Object Detection and Classification Using CNNs

Yuanyi Zhang, Taiming Chen, Yiling Liu

Introduction

The rapid evolution of machine learning and computer vision has brought about transformative changes in how machines interpret images and videos. Our project sought to harness the power of convolutional neural networks (CNNs) for detecting and classifying objects in complex scenes. Specifically, we focused on five object categories including person, car, bird, traffic light, and cell phone, using the Common Objects in Context (COCO) dataset. This report outlines our approach, the implementation, and an analysis of the results obtained.

Dataset Preparation and Balancing

The COCO dataset provided a rich and varied set of images annotated with bounding boxes and segmentation maps. From its 91 object categories in total, we extracted a subset containing only the five targeted classes: person, car, bird, traffic light, and cell phone. We found that the number of images with the label ‘bird’, ‘traffic light’, and ‘cell phone’ is around 4,000 in the training image set, while the class ‘person’ has around 100,000 images and the class ‘car’ has 60,000 images. If we fed all the images with respective labels, the model would be biased: it would perform better at classifying the images with person and car than the images with the other three labels. Therefore, for balanced training and time efficiency, the number of images with ‘person’ was trimmed to 7,000 and ‘car’ was trimmed to 6000. The dataset uses the COCO annotation format. Each image contains annotations for object categories and segmentation masks. The segmentation masks were generated using the COCO library, and each category ID was mapped to a sequential index to simplify the segmentation task.

Model Architecture

Our model architecture is called U-Net, which stems from CNNs that is specifically designed to handle multi-object classification and precise localization, making it suitable for segmentation tasks. The architecture included the following key components:

1. Downsampling Path:
    - It consists of convolutional and max-pooling layers to extract features and reduce the spatial dimensions of the images. 🎬
    - Used kernel sizes of 3×3 with ReLU activation.
2. Bottleneck:
    - Serves as a bridge between downsampling and upsampling paths
3. Upsampling Path:

- Consists of upsampling layers and skip connections to recover spatial information and refine the segmentation masks.

4. Output Layer:
    - a 2D convolution layer with a softmax activation for multi-class segmentation

The model was compiled with optimizer Adam (default learning rate = 0.001), sparse categorical cross-entropy as the loss function, and accuracy as the evaluation metrics

Implementation

The model was implemented using Python and TensorFlow. Key implementation details include:

- Batch Size: 8
- Epochs: 5

Since the model was training without GPU acceleration, more epochs should be implemented to achieve better convergence and better fitting.

Results and Analysis

The training performance is shown in Figure 1. The left plot is the loss function over training steps, demonstrating a steady decrease and convergence. The right plot shows the accuracy, which is a simple measure that calculates the ratio of correctly predicted pixels to the total number of pixels in the image. The right plot shows the accuracy rapidly increasing and stabilizing around 87%. The periodic dips in accuracy and spikes in both plots are due to the completion of each epoch.

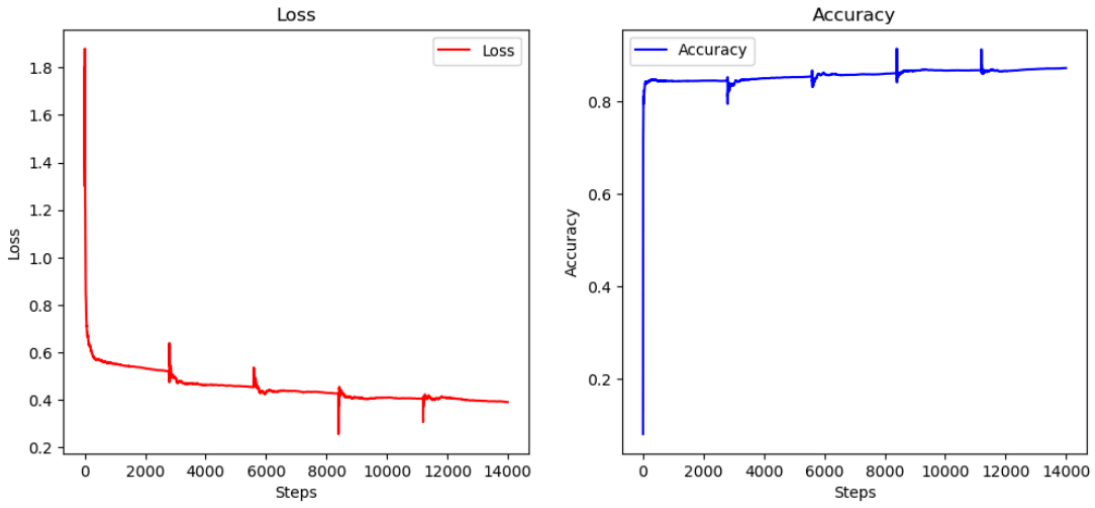

Figure 1:Training performance of the model

The model was evaluated on a test dataset containing around 200 images per target category. The test loss was 0.405 and the test accuracy was 86.82%. However,

these results can't indicate that the model is capable of accurately segmenting the specified categories if most pixels in the image belong to the background. The model can achieve high accuracy by correctly predicting these background pixels, even if it performs poorly on the intended objects (e.g. person, car, bird, etc). Therefore, the model was also evaluated by visualizing the predicted segmentation mask in the testing images, as shown in Figure 2.

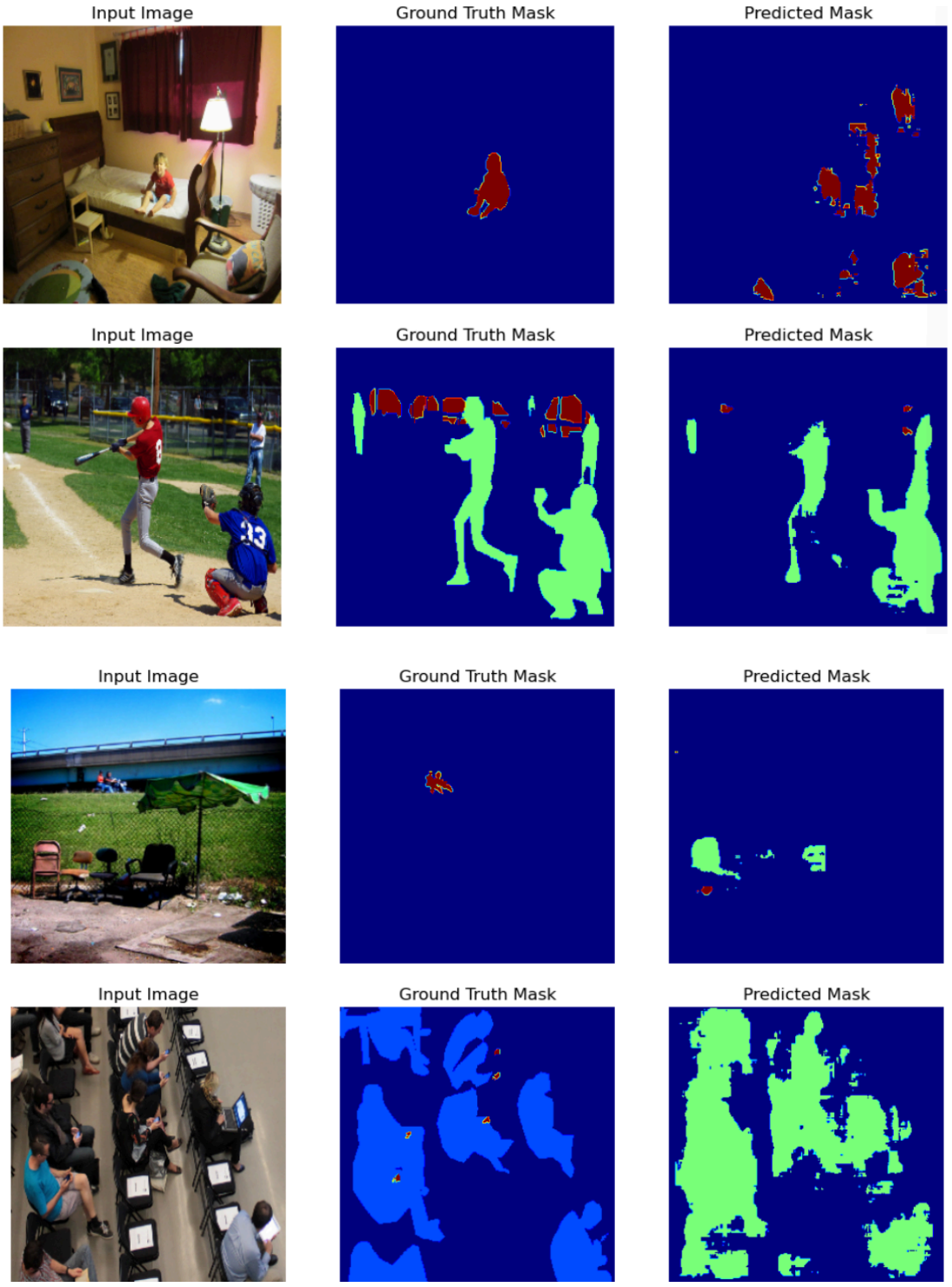

Figure 2: Segmentation results of the model on test dataset.

In Figure 2, the left column is the raw images, which is the original images from the test dataset. The middle column is the actual mask showing the correct segmentation for the labeled objects, which is included in the annotations. The right column is the predicted segmentation mask by the model. The model performs well in identifying and segmenting objects such as people, though some inaccuracies and missed objects are observed, particularly in complex scenes or with smaller objects such as cell phones.

Conclusion

The results demonstrate the efficacy of CNNs in detecting and classifying objects within the challenging COCO dataset. The model achieved a test accuracy of 86.82%, which indicates good generalization. Also, the training process demonstrated steady improvement, as shown by the loss and accuracy curves. However, certain limitations emerged, such as difficulty in recognizing small or heavily occluded objects. Future work could address these by integrating attention mechanisms for better focus on key areas and employing advanced architectures like YOLO or Faster R-CNN, increasing epochs to further improve performance, introducing data augmentation to handle class imbalance, and optimizing learning rate and batch size for better results.

References

- Lin, T.-Y., et al. (2014). Microsoft COCO: Common Objects in Context.
- Krizhevsky, A., et al. (2012). ImageNet Classification with Deep Convolutional Neural Networks.
- Chollet, F. (2017). Deep Learning with Python.