Research on Detectron2

1. What is Detectron2 ?

Detectron2 is an advanced, PyTorch-based open-source library developed by Facebook AI Research (FAIR) for state-of-the-art computer vision tasks. It is the successor to the original Detectron and maskrcnn-benchmark, offering significant enhancements and a more modular design.

Key characteristics of Detectron2:

- **Versatility:** Supports a wide range of tasks like object detection, instance segmentation, panoptic segmentation, and pose estimation.
- **Modularity:** Features a highly flexible design, allowing for easy customization and integration of new components.
- **Performance:** Optimized for fast training and inference, crucial for large-scale and real-time applications.
- **Extensive Model Zoo:** Provides a large collection of pre-trained models (e.g., Faster R-CNN, Mask R-CNN) for fine-tuning.
- **Production Readiness:** Includes Detectron2go for simplified deployment to cloud and mobile devices.

Official resources for Detectron2,

- **GitHub Repository:** The primary source for the Detectron2 library, code, and detailed information is its GitHub repository: https://github.com/facebookresearch/detectron2.

- **Official Documentation:** Comprehensive documentation is available on Read the Docs, providing guides for installation, getting started, and advanced usage: https://detectron2.readthedocs.io/.

2. Detecron2 Fine-Tuning

1 — Prerequisites

Environment: Notebook or script with GPU (e.g., Colab, VS Code + CUDA)
Tools: Label Studio or CVAT for annotation
Libraries: Detectron2, OpenCV, Py Torch, COCO API

2. Data Preparation

- Define your object labels (e.g., table, cell, row)

- Annotate 300+ images using Label Studio.

- Export annotations in **COCO JSON format**
- Organize dataset:

```
bash
Copy code
dataset/
 ├── images/
└── annotations/train. json
```

3. Register Dataset

- Register your dataset with register_ coco _instances ()
- This tells Detectron2 how to load your images and labels.

4. Fine-Tune Model

- Load a pretrained model (e.g. Faster R-CNN from Detectron2 model zoo)
- Set:
    - NUM_CLASSES = your number of labels
    - MAX_ITER = number of training steps
    - BASE_LR = 0.00025
- Train using Default Trainer
- Output: model_final.pth

5. Inference Pipeline

- Load your saved model and config
- Run predictions on new images using Default Predictor
- Output: detected objects with labels and bounding boxes

My Environment

Component	Version	Status
Python	3.11.0	*Not supported* by Detectron2
Py Torch	2.7.1 + CPU	*Too new* + CPU-only
CUDA	Not available	OK for CPU-only, but slower
GPU	None / Not used	Detectron2 works on CPU, but slower

---

1. customdetectron2.ipynb

1. **Installs:** Detectron2, Easy OCR, pdf2image
2. **Converts PDF to Image:** Using pdf2image and saves the first page
3. **Loads Pretrained Model:** mask_rcnn_R_50_FPN_3x from Detectron2 model zoo
4. **Inference:** Runs prediction on the image (page.jpg)
5. **OCR:** Uses Easy OCR to extract text from each detected region (bounding boxes)
6. **Output:** Displays image with visualized boxes and prints text per detected region.

- OCR output was jumbled because bounding boxes were incorrect.
- Output was unstructured and unusable for table extraction.

---

2. detectron2.ipynb

1. **Installs:** Tesseract, pdf2image, layout parser
2. **Uploads a File:** (image or PDF)
3. **Model:** Uses Layout Parser's Detectron2 Table Bank model to detect table cells
4. **OCR:** Performs Tesseract OCR on each detected cell block
5. **Postprocessing:**
    - Clusters detected cells into rows (based on Y-position using DBSCAN)
    - Sorts cells left to right in each row
    - Builds a proper structured table as a pandas Data Frame
6. **Output:** Saves and displays as structured_table.csv

- **OCR failed in small or noisy cells, producing empty or incorrect text.**
- **Output table was partially structured, but not always accurate.**
