19z510 – Machine Learning Laboratory

vAnnotate – A UI tool for creating dataset for Object Detection Model

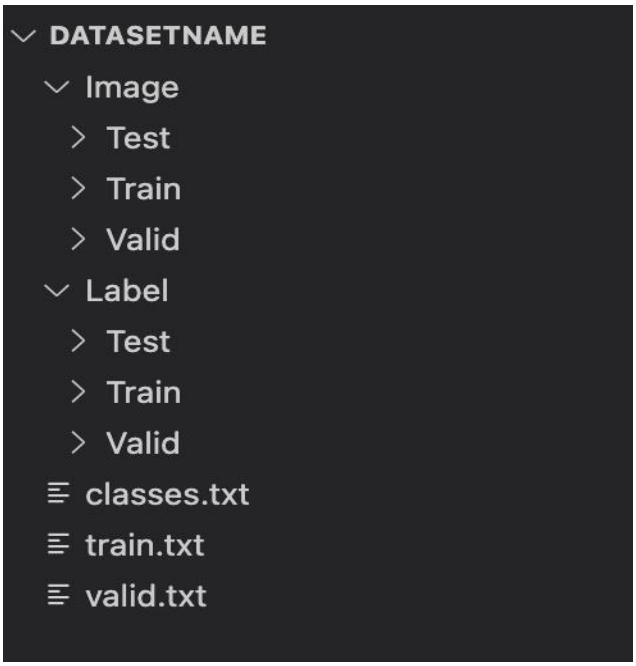
21Z204 – Akil K
21Z216 – Gali Likith Sai
21Z221 – Javagar M
21Z238 – Prithivi Raaj K
21Z266 – Vaikunt Ramakrishnan

Problem Statement:

The field of computer vision has made significant strides, thanks to deep learning techniques[01] and large annotated datasets[02]. However, challenges persist in effectively annotating datasets, managing them efficiently[05], and developing robust object recognition models. Manual annotation methods are labor-intensive, while automated tools often lack the accuracy needed for complex datasets. Dataset management techniques need to balance dataset diversity and class distribution to improve model performance. Object recognition methodologies require advancements to enhance accuracy and robustness, particularly in real-time processing and adaptability to various domains.

Dataset Description:

The dataset to train the YOLO v7 Object Detection Model has been created manually model. The structure of the input for the ML model that is being created is shown in the given picture.



The steps involved in the creation of the dataset have been attached as snapshots in Annexure II.

Methodology:

Object recognition is a fundamental task in computer vision, with applications ranging from object detection and classification to instance segmentation [04]. Deep learning-based approaches, particularly convolutional neural networks (CNNs), have demonstrated remarkable success in object recognition tasks, surpassing traditional computer vision techniques.

Popular CNN architectures[10] like AlexNet, VGG, ResNet, and EfficientNet have been widely adopted for object recognition tasks due to their ability to learn hierarchical features from raw pixel data [06]. These models are typically pretrained on large-scale datasets like ImageNet and then fine-tuned on domain-specific datasets to achieve state-of-the-art performance.

Recent advancements in object recognition include the development of attention mechanisms, multi-scale feature fusion techniques, and context-aware models, which aim to improve the accuracy and robustness of object detection and classification systems. Additionally, techniques like transfer learning and domain adaptation enable models to generalize better to new environments and datasets with limited labeled data.

Tools used:

Tkinter serves as the backbone for our annotation tool's GUI, facilitating interactive image annotation through widget creation, event handling, and layout management. Components like Tk(), Listbox(), Button(), and Canvas() are utilized for crafting a user-friendly interface. Its simplicity, cross-platform compatibility, and Python integration make it an excellent choice for GUI development.

Pillow[09], a PIL fork, plays a crucial role in image handling tasks within the annotation tool. It enables opening, displaying, and saving annotated images, along with functionalities for drawing bounding boxes and integrating with Tkinter's GUI[07]. Features like Image and ImageDraw enhance image manipulation, adding depth to the annotation tool's capabilities.

Beautiful Soup[08] emerges as a valuable tool for image URL extraction from web pages, streamlining the process of sourcing images. Its HTML parsing and URL retrieval capabilities simplify data extraction, seamlessly integrating external image sources into the annotation tool. This simplicity, coupled with robust parsing, makes Beautiful Soup an indispensable asset for augmenting image datasets through web scraping.

YOLO v7[03] is the latest version in the YOLO (You Only Look Once) family of object detection models, which are known for their real-time performance. YOLO v7 is a single-stage detector that uses a fully convolutional neural network (CNN) to process an image. It employs anchor boxes with different aspect ratios to detect objects of various shapes and sizes, reducing false positives.

Challenges Faced:

In the process of image annotation, several challenges were encountered. Firstly, sourcing high-quality images proved to be a significant hurdle, requiring extensive searching and curation. Secondly, the creation of annotations demanded meticulous attention to detail and consumed considerable time. Lastly, ensuring the responsiveness of the graphical user interface (GUI) was crucial for providing users with a smooth and efficient experience. Overcoming these challenges involved strategies such as optimizing image processing techniques, implementing automated annotation algorithms, and refining GUI performance. Efforts in these areas are essential for enhancing the efficiency and effectiveness of image annotation workflows.

Contribution of Team Members:

Roll Number	Name	Contribution
21Z204	Akil K	Dataset Creation Module
21Z216	Gali Likith Sai	Image Search Module
21Z221	Javagar M	Model Creation
21Z238	Prithivi Raaj K	Image Annotations
21Z266	Vaikunt Ramakrishnan	Image Classifier Module

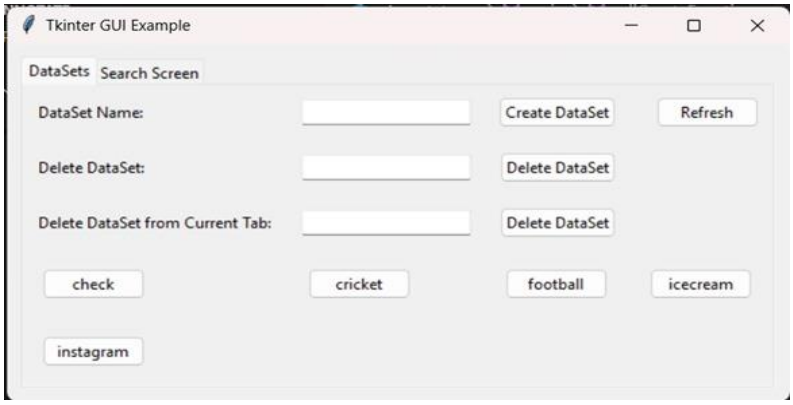
Annexure I: Code

```
# Commented out IPython magic to ensure Python compatibility.
# Download YOLOv7 repository and install requirements
!git clone https://github.com/WongKinYiu/yolov7
# %cd yolov7
!pip install -r requirements.txt

"""# Your Custom Data"""

# Commented out IPython magic to ensure Python compatibility.

# %cd /content
!curl -L "https://github.com/Javagar7/banks/raw/main/v.zip" > v.zip; unzip v.zip; rm v.zip

"""# Prepare image path in txt file"""

import os

train_img_path = "/content/images/train"
val_img_path = "/content/images/val"

# Commented out IPython magic to ensure Python compatibility.
# %cd /content

#Training images
with open('train.txt', "w") as f:
```

```
  img_list = os.listdir(train_img_path)
  for img in img_list:
    f.write(os.path.join(train_img_path,img+'\n'))
  print("Done")

# Validation Image
with open('val.txt', "w") as f:
  img_list = os.listdir(val_img_path)
  for img in img_list:
    f.write(os.path.join(val_img_path,img+'\n'))
  print("Done")

# Commented out IPython magic to ensure Python compatibility.
# %cp /content/yolov7/data/coco.yaml /content/yolov7/data/custom.yaml

"""# Download Pretrained weight"""

# Commented out IPython magic to ensure Python compatibility.
# download COCO starting checkpoint
# %cd /content/yolov7
!wget "https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt"

# Commented out IPython magic to ensure Python compatibility.
# %cp /content/yolov7/cfg/training/yolov7.yaml
/content/yolov7/cfg/training/custom_yolov7.yaml

"""#Begin Training"""

!python train.py --batch 16 --cfg cfg/training/custom_yolov7.yaml --epochs 100 --data
/content/yolov7/data/custom.yaml --weights 'yolov7.pt' --device 0

"""## Evaluation & Inference"""

# Run
!python detect.py --weights /content/yolov7/runs/train/exp/weights/best.pt  --source
/content/yolov7/RTX5EKA3.jpg

#display inference on ALL test images

import glob
from IPython.display import Image, display

i = 0
limit = 10000 # max images to print
for imageName in glob.glob('/content/yolov7/runs/detect/exp2/RTX5EKA3.jpg'): #assuming
JPG
    if i < limit:
      display(Image(filename=imageName))
      print("\n")
    i = i + 1
```

Annexure II: Snapshots of the Output

Creating Dataset



Images from dataset

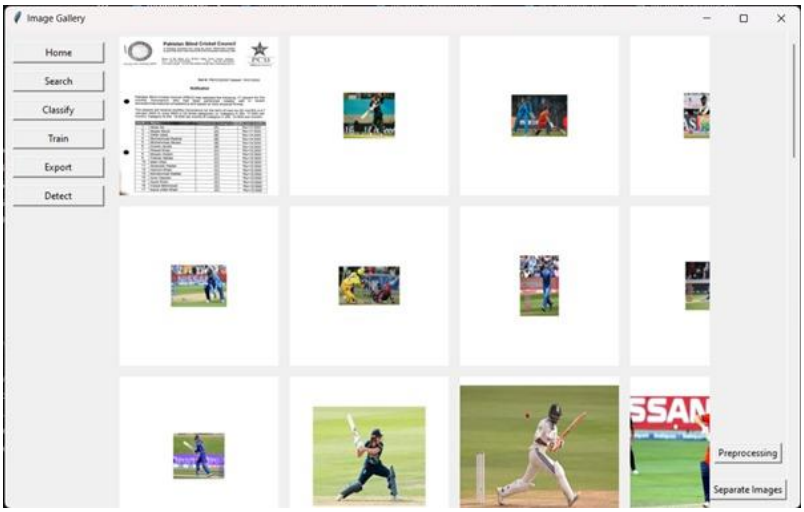
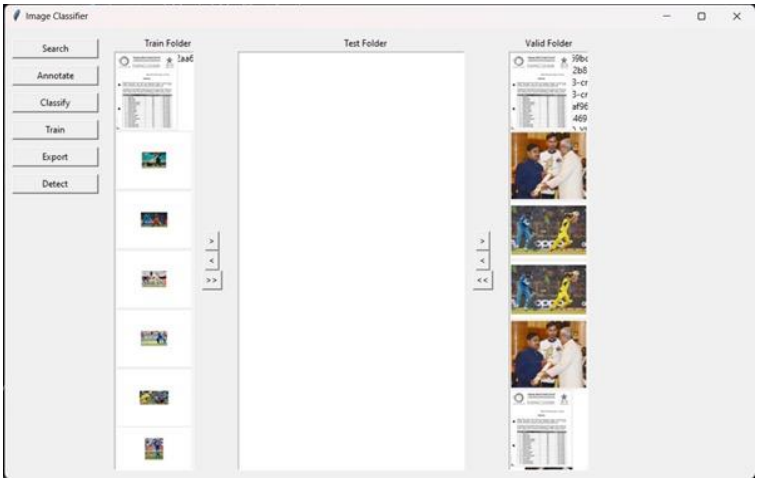
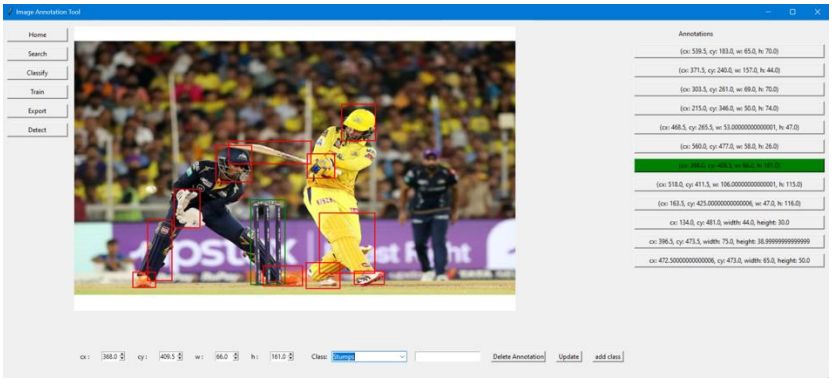


Image Classifier module



Annotations



Final output

References:

[01] https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf

[02] https://arxiv.org/abs/1409.0575

[03] https://www.v7labs.com/blog/yolo-object-detection

[04] https://arxiv.org/abs/1906.11172

[05] https://arxiv.org/abs/1506.02106

[06] https://arxiv.org/abs/1409.1556

[07] https://docs.python.org/3/library/tkinter.html

[08] https://realpython.com/beautiful-soup-web-scraper-python/

[09] https://pypi.org/project/pillow/

[10] https://www.geeksforgeeks.org/introduction-convolution-neural-network/