Table 1: Current layout detection models in the LayoutParser model zoo

| Dataset | Base Model[1] | Large Model | Notes |
|---|---|---|---|
| PubLayNet [38] | F / M | M | Layouts of modern scientific documents |
| PRImA [3] | M | - | Layouts of scanned modern magazines and scientific reports |
| Newspaper [17] | F | - | Layouts of scanned US newspapers from the 20th century |
| TableBank [18] | F | F | Table region on modern scientific and business document |
| HJDataset [31] | F / M | - | Layouts of history Japanese documents |

[1] For each dataset, we train several models of different sizes for different needs (the trade-off between accuracy vs. computational cost). For "base model" and "large model", we refer to using the ResNet 50 or ResNet 101 backbones [13], respectively. One can train models of different architectures, like Faster R-CNN [28] (F) and Mask R-CNN [12] (M). For example, an F in the Large Model column indicates it has a Faster R-CNN model trained using the ResNet 101 backbone. The platform is maintained and a number of additions will be made to the model zoo in coming months.

*layout data structures*, which are optimized for efficiency and versatility. 3) When necessary, users can employ existing or customized OCR models via the unified API provided in the *OCR module*. 4) LayoutParser comes with a set of utility functions for the *visualization and storage* of the layout data. 5) LayoutParser is also highly customizable, via its integration with functions for *layout data annotation and model training*. We now provide detailed descriptions for each component.

### 3.1 Layout Detection Models

In LayoutParser, a layout model takes a document image as an input and generates a list of rectangular boxes for the target content regions. Different from traditional methods, it relies on deep convolutional neural networks rather than manually curated rules to identify content regions. It is formulated as an object detection problem and state-of-the-art models like Faster R-CNN [28] and Mask R-CNN [12] are used. This yields prediction results of high accuracy and makes it possible to build a concise, generalized interface for layout detection. LayoutParser, built upon Detectron2 [35], provides a minimal API that can perform layout detection with only four lines of code in Python:

```
import layoutparser as lp
image = cv2.imread("image_file") # load images
model = lp.Detectron2LayoutModel(
    "lp://PubLayNet/faster_rcnn_R_50_FPN_3x/config")
layout = model.detect(image)
```

LayoutParser provides a wealth of pre-trained model weights using various datasets covering different languages, time periods, and document types. Due to domain shift [7], the prediction performance can notably drop when models are applied to target samples that are significantly different from the training dataset. As document structures and layouts vary greatly in different domains, it is important to select models trained on a dataset similar to the test samples. A semantic syntax is used for initializing the model weights in LayoutParser, using both the dataset name and model name `lp://<dataset-name>/<model-architecture-name>`.