```
|   |   ├── img_001.jpg
|   |   ├── img_001.txt
|   |   └── ...
|   └── val/
|       ├── img_101.jpg
|       ├── img_101.txt
|       └── ...
└── rec/
    ├── train/
    |   ├── word_001.jpg
    |   └── ...
    ├── val/
    |   ├── word_201.jpg
    |   └── ...
    └── label.txt
```

⚡ Deliverables

1. **Runnable Colab** .ipynb → installs PaddleOCR, downloads ICDAR subset, trains detection+recognition.
2. **Runnable Kaggle** .ipynb → same pipeline, adapted for Kaggle dataset mounting.
3. **Results CSV template + helper code** → auto-formats eval metrics into a table.
4. **Export-ready synthetic dataset (zip)** for quick experiments without large downloads.

-------- A runnable .ipynb that includes: -----

1. **Install & Setup** (PaddlePaddle GPU + PaddleOCR).
2. **Dataset Prep** (download a small ICDAR 2015/2019 subset for quick demo, convert to PP-OCR format).
3. **Training** (short run: ~10 epochs, PP-OCRv3 lightweight).
4. **Evaluation** (precision, recall, F1, accuracy, NED).
5. **Visualization** (sample predictions with bounding boxes + recognized text).
6. **Artifacts Saving** (weights, logs, metrics).

--------- Deliverables ---------

📝 Report Outline (PP-OCR Study & Training Workflow)

1. Introduction

- Purpose: end-to-end exploration of PaddleOCR (PP-OCRv3 → PP-OCRv5).
- Scope: architecture review, dataset formatting, training pipelines, reproducible notebooks.

2. Sources Consulted

- **PaddleOCR GitHub** (docs, configs, release notes up to v3.2.0 with PP-OCRv5).
- **Research Papers**: DBNet (detector), CRNN (recognizer), PP-OCR system papers.
- **Datasets**: COCO-Text v2.0, ICDAR 2015, ICDAR 2019 MLT, LSVT, RCTW-17, MTWI.
- **Community Resources**: Colab/Kaggle tutorials, PaddleOCR issues, blogs.

3. PP-OCR Architectures (v3 → v5)

- Detector: **DB-based** (backbone → neck → head).
- Angle Classifier: lightweight CNN.
- Recognizer: CRNN → SVTR lightweight variants.
- Mobile vs Server splits.
- Key improvements in **PP-OCRv5**: multilingual support, accuracy/efficiency trade-offs.
- Include **pipeline diagram** (det → cls → rec).

4. Dataset Preparation

- Detection format: quadrilaterals + transcription.
- Recognition format: cropped words + `label.txt`.
- Conversion script (COCO-Text → PP-OCR format).
- PPOCRLabel tool usage.

- Dataset trade-offs:
    - COCO-Text → large, English-dominant.
    - ICDAR 2019 → multilingual.
    - Synthetic datasets → quick tests.

5. Training Pipelines

- Lightweight vs strong configs.
- Training with `tools/train.py`.
- Hyperparameters (batch size, epochs, learning rate, optimizer, augmentations).
- Mixed precision training for speed.
- Logging: TensorBoard / W&B.

6. Practice Attempts

- Label conversion script (COCO → PP-OCR).
- Colab demo (ICDAR subset).
- Synthetic dataset mini-run.
- Visualization of predictions.

7. Insights & Conclusions

- Efficiency gains in **PP-OCRv5** (faster + more accurate).
- Multilingual data is the main bottleneck (annotation formats vary).
- Lightweight configs are ideal for Colab/Kaggle quick runs; server configs for final training.
- Reproducibility requires saving configs, weights, and dataset splits.

8. References

- PaddleOCR official repo + releases.
- DBNet and CRNN original papers.
- ICDAR, COCO-Text, RCTW datasets.

📂 Deliverable

Now **generate a LaTeX → PDF report** with this structure.
It will include:

- Proper **sections** & formatting.
- **Architecture diagram** (pipeline).
- **Tables** (datasets & trade-offs, results template).
- **Code snippets** (conversion scripts, Colab prep).

----- Repository Layout

```
ppocr-study/
 ├— report/
|   ├— PP-OCR_Workflow_Report.pdf
|   └— PP-OCR_Workflow_Report.tex
 ├— code/
|   ├— convert_coco_det.py      # COCO-Text → PP-OCR det
|   ├— convert_rec.py           # crops + label.txt
|   ├— train_det.py             # launcher for detection training
|   ├— train_rec.py             # launcher for recognition training
|   ├— eval_det.py              # eval metrics for detection
|   ├— eval_rec.py              # eval metrics for recognition
|   ├— visualize_preds.py       # draw det/rec results
|   └— utils.py                 # shared helpers
 ├— datasets/
|   ├— synthetic_demo/          # small package (images + txt + crops)
|   └— icdar2019_subset/        # downloaded & converted split
 ├— outputs/
|   ├— weights/                 # trained model snapshots
|   ├— metrics.csv              # precision/recall/F1/acc/NED
|   └— samples/                 # visualized predictions
└— notebooks/
   ├— colab_pipeline.ipynb
   └— kaggle_pipeline.ipynb
```

Code Deliverables

1. Dataset Conversion

```
# convert_coco_det.py
"""
Convert COCO-Text JSON into PaddleOCR detection format (.txt per image).
"""
import json, os

def coco_to_ppocr(coco_json, output_dir):
    os.makedirs(output_dir, exist_ok=True)
    coco = json.load(open(coco_json, "r", encoding="utf-8"))

    img_map = {img["id"]: img["file_name"] for img in coco["images"]}
    anns_by_img = {}
    for ann in coco["annotations"]:
        anns_by_img.setdefault(ann["image_id"], []).append(ann)

    for img_id, anns in anns_by_img.items():
        out_file = os.path.join(output_dir, os.path.splitext(img_map[img_id])[0] + ".txt")
        lines = []
        for ann in anns:
            seg = ann.get("segmentation", [[]])[0]
            if len(seg) < 8: continue
            points = [str(int(p)) for p in seg[:8]]
            text = ann.get("utf8_string", "###")
            lines.append(",".join(points) + "," + text)
        with open(out_file, "w", encoding="utf-8") as f:
            f.write("\n".join(lines))

# convert_rec.py
"""
Generate recognition crops and label.txt file from detection boxes + images.
"""
import cv2, os, json

def generate_recognition_data(det_dir, img_dir, out_img_dir, out_label_file):
    os.makedirs(out_img_dir, exist_ok=True)
    label_lines = []
    idx = 0
    for det_file in os.listdir(det_dir):
        if not det_file.endswith(".txt"): continue
        img_file = det_file.replace(".txt", ".jpg")
        img = cv2.imread(os.path.join(img_dir, img_file))
        for line in open(os.path.join(det_dir, det_file), "r", encoding="utf-8"):
            parts = line.strip().split(",")
            if len(parts) < 9: continue
            pts, text = list(map(int, parts[:8])), parts[8]
            poly = [(pts[i], pts[i+1]) for i in range(0,8,2)]
            rect = cv2.boundingRect(np.array(poly))
            crop = img[rect[1]:rect[1]+rect[3], rect[0]:rect[0]+rect[2]]
            crop_name = f"rec_{idx}.jpg"
            cv2.imwrite(os.path.join(out_img_dir, crop_name), crop)
            label_lines.append(f"{crop_name}\t{text}")
            idx += 1
    open(out_label_file,"w",encoding="utf-8").write("\n".join(label_lines))
```

2. **Training Launchers**

```
# train_det.py
"""
Wrapper around PaddleOCR training for detection.
"""
import os
os.system("python3 tools/train.py -c configs/det/det_mv3_db.yml -o Global.epoch_num=10
Global.save_model_dir=./outputs/det_model")

# train_rec.py
"""
Wrapper around PaddleOCR training for recognition.
"""
import os
os.system("python3 tools/train.py -c configs/rec/rec_mv3_none_bilstm_ctc.yml -o Global.epoch_num=10
Global.save_model_dir=./outputs/rec_model")
```

3. **Evaluation**

```
# eval_det.py
os.system("python3 tools/eval.py -c configs/det/det_mv3_db.yml -o
Global.pretrained_model=./outputs/det_model/best_accuracy")

# eval_rec.py
os.system("python3 tools/eval.py -c configs/rec/rec_mv3_none_bilstm_ctc.yml -o
Global.pretrained_model=./outputs/rec_model/best_accuracy")
```

4. **Visualization**

```
# visualize_preds.py
"""
Overlay detection boxes + recognized text on images.
```

```
""
import cv2, json
def draw_results(img_path, results, out_path):
   img = cv2.imread(img_path)
   for box, txt in results: # box = [[x1,y1],[x2,y2],[x3,y3],[x4,y4]]
      pts = np.array(box, np.int32).reshape((-1,1,2))
      cv2.polylines(img,[pts],True,(0,255,0),2)
      cv2.putText(img, txt, (box[0][0], box[0][1]-5), cv2.FONT_HERSHEY_SIMPLEX, 0.6,(0,0,255),2)
   cv2.imwrite(out_path, img)
```

📊 Results Deliverables

- **Trained weights**:
  - outputs/weights/det_model/
  - outputs/weights/rec_model/
- **Metrics** (metrics.csv):

Model	Precision	Recall	F1	Accuracy	NED	FPS
Det (PP-OCRv3-lite)	0.78	0.73	0.75	-	-	25
Rec (CRNN-lite)	-	-	-	0.82	0.11	200

- **Qualitative Samples** (outputs/samples/):
  - det_result_img001.jpg → bounding boxes
  - rec_result_img001.jpg → recognized text overlays

📂 Repo Documentation Structure

```
ppocr-study/
├── README.md            # main overview
├── report/README.md        # summary of PDF report
├── code/README.md         # usage of scripts + training pipeline
├── datasets/README.md       # dataset prep, trade-offs, label formats
└── outputs/README.md       # trained weights, metrics, visualizations
```

Main README.md

# PP-OCR Study & Training Workflow

This repository documents an end-to-end exploration of PaddleOCR's PP-OCR system (v3 → v5), covering:
- **Architectures** (detector → angle classifier → recognizer)
- **Datasets** (COCO-Text, ICDAR 2015/2019 MLT, synthetic samples)
- **Training Pipelines** (PaddleOCR training scripts, configs, hyperparameters)
- **Evaluation** (precision, recall, F1, accuracy, NED)
- **Reproducible Notebooks** (Colab and Kaggle)

## 📌 Pipeline Overview
PP-OCR consists of three major components:
1. **Text Detection** (DB-based detector: backbone → neck → head)
2. **Angle Classifier** (lightweight CNN for orientation correction)
3. **Text Recognition** (CRNN/SVTR-based recognizer)

<p align="center">
  <img src="report/pipeline_diagram.png" width="600"/>
</p>

## 📂 Repo Structure
- `report/` → comprehensive PDF + LaTeX source
- `code/` → dataset converters, training/eval scripts, visualization utils
- `datasets/` → prepped ICDAR/COCO subsets + synthetic demo
- `outputs/` → weights, metrics, and sample predictions
- `notebooks/` → runnable Colab & Kaggle notebooks

## 🚀 Quickstart
1. Install dependencies:
   ```bash
   pip install paddlepaddle-gpu==2.5.2 paddleocr==2.7

2. Run dataset conversion:

python code/convert_coco_det.py
python code/convert_rec.py

3. Train detection:

python code/train_det.py

4. Train recognition:

python code/train_rec.py

5. Evaluate:
   ```

```
python code/eval_det.py
python code/eval_rec.py
```

Results (Sample)

Model	Precision	Recall	F1	Accuracy	NED	FPS
PP-OCRv3-lite Det	0.78	0.73	0.75	-	-	25
PP-OCRv3-lite Rec	-	-	-	0.82	0.11	200

📖 References

- PaddleOCR GitHub
- DBNet, CRNN papers
- COCO-Text, ICDAR, RCTW, LSVT datasets

# 📝 `datasets/README.md`

```markdown
# Datasets & Label Formats

## 📂 Detection Format
ICDAR-style: quadrilateral points + transcription.

x1,y1,x2,y2,x3,y3,x4,y4,text
34,56,120,50,122,80,36,86,OPEN

Use `###` for illegible text.

## 📂 Recognition Format
Cropped word/line images + `label.txt`:

word_001.jpg HELLO
word_002.jpg WORLD

## 📊 Dataset Trade-Offs
- **COCO-Text V2.0**
  - 63k images, 239k text instances, mask annotations.
  - Strong for English scene text, weaker multilingual coverage.

- **ICDAR 2019 MLT**
```

  - Multilingual dataset (Latin, Chinese, Arabic, etc.).
  - Ideal for multilingual training, heavier and slower.

- **ICDAR 2015**
  - Small but popular benchmark (oriented scene text).

- **Synthetic Dataset (this repo)**
  - 5–10 demo images for quick runs in Colab.
  - Useful for pipeline validation.

Generalization vs Speed
- Larger, multilingual sets improve robustness but slow training.
- COCO-Text faster for prototyping, ICDAR-MLT better for deployment-level multilingual OCR.

code/README.md

# Code & Scripts

## 🛠 Scripts
- `convert_coco_det.py` → Convert COCO-Text JSON → PP-OCR detection `.txt`
- `convert_rec.py` → Generate crops + `label.txt` for recognition
- `train_det.py` → Launcher for detection training
- `train_rec.py` → Launcher for recognition training
- `eval_det.py` → Evaluate detection precision/recall/F1
- `eval_rec.py` → Evaluate recognition accuracy/NED
- `visualize_preds.py` → Overlay detection boxes + recognized text

## 🔧 Usage
```bash
python code/convert_coco_det.py
python code/train_det.py
python code/eval_det.py
```

See notebooks/colab_pipeline.ipynb for a runnable example.

# 📝 `outputs/README.md`

```markdown
# Outputs

## 📂 Contents
- `weights/` → trained models
```

- `metrics.csv` → benchmarked precision, recall, F1, accuracy, NED
- `samples/` → visualized predictions (detection boxes + recognized text)

## 📊 Example Metrics

Model	Precision	Recall	F1	Accuracy	NED
Det (PP-OCRv3)	0.78	0.73	0.75	-	-
Rec (CRNN-lite)	-	-	-	0.82	0.11

Example (Markdown snippet for README.md):

## 📖 References

- PaddleOCR GitHub. *PaddleOCR: End-to-End OCR Toolkit*.
  https://github.com/PaddlePaddle/PaddleOCR (accessed Sept 2025).

- PaddleOCR Docs. *Installation, Datasets, Training*.
  https://paddlepaddle.github.io/PaddleOCR/main/en/index.html.

- COCO-Text V2.0 Dataset. *COCO-Text: Dataset for Text Detection and Recognition in Natural Images*.

- ICDAR 2019 MLT Dataset. *ICDAR 2019 Robust Reading Challenge on Multi-lingual Scene Text Detection and Recognition*.

- Example tutorials on Kaggle & Colab (multilingual OCR training, fine-tuning, W&B logging).