# Layout Parser

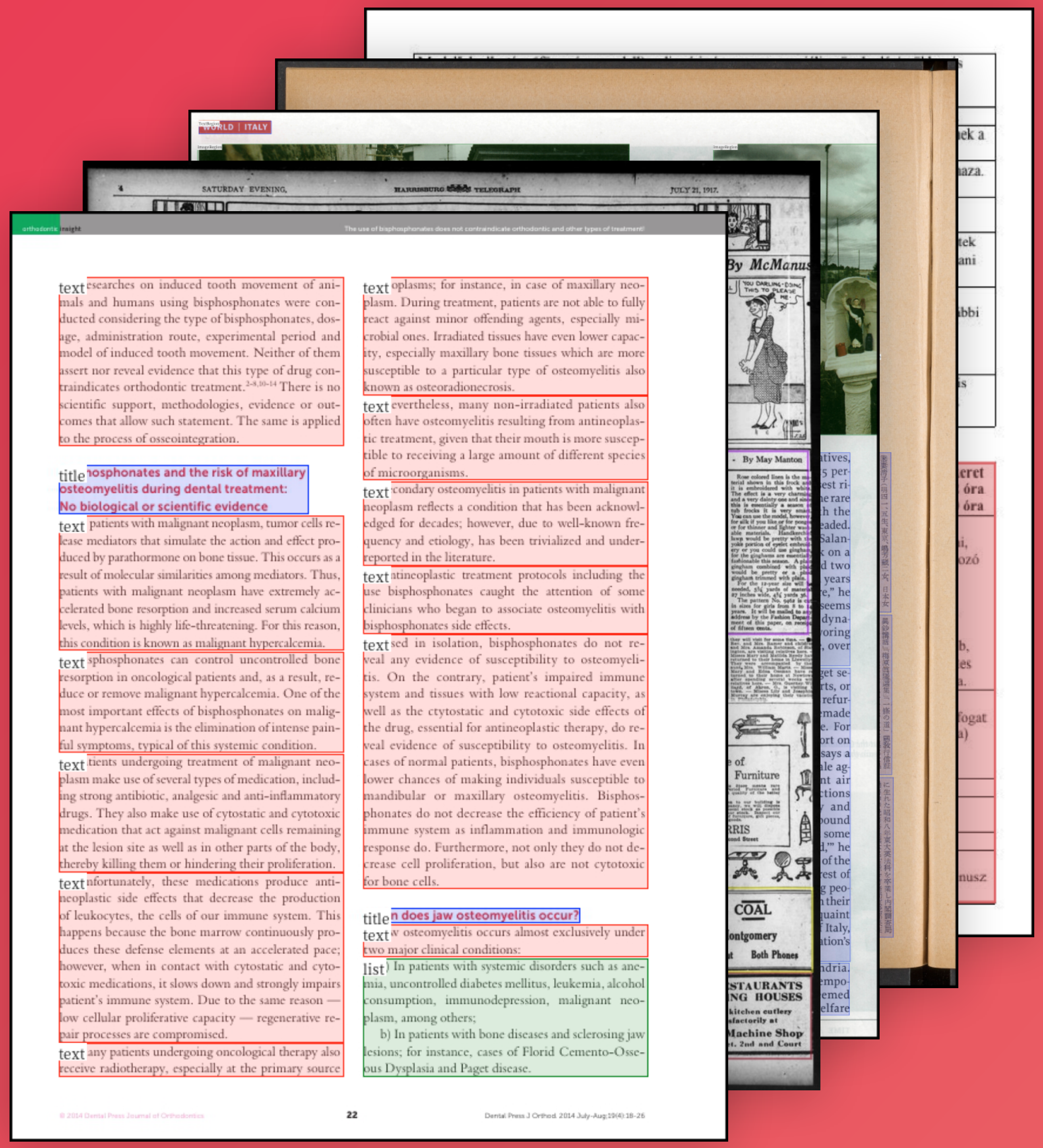
## A Unified Toolkit for Deep Learning Based Document Image Analysis

*Zejiang Shen, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson, Weining Li*

- Motivation
- Demo
- Design & Implementation
- Future Work
- Community

# Deep Learning for Document Image Analysis

The Task | Exciting Progress | Challenges



Input: Doc Images


{
"title": "Construction of the Literature Graph in Semantic Scholar",
"authors": "Waleed Ammar et. al.",
"abstract": "We describe a deployed scalable system for organizing published scientific literature into a heterogeneous graph to facilitate algorithmic manipulation and ...",
"sections": ["..."]
}

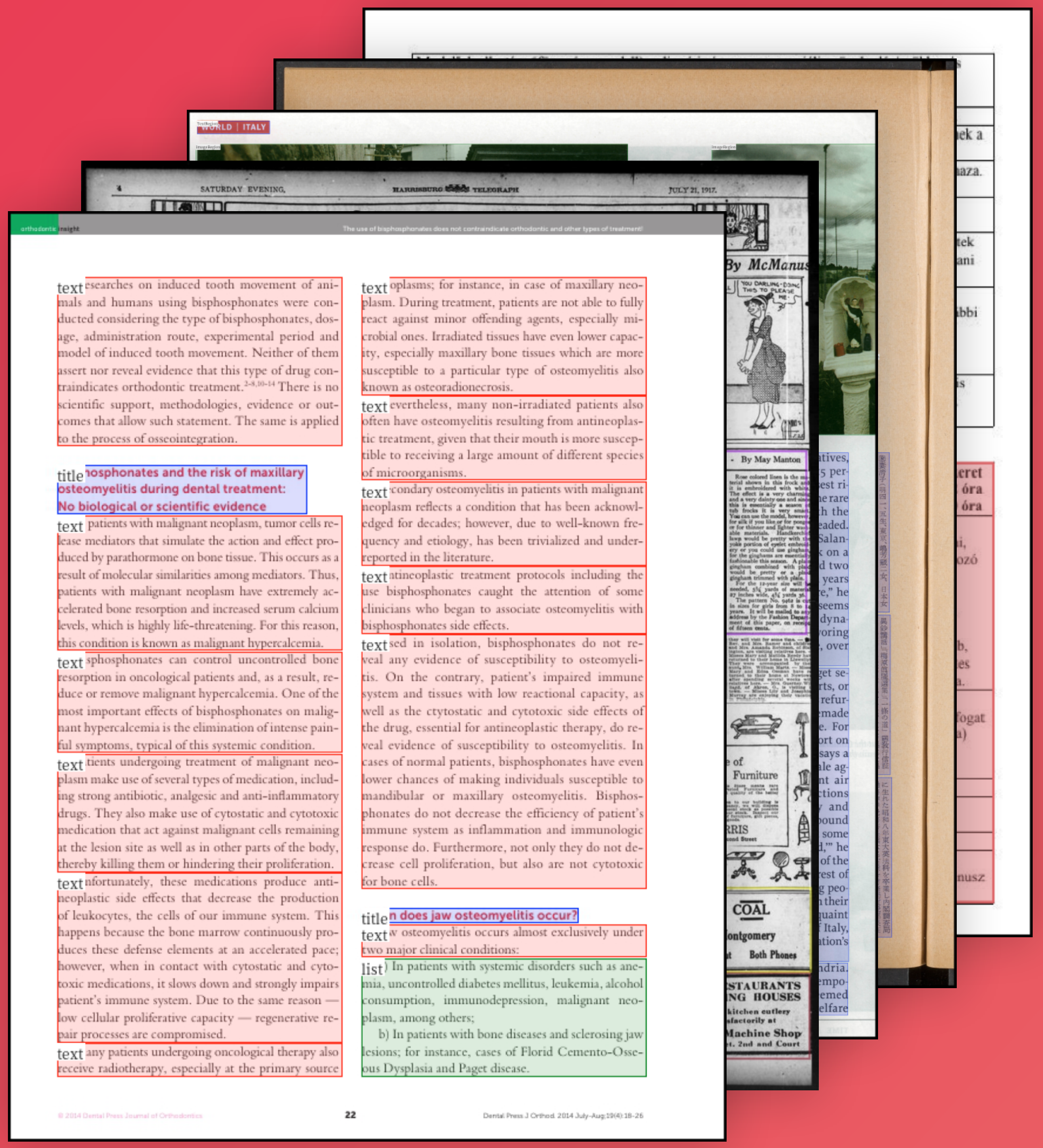

Output: Layout/Text

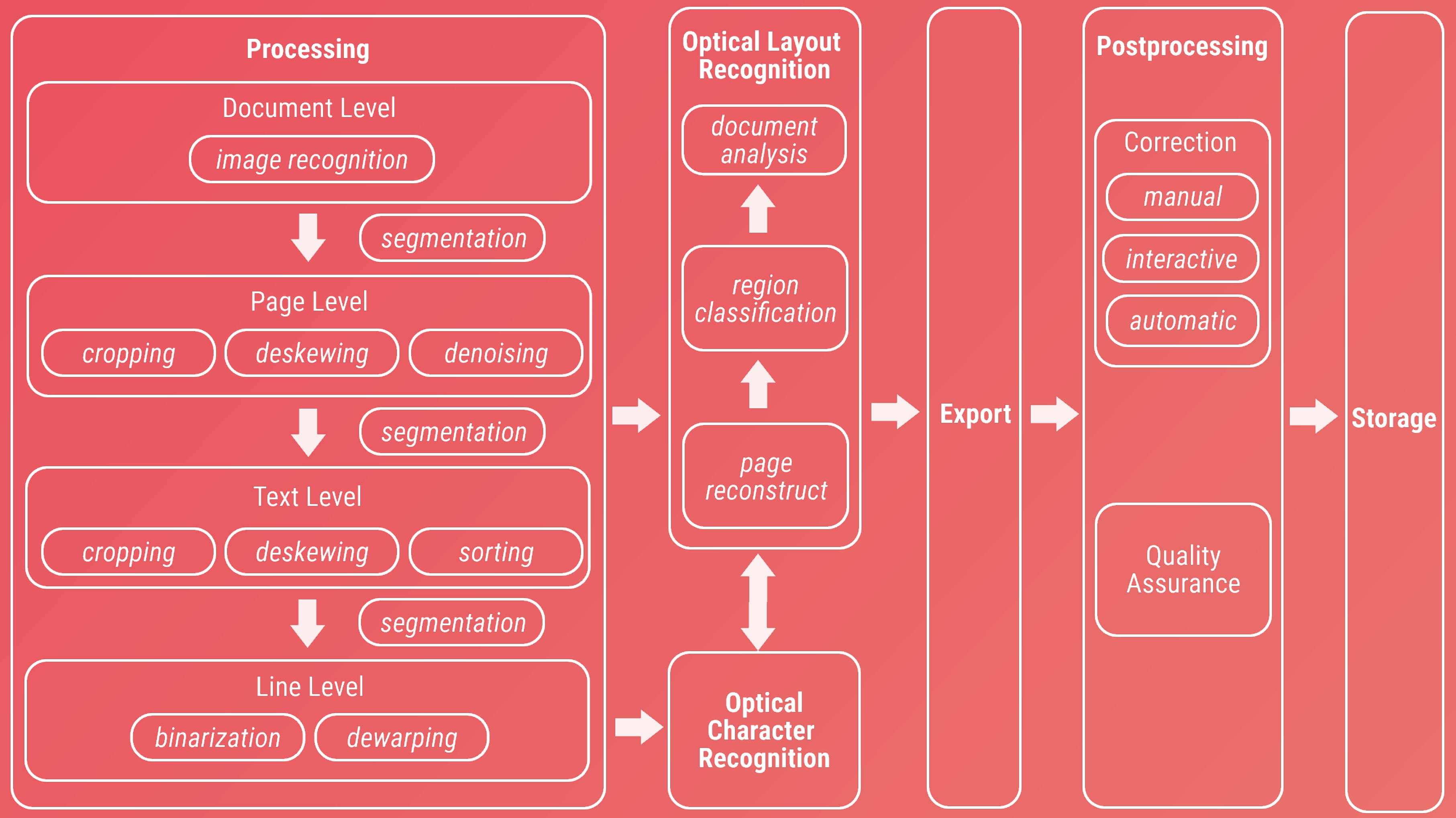
# Deep Learning for Document Image Analysis

The Task

Exciting Progress

Challenges







{
"title": "Construction of the Literature Graph in Semantic Scholar",
"authors": "Waleed Ammar et. al.",
"abstract": "We describe a deployed scalable system for organizing published scientific literature into a heterogeneous graph to facilitate algorithmic manipulation and ...",
"sections": ["..."]
}


Input: Doc Images

DIA Pipeline

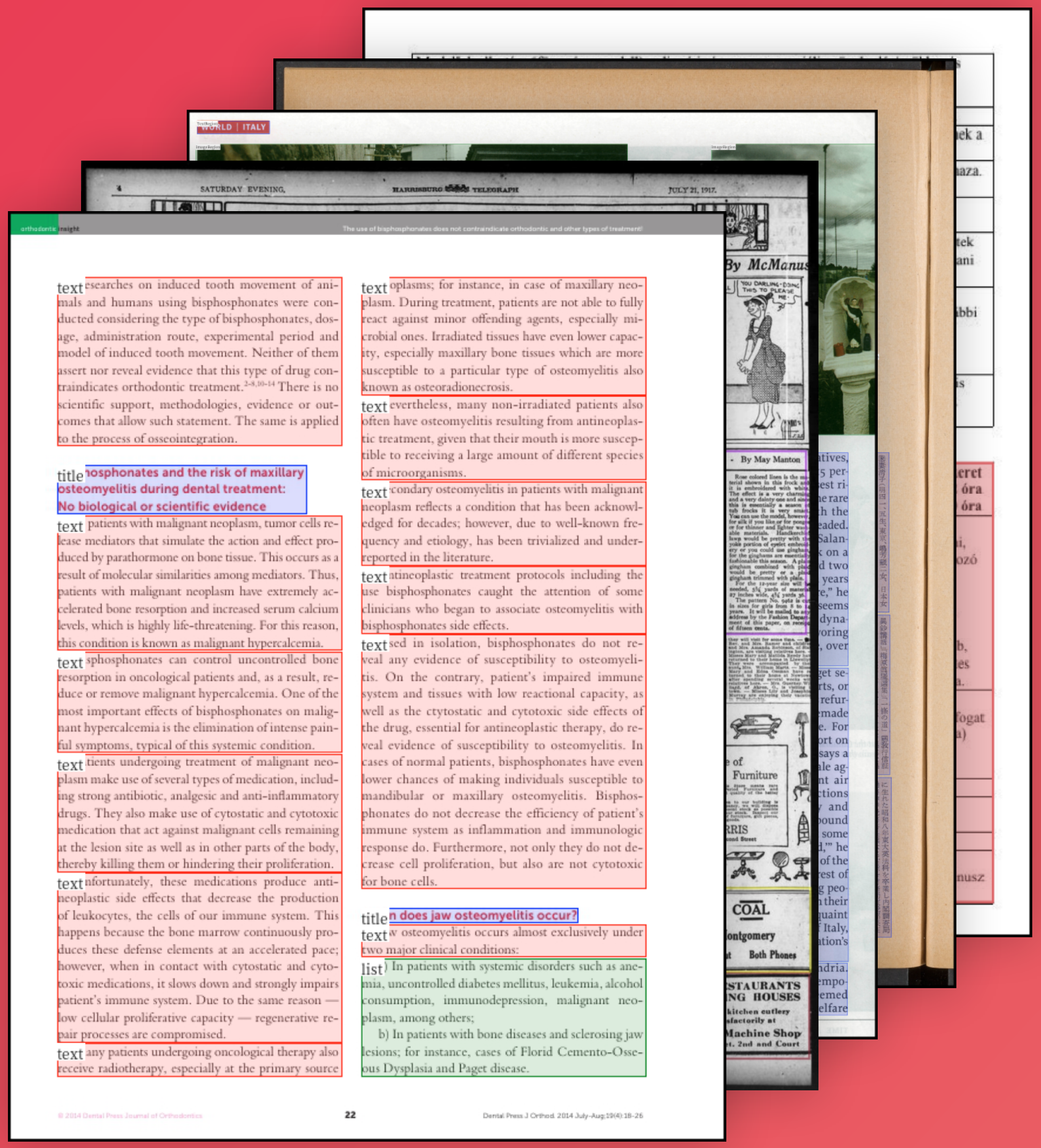
Output: Layout/Text

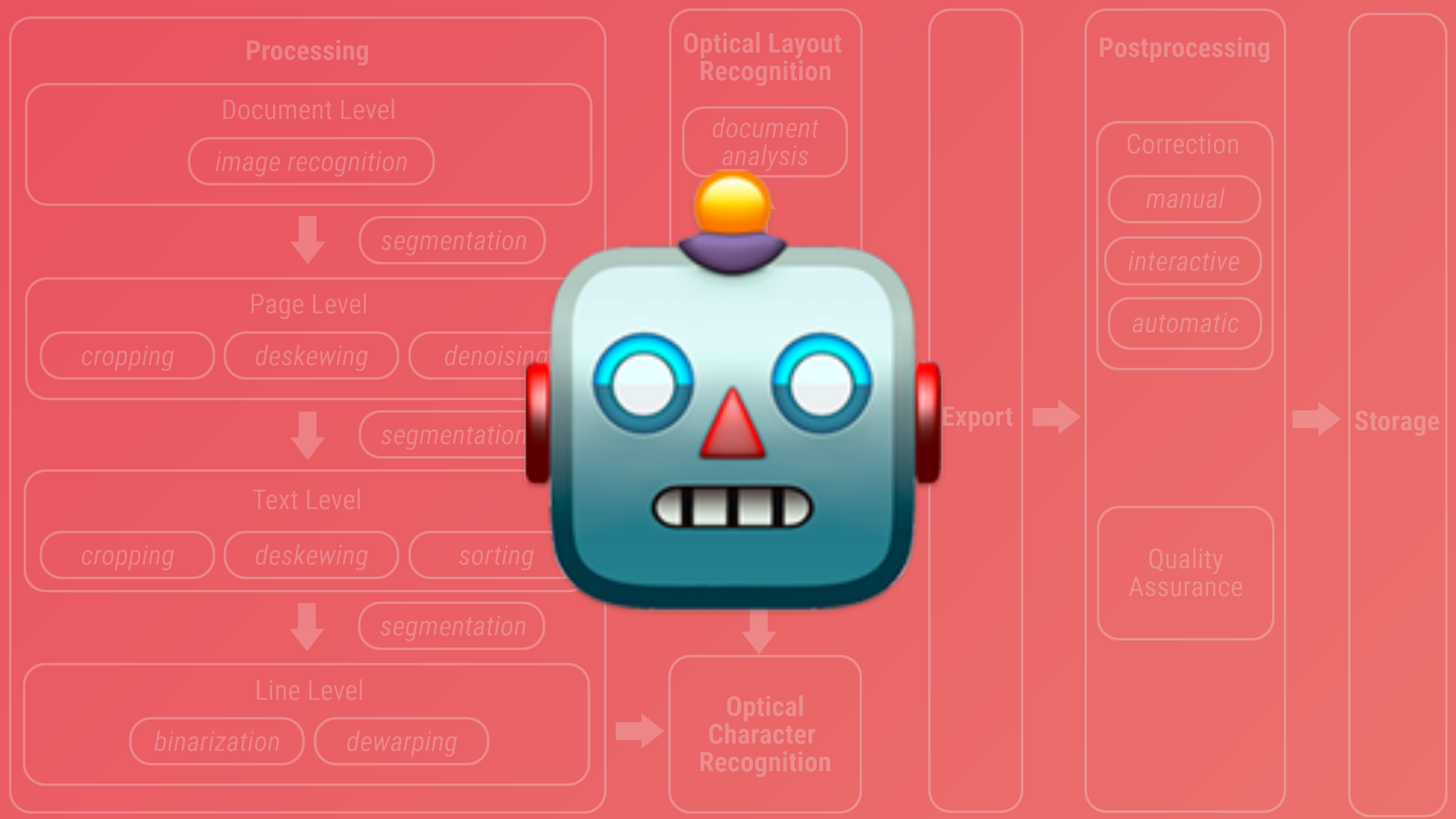
# Deep Learning for Document Image Analysis

The Task

Exciting Progress

Challenges






{
"title": "Construction of the Literature Graph in Semantic Scholar",
"authors": "Waleed Ammar et. al.",
"abstract": "We describe a deployed scalable system for organizing published scientific literature into a heterogeneous graph to facilitate algorithmic manipulation and ...",
"sections": ["..."]
}

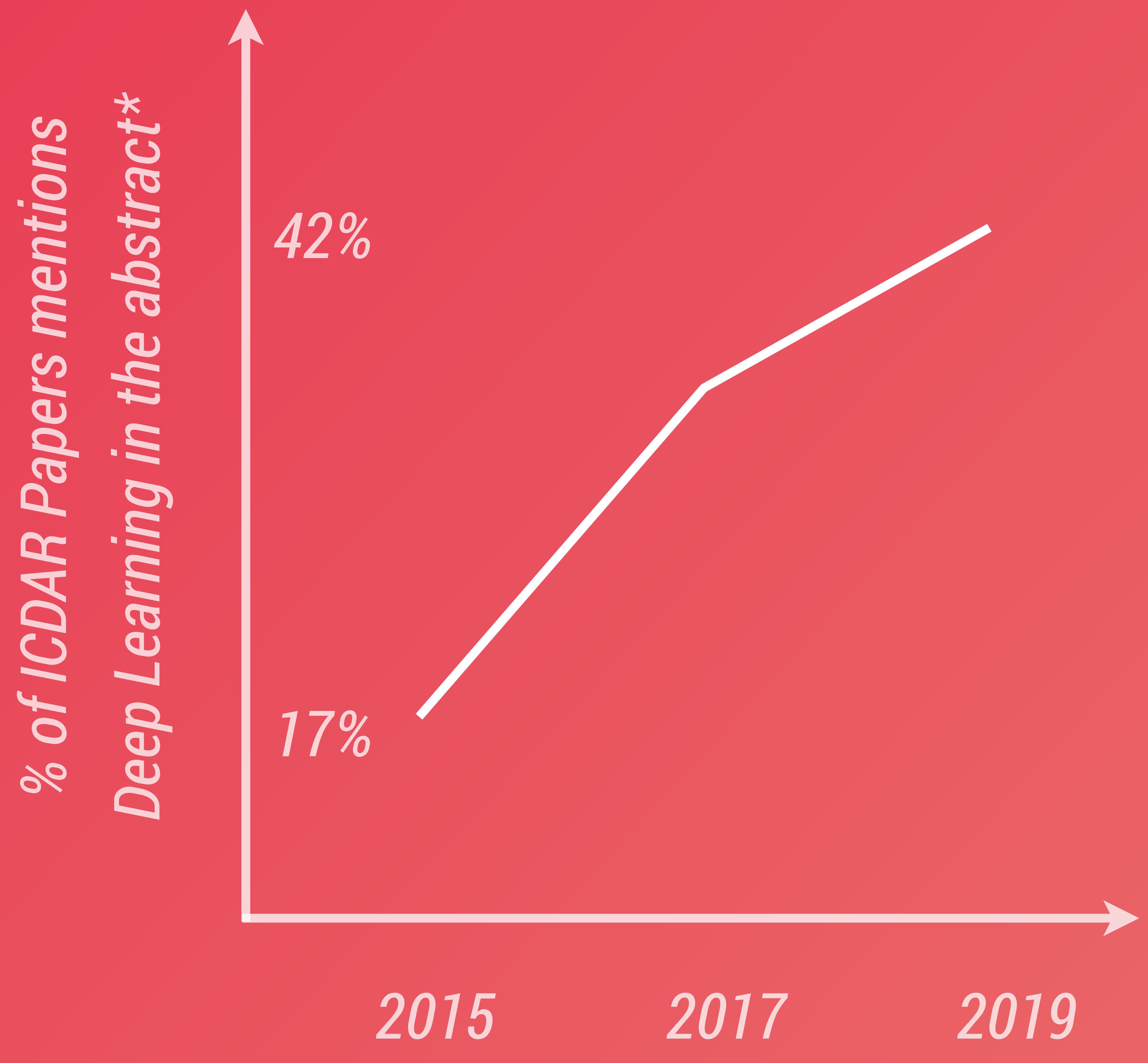
Input: Doc Images

Deep Learning Models

Output: Layout/Text

# Deep Learning for Document Image Analysis

The Task | Exciting Progress | Challenges




Large datasets

Better computation infrastructure

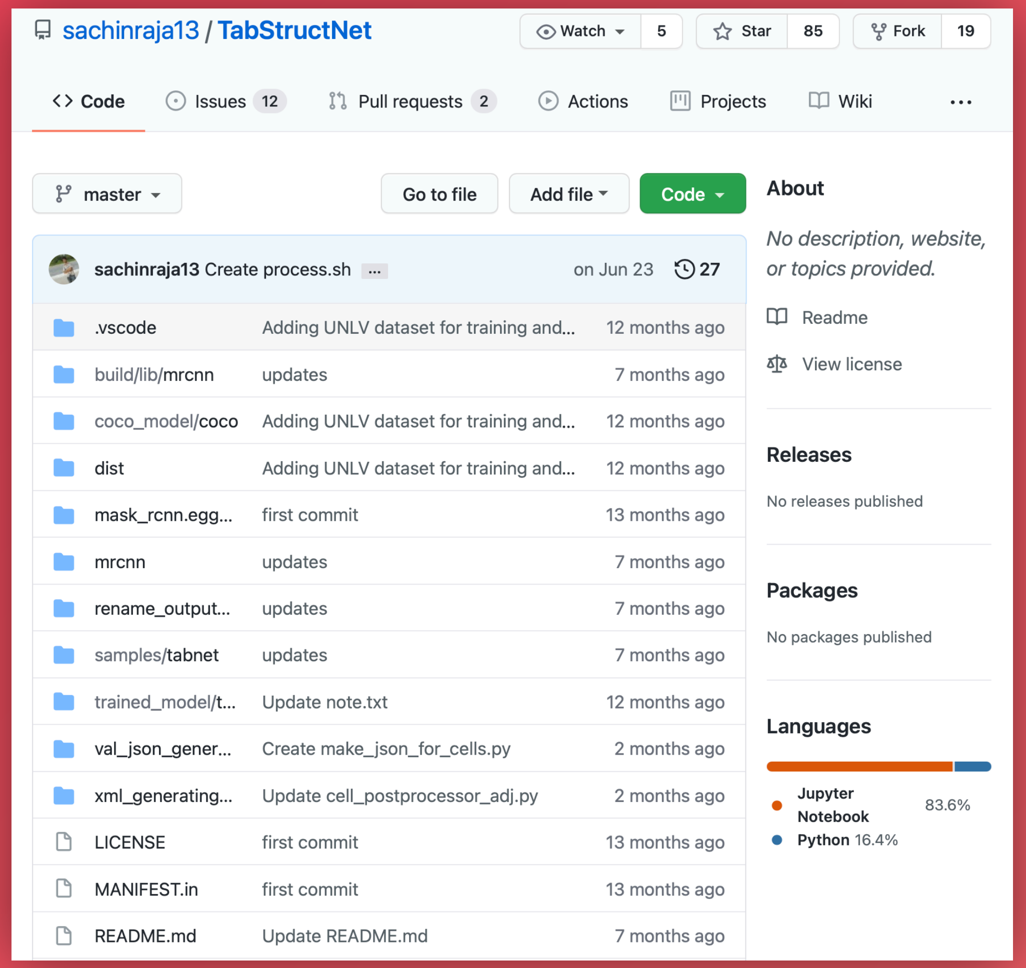
More work focus on DL, pushing SOTA

* We count papers with either Deep Learning, DNN, or Neural Network appeared in the abstract. Source data is from Semantic Scholar.

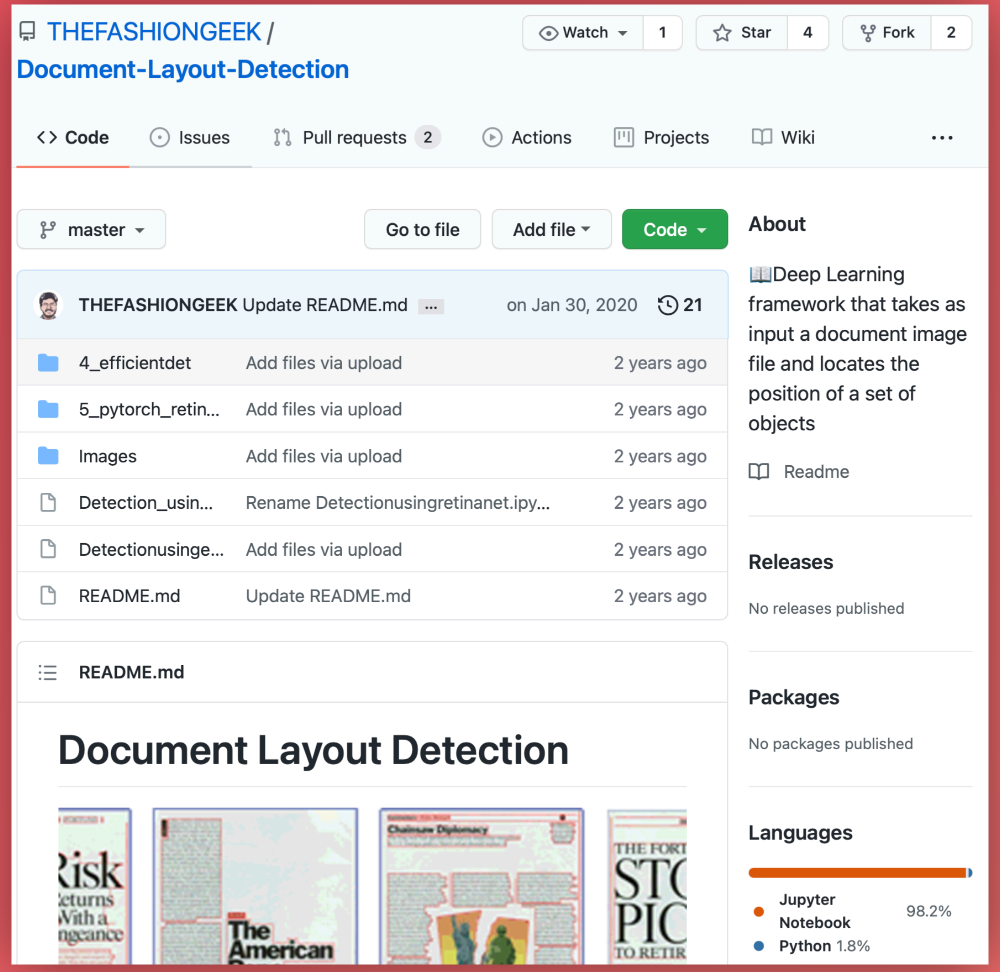
# Deep Learning for Document Image Analysis

The Task | Exciting Progress | **Challenges**

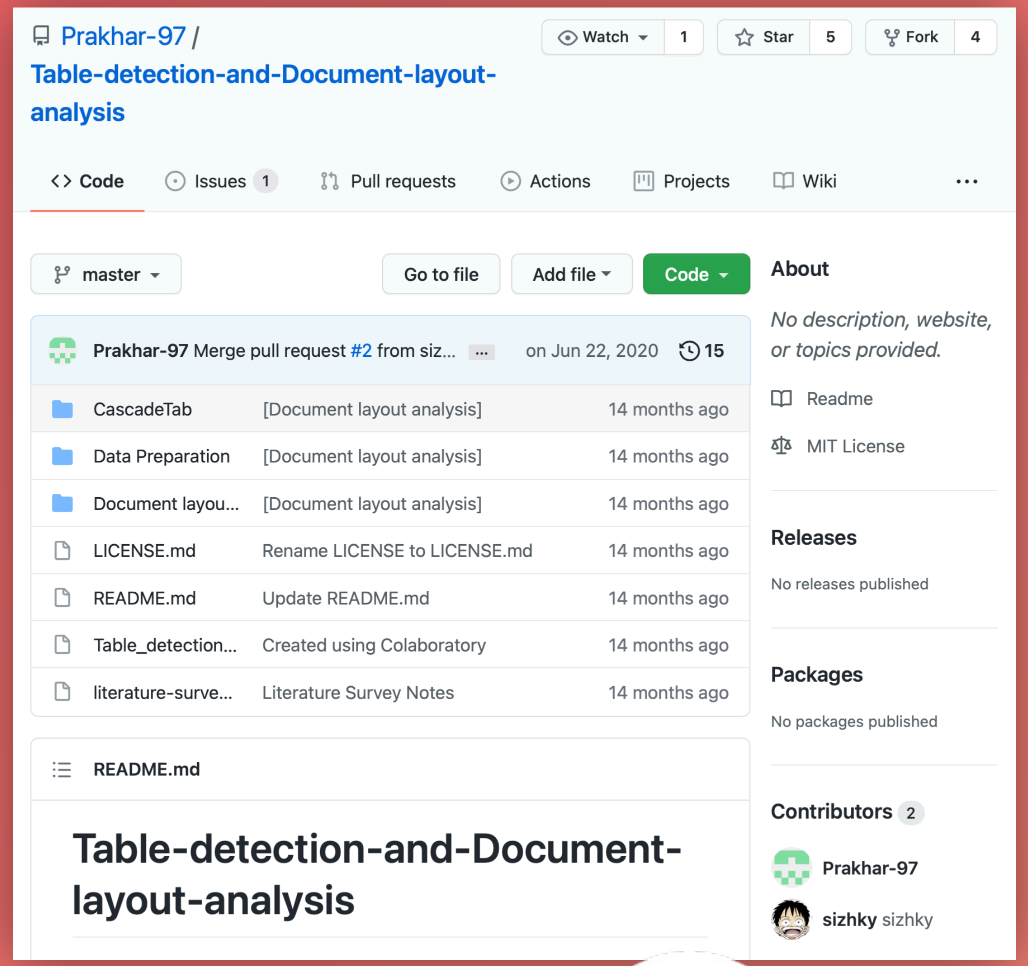
## Model code exists in different GitHub repos, using inconsistent DL backends & APIs




PyTorch




TensorFlow




mxnet

# Deep Learning for Document Image Analysis

The Task | Exciting Progress | **Challenges**

## Hard to be incorporated to existing DIA pipelines

| | | |
|---|---|---|
| *Preprocessing* | *Github Repo A* | *Java \| Environment A* |
| *Layout Detection* | *Github Repo B* | *Python \| Environment B* |
| *Character Recognition* | *Tesseract* | *C++/bash* |
| *Postprocessing* | *Github Repo C* | *MATLAB* |
| *Storage* | *PAGE Exporter* | *C++* |

# Deep Learning for Document Image Analysis

The Task | Exciting Progress | **Challenges**

## Hard to be incorporated to existing DIA pipelines

| | | |
|---|---|---|
| *Preprocessing* | *Github Repo A* | *Java \| Environment A* |
| *Layout Detection* | *Github Repo B* | *Python \| Environment B* |
| *Character Recognition* | *Tesseract* | *C++/bash* |
| *Postprocessing* | *Github Repo C* | *MATLAB* |
| *Storage* | *PAGE Exporter* | *C++* |

# Deep Learning for Document Image Analysis

The Task | Exciting Progress | **Challenges**

## Hard to be incorporated to existing DIA pipelines

| | | |
|---|---|---|
| *Preprocessing* | *Github Repo A* | *Java \| Environment A* |
| ***Layout Detection*** | ***Github Repo B*** | ***Python \| Environment B*** |
| *Character Recognition* | *Tesseract* | *C++/bash* |
| *Postprocessing* | *Github Repo C* | *MATLAB* |
| *Storage* | *PAGE Exporter* | *C++* |

# Deep Learning for Document Image Analysis

The Task | Exciting Progress | **Challenges**

## Hard to be incorporated to existing DIA pipelines

| | | |
|---|---|---|
| *Preprocessing* | *Github Repo A* | *Java \| Environment A* |
| *Layout Detection* | *Github Repo B* | *Python \| Environment B* |
| ***Character Recognition*** | ***Tesseract*** | ***C++/bash*** |
| *Postprocessing* | *Github Repo C* | *MATLAB* |
| *Storage* | *PAGE Exporter* | *C++* |

# Deep Learning for Document Image Analysis

The Task | Exciting Progress | **Challenges**

## Hard to be incorporated to existing DIA pipelines

| | | |
|---|---|---|
| *Preprocessing* | *Github Repo A* | *Java \| Environment A* |
| *Layout Detection* | *Github Repo B* | *Python \| Environment B* |
| *Character Recognition* | *Tesseract* | *C++/bash* |
| ***Postprocessing*** | ***Github Repo C*** | ***MATLAB*** |
| *Storage* | *PAGE Exporter* | *C++* |

# Deep Learning for Document Image Analysis

The Task | Exciting Progress | **Challenges**

## Hard to be incorporated to existing DIA pipelines

| | | |
|---|---|---|
| *Preprocessing* | *Github Repo A* | *Java \| Environment A* |
| *Layout Detection* | *Github Repo B* | *Python \| Environment B* |
| *Character Recognition* | *Tesseract* | *C++/bash* |
| *Postprocessing* | *Github Repo C* | *MATLAB* |
| ***Storage*** | ***PAGE Exporter*** | ***C++*** |

# Deep Learning for Document Image Analysis

The Task | Exciting Progress | Challenges

## The research advances becomes less accessible

### for DIA researchers

*hard to reproduce the results & improve models*

### for end users

*who might come from non-technical backgrounds*

# What should an ideal toolkit be?

Simple

Comprehensive

Customizable

Extensible

Open Platform

# Layout Parser

Simple

Comprehensive

Customizable

Extensible

Open Platform

Motivation

Demo

Design & Implementation

Future Work

Community

# Layout Parser usage example

Installation | Layout Detection | Optical Character Recognition

```
$ pip install layoutparser
$ python
>>> import layoutparser as lp
>>> # Ready to go!
```

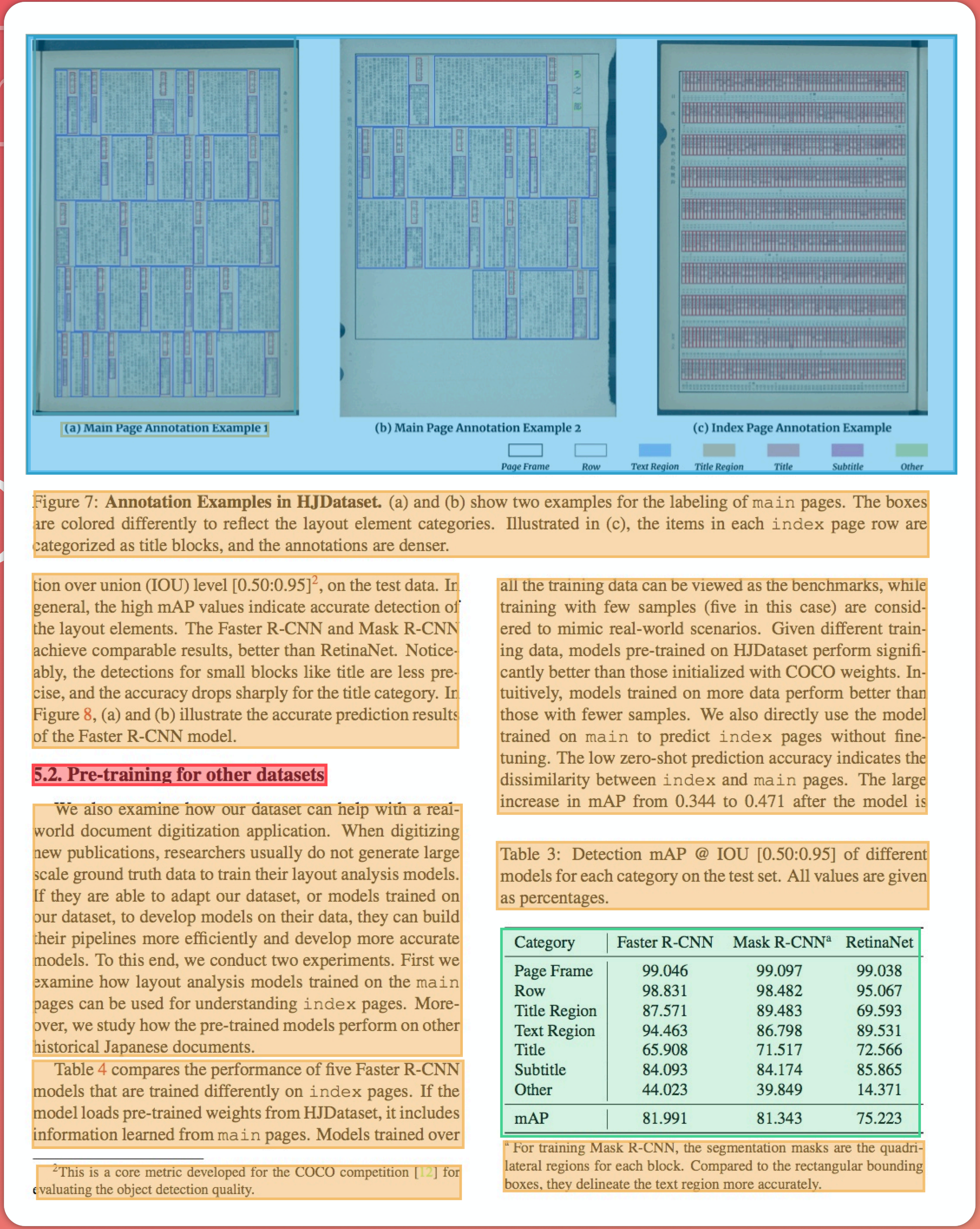
# Layout Parser usage example

Installation | Layout Detection | Optical Character

```
>>> model =
    lp.Detectron2LayoutModel()

>>> image = load_image()

>>> layout =
    model.detect(image)
```



(a) Main Page Annotation Example 1 (b) Main Page Annotation Example 2 (c) Index Page Annotation Example

Page Frame Row Text Region Title Region Title Subtitle Other

Figure 7: **Annotation Examples in HJDataset.** (a) and (b) show two examples for the labeling of `main` pages. The boxes are colored differently to reflect the layout element categories. Illustrated in (c), the items in each `index` page row are categorized as title blocks, and the annotations are denser.

tion over union (IOU) level [0.50:0.95][2], on the test data. In general, the high mAP values indicate accurate detection of the layout elements. The Faster R-CNN and Mask R-CNN achieve comparable results, better than RetinaNet. Noticeably, the detections for small blocks like title are less precise, and the accuracy drops sharply for the title category. In Figure 8, (a) and (b) illustrate the accurate prediction results of the Faster R-CNN model.

### 5.2. Pre-training for other datasets

We also examine how our dataset can help with a real-world document digitization application. When digitizing new publications, researchers usually do not generate large scale ground truth data to train their layout analysis models. If they are able to adapt our dataset, or models trained on our dataset, to develop models on their data, they can build their pipelines more efficiently and develop more accurate models. To this end, we conduct two experiments. First we examine how layout analysis models trained on the `main` pages can be used for understanding `index` pages. Moreover, we study how the pre-trained models perform on other historical Japanese documents.

Table 4 compares the performance of five Faster R-CNN models that are trained differently on `index` pages. If the model loads pre-trained weights from HJDataset, it includes information learned from `main` pages. Models trained over

[2]This is a core metric developed for the COCO competition [12] for evaluating the object detection quality.

all the training data can be viewed as the benchmarks, while training with few samples (five in this case) are considered to mimic real-world scenarios. Given different training data, models pre-trained on HJDataset perform significantly better than those initialized with COCO weights. Intuitively, models trained on more data perform better than those with fewer samples. We also directly use the model trained on `main` to predict `index` pages without fine-tuning. The low zero-shot prediction accuracy indicates the dissimilarity between `index` and `main` pages. The large increase in mAP from 0.344 to 0.471 after the model is

Table 3: Detection mAP @ IOU [0.50:0.95] of different models for each category on the test set. All values are given as percentages.

| Category | Faster R-CNN | Mask R-CNN[a] | RetinaNet |
|---|---|---|---|
| Page Frame | 99.046 | 99.097 | 99.038 |
| Row | 98.831 | 98.482 | 95.067 |
| Title Region | 87.571 | 89.483 | 69.593 |
| Text Region | 94.463 | 86.798 | 89.531 |
| Title | 65.908 | 71.517 | 72.566 |
| Subtitle | 84.093 | 84.174 | 85.865 |
| Other | 44.023 | 39.849 | 14.371 |
| mAP | 81.991 | 81.343 | 75.223 |

[a] For training Mask R-CNN, the segmentation masks are the quadrilateral regions for each block. Compared to the rectangular bounding boxes, they delineate the text region more accurately.

# Layout Parser usage example

Installation | Layout Detection | Optical Character

```
>>> ocr_model =
    lp.TesseractAgent()

>>> ocr_text =
    ocr_model.detect(image)

>>> ocr_text.to_json()
```

(a) Main Page Annotation Example 1 (b) Main Page Annotation Example 2 (c) Index Page Annotation Example

Co Co PageFrame «Row —Text Region TitleRegion —Title. = Subtitle Other

Figure 7: Annotation Examples in HJDataset. (a) and (b) show two examples for the labeling of main pages. The boxes are colored differently to reflect the layout element categories. Illustrated in (c), the items in each index page row are categorized as title blocks, and the annotations are denser.

tion over union (IOU) level [0.50:0.95]', on the test data. In general, the high mAP values indicate accurate detection of the layout elements. The Faster R-CNN and Mask R-CNN achieve comparable results, better than RetinaNet. Noticeably, the detections for small blocks like title are less precise, and the accuracy drops sharply for the title category. In Figure 8, (a) and (b) illustrate the accurate prediction results of the Faster R-CNN model.

5.2. Pre-training for other datasets

We also examine how our dataset can help with a real-world document digitization application. When digitizing new publications, researchers usually do not generate large scale ground truth data to train their layout analysis models. If they are able to adapt our dataset, or models trained on our dataset, to develop models on their data, they can build their pipelines more efficiently and develop more accurate models. To this end, we conduct two experiments. First we examine how layout analysis models trained on the main pages can be used for understanding index pages. Moreover, we study how the pre-trained models perform on other historical Japanese documents.

Table 4 compares the performance of five Faster R-CNN models that are trained differently on index pages. If the model loads pre-trained weights from HJDataset, it includes information learned from main pages. Models trained over

2This is a core metric developed for the COCO competition [] for evaluating the object detection quality.

all the training data can be viewed as the benchmarks, while training with few samples (five in this case) are considered to mimic real-world scenarios. Given different trai ing data, models pre-trained on HJDataset perform significantly better than those initialized with COCO weights. Intuitively, models trained on more data perform better than those with fewer samples. We also directly use the model trained on main to predict index pages without fine-tuning. The low zero-shot prediction accuracy indicates the dissimilarity between index and main pages. The large increase in mAP from 0.344 to 0.471 after the model is

Table 3: Detection mAP @ IOU [0.50:0.95] of different models for each category on the test set. All values are given as percentages.

| Category | FasterR-CNN | Mask R-CNN* | RetinaNet |
|---|---|---|---|
| Page Frame | 99.046 | 99.097 | 99.038 |
| Row | 98.831 | 98.482 | 95.067 |
| Title Region | 87.571 | 89.483 | 69.593 |
| Text Region | 94.463 | 86.798 | 89.531 |
| Title | 65.908 | 71.517 | 72.566 |
| Subtitle | 84.093 | 84.174 | 85.865 |
| Other | 44.023 | 39.849 | 14.371 |
| mAP | 81.991 | 81.343 | 75.223 |

*For training Mask R-CNN, the segmentation masks are the quadrilateral regions for each block. Compared to the rectangular bounding boxes, they delineate the text region more accurately.

Motivation

Demo

**Design & Implementation**

Future Work

Community

Deep Layout Models
Rich Repository of Pre-trained Models
Simple Model Usage

Layout Data Structure
Layout Visualization
OCR Engine Support
Data Import and Export

Open-the-box Usage

# *Design & Implementation*

Modularized Design

Layout Data Annotation
DL Models Training
Multi-backend support

Tutorials & Examples
Sharing Platform
Community Support

*Layout Data Annotation*

*Deep Layout Models*

*Sharing Platform*

*DL Models Training*

*Simple Model Usage*

*Tutorials & Examples*

*Multi-backend support*

*Layout Model Zoo*

*Community Support*

*Layout Data Structure*

*Layout Visualization*

*OCR Engine Support*

*Data Import and Export*

Layout Data Annotation

**Model Customization**

DL Models Training

Multi-backend support

**Deep Learning Models for Layout Detection**

Deep Layout Models

Sharing Platform

**Layout Parser Open Platform**

Tutorials & Examples

Community Support

Layout Data Structure

**Infrastructure APIs**

Layout Visualization

OCR Engine Support

Data Import and Export

# Deep Learning Models for Layout Detection

Layout Data Annotation
Deep Learning Model Customization
DL Models Training
Multi-backend support

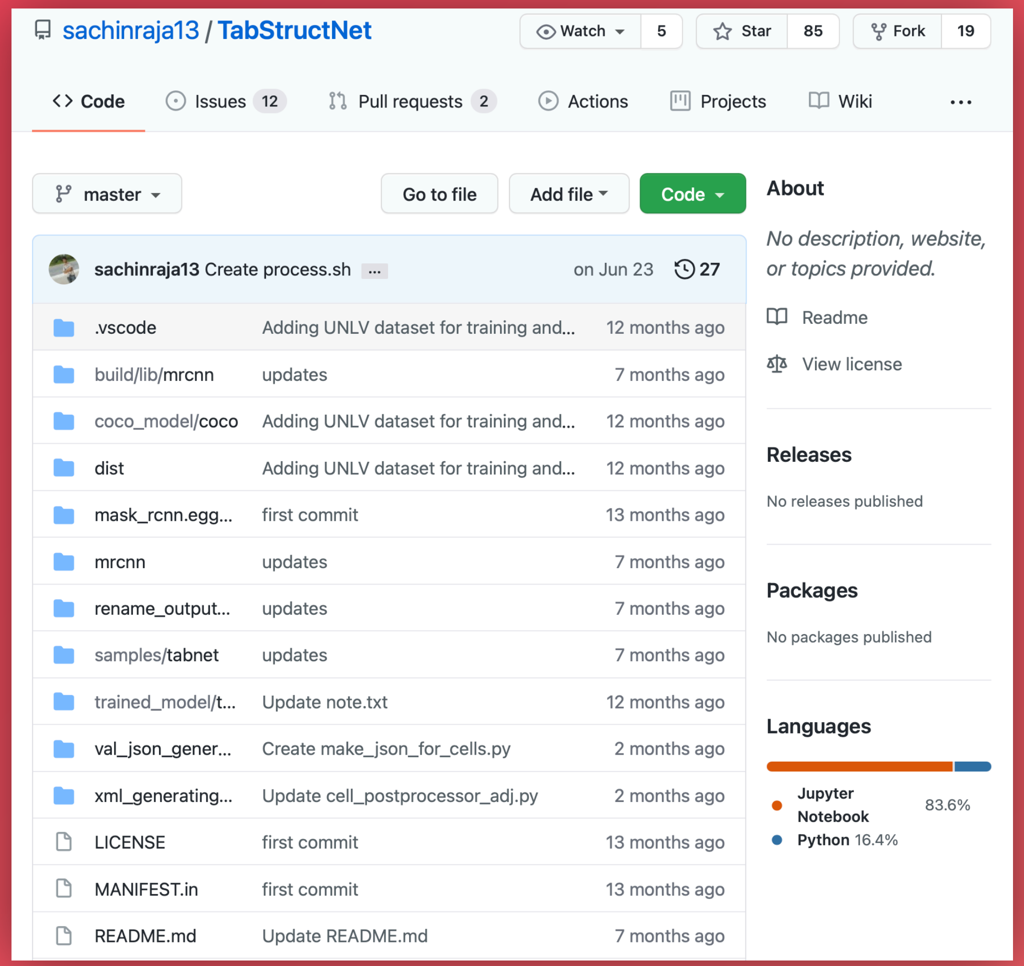
Sharing Platform
Layout Parser Open Platform
Tutorials & Examples
Community Support

Layout Data Structure
Core APIs
Layout Visualization
OCR Engine Support
Data Import and Export

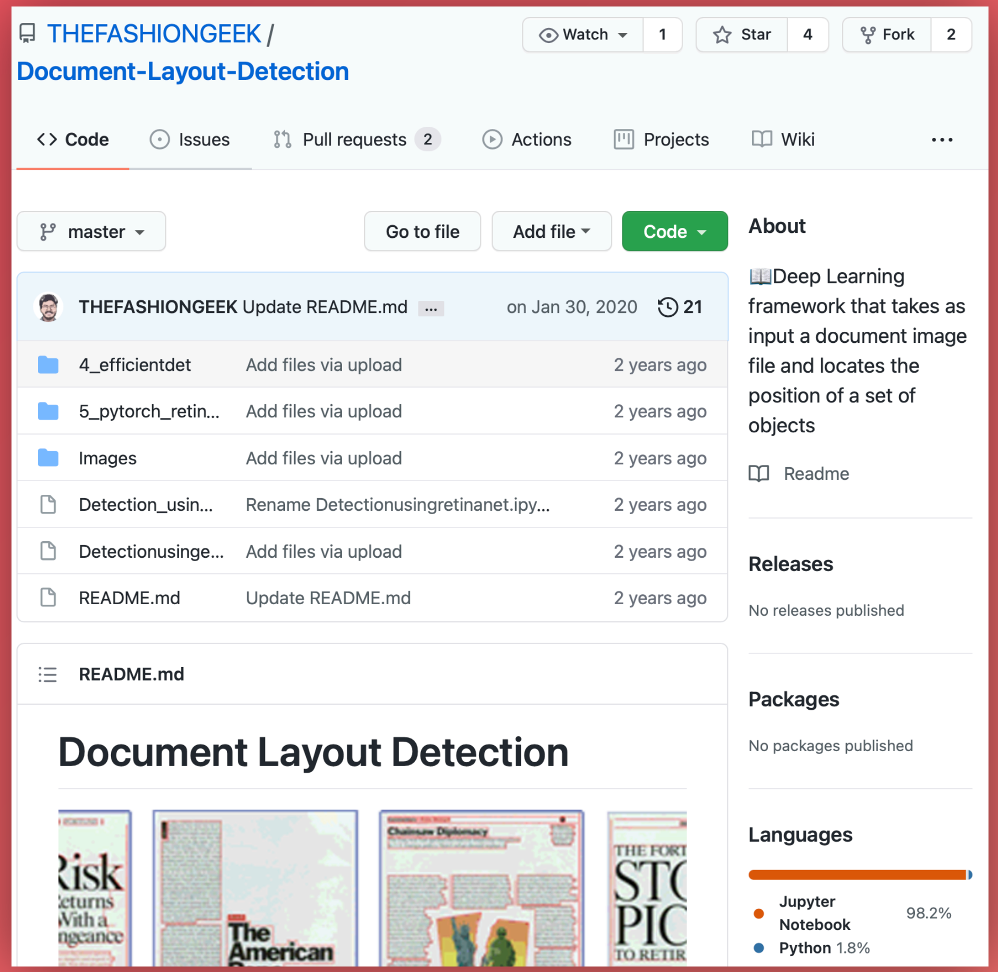
# Deep Learning Models for Layout Detection

Challenges | Standardized Modeling API | Model Zoo

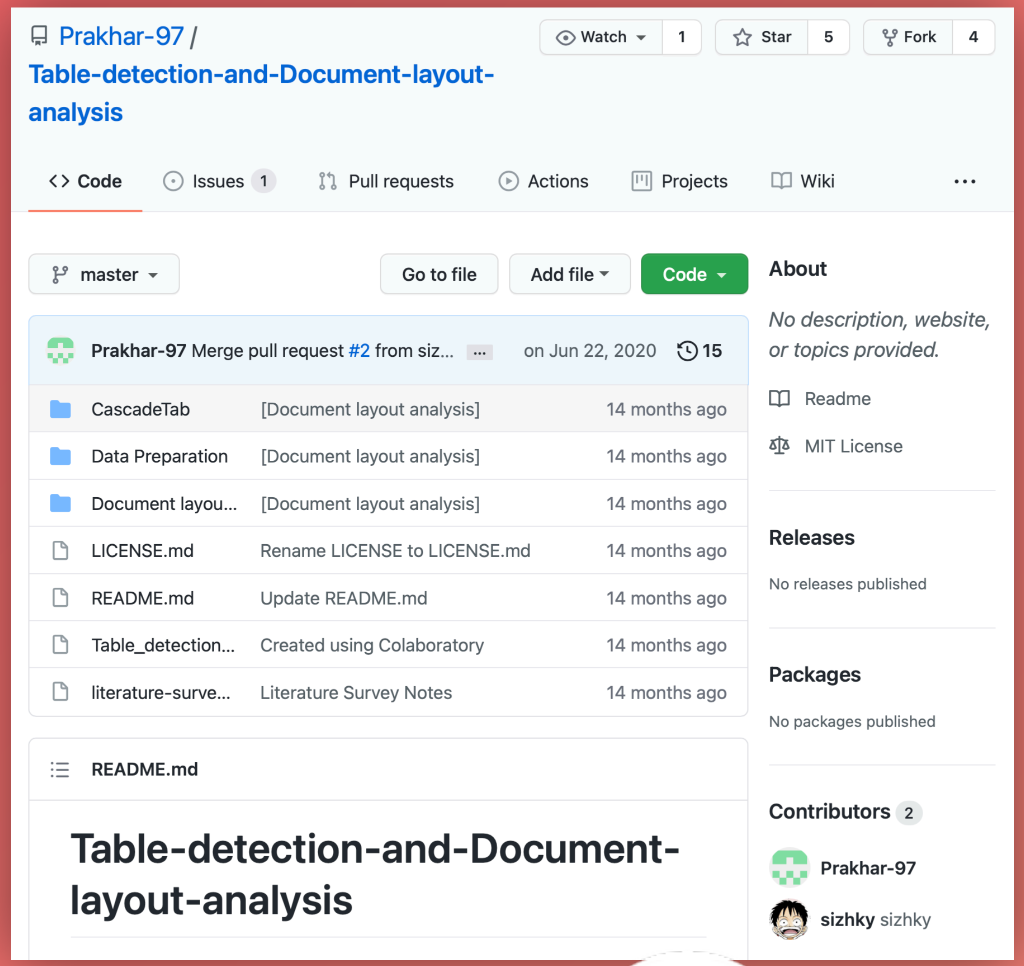
## No standard way for sharing and re-using existing models




PyTorch




TensorFlow




mxnet

# Deep Learning Models for Layout Detection

Challenges

Standardized Modeling API

Model Zoo

# Deep Learning Models for Layout Detection

Challenges | **Standardized Modeling API** | Model Zoo

```
>>> config = "lp://PubLayNet/mask_rcnn_R_50_FPN_3x/config"
>>> model = lp.Detectron2LayoutModel(config)
>>> layout = model.detect(image)
```

# Deep Learning Models for Layout Detection

Challenges | Standardized Modeling API | Model Zoo

▼ *Specify the model configuration*

```
>>> config = "lp://PubLayNet/mask_rcnn_R_50_FPN_3x/config"
>>> model = lp.Detectron2LayoutModel(config)
>>> layout = model.detect(image)
```

# Deep Learning Models for Layout Detection

Challenges | **Standardized Modeling API** | Model Zoo

▼ *Training Dataset Name*

```
>>> config = "lp://PubLayNet/mask_rcnn_R_50_FPN_3x/config"
>>> model = lp.Detectron2LayoutModel(config)
>>> layout = model.detect(image)
```

# Deep Learning Models for Layout Detection

Challenges | **Standardized Modeling API** | Model Zoo

▼ *Model Architecture Name*

```
>>> config = "lp://PubLayNet/mask_rcnn_R_50_FPN_3x/config"

>>> model = lp.Detectron2LayoutModel(config)

>>> layout = model.detect(image)
```

# Deep Learning Models for Layout Detection

Challenges | **Standardized Modeling API** | Model Zoo

```
>>> config = "lp://PubLayNet/mask_rcnn_R_50_FPN_3x/config"
>>> model = lp.Detectron2LayoutModel(config)
>>> layout = model.detect(image)
```

▲ *Standardized Model Initalization*

# Deep Learning Models for Layout Detection

Challenges | Standardized Modeling API | Model Zoo

```
>>> config = "lp://PubLayNet/mask_rcnn_R_50_FPN_3x/config"
>>> model = lp.Detectron2LayoutModel(config)
>>> layout = model.detect(image)
```

▲ *The Deep Learning Backend Name*

# Deep Learning Models for Layout Detection

Challenges | **Standardized Modeling API** | Model Zoo

```
>>> config = "lp://PubLayNet/mask_rcnn_R_50_FPN_3x/config"
>>> model = lp.Detectron2LayoutModel(config)
>>> layout = model.detect(image)
```

▲ *Standardized Model Detection API*

# Deep Learning Models for Layout Detection

Challenges | Standardized Modeling API | Model Zoo

## What if we want to make some changes?

```
>>> config = "lp://PubLayNet/mask_rcnn_R_50_FPN_3x/config"
>>> model = lp.Detectron2LayoutModel(config)
>>> layout = model.detect(image)
```

# Deep Learning Models for Layout Detection

Challenges | Standardized Modeling API | Model Zoo

## Switch to a different model architecture?

```
>>> config = "lp://PubLayNet/faster_rcnn_R_50_FPN_3x/config"
>>> model = lp.Detectron2LayoutModel(config)
>>> layout = model.detect(image)
```

# Deep Learning Models for Layout Detection

Challenges | Standardized Modeling API | Model Zoo

Switch to models trained on another dataset?

```
>>> config = "lp://PrimaLayout/mask_rcnn_R_50_FPN_3x/config"
>>> model = lp.Detectron2LayoutModel(config)
>>> layout = model.detect(image)
```

# Deep Learning Models for Layout Detection

Challenges | **Standardized Modeling API** | Model Zoo

Switch to a different model architecture from another DL backend?

```
>>> config = "lp://PubLayNet/ppyolov2_r50vd_dcn_365e/config"
>>> model = lp.PaddleDetectionLayoutModel(config)
>>> layout = model.detect(image)
```

# Deep Learning Models for Layout Detection

Challenges | Standardized Modeling API | Model Zoo

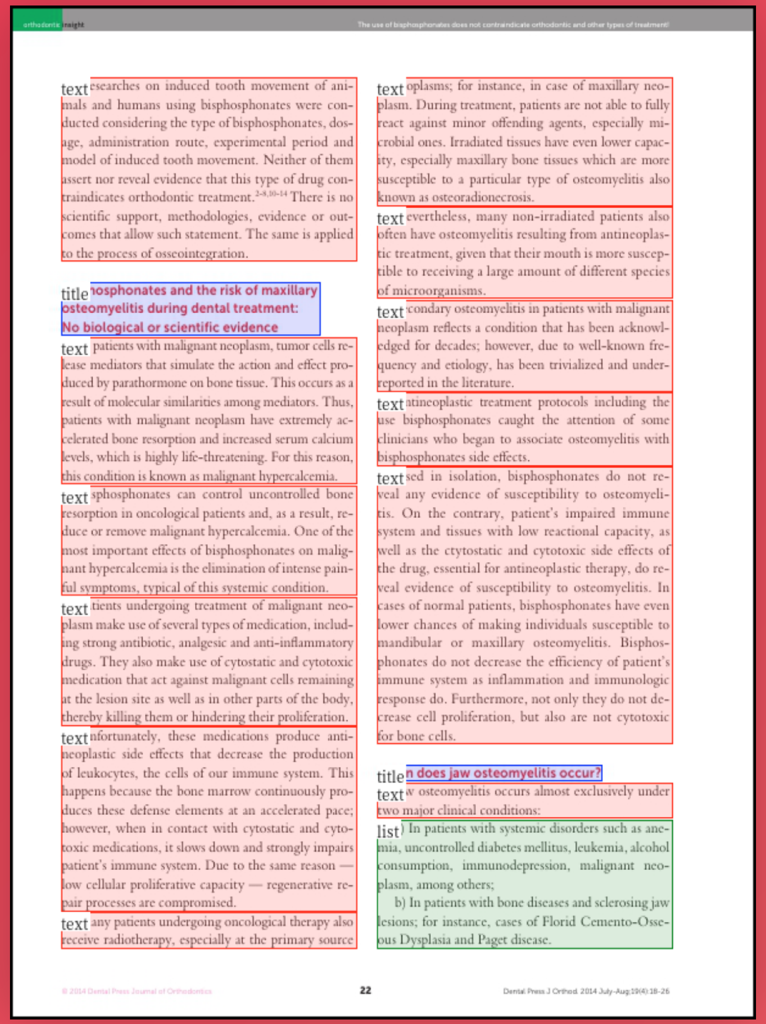
Even simpler!

```
>>> model = lp.AutoLayoutModel("lp://detectron2/publaynet")
```

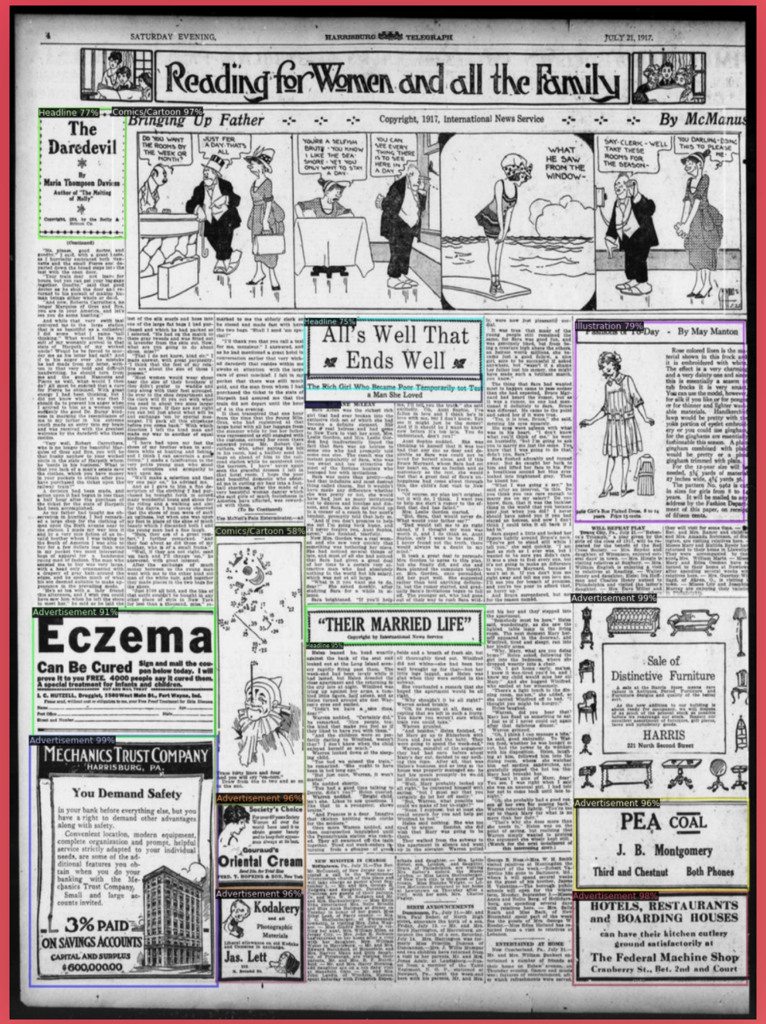
# Deep Learning Models for Layout Detection

Challenges | Standardized Modeling API | Model Zoo

## Layout Parser has pre-trained models on 6 datasets, including:

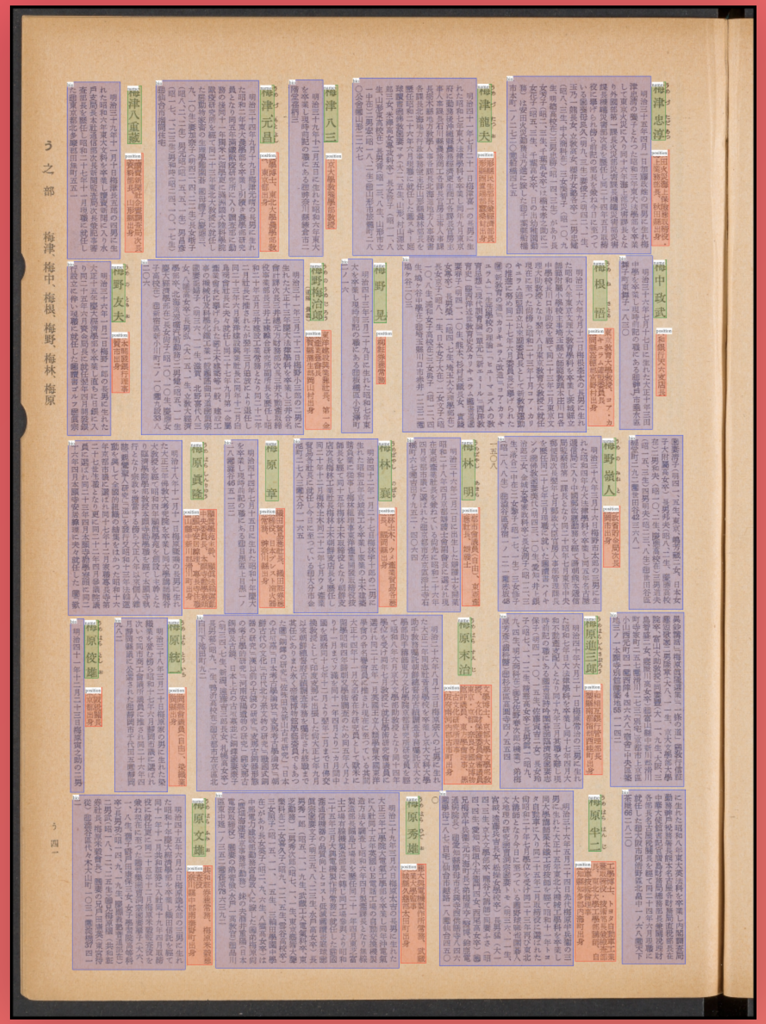


PubLayNet

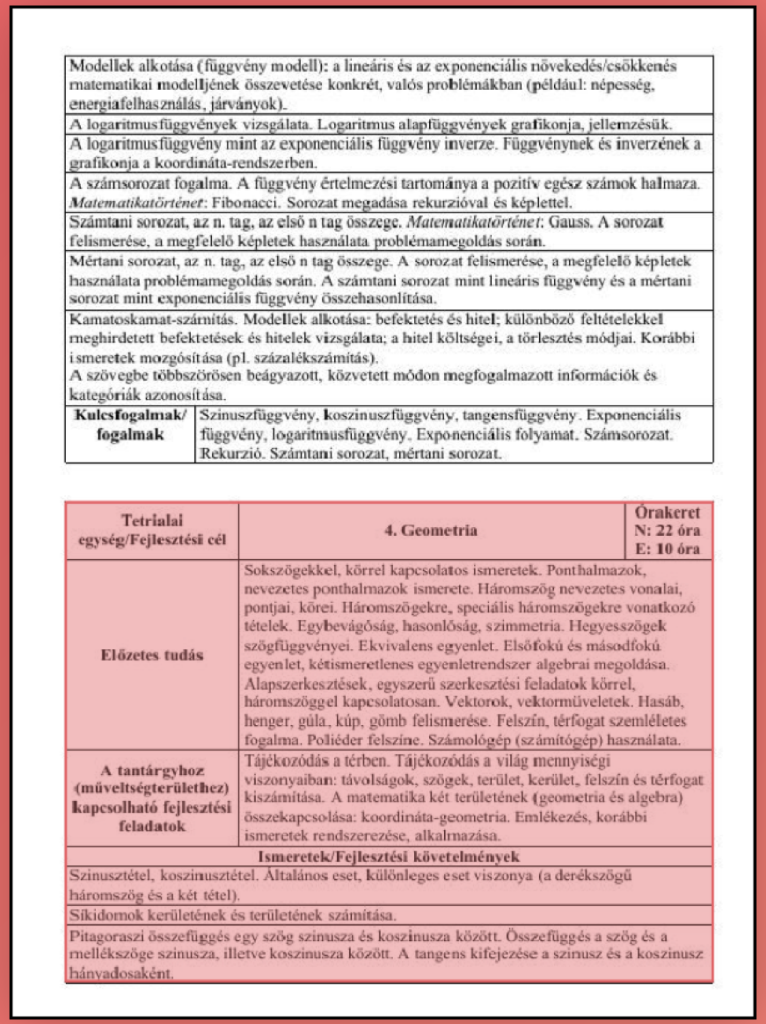


Newspaper Navigator

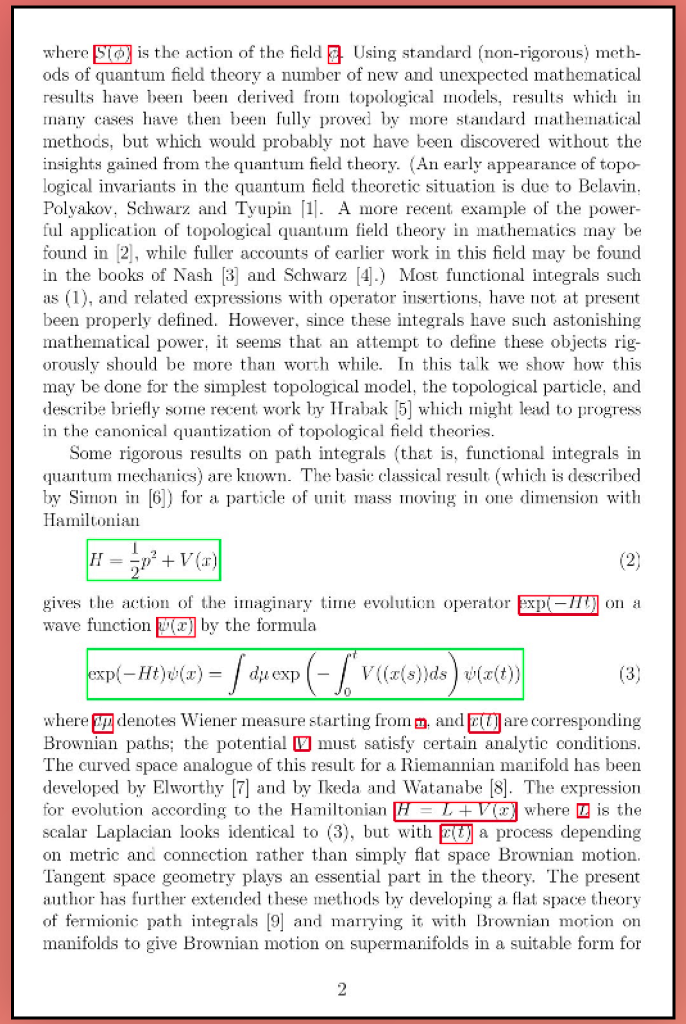
PRImA Layout



HJDataset



TableBank



Math Formula Detection

*What if we need to post-process model outputs?*

Deep Learning Model Customization

Layout Data Annotation

DL Models Training

Multi-backend support

**Deep Learning Models for Layout Detection**

Deep Layout Models

Simple Model Usage

Layout Model Zoo

Layout Parser Open Platform

Sharing Platform

Tutorials & Examples

Community Support

**Infrastructure APIs**

Layout Data Structure

Layout Visualization

OCR Engine Support

Data Import and Export

# Layout Parser Infrastructure APIs

Layout Data Structures | OCR Engine Support | Layout Visualization | Load & Storage

## Convenient APIs that simplify postprocessing

```
>>> layout = model.detect(image)
```

# Layout Parser Infrastructure APIs

Layout Data Structures | OCR Engine Support | Layout Vi

## Select layout regions in the left column / Coherent APIs that simplify postprocessing

```
>>> layout = model.detect(image)

>>> width, height = image.size

>>> left_column = \
    lp.Interval(0, width/2, axis="x")
```

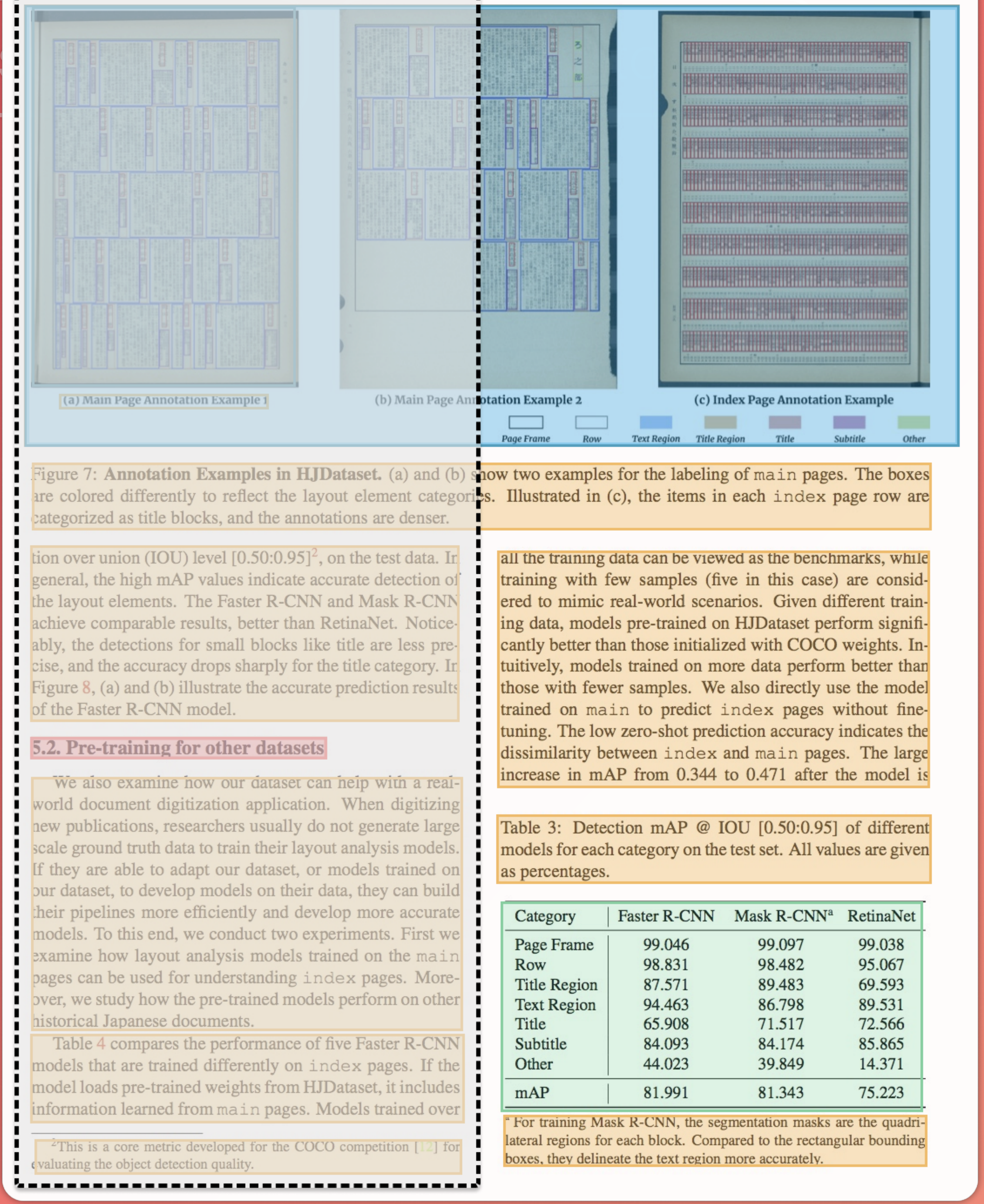



Figure 7: Annotation Examples in HJDataset. (a) and (b) show two examples for the labeling of main pages. The boxes are colored differently to reflect the layout element categories. Illustrated in (c), the items in each index page row are categorized as title blocks, and the annotations are denser.

tion over union (IOU) level [0.50:0.95][2], on the test data. In general, the high mAP values indicate accurate detection of the layout elements. The Faster R-CNN and Mask R-CNN achieve comparable results, better than RetinaNet. Noticeably, the detections for small blocks like title are less precise, and the accuracy drops sharply for the title category. In Figure 8, (a) and (b) illustrate the accurate prediction results of the Faster R-CNN model.

### 5.2. Pre-training for other datasets

We also examine how our dataset can help with a real-world document digitization application. When digitizing new publications, researchers usually do not generate large scale ground truth data to train their layout analysis models. If they are able to adapt our dataset, or models trained on our dataset, to develop models on their data, they can build their pipelines more efficiently and develop more accurate models. To this end, we conduct two experiments. First we examine how layout analysis models trained on the main pages can be used for understanding index pages. Moreover, we study how the pre-trained models perform on other historical Japanese documents.

Table 4 compares the performance of five Faster R-CNN models that are trained differently on index pages. If the model loads pre-trained weights from HJDataset, it includes information learned from main pages. Models trained over all the training data can be viewed as the benchmarks, while training with few samples (five in this case) are considered to mimic real-world scenarios. Given different training data, models pre-trained on HJDataset perform significantly better than those initialized with COCO weights. Intuitively, models trained on more data perform better than those with fewer samples. We also directly use the model trained on main to predict index pages without fine-tuning. The low zero-shot prediction accuracy indicates the dissimilarity between index and main pages. The large increase in mAP from 0.344 to 0.471 after the model is

Table 3: Detection mAP @ IOU [0.50:0.95] of different models for each category on the test set. All values are given as percentages.

| Category | Faster R-CNN | Mask R-CNN[a] | RetinaNet |
|---|---|---|---|
| Page Frame | 99.046 | 99.097 | 99.038 |
| Row | 98.831 | 98.482 | 95.067 |
| Title Region | 87.571 | 89.483 | 69.593 |
| Text Region | 94.463 | 86.798 | 89.531 |
| Title | 65.908 | 71.517 | 72.566 |
| Subtitle | 84.093 | 84.174 | 85.865 |
| Other | 44.023 | 39.849 | 14.371 |
| mAP | 81.991 | 81.343 | 75.223 |

[a] For training Mask R-CNN, the segmentation masks are the quadrilateral regions for each block. Compared to the rectangular bounding boxes, they delineate the text region more accurately.

[2] This is a core metric developed for the COCO competition [ ] for evaluating the object detection quality.

# Layout Parser Infrastructure APIs

Layout Data Structures | OCR Engine Support | Layout Vi

## Select layout regions in the left column:

```
>>> selected = layout.filter_by(
    left_column, center=True)

>>> selected.sort()
```

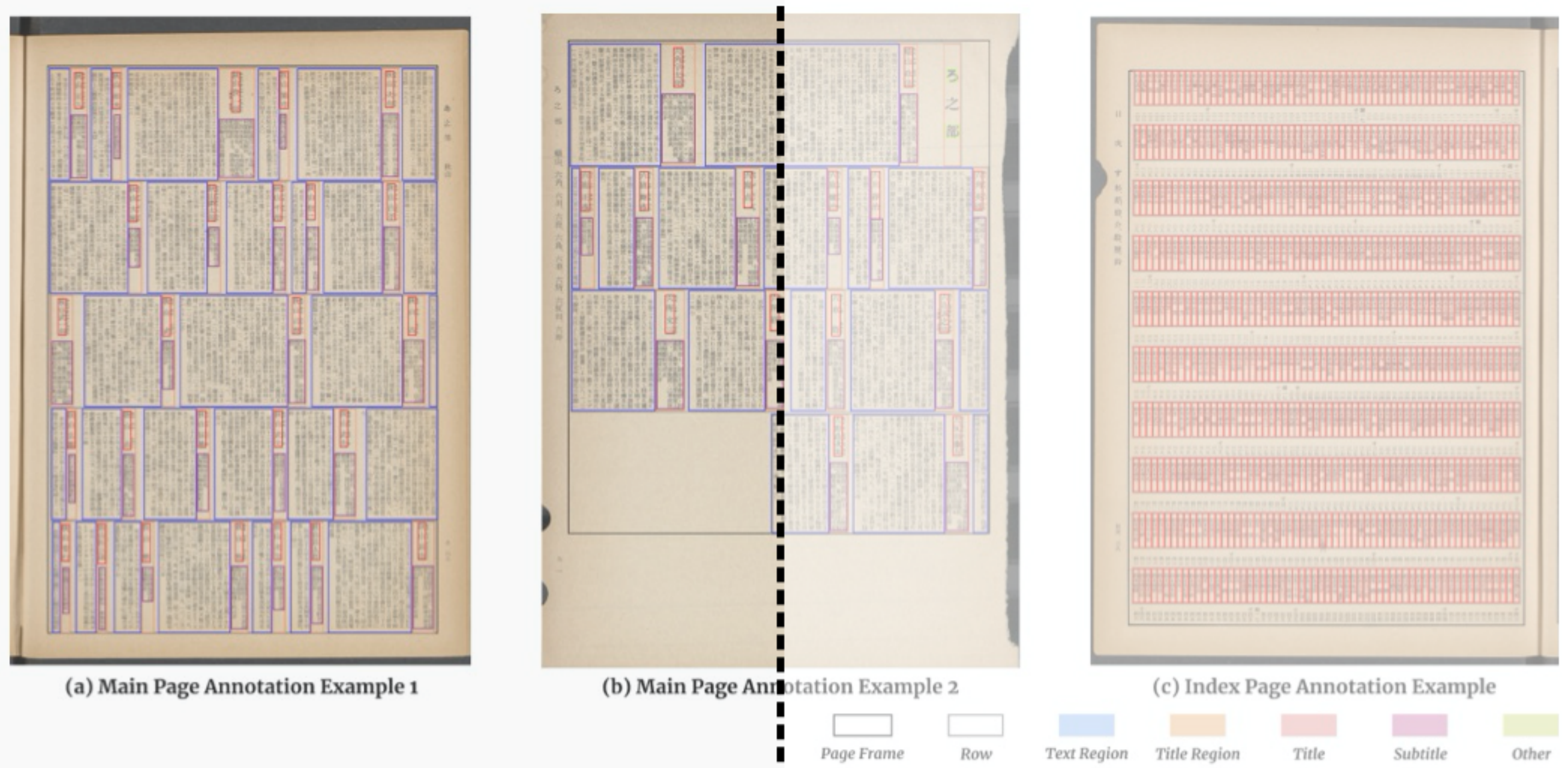



Figure 7: **Annotation Examples in HJDataset.** (a) and (b) show two examples for the labeling of `main` pages. The boxes are colored differently to reflect the layout element categories. Illustrated in (c), the items in each `index` page row are categorized as title blocks, and the annotations are denser.

tion over union (IOU) level [0.50:0.95][2], on the test data. In general, the high mAP values indicate accurate detection of the layout elements. The Faster R-CNN and Mask R-CNN achieve comparable results, better than RetinaNet. Noticeably, the detections for small blocks like title are less precise, and the accuracy drops sharply for the title category. In Figure 8, (a) and (b) illustrate the accurate prediction results of the Faster R-CNN model.

### 5.2. Pre-training for other datasets

We also examine how our dataset can help with a real-world document digitization application. When digitizing new publications, researchers usually do not generate large scale ground truth data to train their layout analysis models. If they are able to adapt our dataset, or models trained on our dataset, to develop models on their data, they can build their pipelines more efficiently and develop more accurate models. To this end, we conduct two experiments. First we examine how layout analysis models trained on the `main` pages can be used for understanding `index` pages. Moreover, we study how the pre-trained models perform on other historical Japanese documents.

Table 4 compares the performance of five Faster R-CNN models that are trained differently on `index` pages. If the model loads pre-trained weights from HJDataset, it includes information learned from `main` pages. Models trained over all the training data can be viewed as the benchmarks, while training with few samples (five in this case) are considered to mimic real-world scenarios. Given different training data, models pre-trained on HJDataset perform significantly better than those initialized with COCO weights. Intuitively, models trained on more data perform better than those with fewer samples. We also directly use the model trained on `main` to predict `index` pages without fine-tuning. The low zero-shot prediction accuracy indicates the dissimilarity between `index` and `main` pages. The large increase in mAP from 0.344 to 0.471 after the model is

[2]This is a core metric developed for the COCO competition [12] for evaluating the object detection quality.

Table 3: Detection mAP @ IOU [0.50:0.95] of different models for each category on the test set. All values are given as percentages.

| Category | Faster R-CNN | Mask R-CNN[a] | RetinaNet |
|---|---|---|---|
| Page Frame | 99.046 | 99.097 | 99.038 |
| Row | 98.831 | 98.482 | 95.067 |
| Title Region | 87.571 | 89.483 | 69.593 |
| Text Region | 94.463 | 86.798 | 89.531 |
| Title | 65.908 | 71.517 | 72.566 |
| Subtitle | 84.093 | 84.174 | 85.865 |
| Other | 44.023 | 39.849 | 14.371 |
| mAP | 81.991 | 81.343 | 75.223 |

[a] For training Mask R-CNN, the segmentation masks are the quadrilateral regions for each block. Compared to the rectangular bounding boxes, they delineate the text region more accurately.

# Layout Parser Infrastructure APIs

Layout Data Structures | OCR Engine Support | Layout Visualization | Load & Storage

## Interaction between layout regions and OCR

```
>>> layout = model.detect(image)

>>> ocr_text = ocr_model.detect(image)
```

# Layout Parser Infrastructure APIs

Layout Data Structures | OCR Engine Support | Layout Visualization | Load & Storage

## Perform OCR for each Block

## Interaction Between Layout Regions and OCR

```
>>> layout = model.detect(image)

>>> for block in layout:
        segment = block.crop_image(segment)
        block.text = ocr_agent.detect(segment)
```

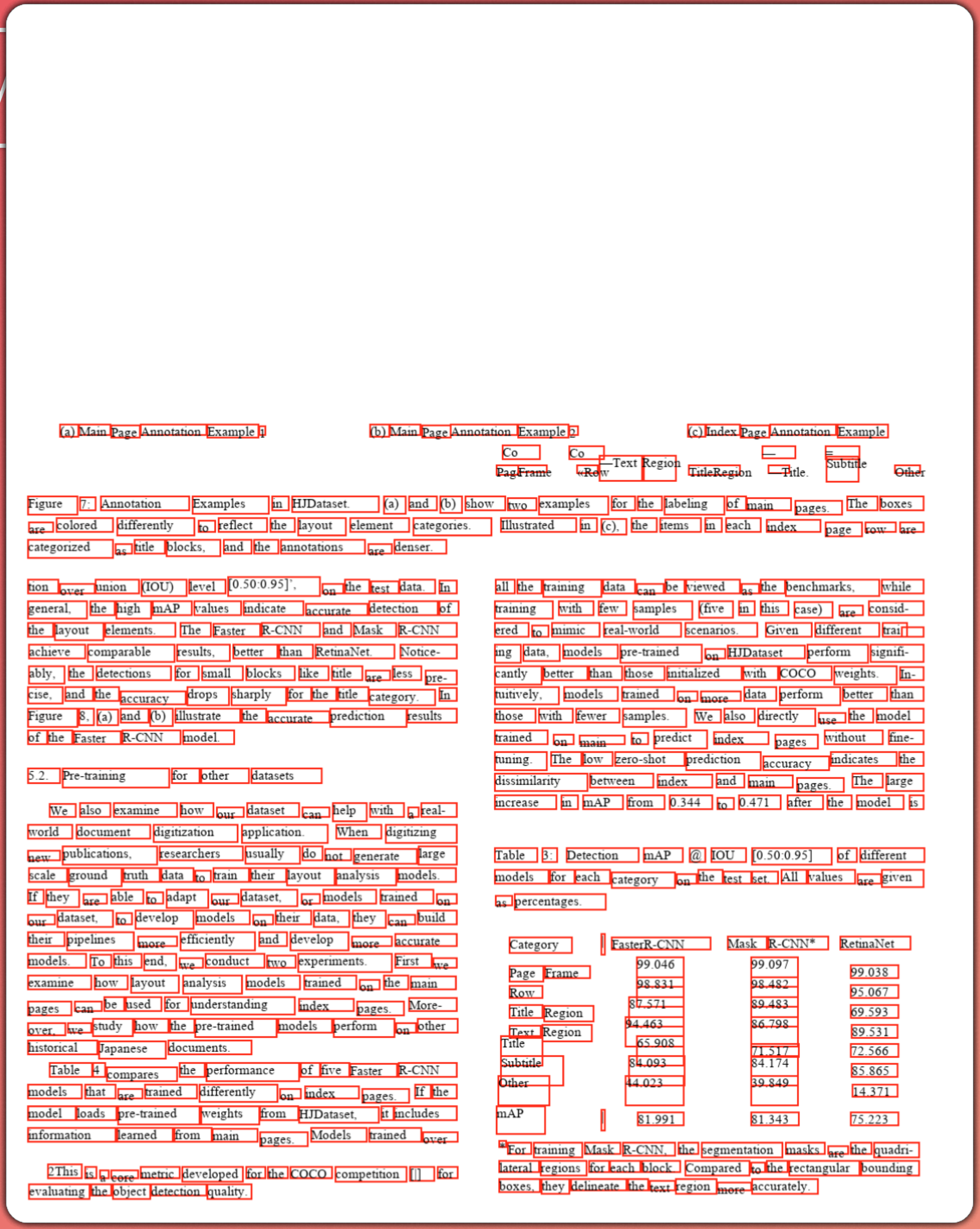
# Layout Parser Infrastructure APIs

Layout Data Structures | OCR Engine Support | Layout V

## Highly configurable visualization

```
>>> draw_box(image, layout,
    show_element_type=True,
    show_element_id=True,
    box_width=4,
    color_map={...})

>>> draw_text(image, ocr_text,
    with_box_on_text=True, ...)
```



# Layout Parser Infrastructure APIs

Layout Data Structures | OCR Engine Support | Layout Visualization | **Load & Storage**

## Exporting

```
>>> layout.to_csv()
>>> layout.to_json()
```

## Loading

```
>>> layout = lp.load_csv()
>>> layout = lp.load_json()
```

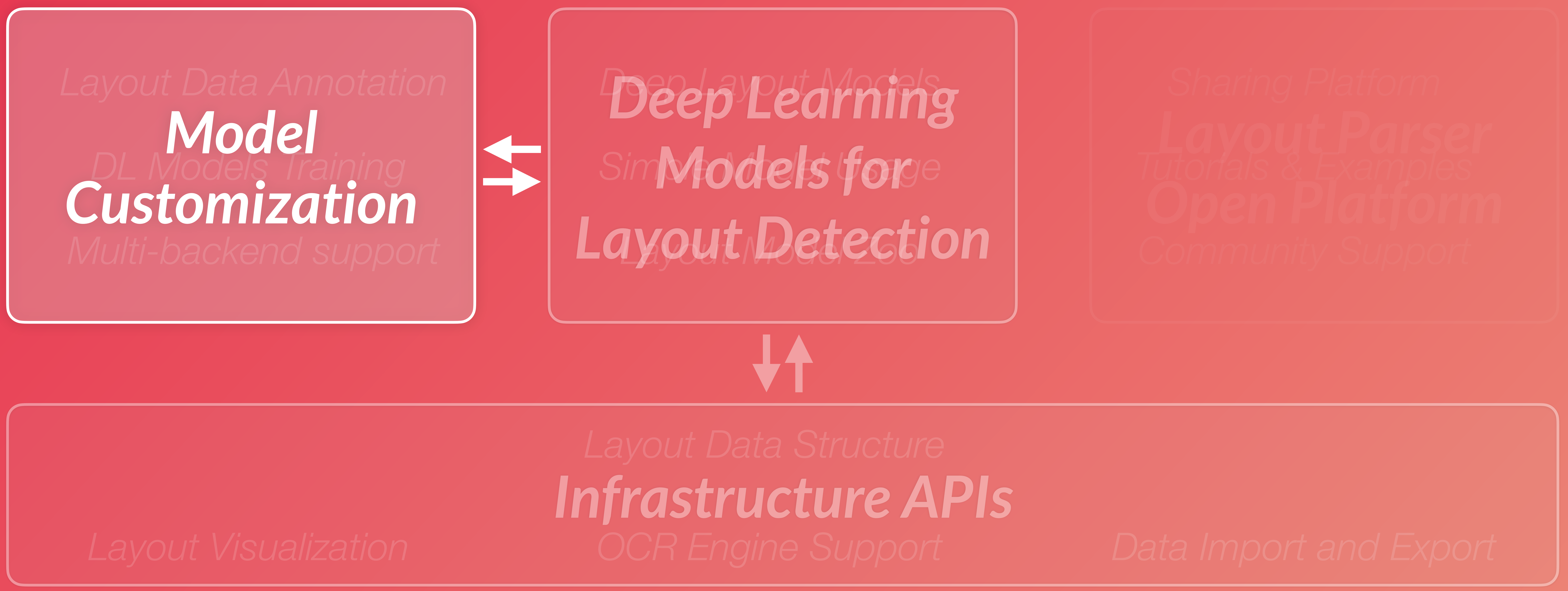
## Currently supports

**CSV** **JSON**

## More formats will be added:

*PAGES* *METS/ALTO* *hOCR* *...*

*What if we want better models?*


Layout Data Annotation
Model Customization
DL Models Training
Multi-backend support
Deep Layout Models
Deep Learning Models for Layout Detection
Layout Model Zoo
Sharing Platform
Layout Parser Open Platform
Tutorials & Examples
Community Support
Layout Data Structure
Infrastructure APIs
Layout Visualization
OCR Engine Support
Data Import and Export

# Deep Learning Model Customization

Why

How different is the target data?

*Very Similar* — *Very Different*

# Deep Learning Model Customization

Why

How different is the target data?

*Very Similar* *Very Different*

Accuracy/efficiency requirements?

*Very Accurate* *Very Efficient*

# Deep Learning Model Customization

Why

How different is the target data?

Very Similar — Very Different

Accuracy/efficiency requirements?

Very Accurate — Very Efficient

How much training data is available?

None — More

# Deep Learning Model Customization

Target Data Difference

*Similar* *Different*

Accuracy/efficiency trade-off

*Accurate* *Efficient*

Available training data

*None* *More*

# Deep Learning Model Customization

Target Data Difference

*Similar* *Different*

Accuracy/efficiency trade-off

*Accurate* *Efficient*

Available training data

*None* *More*

Multi-backend Support

Detectron2 — *Accurate*

EfficientDet — *Efficient*

PaddleDetection

*And more in the future*

# Deep Learning Model Customization

Target Data Difference

*Similar* *Different*

Accuracy/efficiency trade-off

*Accurate* *Efficient*

Available training data

*None* *More*

# Deep Learning Model Customization

Target Data Difference

*Similar* *Different*

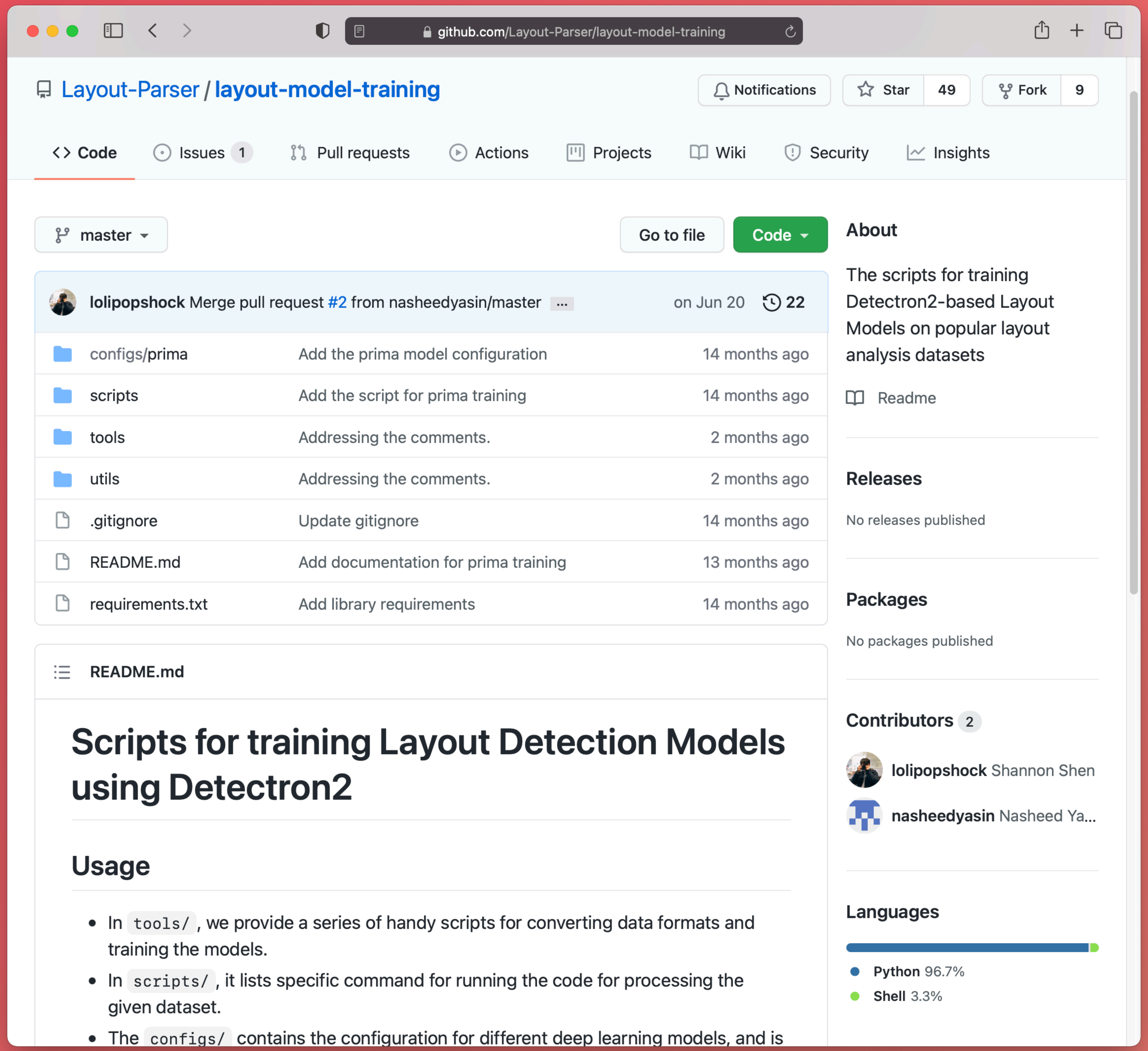
Accuracy/efficiency trade-off

*Accurate* *Efficient*

Available training data

*None* *More*

Model Fine-tuning




▲ *Layout Parser comes with the script that fine-tunes existing models to new datasets.*

# Deep Learning Model Customization

Target Data Difference

*Similar* *Different*

Accuracy/efficiency trade-off

*Accurate* *Efficient*

Available training data

*None* *More*

# Deep Learning Model Customization

## Target Data Difference

*Similar* *Different*

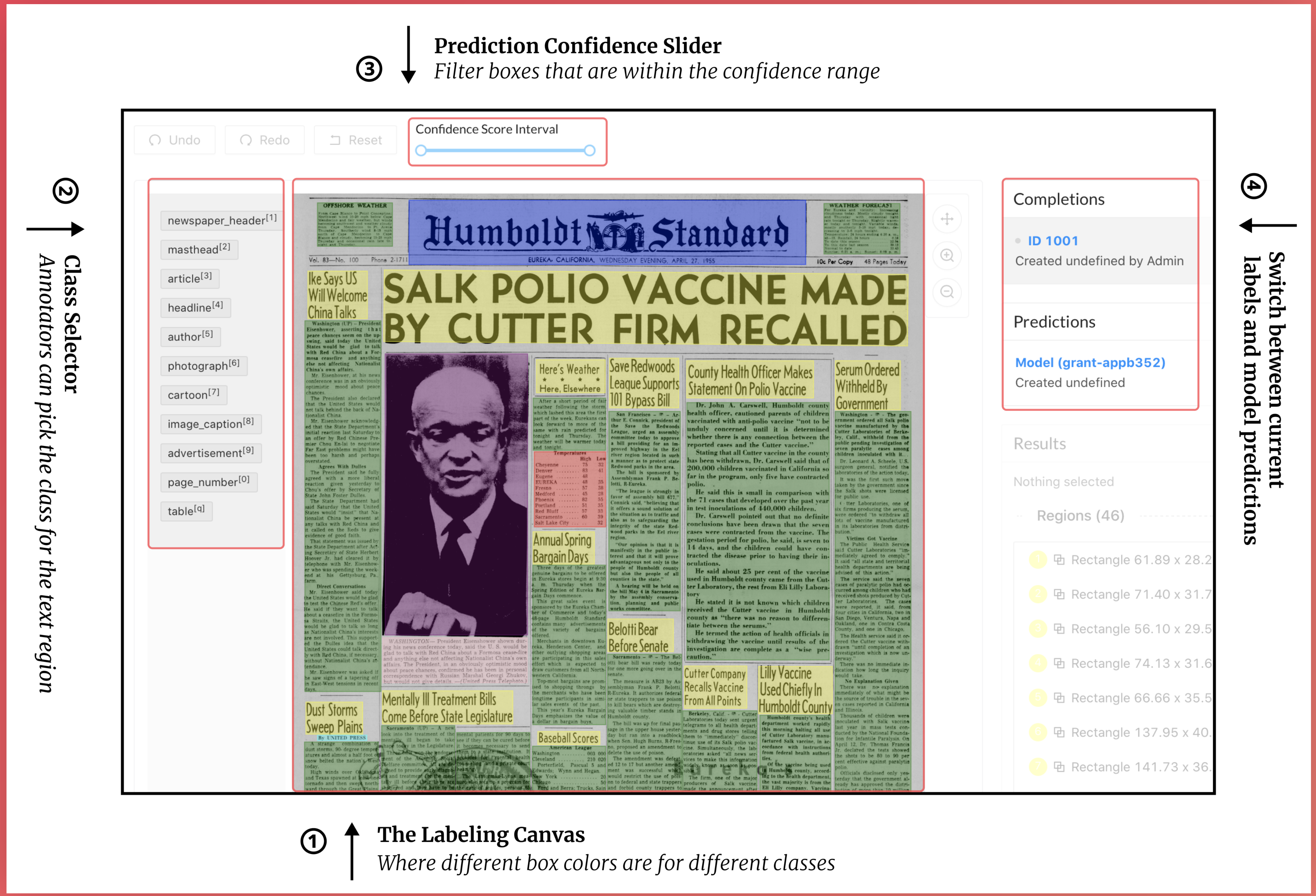
## Accuracy/efficiency trade-off

*Accurate* *Efficient*

## Available training data

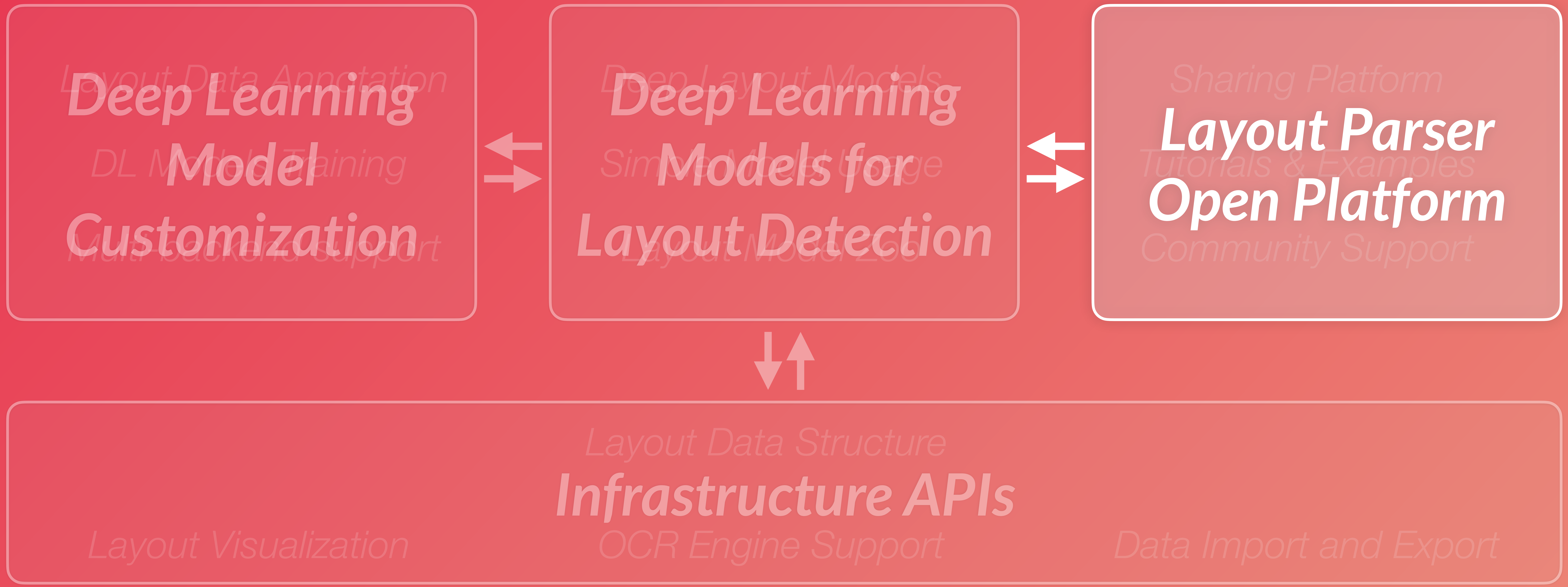
*None* *More*

## Annotation & Model Retraining




*▲ Layout Parser incorporates labeling toolkits from existing resources to streamline the labeling and improve efficiency.*

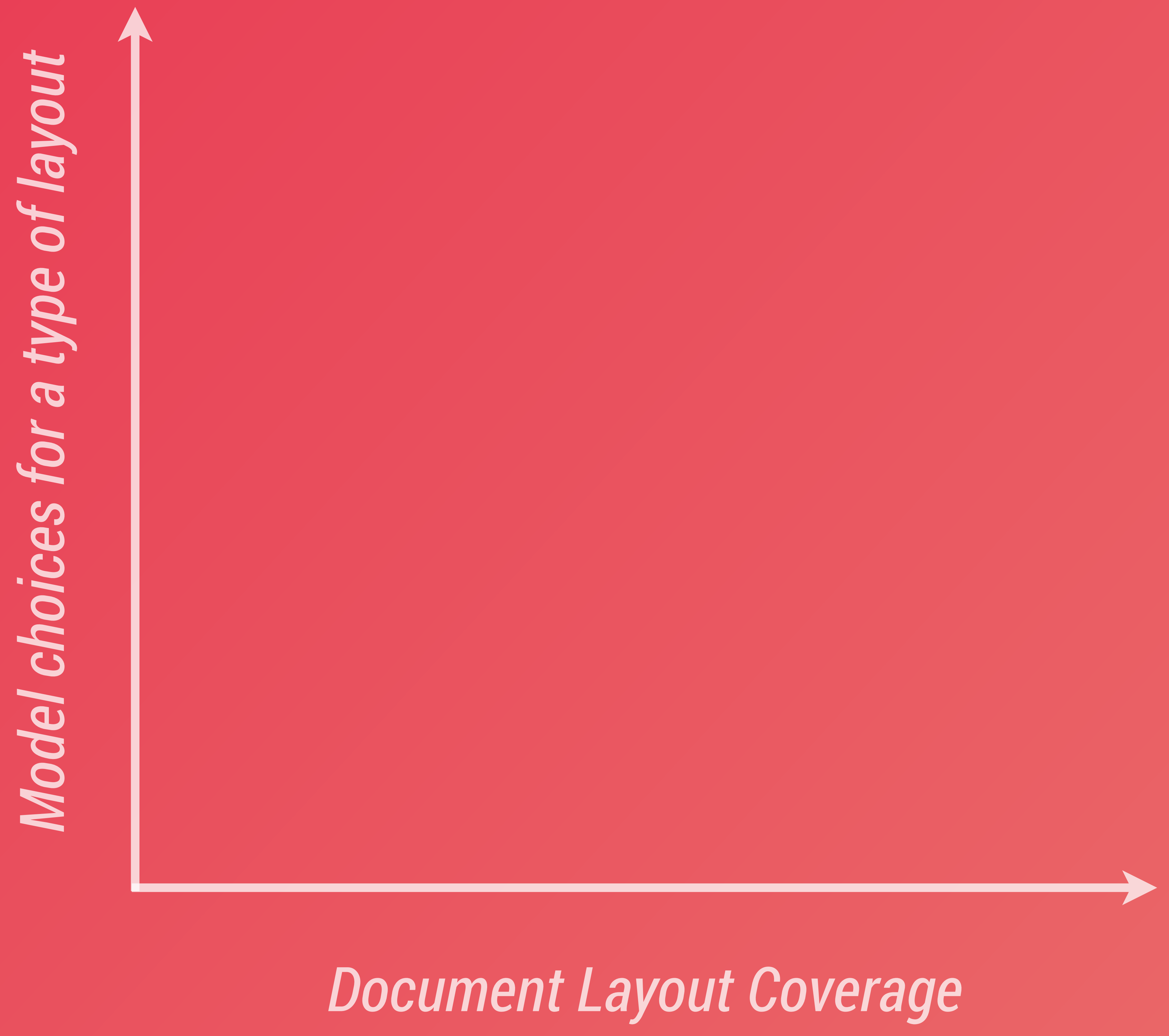
*How about sharing your work with the community?*


Deep Learning Model Customization
Layout Data Annotation
DL Model Training
Multi-backend Support
Deep Learning Models for Layout Detection
Deep Layout Models
Simple Model Usage
Layout Model Zoo
Layout Parser Open Platform
Sharing Platform
Tutorials & Examples
Community Support
Infrastructure APIs
Layout Data Structure
Layout Visualization
OCR Engine Support
Data Import and Export

# Layout Parser Open Platform

Share Layout Models

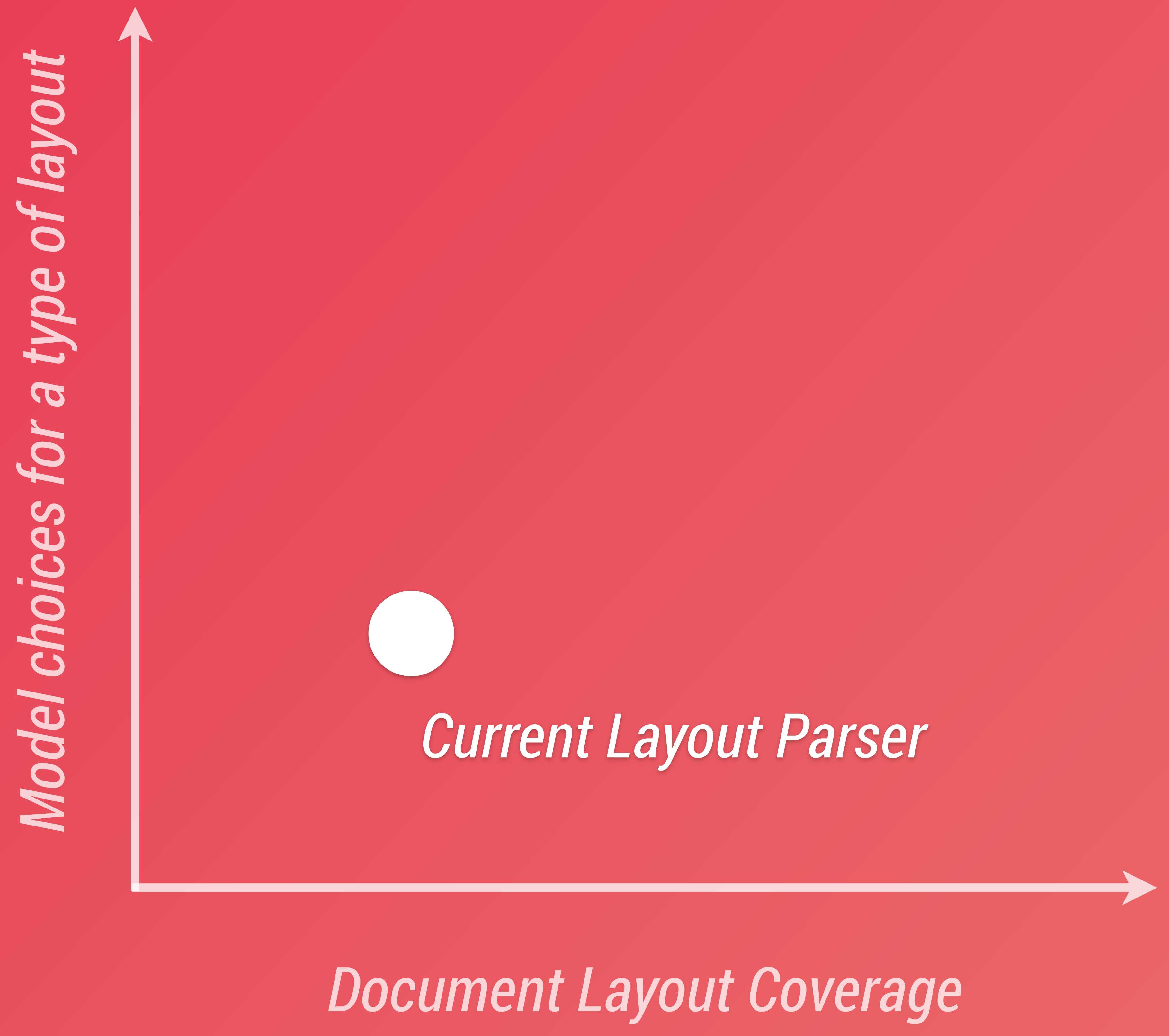
Share DIA Pipelines



# Layout Parser Open Platform

Share Layout Models

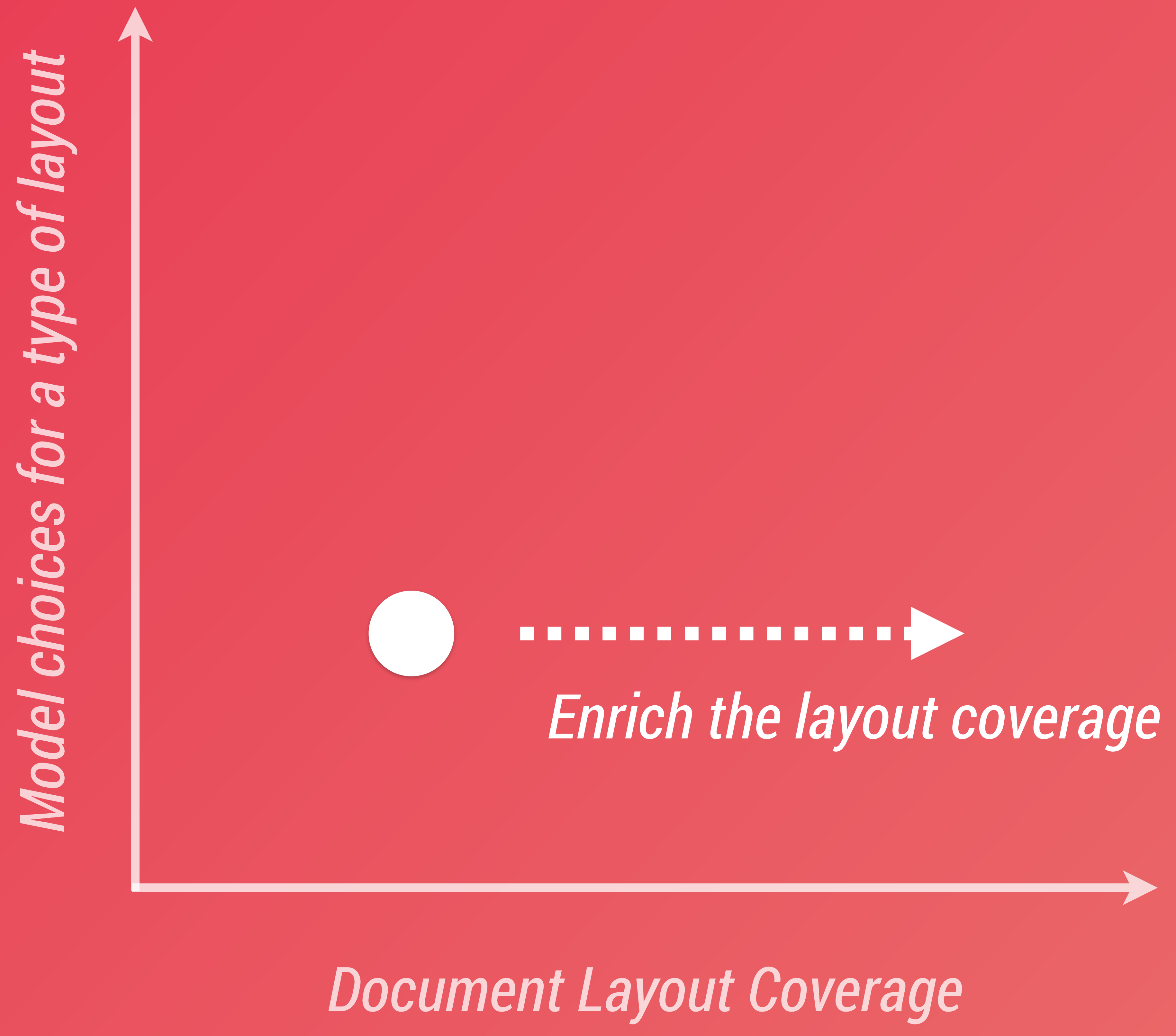
Share DIA Pipelines



# Layout Parser Open Platform

Share Layout Models

Share DIA Pipelines

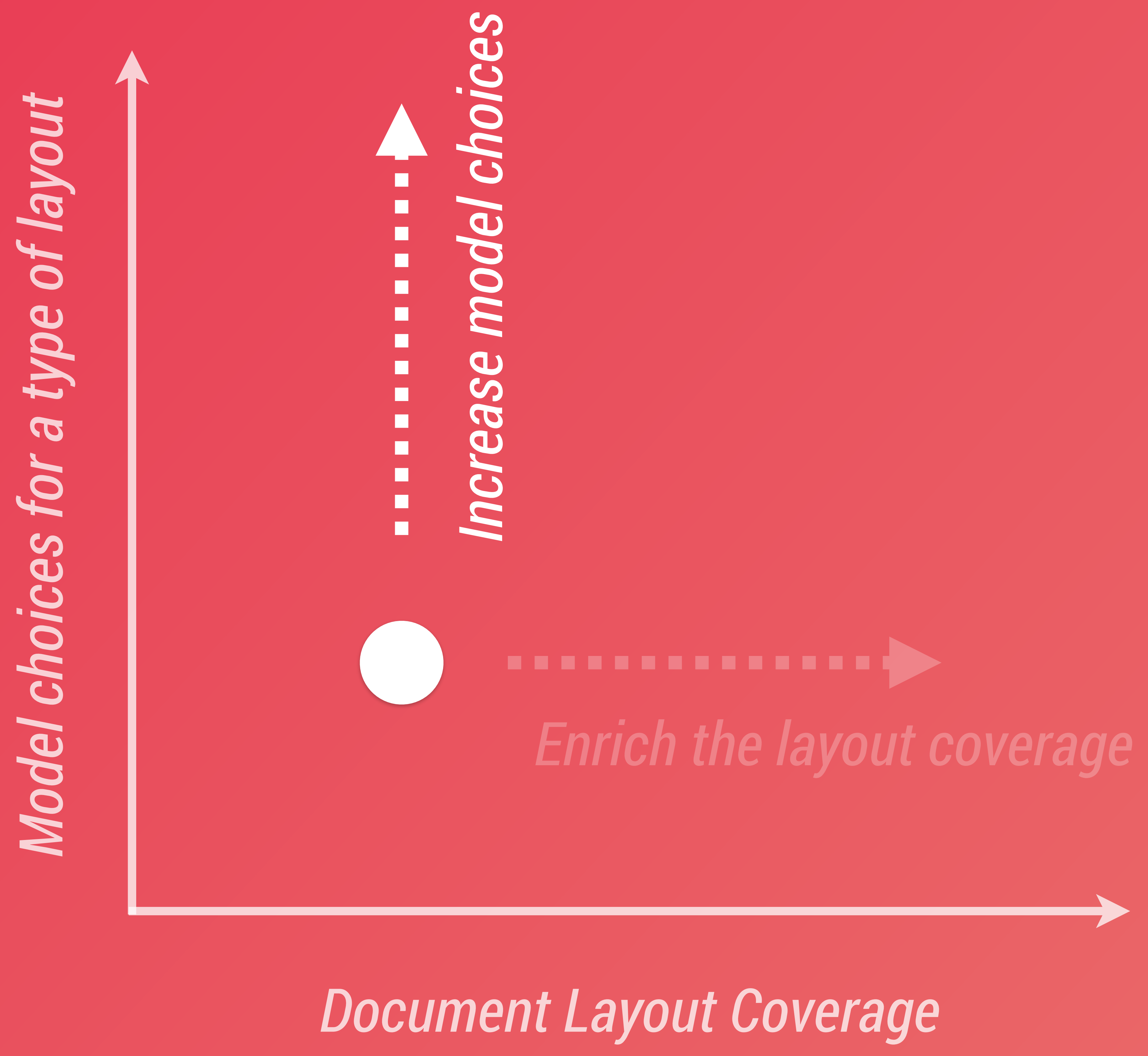



If we share:

Models trained on different datasets

# Layout Parser Open Platform

Share Layout Models | Share DIA Pipelines




If we share:

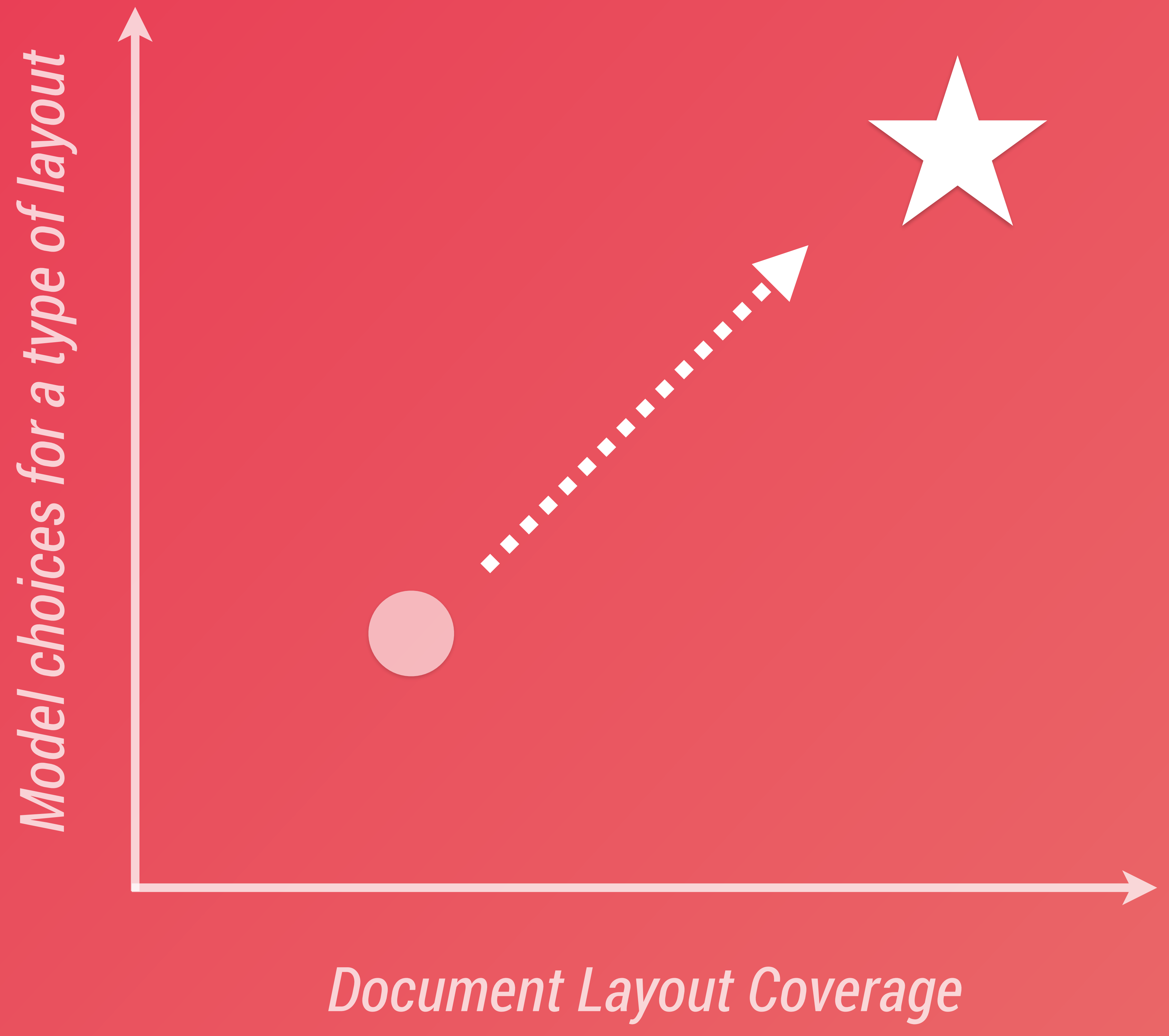
Models trained on different datasets

Models of different architecture/backend

# Layout Parser Open Platform

Share Layout Models

Share DIA Pipelines




Ultimately:

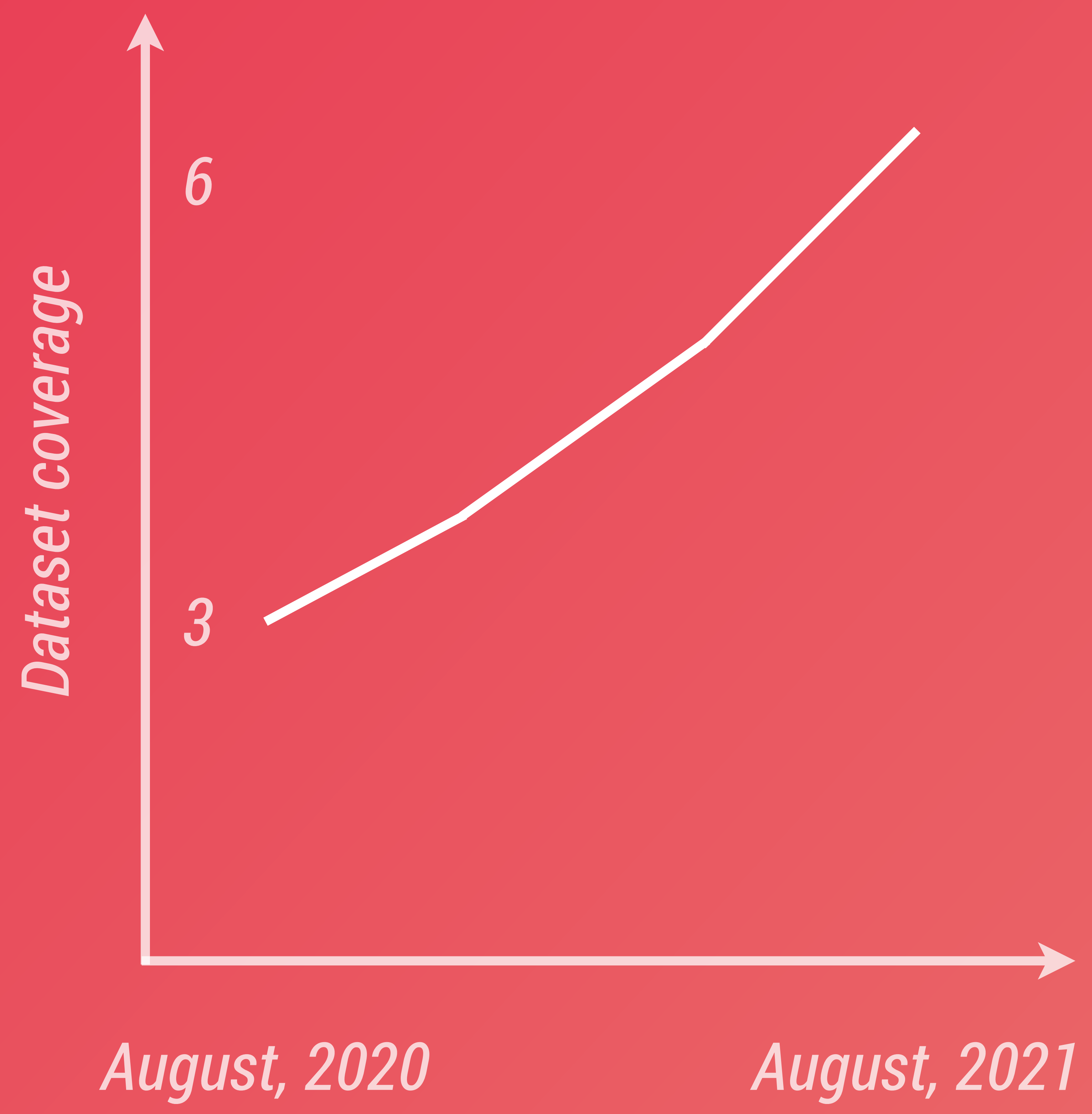
Make it easier to find the ideal model

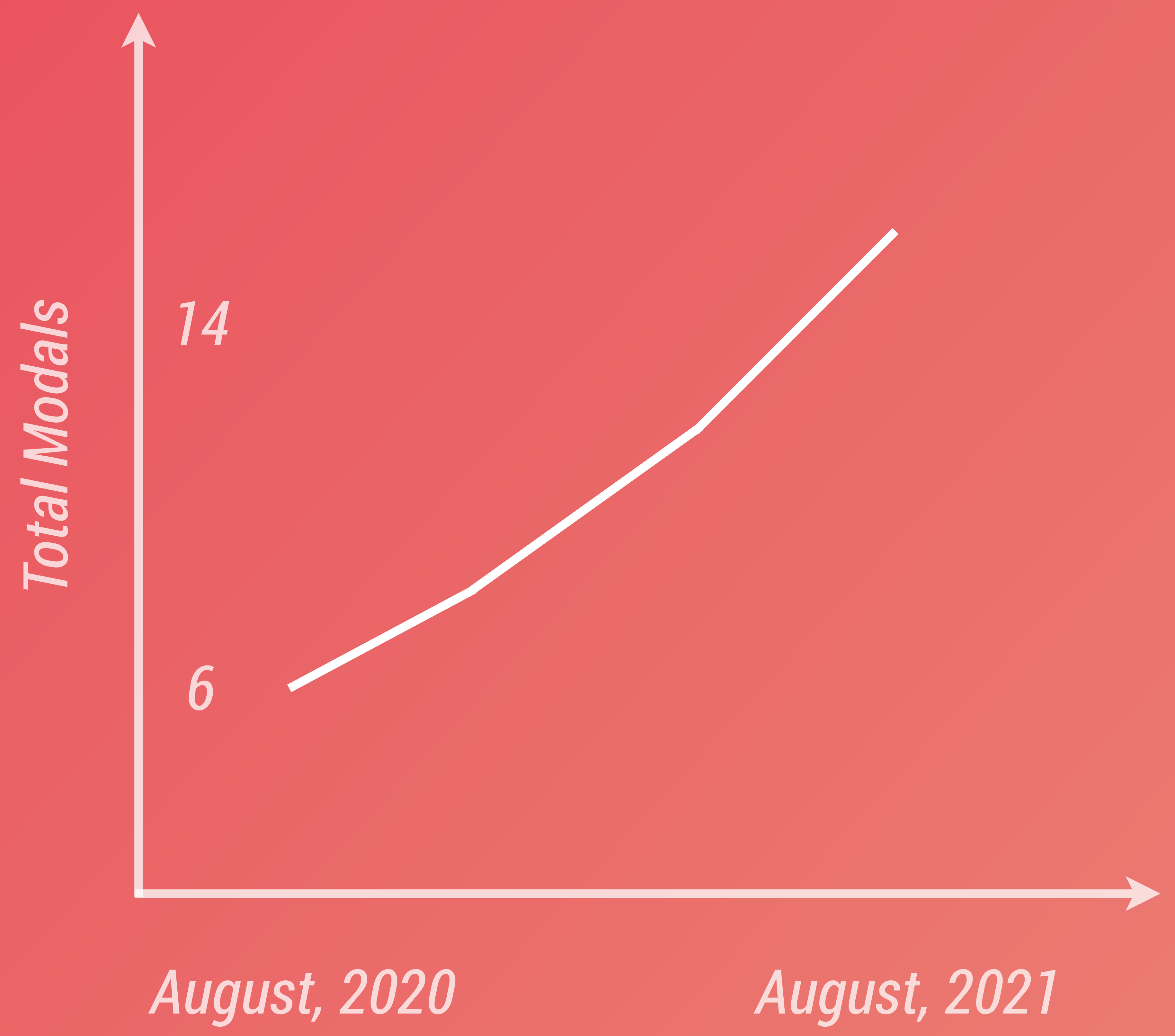
# Layout Parser Open Platform

Share Layout Models

Share DIA Pipelines

## 14 models for 6 datasets, 2x in the past year, with the help of community

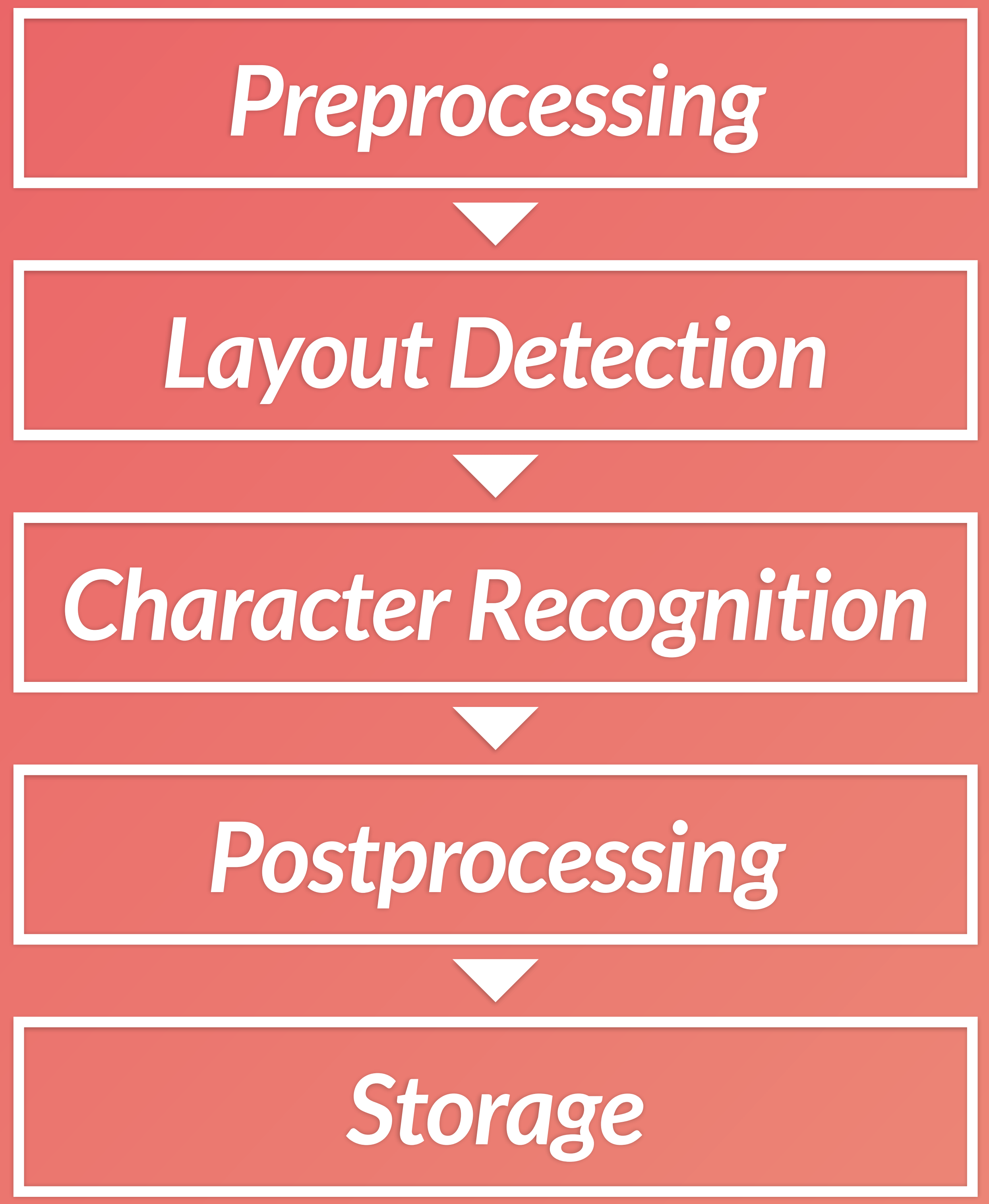


# Layout Parser Open Platform

Share Layout Models | Share DIA Pipelines

DIA pipelines have multiple steps:



# Layout Parser Open Platform

Share Layout Models | Share DIA Pipelines

DIA pipelines have multiple steps:

*Preprocessing*

***Layout Detection***

*Character Recognition*

*Postprocessing*

*Storage*

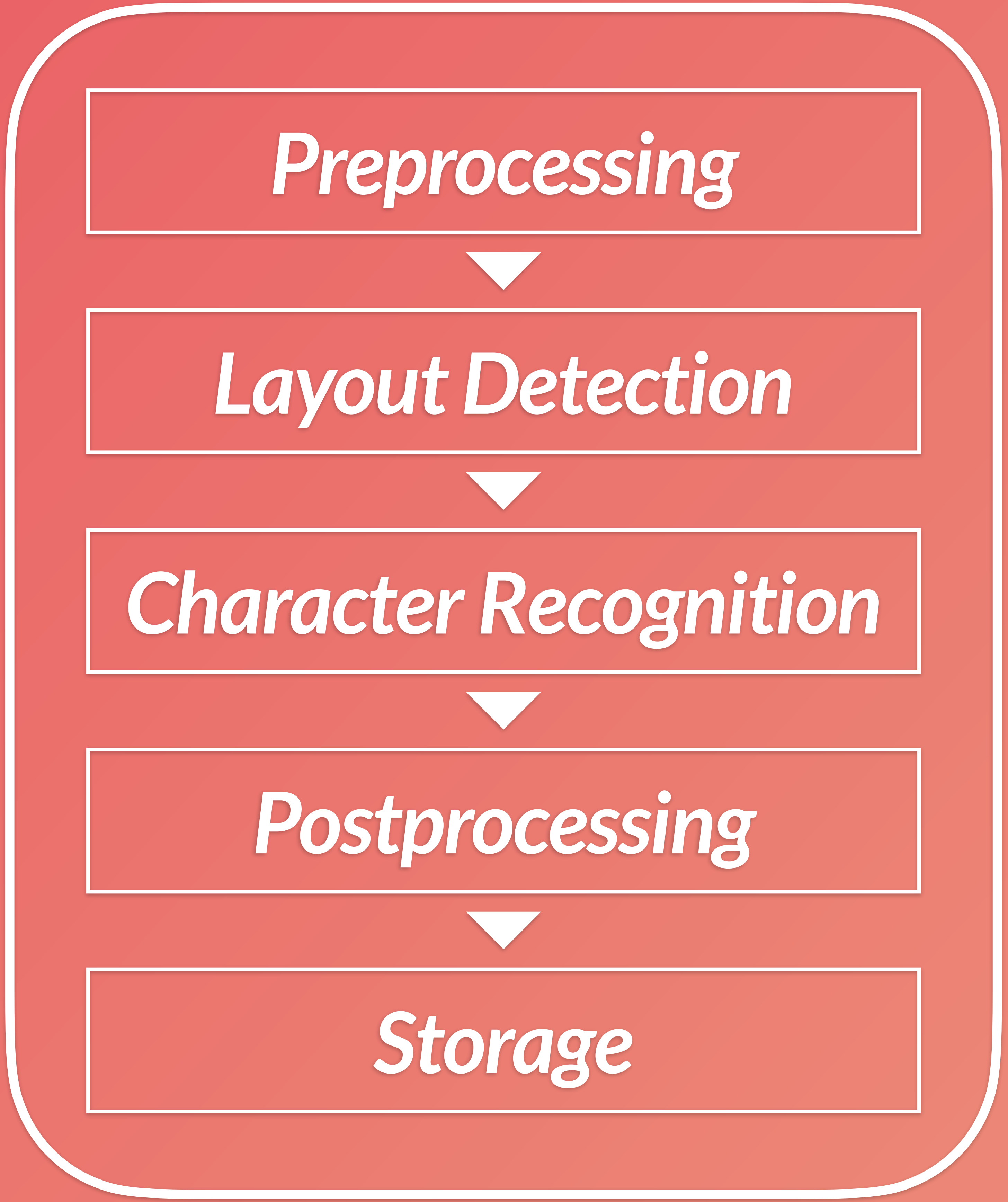
# Layout Parser Open Platform

Share Layout Models

Share DIA Pipelines

DIA pipelines have multiple steps:

Can we share them as a whole?

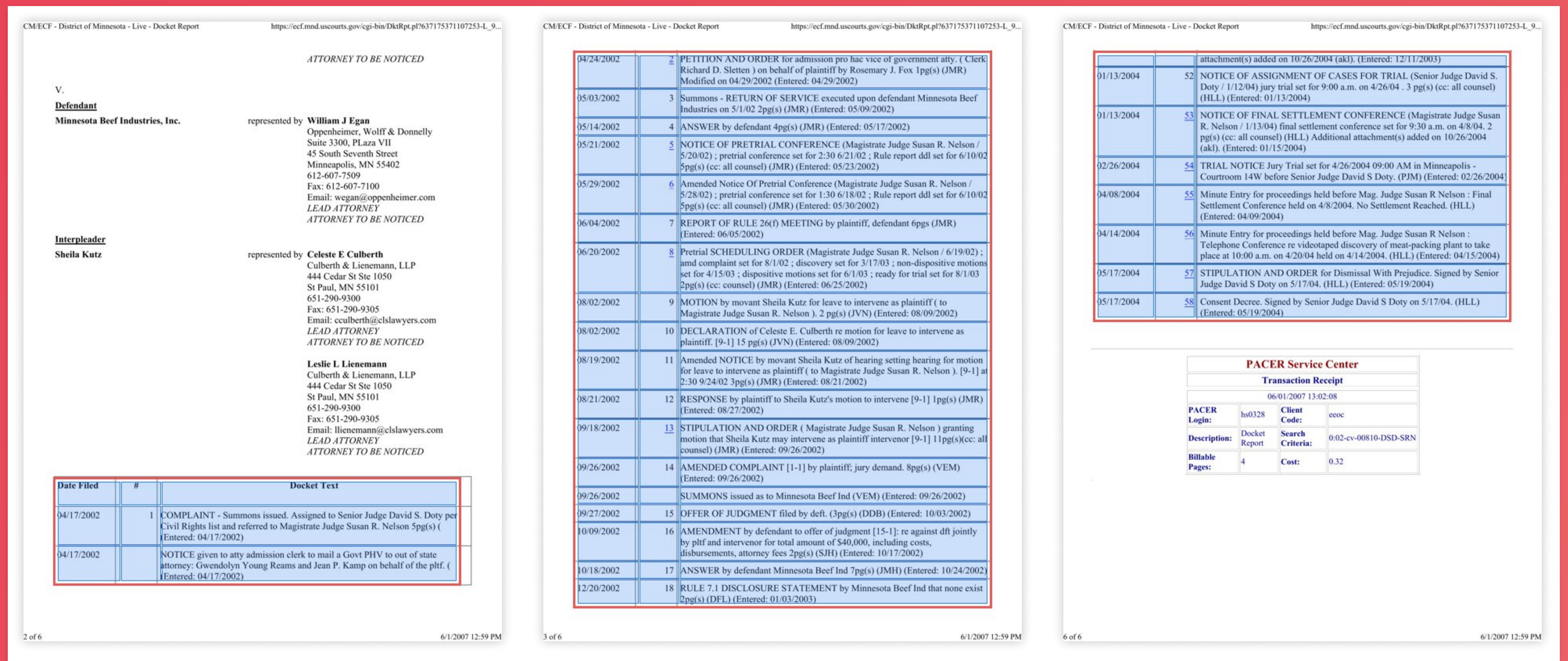


# Layout Parser Open Platform

Share Layout Models | Share DIA Pipelines

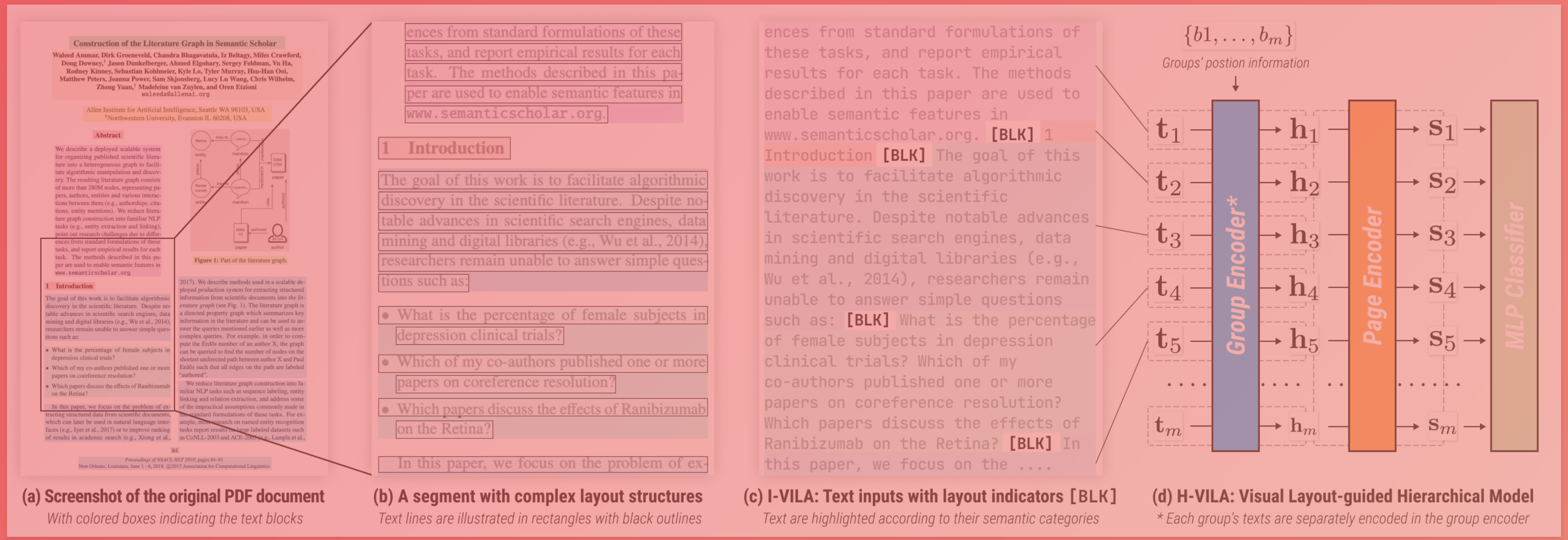
## Examples:

### *Table Extraction*



▲ *Layout parser has been used for extraction tables and other layout elements in legal documents.*

### *Scientific Document Parsing*



(a) Screenshot of the original PDF document — With colored boxes indicating the text blocks
(b) A segment with complex layout structures — Text lines are illustrated in rectangles with black outlines
(c) I-VILA: Text inputs with layout indicators [BLK] — Text are highlighted according to their semantic categories
(d) H-VILA: Visual Layout-guided Hierarchical Model — * Each group's texts are separately encoded in the group encoder

▲ *Layout parser is used in a research project that studies how layout information can improve scientific document analysis.*

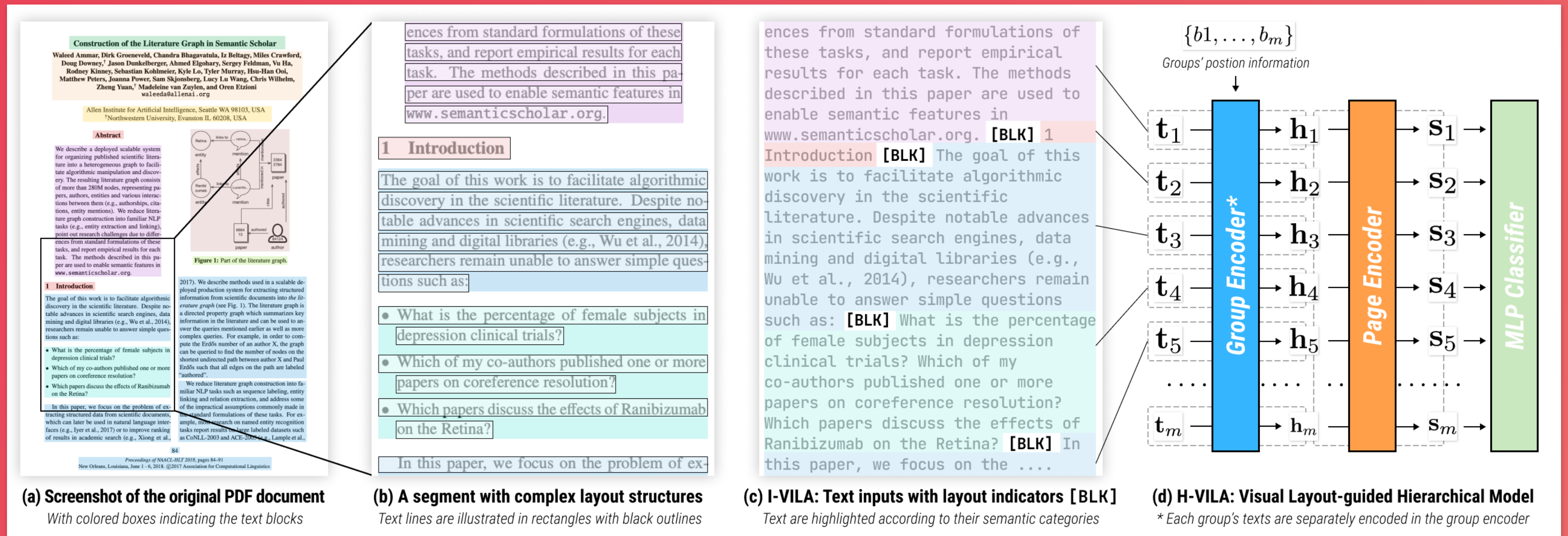
# Layout Parser Open Platform

Share Layout Models

Share DIA Pipelines

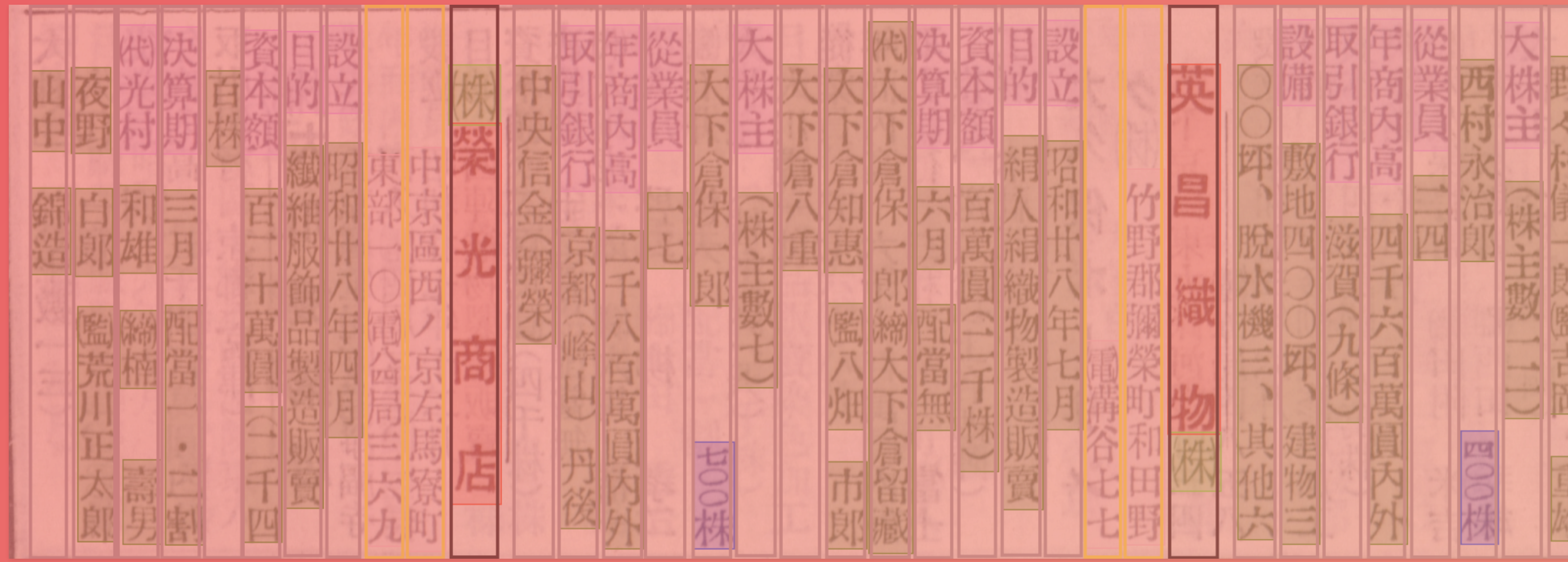
## Examples:

### *Scientific Document Parsing*




*▲ Layout parser is used in a research project that studies how layout information can improve scientific document analysis.*

### *Historical Document Analysis*



*▲ Layout parser is also used for large-scale digitization of Historical Japanese Documents*

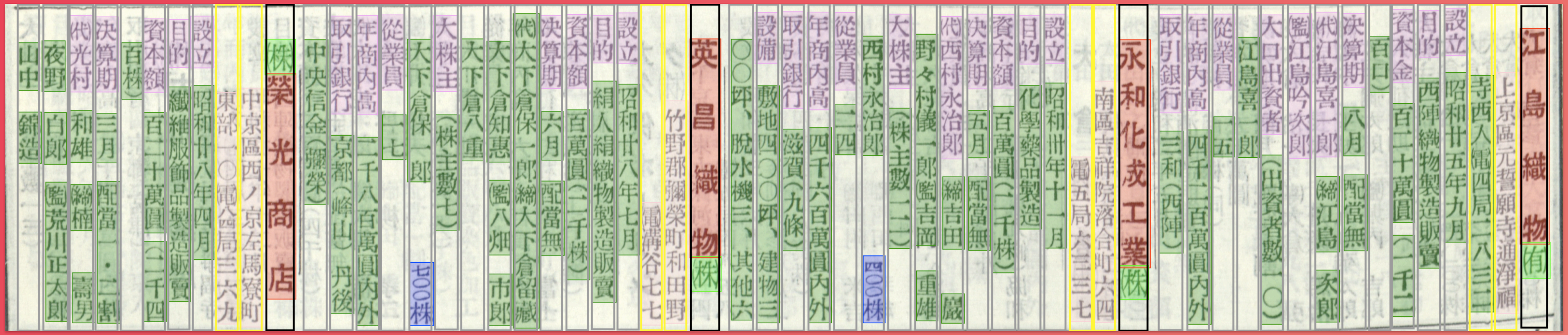
# Layout Parser Open Platform

Share Layout Models

Share DIA Pipelines

## Examples:

### *Historical Document Analysis*



*▲ Layout parser is also used for large-scale digitization of Historical Japanese Documents*

Layout Data Annotation

DL Models Training

Multi-backend support

Deep Layout Models

Simple Model Usage

Layout Model Zoo

Sharing Platform

Tutorials & Examples

Community Support

Layout Data Structure

Layout Visualization

OCR Engine Support

Data Import and Export

Layout Data Annotation

DL Models Training

Multi-backend support

Deep Layout Models

# Deep Learning Models for Layout Detection

Sharing Platform

Tutorials & Examples

Community Support

Layout Data Structure

Layout Visualization

OCR Engine Support

Data Import and Export

Layout Data Annotation

DL Models Training

Multi-backend support

**Deep Learning Models for Layout Detection**

Sharing Platform

Tutorials & Examples

Community Support

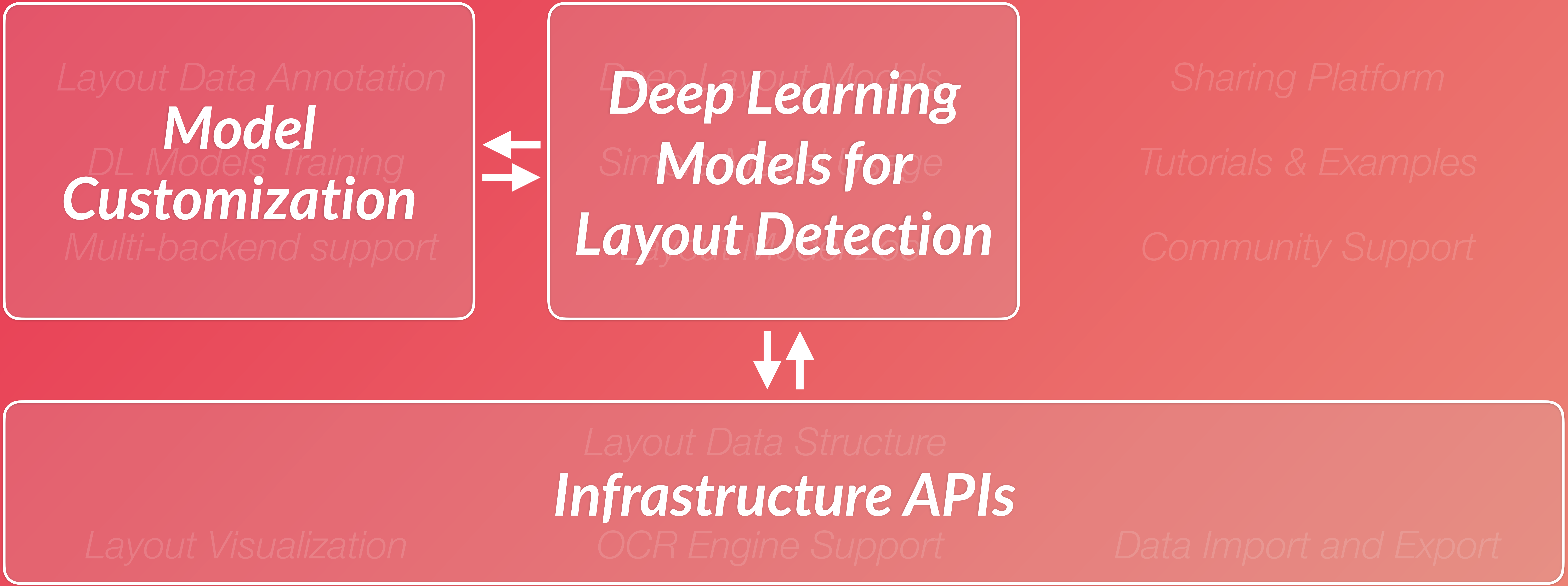
Layout Data Structure

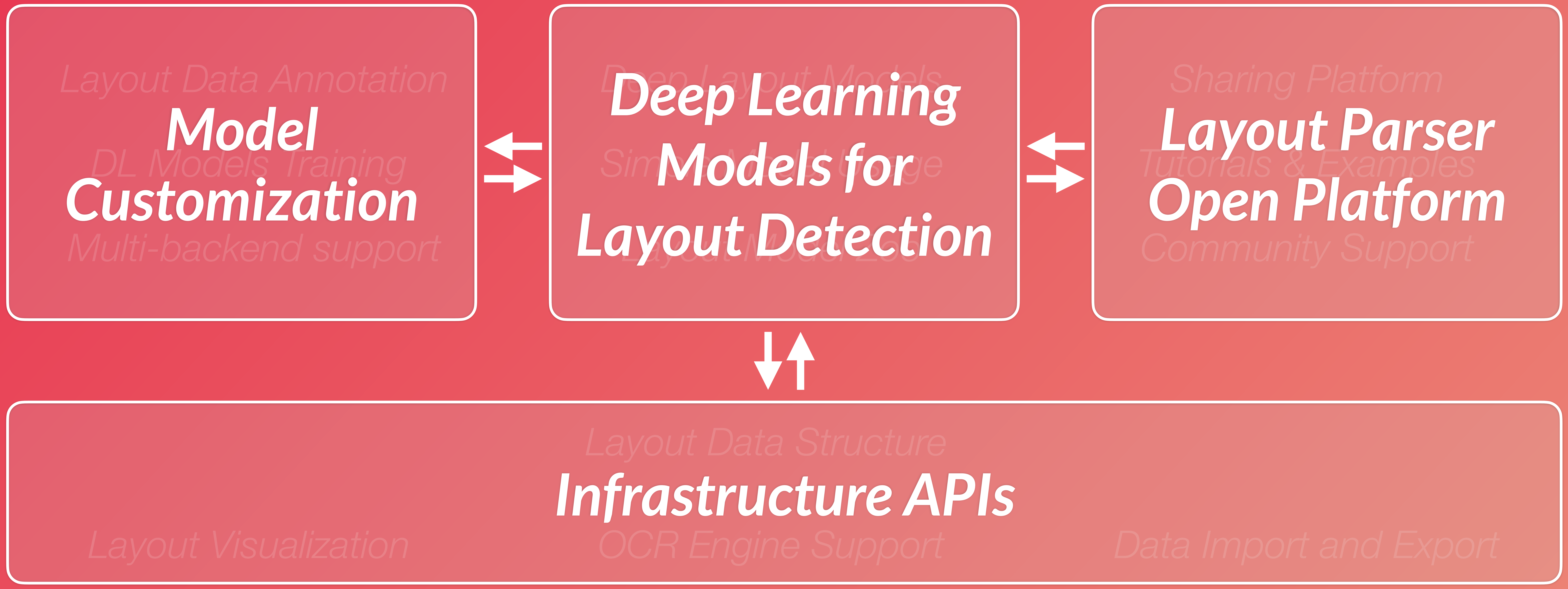
**Infrastructure APIs**

Layout Visualization

OCR Engine Support

Data Import and Export


Layout Data Annotation
Model Customization
DL Models Training
Multi-backend support
Deep Layout Models
Deep Learning Models for Layout Detection
Sharing Platform
Tutorials & Examples
Community Support
Layout Data Structure
Infrastructure APIs
Layout Visualization
OCR Engine Support
Data Import and Export


Layout Data Annotation
Model Customization
DL Models Training
Multi-backend support
Deep Layout Models
Deep Learning Models for Layout Detection
Sharing Platform
Layout Parser Open Platform
Tutorials & Examples
Community Support
Layout Data Structure
Infrastructure APIs
Layout Visualization
OCR Engine Support
Data Import and Export

Motivation

Demo

Design & Implementation

Future Work

Community

# Future Work

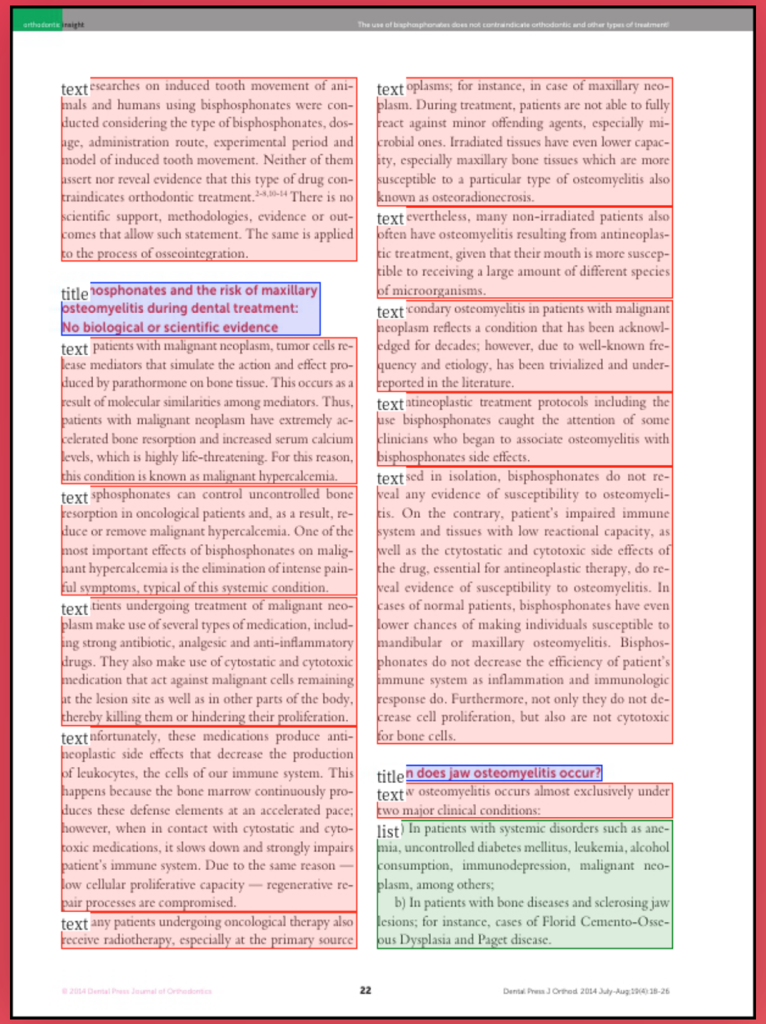
Generalized Models

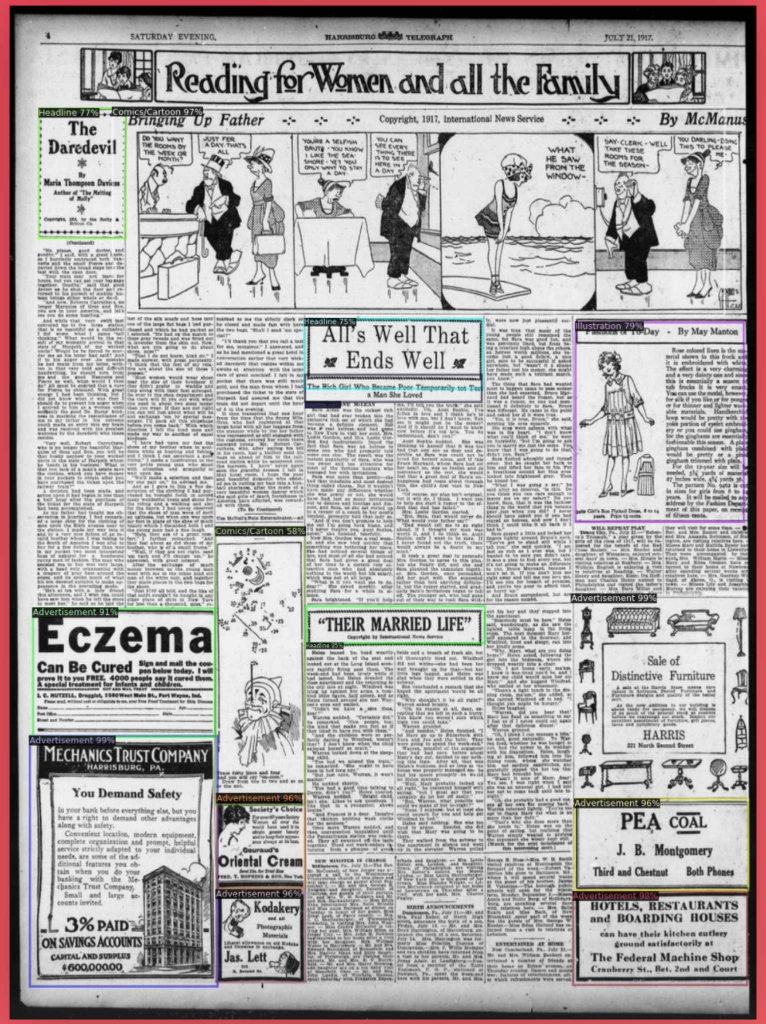
Multimodal Modeling

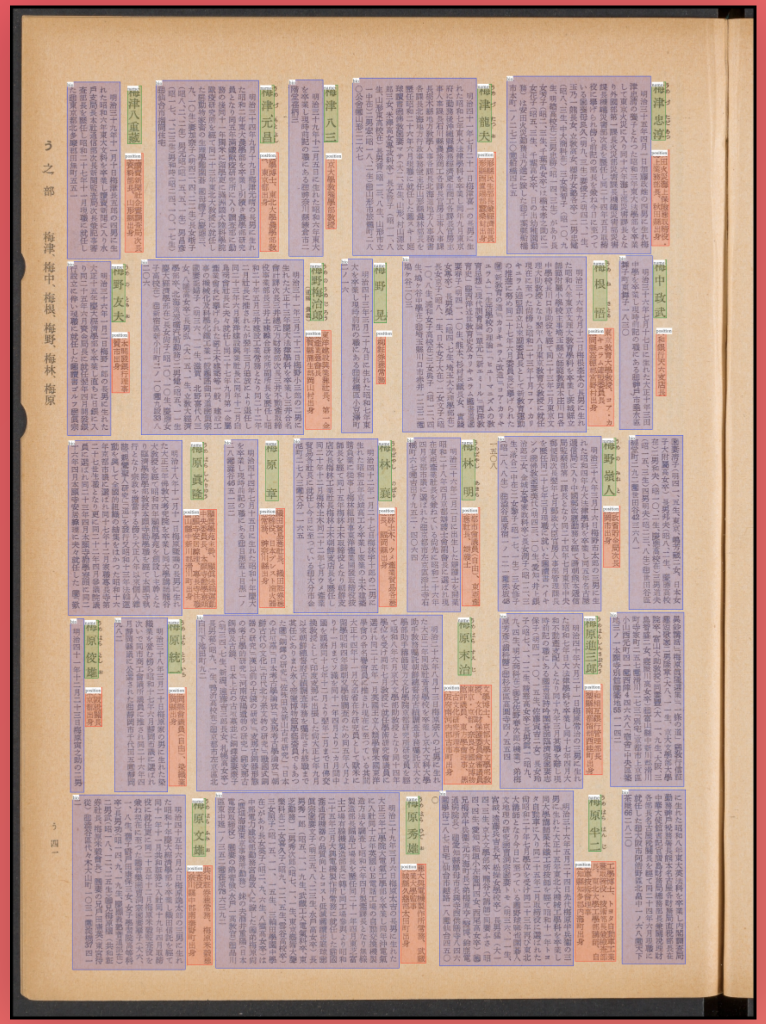
# Future Work

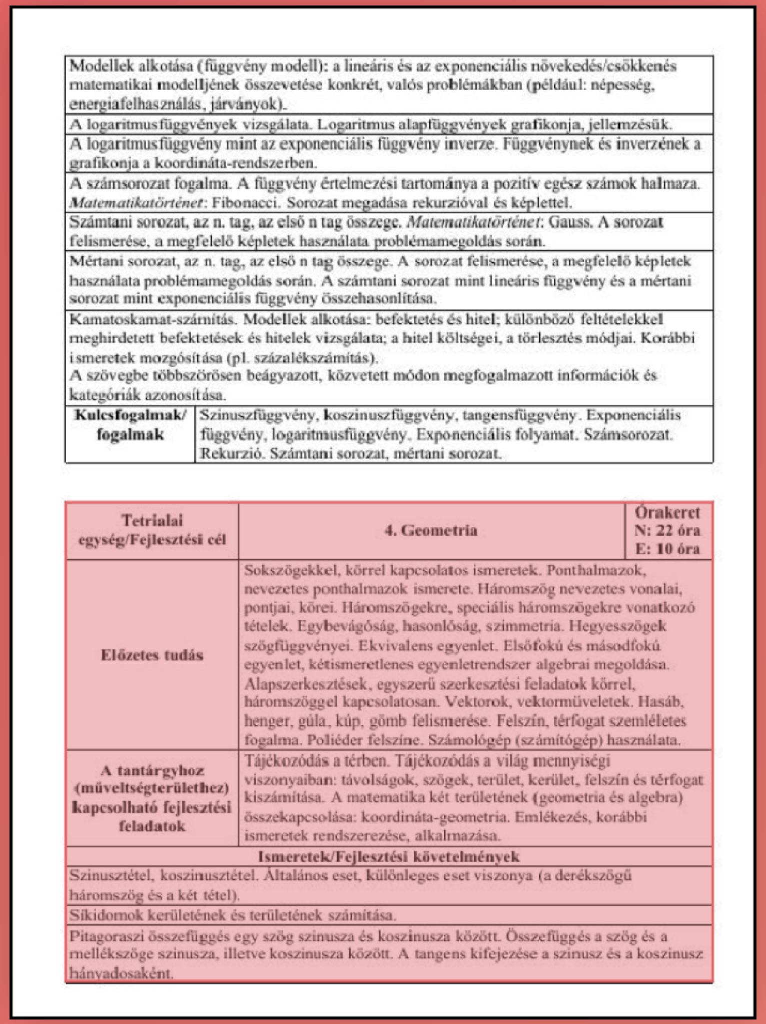
Generalized Models

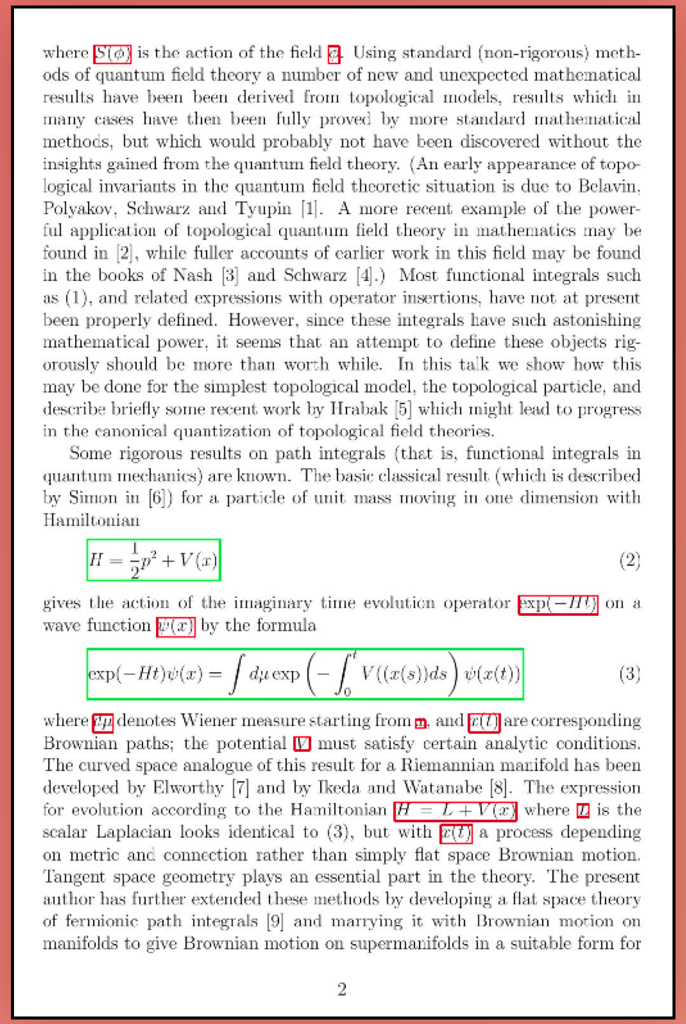
Multimodal Modeling









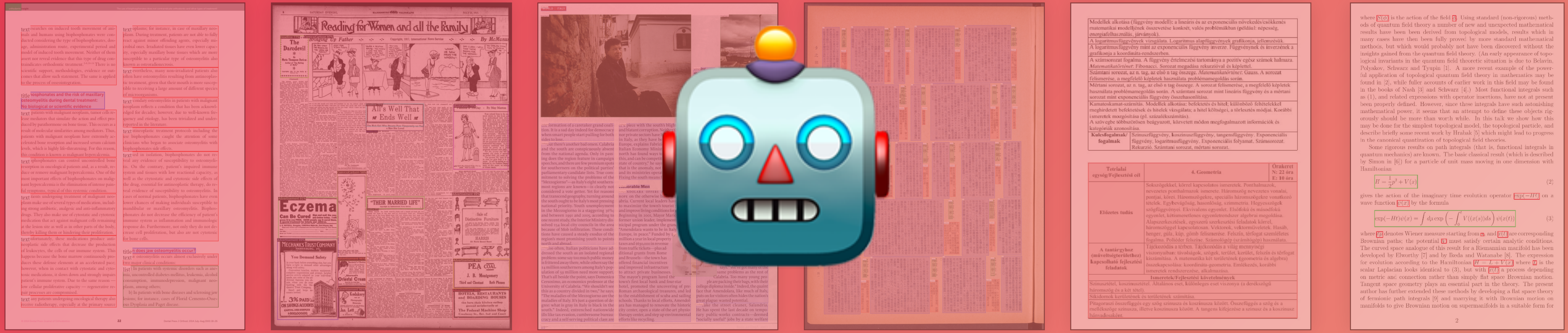


# Future Work

Generalized Models

Multimodal Modeling

## Can we have a single model for multiple layouts?



# Future Work

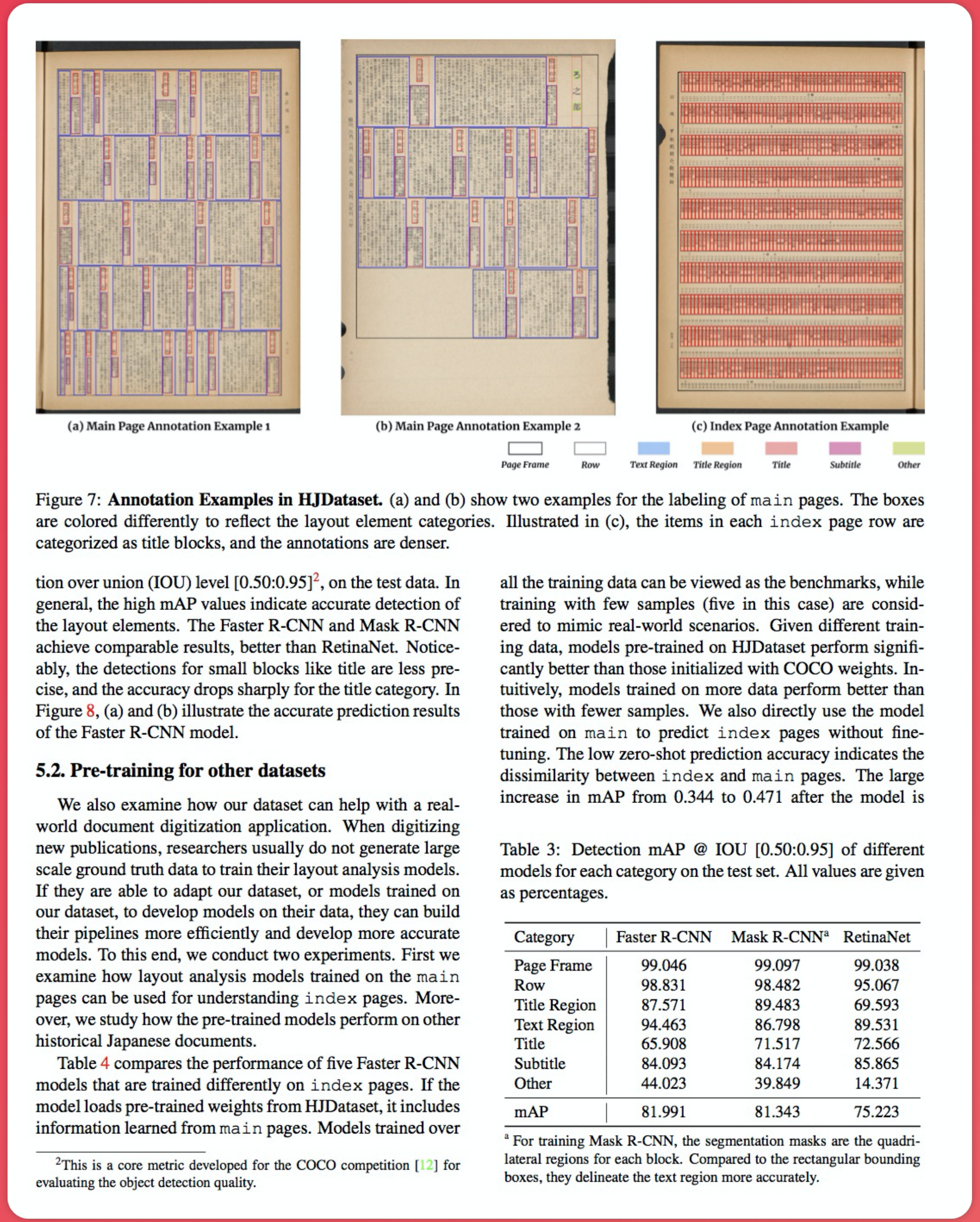
Generalized Models

Multimodal Modeling

# Future Work

Generalized Models

Multimodal Modeling



(a) Main Page Annotation Example 1 (b) Main Page Annotation Example 2 (c) Index Page Annotation Example

Page Frame Row Text Region Title Region Title Subtitle Other

Figure 7: **Annotation Examples in HJDataset.** (a) and (b) show two examples for the labeling of `main` pages. The boxes are colored differently to reflect the layout element categories. Illustrated in (c), the items in each `index` page row are categorized as title blocks, and the annotations are denser.

tion over union (IOU) level [0.50:0.95][2], on the test data. In general, the high mAP values indicate accurate detection of the layout elements. The Faster R-CNN and Mask R-CNN achieve comparable results, better than RetinaNet. Noticeably, the detections for small blocks like title are less precise, and the accuracy drops sharply for the title category. In Figure 8, (a) and (b) illustrate the accurate prediction results of the Faster R-CNN model.

**5.2. Pre-training for other datasets**

We also examine how our dataset can help with a real-world document digitization application. When digitizing new publications, researchers usually do not generate large scale ground truth data to train their layout analysis models. If they are able to adapt our dataset, or models trained on our dataset, to develop models on their data, they can build their pipelines more efficiently and develop more accurate models. To this end, we conduct two experiments. First we examine how layout analysis models trained on the `main` pages can be used for understanding `index` pages. Moreover, we study how the pre-trained models perform on other historical Japanese documents.

Table 4 compares the performance of five Faster R-CNN models that are trained differently on `index` pages. If the model loads pre-trained weights from HJDataset, it includes information learned from `main` pages. Models trained over all the training data can be viewed as the benchmarks, while training with few samples (five in this case) are considered to mimic real-world scenarios. Given different training data, models pre-trained on HJDataset perform significantly better than those initialized with COCO weights. Intuitively, models trained on more data perform better than those with fewer samples. We also directly use the model trained on `main` to predict `index` pages without fine-tuning. The low zero-shot prediction accuracy indicates the dissimilarity between `index` and `main` pages. The large increase in mAP from 0.344 to 0.471 after the model is

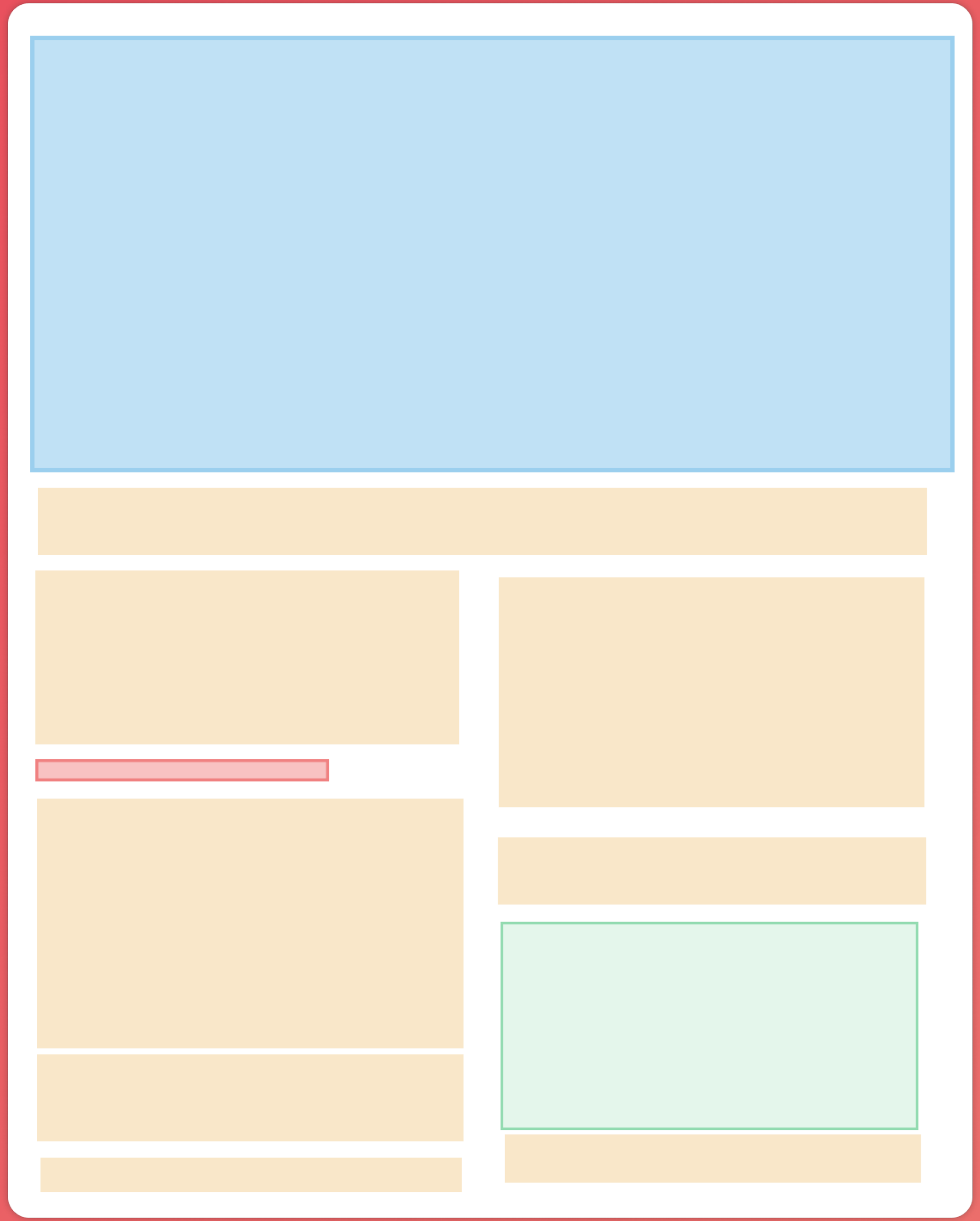
Table 3: Detection mAP @ IOU [0.50:0.95] of different models for each category on the test set. All values are given as percentages.

| Category | Faster R-CNN | Mask R-CNN[a] | RetinaNet |
|---|---|---|---|
| Page Frame | 99.046 | 99.097 | 99.038 |
| Row | 98.831 | 98.482 | 95.067 |
| Title Region | 87.571 | 89.483 | 69.593 |
| Text Region | 94.463 | 86.798 | 89.531 |
| Title | 65.908 | 71.517 | 72.566 |
| Subtitle | 84.093 | 84.174 | 85.865 |
| Other | 44.023 | 39.849 | 14.371 |
| mAP | 81.991 | 81.343 | 75.223 |

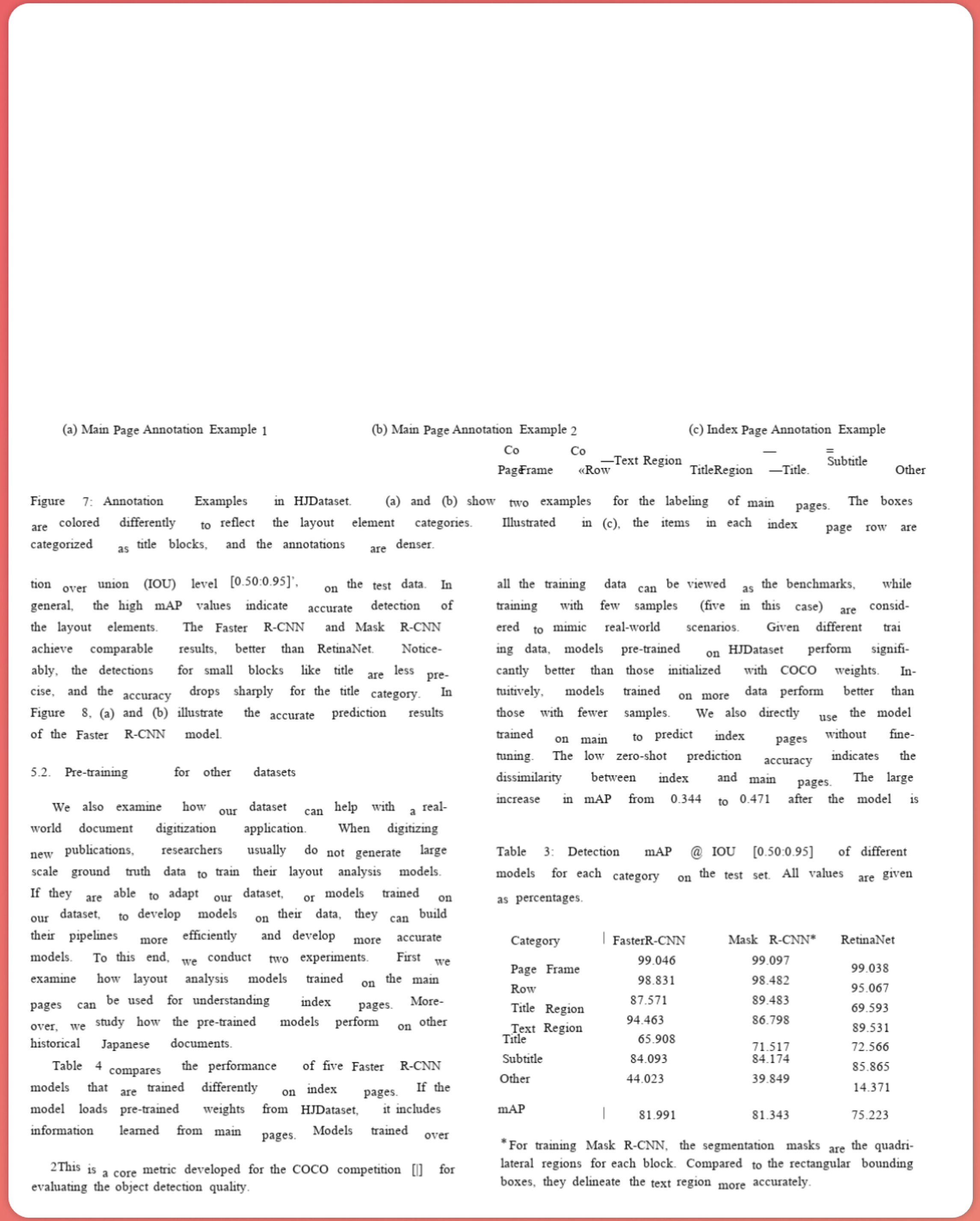
[a] For training Mask R-CNN, the segmentation masks are the quadrilateral regions for each block. Compared to the rectangular bounding boxes, they delineate the text region more accurately.

[2]This is a core metric developed for the COCO competition [12] for evaluating the object detection quality.

Image



Layout



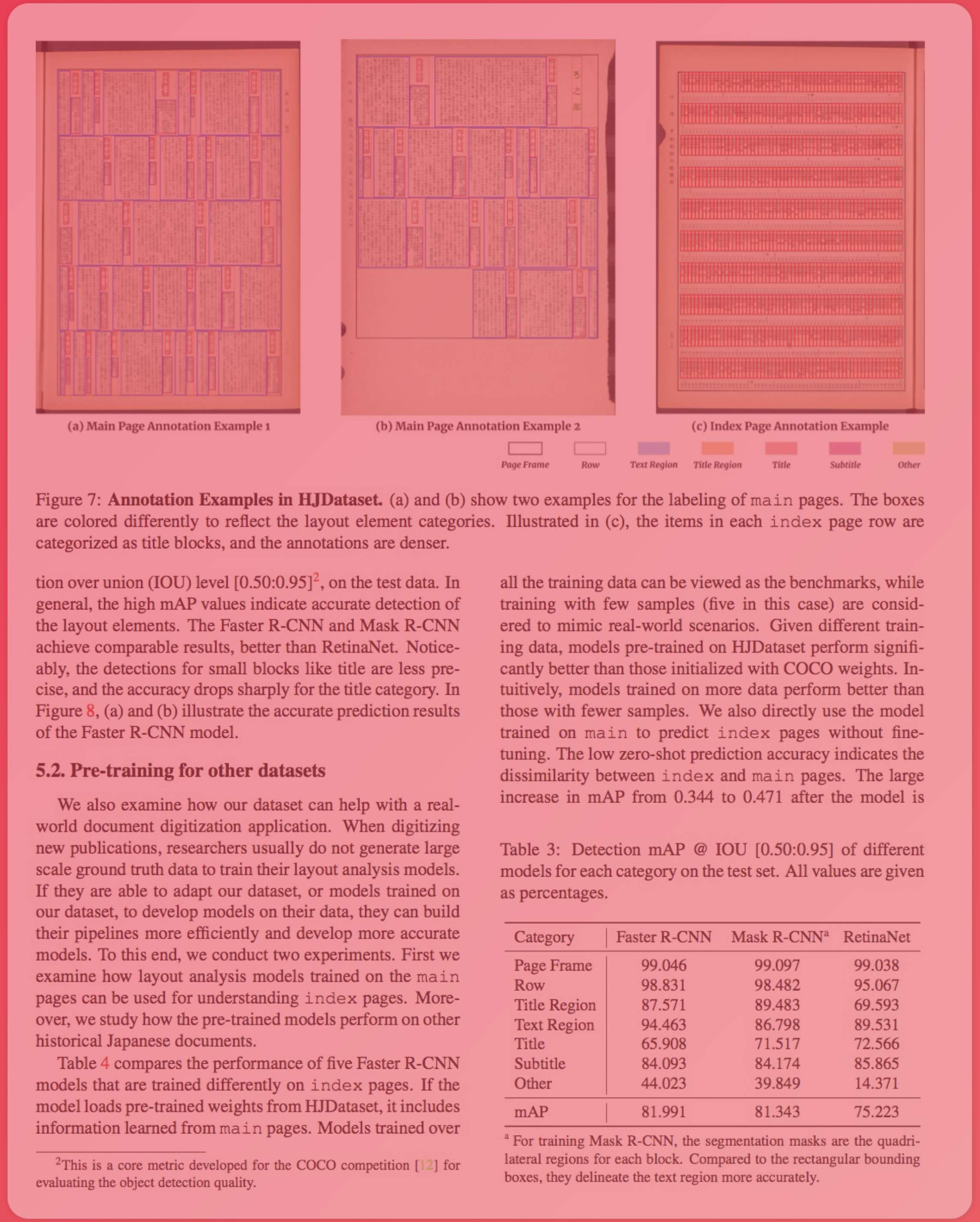
Text

# Future Work

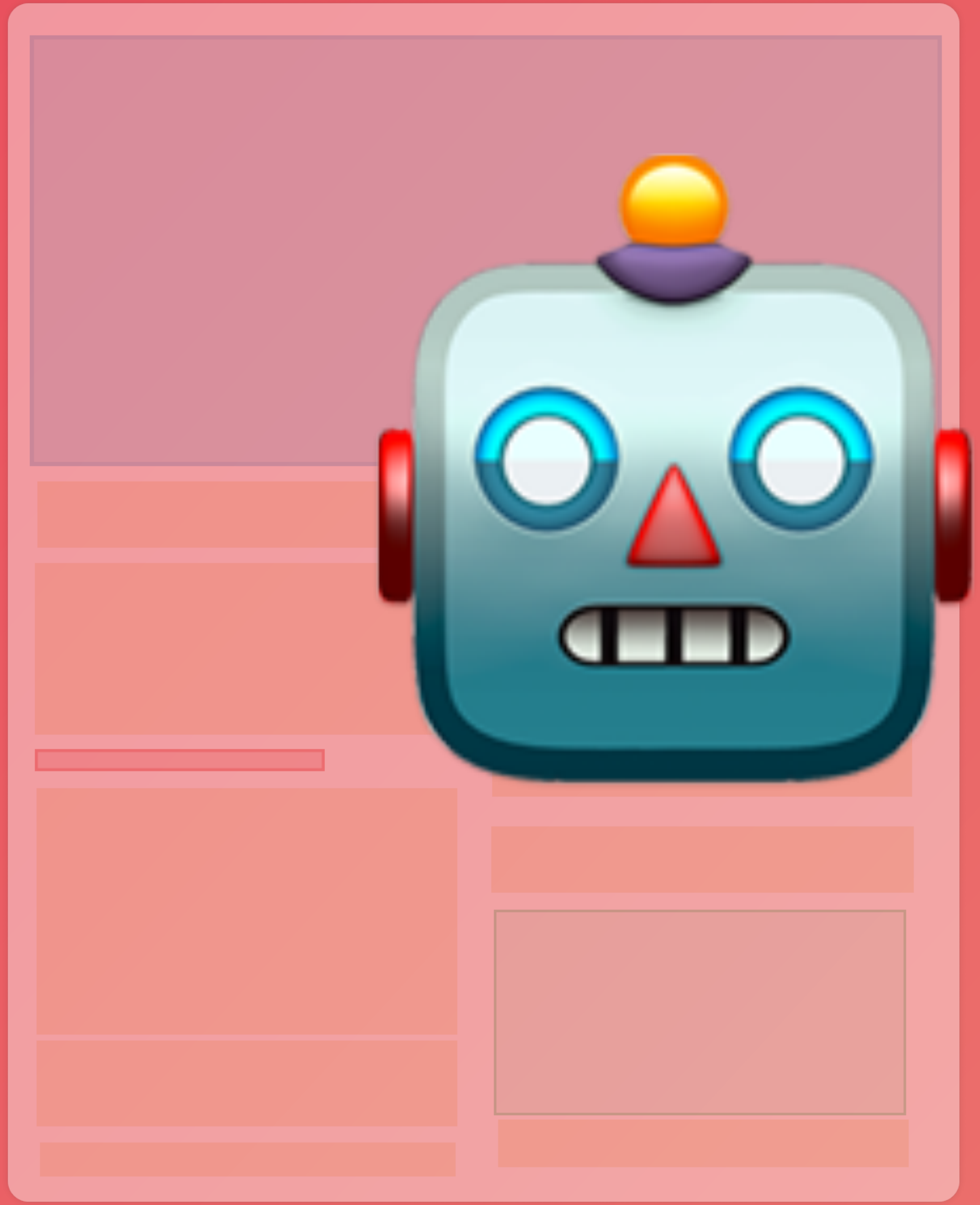
Generalized Models

Multimodal Modeling

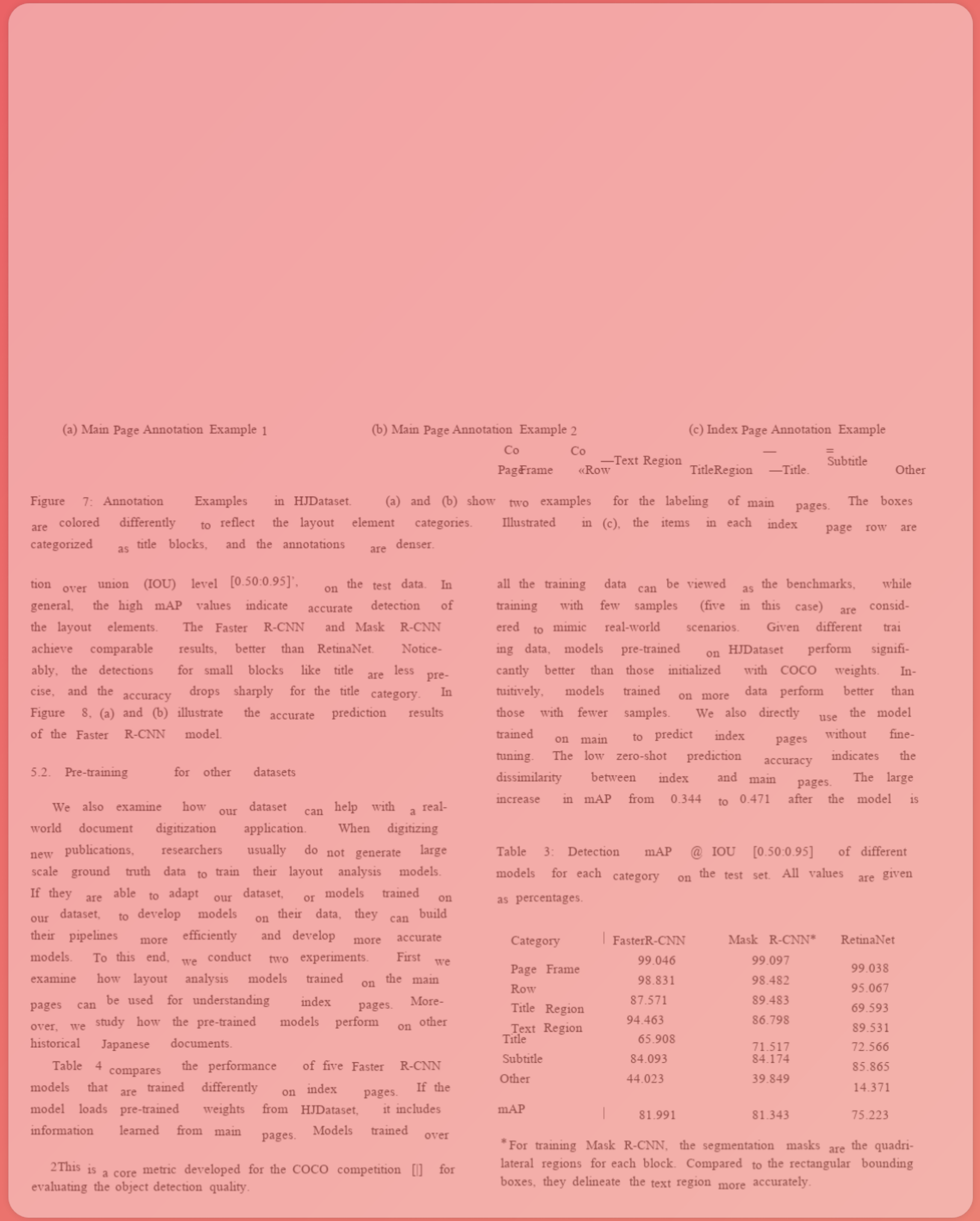
## Can we design better ways that model doc image, layout, and text together?



Image



Layout



Text

# Future Work

Generalized Models

Multimodal Modeling

**MMDA**

*Multi Model Document Analysis*

# Our Contributions Layout Parser

Layout Parser

# Our Contributions

## A unified DIA toolkit

Layout Parser

A unified DIA toolkit

# Our Contributions

Open the box usage

Layout Parser

A unified DIA toolkit

Open the box usage

# Our Contributions

Deep learning integration

Layout Parser

A unified DIA toolkit

Open the box usage

Deep learning integration

Our Contributions

Simple APIs + customization

Layout Parser

A unified DIA toolkit

Open the box usage

Deep learning integration

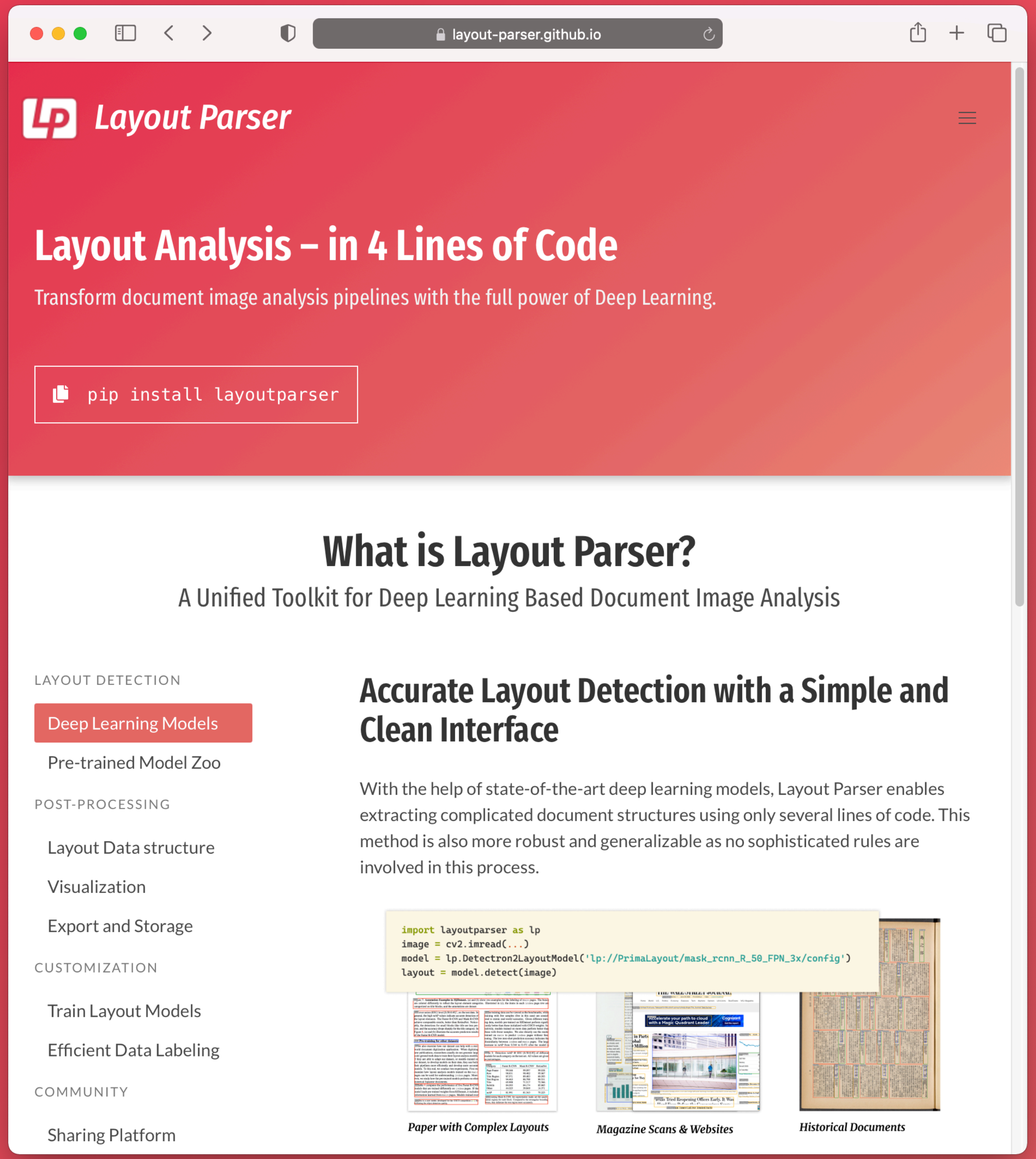
Simple APIs + customization

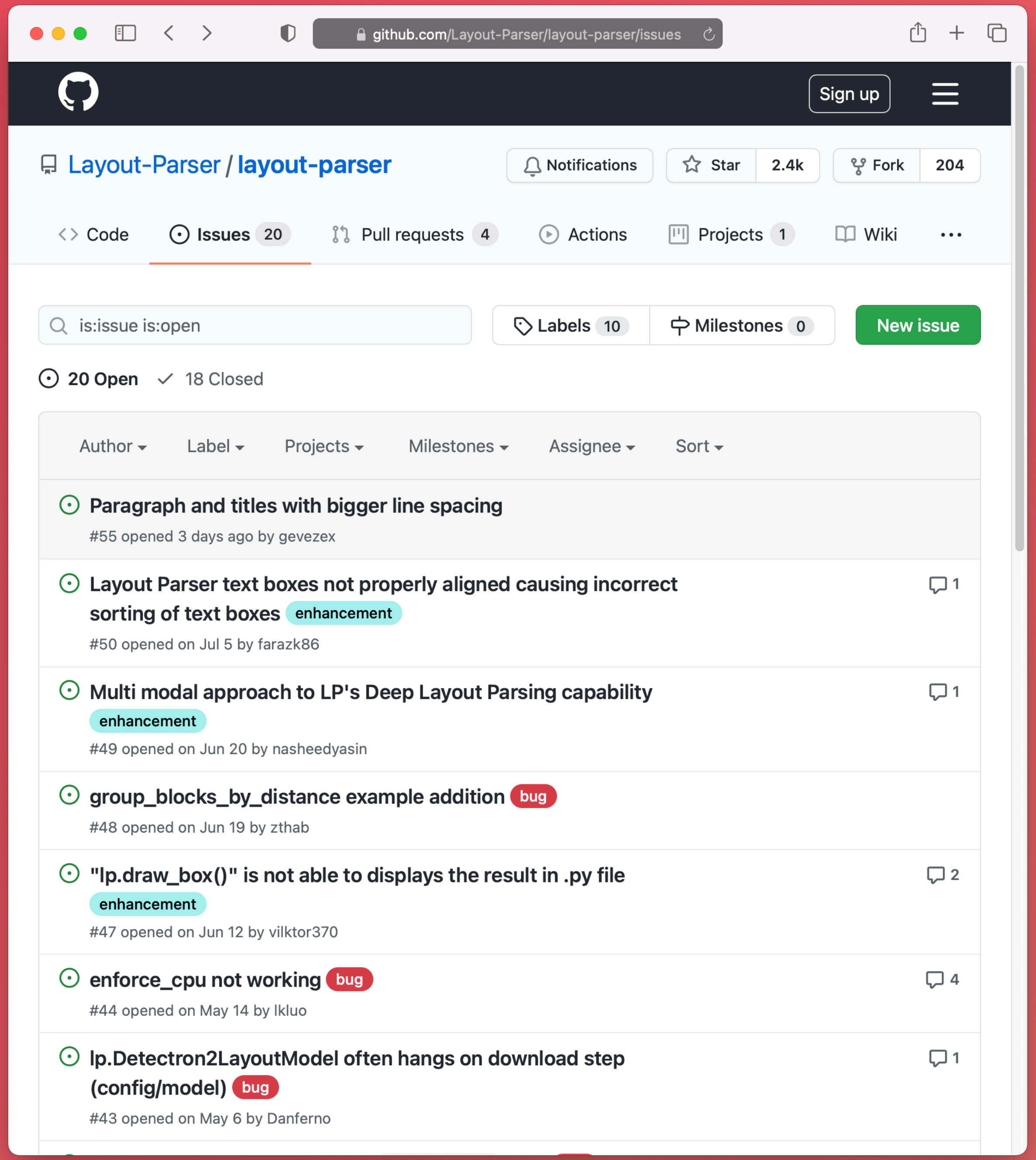
# Our Contributions

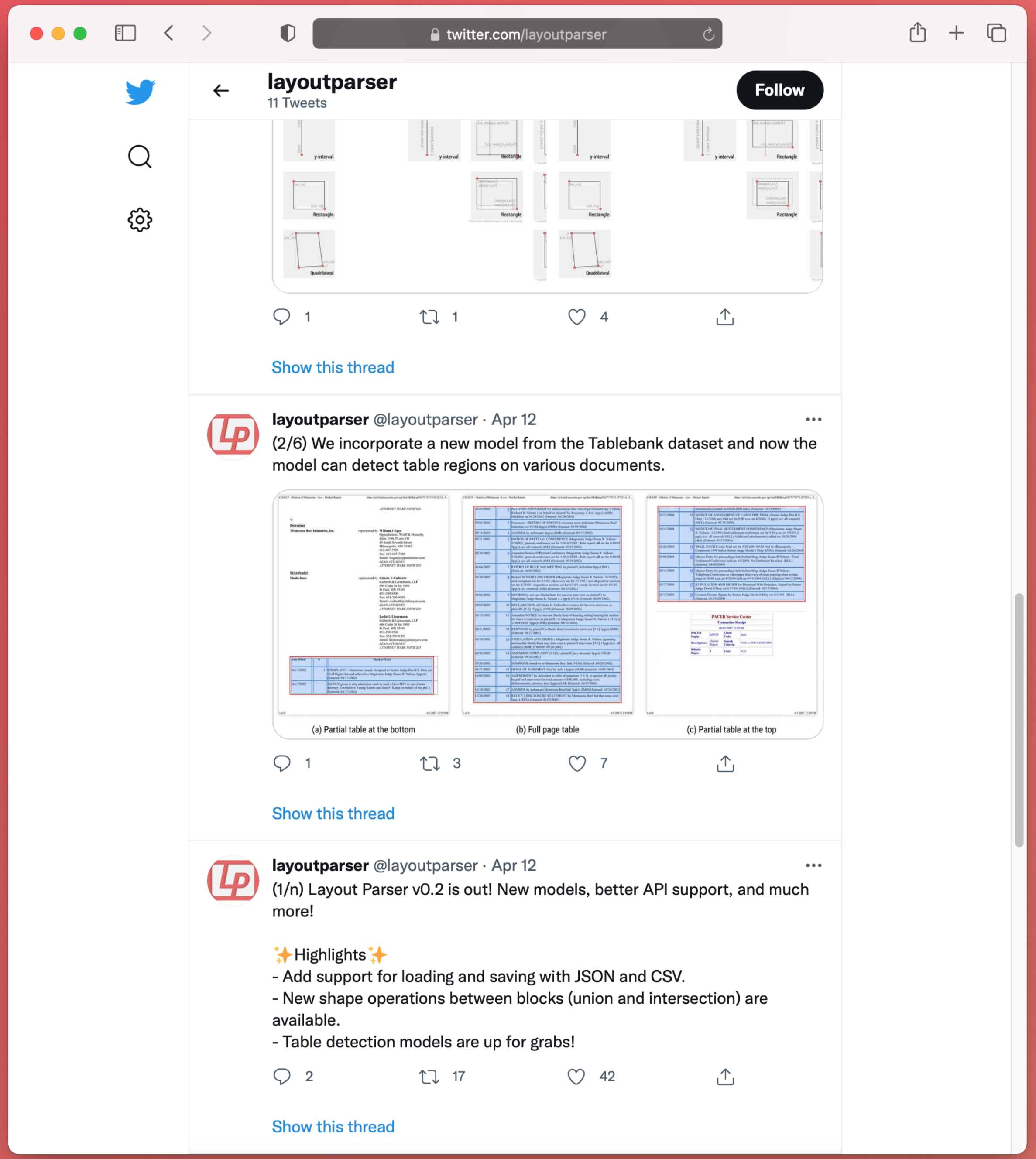
Open platform & community

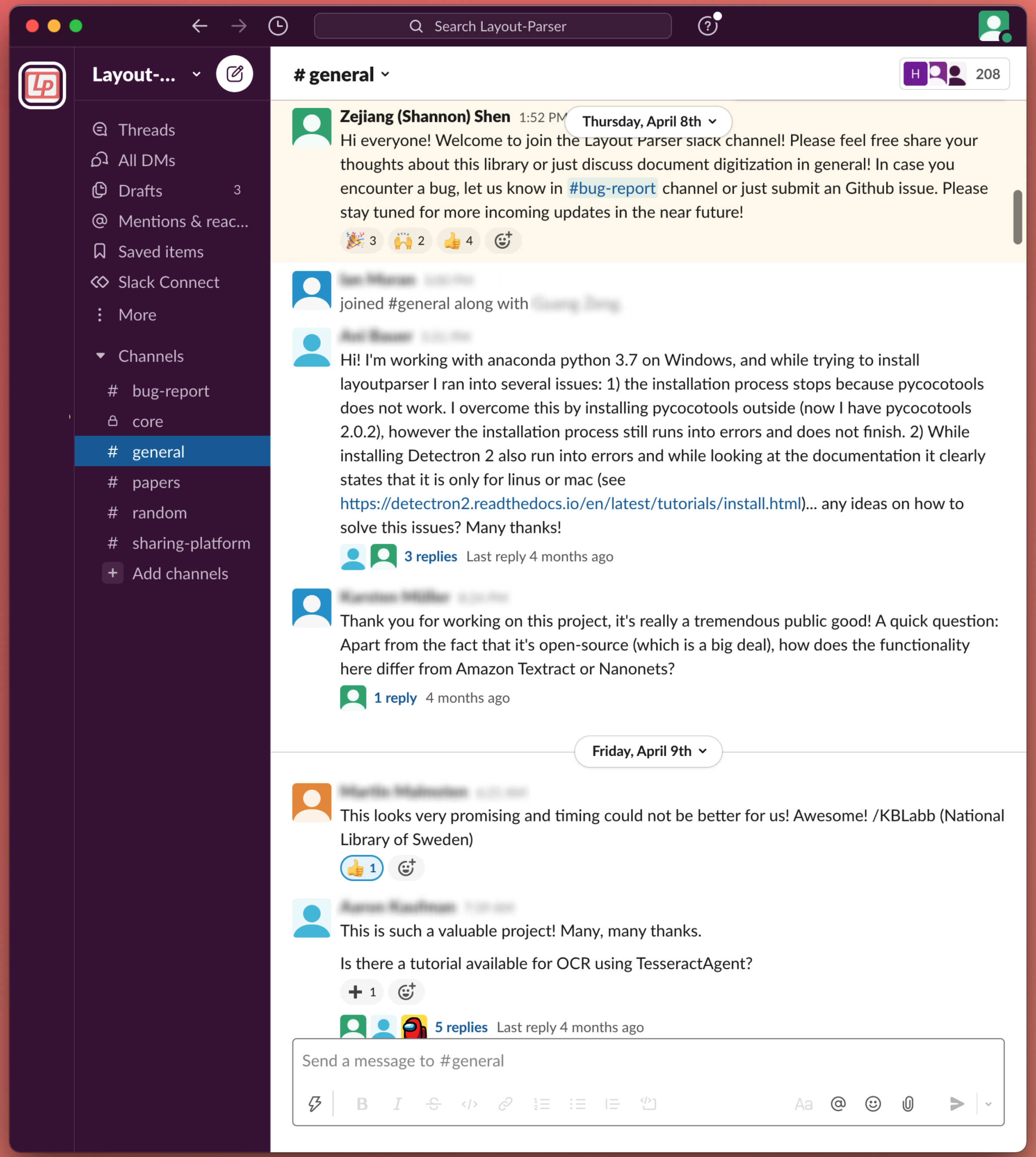
# Layout Parser

## Community & Discussion




**Website**
*layout-parser.github.io*

**Github**
*@layout-parser*

**Twitter**
*@layoutparser*

**Slack**
*layout-parser.slack.com*

** No "-" for twitter account name, as it disallows...*

Open-the-box Usage

Modularized Design

Open Sharing Platform

Layout Visualization

Deep Learning Integration

DL Models Customization

OCR Engine Support

Data Import and Export

Layout Data Annotation

Multi-backend support

Modeling Tutorials

Commandline Tools*

# Layout Parser

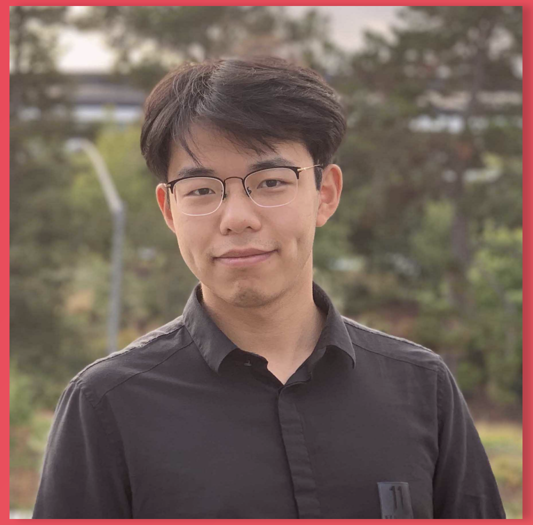
**Website**
layout-parser.github.io

**Github**
@layout-parser

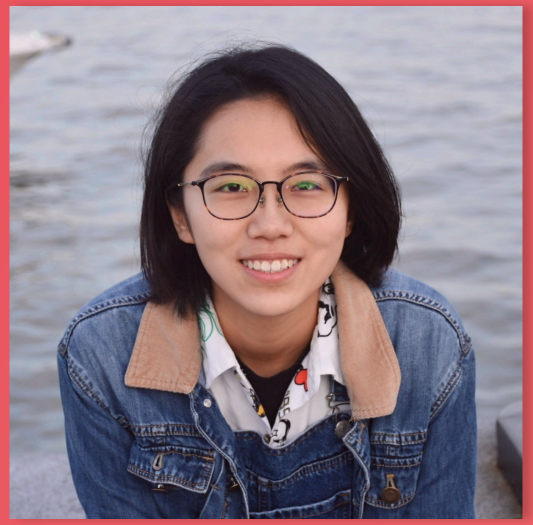
**Twitter**
@layoutparser

**Slack**
layout-parser.slack.com

---



Zejiang Shen
@ZejiangS



Ruochen Zhang
@ruochenz_

Melissa Dell
@MelissaLDell

Benjamin Lee
@lee_bcg

Jacob Carlson
@J_S_Carlson

Weining Li
@WeiningLiCA