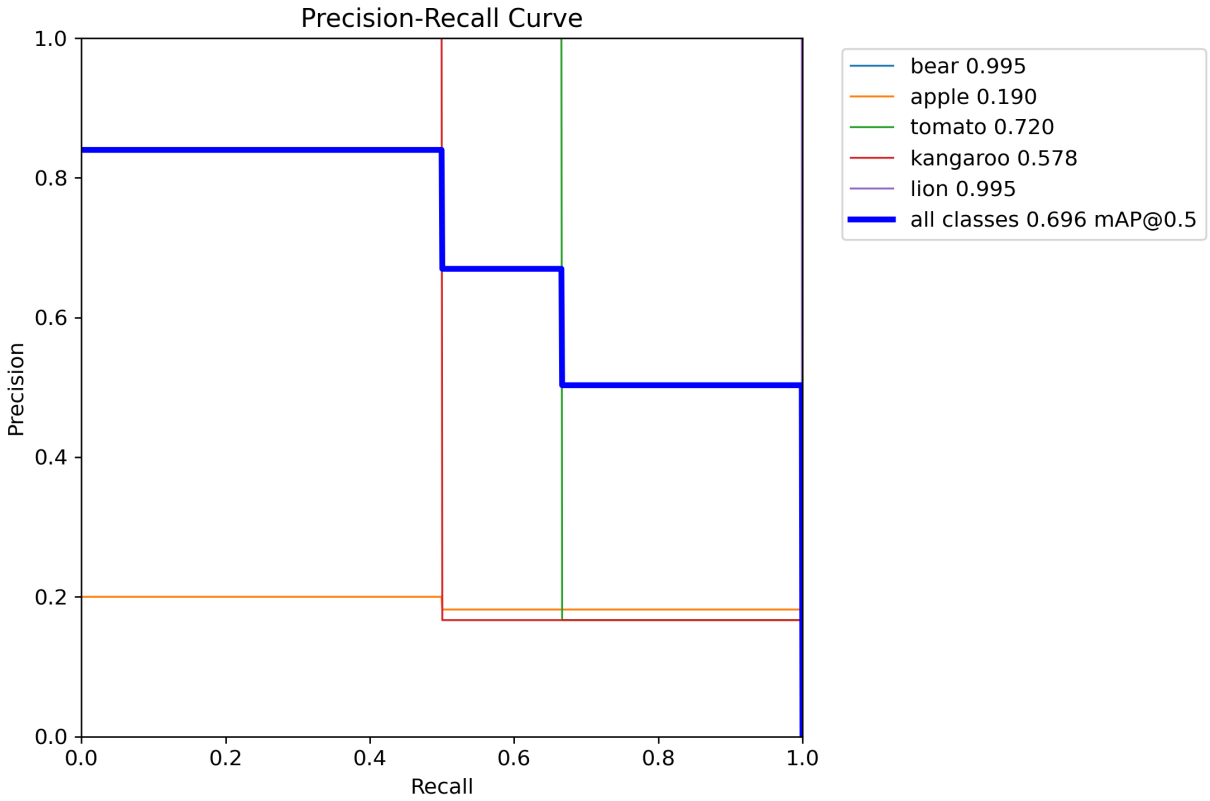

Figure 14: Precision-Recall Curve

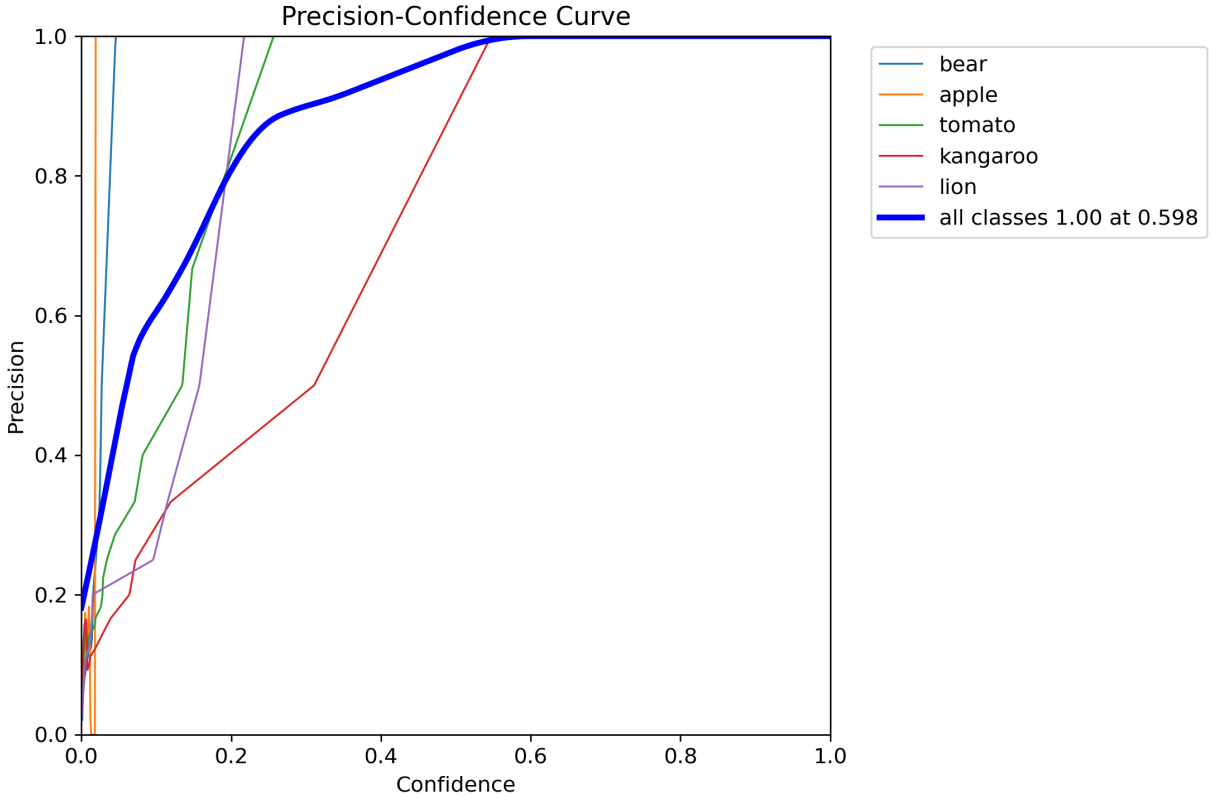

Figure 15: Precision Curve

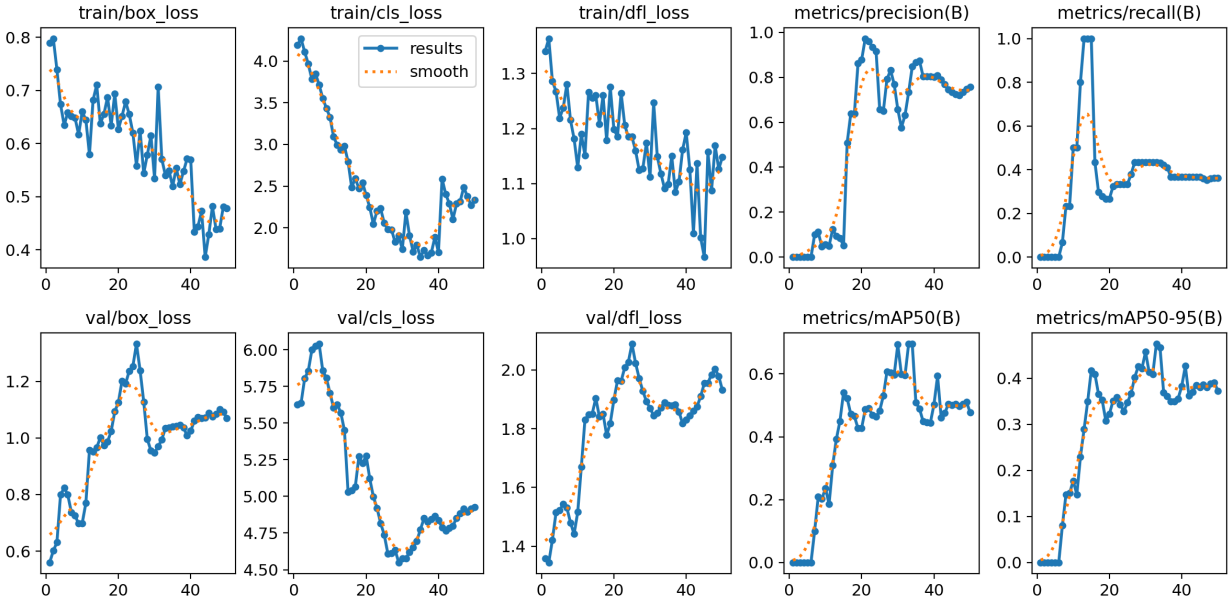

Figure 16: Results Image

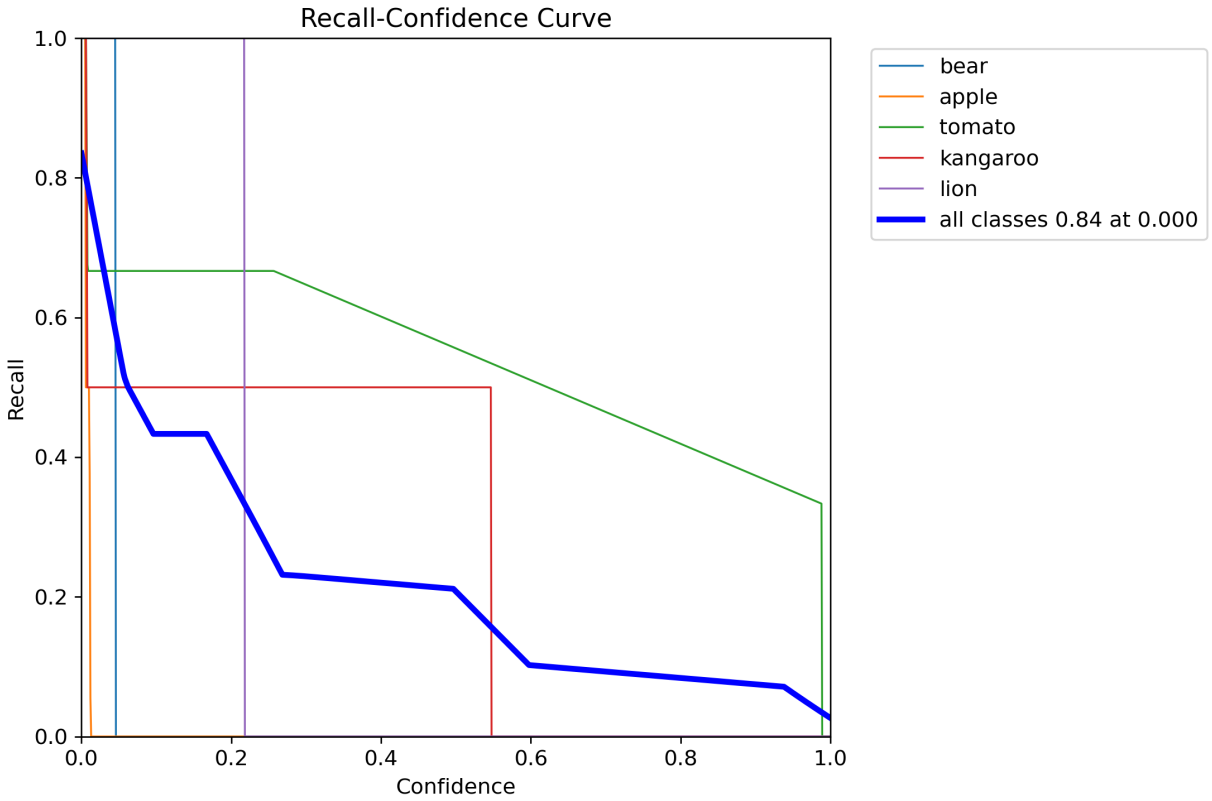

Figure 17: Recall Curve

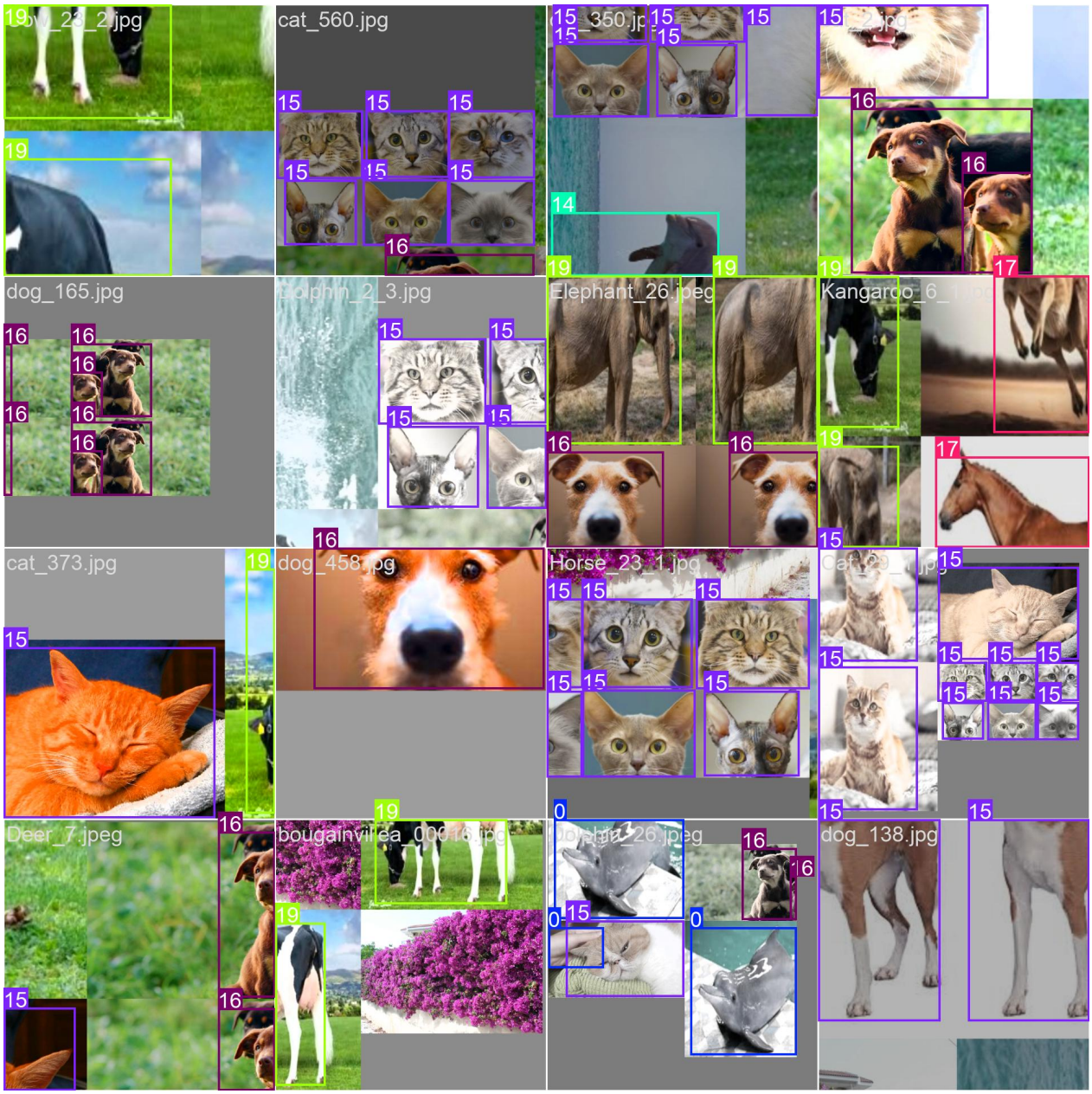

Figure 18: Training Batch 0

Figure 19: Training Batch 1

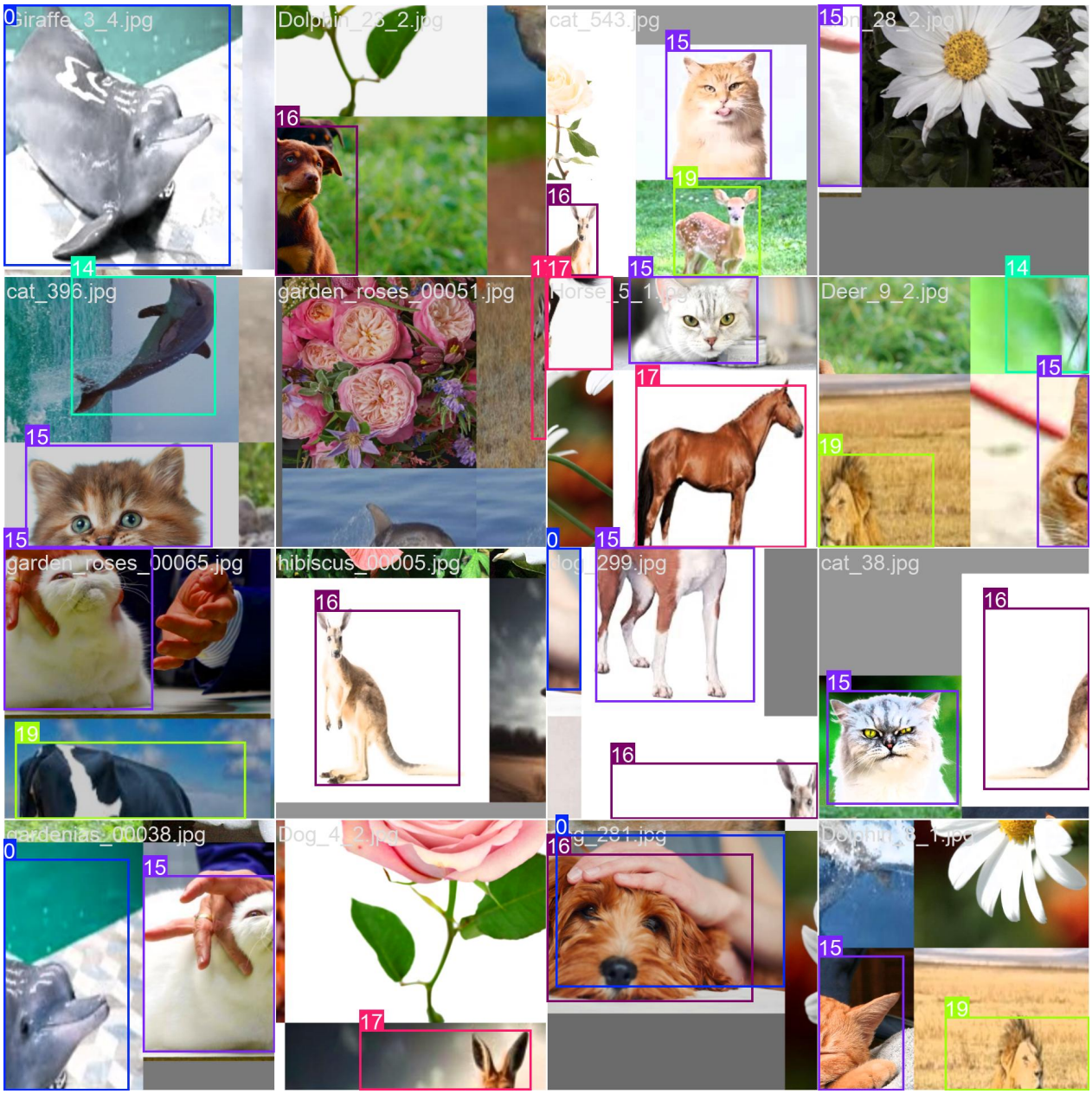

Figure 20: Training Batch 2

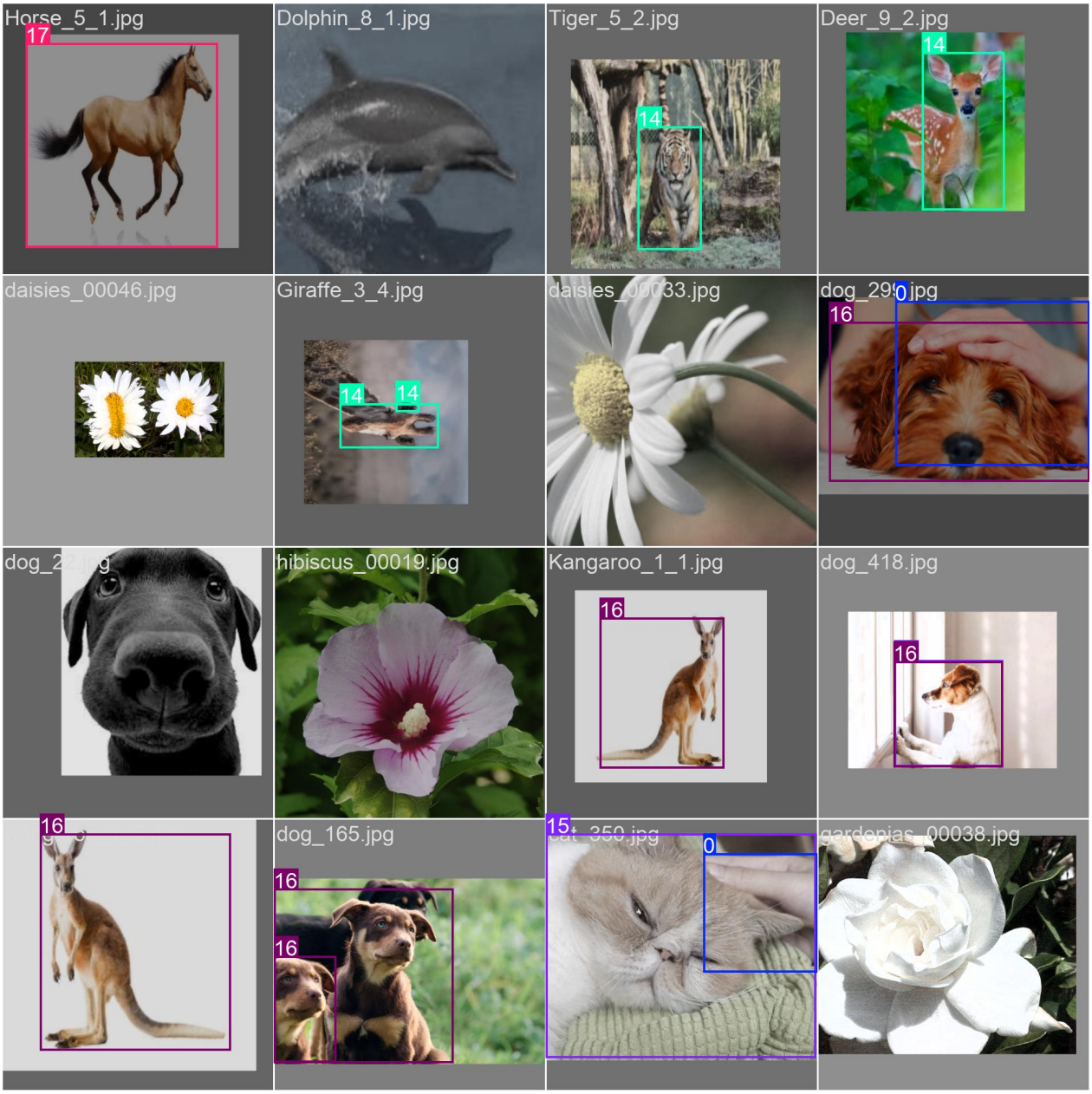

Figure 21: Training Batch 160

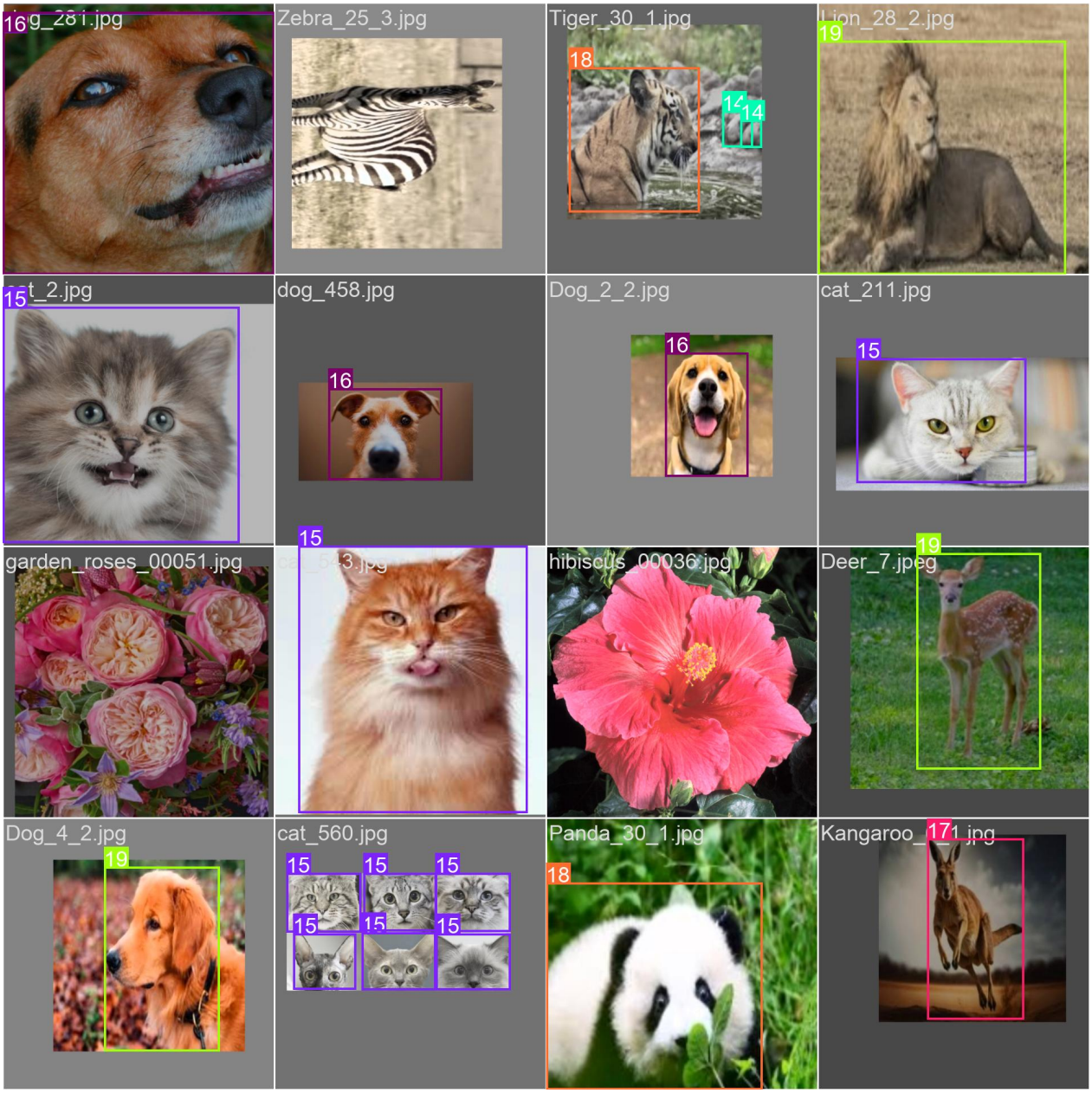

Figure 22: Training Batch 161

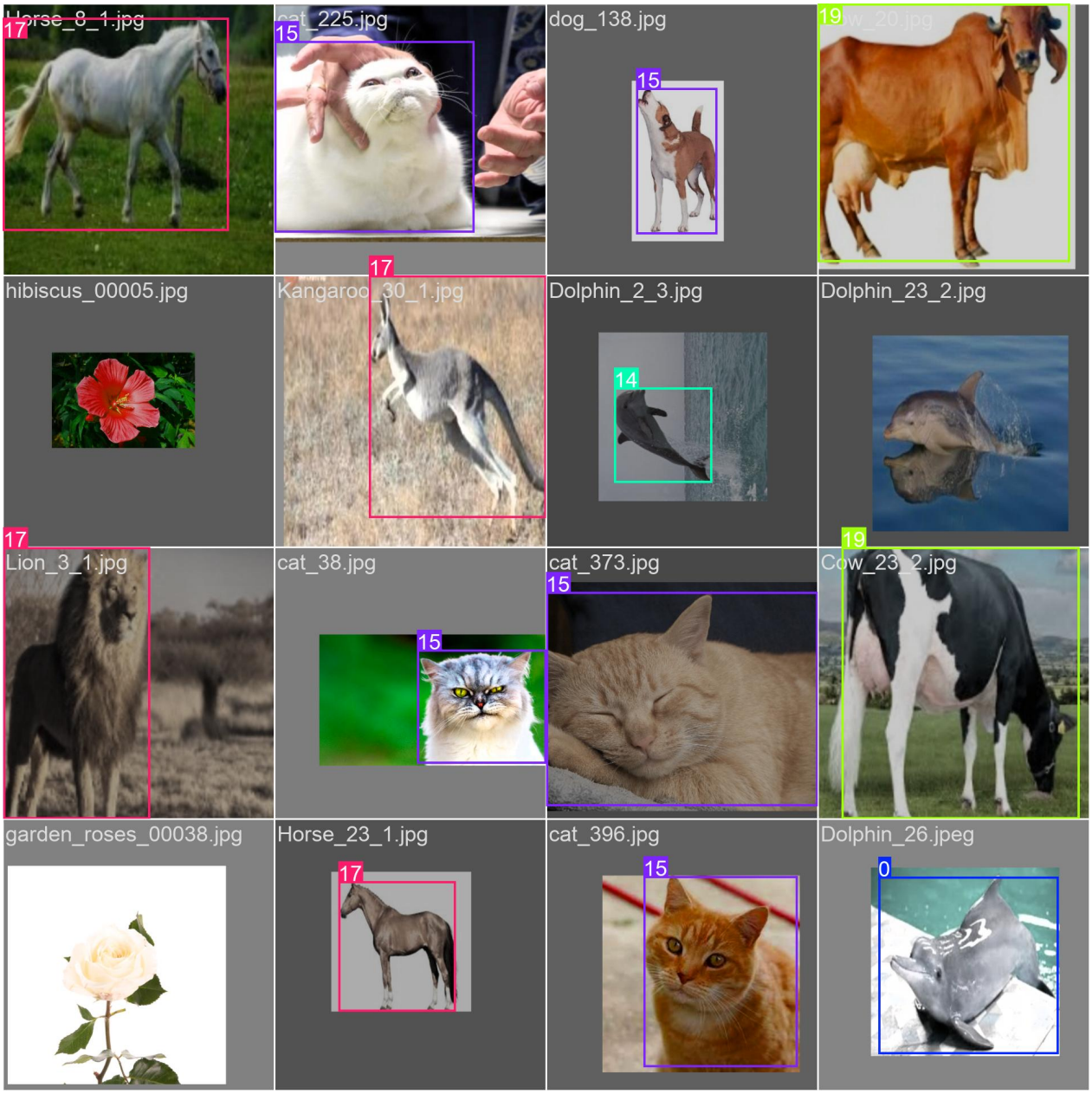

Figure 23: Training Batch 162

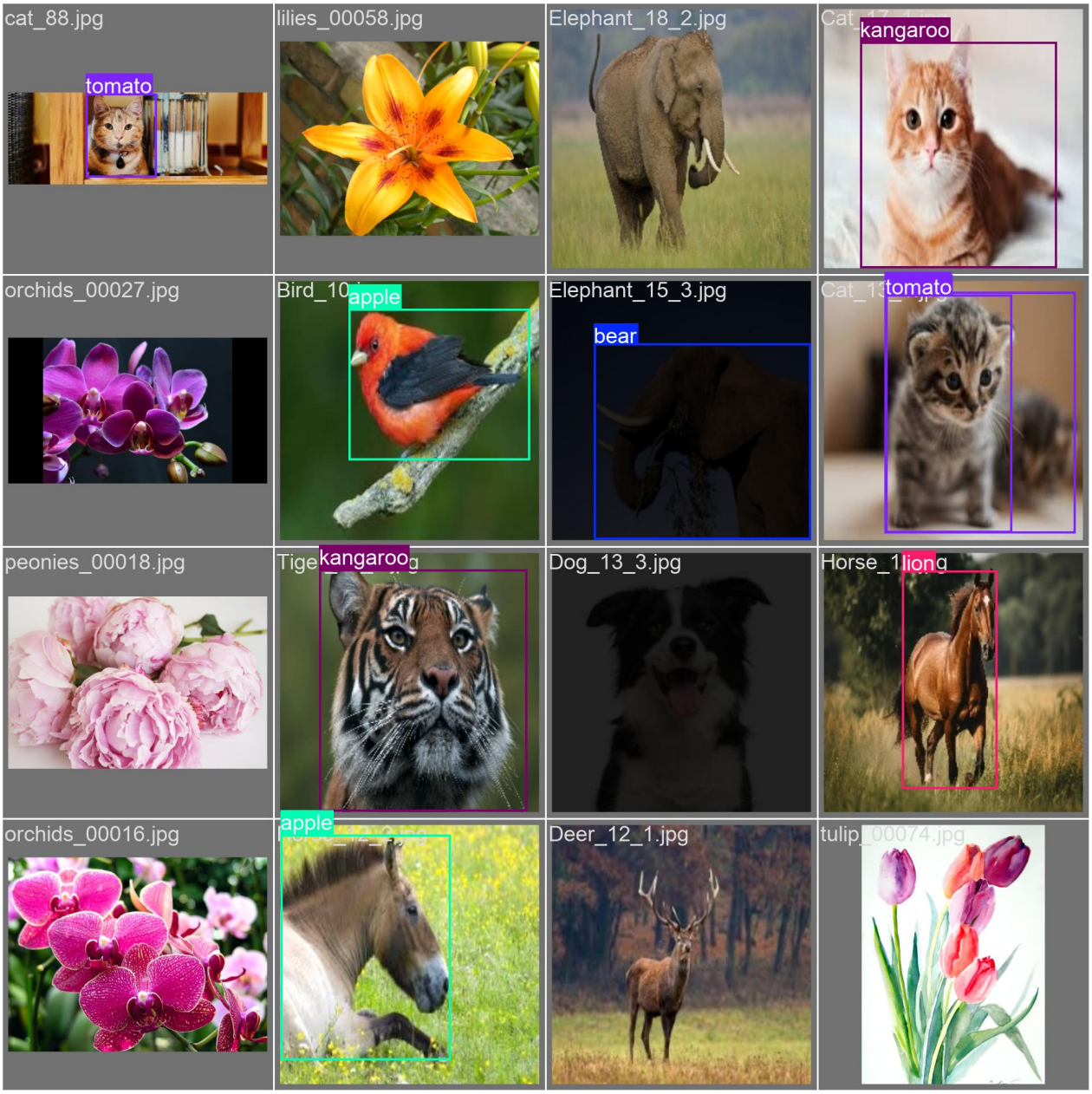

Figure 24: Validation Batch 0 Labels

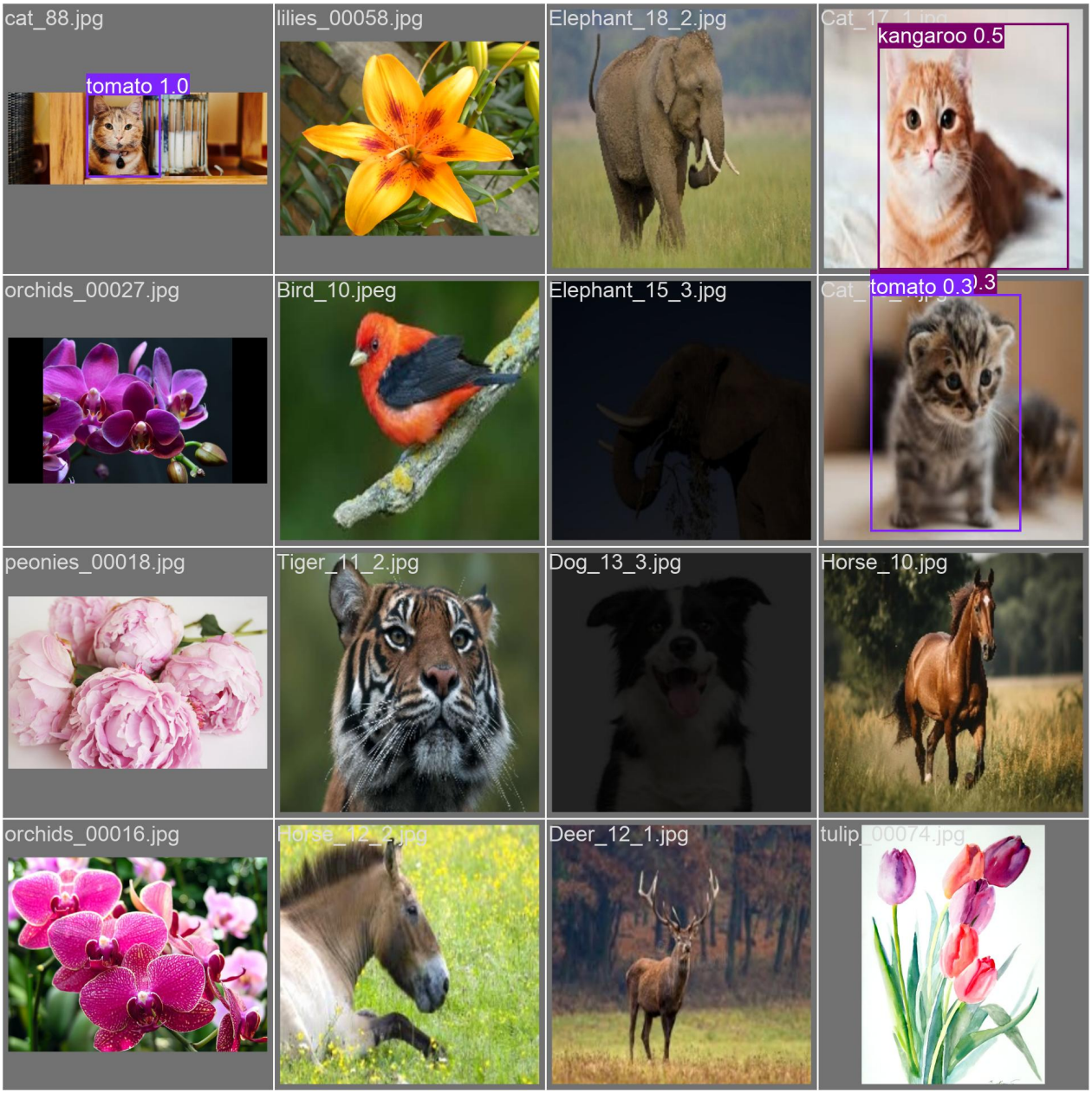

Figure 25: Validation Batch 0 Predictions

2 Theory Assignment

2.1 Question 1

How does YOLOv1 process input images? Describe the architecture of YOLOv1.

Answer

How YOLOv1 Processes Input Images and Its Architecture

YOLOv1 is a fast and clever way for a computer to find objects in pictures. Instead of looking at parts of the image one by one, YOLOv1 looks at the whole image just once and tries to find all the objects at the same time.

How It Works

First, YOLOv1 takes any input image and resizes it to a standard size so it can process it easily. Then, it divides this image into a grid made up of small squares. Each square is responsible for checking if there is an object inside it.

For each square, the model guesses a few boxes where objects might be, and also how confident it is about those guesses. It also predicts what kind of object is inside each box, like a dog, a car, or a person.

After making all these guesses for every square, YOLOv1 filters out overlapping boxes and keeps the best ones. This way, it quickly finds and labels objects in the image in just one go.

The Architecture

YOLOv1 uses a deep neural network made up of many layers that help it understand the image:

- It has a series of layers called convolutional layers that look for important features like edges and shapes.
- These layers shrink the image step-by-step while keeping the important details.
- After these, there are a couple of fully connected layers that take all the information from the previous layers and make the final predictions about where objects are and what they are.
- The network uses special functions that help it learn complex patterns from images.

Because the network looks at the entire image at once and predicts all objects in one step, YOLOv1 is very fast fast enough to work in real-time applications like video surveillance or self-driving cars.

In Simple Terms

- YOLOv1 splits the image into a grid.
- Each grid cell guesses if there is an object inside it.
- It predicts boxes around objects and what those objects are.
- It does all this in a single pass, making it very fast.

This approach makes YOLOv1 different from older methods that needed to look at the image many times or in pieces. Its like glancing at a photo once and instantly knowing whats in it and where.

Detailed Architecture of YOLOv1

YOLOv1 is a single neural network designed to detect objects in images quickly and accurately by looking at the whole image at once. Its architecture is inspired by a well-known image classification network called GoogLeNet, but it has been adapted for the task of object detection.

Key Components of the Network

- **Input:** The network takes an image resized to 448 by 448 pixels as input, ensuring consistent processing regardless of the original image size.
- **Convolutional Layers:** YOLOv1 has **24 convolutional layers** that extract important features from the image. These layers scan the image for patterns such as edges, textures, and shapes that help identify objects.
- **Max-Pooling Layers:** Interspersed between convolutional layers are **max-pooling layers**, which reduce the spatial size of the feature maps. This helps the network focus on the most important information while reducing computational load.
- **1x1 and 3x3 Convolutions:** The network cleverly alternates between small 1x1 convolutions, which reduce the number of channels (making the network more efficient), and 3x3 convolutions, which capture spatial information.
- **Activation Functions:** Most layers use a **Leaky ReLU** activation function, which helps the network learn complex patterns by allowing small gradients when neurons are not active.
- **Fully Connected Layers:** After the convolutional layers, there are **2 fully connected layers** that take all the extracted features and make the final predictions. These layers output a fixed-size tensor containing bounding box coordinates, confidence scores, and class probabilities for each grid cell.
- **Output Tensor:** The output is structured as a grid (usually 7x7). Each grid cell predicts:
  - Multiple bounding boxes (usually 2 per cell), each with coordinates and a confidence score indicating how likely it is that the box contains an object.
  - Class probabilities indicating what type of object might be inside the bounding boxes.
- **Global Reasoning:** Because the fully connected layers see the entire images features at once, the network can reason about the global context understanding how objects relate to each other across the whole image.

Training Details and Techniques

- The network was initially pretrained on a large image classification dataset (ImageNet) to learn general visual features.
- It was then fine-tuned on object detection datasets like PASCAL VOC, where it learned to predict bounding boxes and classes.
- Techniques like **dropout** and **data augmentation** were used to prevent overfitting, helping the model generalize better to new images.
- The loss function combines errors from bounding box coordinates, confidence scores, and classification, training the network end-to-end as a single regression problem.

Why This Architecture Matters - The combination of convolutional layers and fully connected layers allows YOLOv1 to detect multiple objects in an image simultaneously and quickly. - The use of 1x1 convolutions reduces the number of parameters and speeds up computation without sacrificing accuracy. - Processing the entire image in one pass (instead of scanning parts individually) makes YOLOv1 much faster than previous object detection methods. - The networks design balances accuracy and speed, enabling real-time object detection on standard hardware.

Limitations to Keep in Mind - YOLOv1 can struggle with detecting very small objects because the grid cells are relatively large. - It can detect only a limited number of objects per image due to the fixed grid and bounding box predictions. - Nearby or overlapping objects can be hard to separate because each grid cell predicts only a few bounding boxes.

—

In summary, YOLOv1s architecture is a smart blend of deep convolutional layers and fully connected layers that work together to quickly and accurately detect objects across the entire image in one go. This design was groundbreaking at the time and laid the foundation for many improved versions of YOLO that followed.

2.2 Question 2

How has the architecture evolved in YOLO different versions over the last decade?

Answer

How YOLO Architecture Has Evolved Over Time

Over the past ten years, YOLOs design has changed a lot. Each new version brought improvements that help it find objects in images faster, more accurately, and in a wider variety of situations. Heres a simple overview of how YOLOs design has improved:

YOLOv1: The Beginning YOLOv1 introduced the idea of detecting objects by looking at the whole image once. It divided the image into a grid and predicted bounding boxes and object types for each grid cell all at once. It was fast and straightforward but had trouble with small objects and precise localization.

YOLOv2: Introducing Anchor Boxes YOLOv2 added anchor boxes, which helped it better detect objects of different shapes and sizes. It used a new backbone network called Darknet-19, which improved feature extraction. It also included batch normalization for faster, more stable training and trained on images of varying sizes to be more flexible.

YOLOv3: Multi-Scale Detection YOLOv3 upgraded to a deeper backbone called Darknet-53 and started making predictions at multiple scales, which helped it detect small objects better. It kept its speed while becoming much more accurate.

YOLOv4: Better Feature Fusion YOLOv4 introduced new techniques like Cross-Stage Partial connections (CSP) and Spatial Pyramid Pooling (SPP) to combine features more effectively. It also used a Path Aggregation Network (PAN) to improve information flow. These changes balanced speed and accuracy well.

YOLOv5: Easier to Use YOLOv5 switched to the PyTorch framework, making it easier to train and customize. It used a CSPDarknet53 backbone and offered different model sizes to balance speed and accuracy for different needs.

YOLOv6 and YOLOv7: Focus on Efficiency YOLOv6 introduced a RepVGG-style backbone for faster inference and better accuracy. YOLOv7 improved layer connections further, achieving top performance for real-time detection.

YOLOv8 and Beyond The latest versions continue refining the architecture and training methods to make the models faster, more accurate, and more adaptable to various tasks.

Version	**Main Design Changes**	**Improvements**
YOLOv1	Single CNN with grid-based predictions	Fast and simple detection
YOLOv2	Anchor boxes, Darknet-19 backbone, batch normalization	Better accuracy and flexibility
YOLOv3	Darknet-53 backbone, multi-scale predictions	Improved small object detection and accuracy
YOLOv4	CSP connections, SPP, PAN	Balanced speed and accuracy
YOLOv5	PyTorch implementation, CSPDarknet53, multiple model sizes	Easier training and deployment
YOLOv6	RepVGG backbone, improved blocks	Faster inference and higher accuracy
YOLOv7	Enhanced layer aggregation (E-ELAN)	State-of-the-art real-time performance
YOLOv8+	Lightweight, refined architecture	Better adaptability and deployment

In short, YOLOs design has grown from a simple, one-step detector into a series of highly optimized models. Each new version builds on the last to make object detection faster, more accurate, and easier to use in real-world applications.

3 End of Document