A Brief Description of Data Analysis, Model Creation and Performance

Data collection and preprocessing

Here is the explanation for my data collection and preprocessing steps,

1. **Data Source:**
   I have downloaded the Koala dataset from the provided google drive link.

2. **Data Exploration:**
   I have found, the dataset contains a mix of realistic and cartoon images of Koalas.

3. **Identification of Unnecessary Images or files:**
   During the exploration of the dataset, I identified some images that were not koala images. So I performed a data cleaning step by removing these unnecessary images. This ensures that only relevant and desired images for further analysis.

4. **Data Filtering:**
   After cleaning unnecessary images, I found a total 164 images of Koala. Of them, 144 were realistic and 20 were cartoon image. I only used the realistic image set for further preprocessing.

5. **Resize and Rename:**
   I have resized the 164 images so that all the images are in same size and rename them for giving each image a unique name.

6. **Data Annotation:**
   I used Labelme tool to annotate these 144 realistic images. Labelme tool provides the bounding box position in .json format. All the bounding box represents only a single class ‘Koala’.

7. **Conversion to YOLO format:**
   I have converted the annotated data from Labelme tool’s .json format to YOLO (You Only Look Once) format. YOLO format typically requires a .txt file for each corresponding image, where each line in the .txt file represents a bounding box in the YOLO format. The YOLO format for a bounding box in a .txt file is often represented as: “class x_center y_center width height”. The conversion code is given the github repository.

8. **Albumentation for Data Augmentation:**
   To increase data, I have used albumentation tool for data augmentation. Using Albumentation gives a scope to apply augmentation in images and adjust the corresponding YOLO-formatted .txt files to reflect the changes in the bounding box coordinates. For data augmentation I used the following methods,
   - Horizontal Flip,
   - Vertical Flip,
   - Rotate,
   - Random Brightness Contrast.

   The code is given in the github repository.

9. **Varify and Adjust Annotation:**
   I wrote a python script to inspect the augmented images and their corresponding annotations to ensure that the bounding boxes accurately represent the koala objects after augmentation. The code is given in the github repository.

10. **Data Splitting:**
    I have split the dataset with a ration of 70% for training, 15% for validation and 15% for testing. Before splitting, I shuffled the dataset to ensure a random distribution of data points across the sets. The code is given in the github repository.

Finally, the dataset is ready for training.

Training/Validation and Testing:

After preparing data I have followed this link (https://github.com/ultralytics/ultralytics) to install YOLOv8 model and followed the code given in the link for training, validation and testing.

Before training I have modified the coco.yaml file and create a custom.yaml file which contains the directory of my train, test and validation data path. I also mentioned the number of class with class name in the custom.yaml file.

Observation and Evaluation Report

I have trained the yolov8 model for 2 times with 2 different dataset. I am giving a brief description here,

Dataset 1	Dataset 2
Dataset Overview: • Total Images : 144 • Training Set: 100 images • Validation Set: 22 images • Test Set: 22 images	**Dataset Overview:** • Total Images : 864 • Training Set: 604 images • Validation Set: 129 images • Test Set: 131 images

Training Metrics:
• Box Loss
  ◦ Minimum: 0.35185
  ◦ Maximum: 1.4919
• Class Loss
  ◦ Minimum: 0.37036
  ◦ Maximum: 2.8689
• DFL Loss
  ◦ Minimum: 0.88047
	Training Metrics:
	• Box Loss
	  ◦ Minimum: 0.42743
	  ◦ Maximum: 1.4359
	• Class Loss
	  ◦ Minimum: 0.28883
	  ◦ Maximum: 2.1334
	• DFL Loss
	  ◦ Minimum: 0.92388

◦ Maximum: 1.5654	◦ Maximum: 1.5145
Validation Metrics: • Box Loss ◦ Minimum: 1.0347 ◦ Maximum: 2.7972 • Class Loss ◦ Minimum: 0.81269 ◦ Maximum: 3.0601 • DFL Loss ◦ Minimum: 1.1254 ◦ Maximum: 2.9755	**Validation Metrics:** • Box Loss ◦ Minimum: 0.7005 ◦ Maximum: 2.1855 • Class Loss ◦ Minimum: 0.39907 ◦ Maximum: 3.1106 • DFL Loss ◦ Minimum: 0.86665 ◦ Maximum: 2.1874
Training Metrics(B): • Precision ◦ Minimum: 0.00952 ◦ Maximum: 1.0 • Recall ◦ Minimum: 0.25926 ◦ Maximum: 1.0 • mAP50 ◦ Minimum: 0.17646 ◦ Maximum: 0.94776 • mAP50-95 ◦ Minimum: 0.06356 ◦ Maximum: 0.59151	**Training Metrics(B):** • Precision ◦ Minimum: 0.45624 ◦ Maximum: 1.0 • Recall ◦ Minimum: 0.48849 ◦ Maximum: 0.9863 • mAP50 ◦ Minimum: 0.38823 ◦ Maximum: **0.99466** • mAP50-95 ◦ Minimum: 0.14123 ◦ Maximum: **0.8173**
Inference Result: • Inference result on test set and cartoon set is given in the google drive link. • Runs-144 link	**Inference Result:** • Inference result on test set and cartoon set is given in the google drive link. • Runs-864 link

Evaluation Result:

Dataset 1	Dataset 2
Training Metrics Analysis: • The model shows varying losses during training, with fluctuations in box, class, and DFL losses. • Precision and recall have a wide range, indicating potential challenges in model performance.	**Training Metrics Analysis:** • Dataset 2 shows relatively stable and lower losses during training compared to Dataset 1. • Precision and recall metrics are consistent, indicating more stable performance.

Validation Metrics Analysis: • Validation losses demonstrate a higher range compared to training, suggesting some overfitting or difficulties in generalization.	**Validation Metrics Analysis:** • Validation losses are within an acceptable range, suggesting good generalization to the validation set.
Overall Assessment: • The model trained on Dataset 1 has mixed performance metrics, with notable variations in losses and evaluation metrics. Fine-tuning or adjusting hyperparameters may be beneficial to enhance model performance.	**Overall Assessment:** • The model trained on Dataset 2 demonstrates more stable performance metrics across training and validation. It appears to be better suited for generalization to unseen data.

Summary:

- Dataset 2, with a larger and more diverse dataset, shows more stable and promising results compared to Dataset 1.
- Consider further investigation into Dataset 1 to identify potential challenges and improve model performance.
- Fine-tuning hyperparameters and exploring additional data augmentation techniques may be beneficial for both datasets.