Title: Detection of LEGO Pieces Using Mask R-CNN

Author: [Yuhao Lu]

Link: https://github.com/327840386/lego-detection/blob/main/lego_detection.ipynb

1. Methods

In this lab, we used a deep learning model to detect and count LEGO pieces in images using the Mask R-CNN architecture. The goal was to detect all LEGO pieces in an image using bounding boxes and subsequently count them.

Dataset

The dataset used for this experiment consists of synthetic images of LEGO pieces with annotations in PASCAL VOC format. A subset of the dataset was used for simplicity, consisting of 200 images. The dataset was divided as follows:

- Training set: 70% of images
- Validation set: 15% of images
- Testing set: 15% of images

Only bounding box detection was performed, and the LEGO pieces were treated as a single class.

Model Architecture

The model used for this task is **Mask R-CNN** with a **ResNet-50 FPN** (Feature Pyramid Network) backbone. This architecture is effective for object detection tasks, where it identifies bounding boxes and generates masks for detected objects.

We used a pre-trained Mask R-CNN model from the **Torchvision** library, which was originally trained on the COCO dataset. The final layer of the model was adjusted to classify only two classes: **background** and **LEGO**.

Parameters and Settings

- **Batch size**: 1
- **Learning rate**: 0.005
- **Optimizer**: SGD with momentum of 0.9
- **Weight decay**: 0.0005
- **Learning rate scheduler**: StepLR, with step size of 3 and gamma of 0.1
- **Number of epochs**: 3
- **IoU threshold** for mAP evaluation: Standard mAP without explicit IoU threshold

The data was augmented during training with random horizontal flipping (with a probability of 0.5). Images were converted to PyTorch tensors before being fed to the model.

Training Process

The model was trained for 3 epochs. During each epoch, the training dataset was used to optimize the model weights by minimizing the detection loss. The optimizer used was SGD, which adjusts the model parameters based on computed gradients to reduce the overall loss.

Evaluation

To evaluate the model's performance, **Mean Average Precision (mAP)** was used. This metric is commonly used to evaluate object detection models, as it measures both precision and recall. The evaluation was performed on the validation set.

2. Results and Discussion

Training Loss

Over the course of the 3 epochs, the training loss decreased consistently, indicating that the model was learning to accurately detect the LEGO pieces. However, due to the small dataset size (200 images), the loss may not have reached an optimal point.

Mean Average Precision (mAP)

The **mAP** value achieved on the validation set was computed using the **torchmetrics** library. The resulting mAP score was **0.60**. This value suggests that the model can successfully identify LEGO pieces in the images, though the accuracy might be limited due to the relatively small training dataset.

Visualization

We performed inference on a sample image from the test set and visualized the results. The model successfully drew bounding boxes around detected LEGO pieces. The visualized results show that the model can detect and localize LEGO pieces reasonably well. However, some inaccuracies were observed, particularly in images where multiple LEGO pieces were overlapping or partially obscured.

Limitations

- **Small Dataset**: The dataset used in this experiment was small (200 images), which limits the generalizability of the model. A larger dataset would likely improve performance.
- **Simple Augmentation**: Only horizontal flipping was used for data augmentation. More diverse augmentations could help the model generalize better.

- **Limited Epochs**: The model was trained for only 3 epochs, which may not be sufficient for convergence, especially with a complex architecture like Mask R-CNN.

3. Conclusion

In this lab, we trained a Mask R-CNN model to detect and count LEGO pieces in synthetic images. The model achieved a reasonable mAP score, demonstrating its ability to localize and identify LEGO pieces. However, due to the limited dataset size and training epochs, the model's performance may not be optimal for more complex real-world scenarios.

Future work could include using a larger dataset, employing more sophisticated data augmentation techniques, and training for more epochs to improve the model's accuracy and robustness.

4. References

- **PyTorch Documentation**: https://pytorch.org/docs/stable/index.html
- **Torchvision Models**: https://pytorch.org/vision/stable/models.html
- **COCO Dataset**: https://cocodataset.org/
- **Mean Average Precision Metric (torchmetrics):** https://torchmetrics.readthedocs.io/en/stable/references/modules.html#meanaverageprecision
- **Szeliski, R.: Computer Vision: Algorithms and Applications**
- **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks:** https://arxiv.org/abs/1506.01497
- **Mask R-CNN Paper: https://arxiv.org/abs/1703.06870**
- **Pascal VOC Dataset and Evaluation Metrics:** http://host.robots.ox.ac.uk/pascal/VOC/
- **SGD Optimizer Details:** https://pytorch.org/docs/stable/optim.html#torch.optim.SGD
- **Transfer Learning in PyTorch:** https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html