OLALA: Object-Level Active Learning for Efficient Document Layout Annotation

Zejiang Shen[†*] **Weining Li**[◇] **Jian Zhao**[◇] **Yaoliang Yu**[◇] **Melissa Dell**[♣]
[†]MIT [◇]University of Waterloo [♣]Harvard University
zjshen@mit.edu melissadell@fas.harvard.edu
{w422li,jianzhao,yaoliang.yu}@waterloo.ca

Abstract

Layout detection is an essential step for accurately extracting structured contents from historical documents. The intricate and varied layouts present in these document images make it expensive to label the numerous layout regions that can be densely arranged on each page. Current active learning methods typically rank and label samples at the *image level*, where the annotation budget is not optimally spent due to the overexposure of common *objects* per image. Inspired by recent progress in semi-supervised learning and self-training, we propose OLALA, an **O**bject-**L**evel **A**ctive **L**earning framework for efficient document layout **A**nnotation. OLALA aims to optimize the annotation process by selectively annotating only the most ambiguous regions within an image, while using automatically generated labels for the rest. Central to OLALA is a perturbation-based scoring function that determines which objects require manual annotation. Extensive experiments show that OLALA can significantly boost model performance and improve annotation efficiency, facilitating the extraction of masses of structured text for downstream NLP applications.[1]

1 Introduction

When working with historical documents, social scientists have often used keyword methods that do not require the recognition of structured layouts (see *e.g.* Hanlon and Beach (2022) for a review of the literature on historical newspapers). To apply neural NLP methods to these documents, it is essential to accurately detect the layouts and extract the structured content. For example, a historical newspaper scan contains a mixture of article regions, headlines, captions, advertisements, etc. Commercial OCR software will typically read the multi-

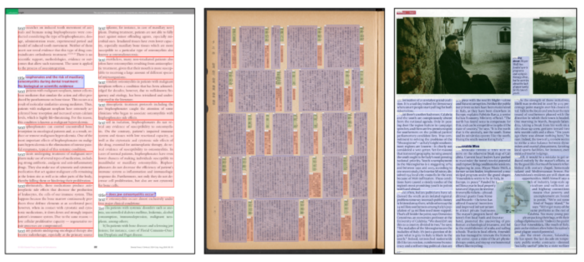


Figure 1: Three exemplar document layouts from PubLayNet (Zhong et al., 2019), HJDataset (Shen et al., 2020), and PRImA (Antonacopoulos et al., 2009). There are numerous layout objects per page, and many of them are very similar. Directly labeling them all will result in wasted labeling budget.

column document as if it is a single column book, unable to distinguish content in different regions and producing scrambled text that leads to poor performance for downstream NLP applications.

Deep learning-based approaches can be used for document layout analysis and content parsing (Shen et al., 2021; Zhong et al., 2019; Schreiber et al., 2017). Figure 1 illustrates that document layout object detection, like image object detection, requires identifying content regions and categories within images. A key distinction, however, is that it is common for dozens to hundreds of content regions to appear on a single page in documents, as opposed to only several objects per image in natural image datasets (*e.g.* 5 on average in the MS-COCO Dataset (Lin et al., 2014)). Additionally, the region category distribution is often heavily imbalanced and requires more pages to be annotated to allow for reasonable exposure of uncommon categories (*e.g.* footnotes, watermarks, or mastheads). Hence, the manual labeling process used on natural images to create high-quality labeled datasets can be prohibitively costly to replicate for documents of central interest to social scientists, who typically have heavily constrained annotation budgets. As a result, extracting structured text is often infeasible,

[*] Work done when working as a Data Science Fellow at Harvard University.

[1]Our source code is available at https://github.com/lolipopshock/detectron2_al.

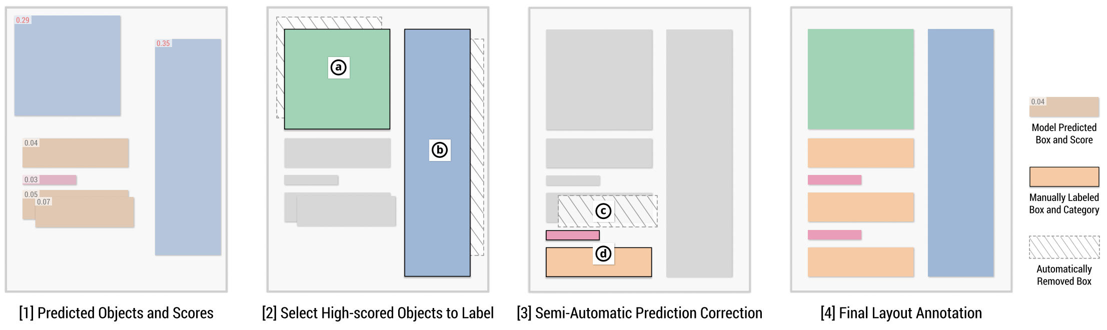

0.29
0.35
0.04
0.03
0.05
0.07
a
b
c
d
0.04
Model Predicted Box and Score
Manually Labeled Box and Category
Automatically Removed Box
[1] Predicted Objects and Scores
[2] Select High-scored Objects to Label
[3] Semi-Automatic Prediction Correction
[4] Final Layout Annotation

Figure 2: Illustration of the `OLALA` framework. [1] During labeling, for an input image, a trained model predicts the layout with various errors. An object scoring function f evaluates the informativeness for each object prediction. [2] `OLALA` selects the regions of top scores and sends them for manual labeling to correct the wrong object category (a) and bounding box (b). [3] A semi-automatic prediction correction algorithm is applied to rectify duplicated objects (c) and recover false-negatives (d) with minimal extra supervision. [4] After this process, the final annotation is obtained from labeling only a portion of the objects.

limiting the application of modern NLP techniques in historical document applications.

Active Learning (AL) has been widely adopted in image object detection for optimizing labeling efficiency via prioritizing the most important samples to annotate (Aghdam et al., 2019; Haussmann et al., 2020; Brust et al., 2018; Roy et al., 2018). However, while the end goal is to annotate individual objects within an image, these AL methods typically score and select samples at the image level, rather than at the object level. Yao et al. (2012) study an annotator-centered labeling cost estimation method and prioritize labeling for high-cost images. In the context of deep learning, image level scores are generated via aggregation of marginal scores for candidate boxes (Brust et al., 2018) or applying query by committee (Seung et al., 1992) to features maps (Roy et al., 2018). Aghdam et al. (2019) propose a pixel level scoring method using convolutional backbones and aggregate them to informativeness scores for image ranking. For category-imbalanced layouts, common in documents, such image level selection can suffer from the over-exposure of common objects.

Recent advances in Semi-Supervised Learning (SSL) and self-training can boost model performance using unlabeled data (Rosenberg et al., 2005; Xie et al., 2020). The Self-supervised Sample Mining (SSM) algorithm (Wang et al., 2018) proposes to stitch high-confidence patches from unlabeled data to labeled data to improve both labeling efficiency and model performance. It enables object-level prediction selection but requires objects to be sparsely distributed, making it inapplicable to our case where content is densely arranged.

To address these challenges, we propose a novel AL framework, `OLALA`, **O**bject-**L**evel **A**ctive **L**earning for efficient layout **A**nnotation. Shown in Figure 2, critical *objects*, rather than *images*, are individually evaluated and selected for labeling. During the labeling process, `OLALA` trains a model to generate object predictions. Within an image, only the most ambiguous predictions are chosen for human inspection and annotation, addressing the inefficient use of annotation budget on common objects or categories. Central to this process is a semi-automatic prediction correction algorithm. Inspired by previous endeavors of automated layout dataset generation (Zhong et al., 2019; Li et al., 2019; Shen et al., 2020), `OLALA` incorporates prior knowledge about layout structures to ensure the high quality of the created dataset. It can identify false-positives and false-negatives in the unselected model predictions, and correct them with minimal extra supervision.

Additionally, we design a novel object-level scoring function governing the region selection process. The perturbation-based scoring method evaluates consistency of both object position and category predictions between the original and perturbed inputs. Compared to prior work, it is carefully designed for layout datasets with unique arrangement of content regions, and can identify errors of critical importance to layout analysis tasks.

In other contexts where predictions on unlabeled images (Wang et al., 2018; Xie et al., 2020) or weak labels (Desai et al., 2019) are used to boost model performance, the predicted labels are discarded after model training. OLALA includes a rigorous process to validate the accuracy of predictions, meaning that the full labeled dataset - created by human and machine - can be released publicly and potentially used by social scientists for transfer learning on other applications.

Through extensive experiments, we study how the proposed approach can improve labeling efficiency in two different scenarios. We show that `OLALA` can create datasets with better trained model performance compared to image-level AL baselines, for a given limited annotation budget. On the other side, as only part of an image requires annotation in our method, we demonstrate that our method can create datasets of the same size with far less human effort.

To the best of our knowledge, this is the first AL method dedicated to document layout analysis. `OLALA` was motivated by our need to automate the extraction of structured text from millions of historical documents, to enable modern NLP analyses on information trapped in hard copy. We are using it extensively on real world documents for this purpose, with the `OLALA` labeling interface described in the supplementary material.

Section 2 introduces the OLALA framework, and Section 3 describes the perturbation based object scoring method. Sections 4 and 5 demonstrate how `OLALA` can improve labeling efficiency and model performance under different scenarios.

2 The `OLALA` Framework

2.1 Object-Level Active Learning Setup

In layout object detection problems, a detection model Θ is trained to identify n_i objects within an input image X_i, where the bounding box b_j and category distribution c_j is estimated for the j-th object. $Y_i = \{(b_j, c_j)\}_{j=1}^{n_i}$ are the object annotations for X_i. Θ is initially trained on a small labeled dataset $\mathcal{L}_0 = \{(X_i, Y_i)\}_{i=1}^{l}$, and it receives a large unlabeled dataset $\mathcal{U}_0 = \{X_i\}_{i=1+l}^{u+l}$.

The goal of typical image-level AL methods (Aghdam et al., 2019; Brust et al., 2018; Roy et al., 2018) is to optimally sample images from $\mathcal{U}$ for annotation to maximally improve the model's performance on given metrics. This process could be iterative: at each round t, it selects m samples $\mathcal{M}_t = \{X_i\}_{i=1}^{m}$ from $\mathcal{U}_{t-1}$ to query labels, obtains the corresponding labeled set $\bar{\mathcal{M}}_t = \{(X_i, Y_i)\}_{i=1}^{m}$, and updates the existing labeled set $\mathcal{L}_t = \mathcal{L}_{t-1} \cup \bar{\mathcal{M}}_t$. The new model Θ_t is obtained by training (or fine-tuning) on $\mathcal{L}_t$. For the next round, the unlabeled set becomes $\mathcal{U}_t = \mathcal{U}_{t-1} \setminus \mathcal{M}_t$.

In this process, annotators need to create all object labels $\bar{Y}_i = Y_i$ for the images in $\mathcal{M}_t$. This is not optimal for layout object detection, where many objects could appear on a single image. Because of the uneven distribution of objects, sometimes only a small portion of object predictions in an image are inaccurate. Labeling whole images wastes budget, which could be otherwise used for labeling less common and accurate objects.

Consider an alternative setup illustrated in Figure 2: the AL agent prioritizes annotation for a portion of objects in Y_i within each image. An *object*-level scoring function f evaluates the ambiguities of predictions generated by Θ. Object regions of top scores, the *selected objects*, will be sent for manual annotation to create labels $\bar{Y}_i$. To wisely use human efforts, the ratio of selected objects r is dynamically adjusted during the labeling process (Section 2.2). And after correcting possible errors (Section 2.3), the remaining *unselected objects* constitute the complement labels $\hat{Y}_i$ and are merged with the human labels. The *Objects Selection Scheduling* and *Semi-automatic Prediction Correction* ensure the combined annotation $\tilde{Y}_i = \bar{Y}_i \cup \hat{Y}_i$ is close to Y_i. Therefore, accurate dataset annotations can be created with only $|\bar{Y}_i|/|\tilde{Y}_i|$ of time ($|\cdot|$ being the cardinality of the set), and more images can be annotated given the same labeling budget. This is our object-centered labeling setup in `OLALA`.

2.2 Objects Selection Scheduling

The ratio of selected objects during training can influence the labeling efficiency as well as the trained model accuracy. A ratio near 1 approximates the full human labeling process (less efficient), while a zero ratio resembles full self-training (Rosenberg et al., 2005) settings (less accurate). To optimally balance efficiency and accuracy, r is dynamically adjusted at different rounds of labeling via a scheduling function. According to Curriculum Learning (Bengio et al., 2009), we set high initial values of r to rely more on human labeling and ease model training in the beginning. Linear or exponential decay is then applied to gradually decrease r, increasing the trust in the model predictions as their accuracy improves during training. From an optimization perspective, r can be seen as a "learning rate" for the `OLALA` AL process. We demonstrate the effectiveness of the proposed scheduling mechanism in the experiments (Section 5.3).

2.3 Semi-automatic Prediction Correction

Compared to recent work (Wang et al., 2018; Xie et al., 2020) using self-training for improving model performance, `OLALA` contains an additional component to fix possible errors in the utilized model predictions. Inspired by recent efforts for creating large-scale layout analysis datasets (Zhong et al., 2019; Li et al., 2019; Shen et al., 2020), we propose a semi-automatic prediction correction algorithm to ensure the quality of the model predictions. This method relies on the unique structures of document data: layout objects are densely arranged, and there is usually no overlap between content regions. It can identify *duplicated predictions* and *false-negative predictions* based on this prior knowledge, and requests minor supervision to fix them. Shown in Section 5.1 and 5.2, this algorithm both improves the final trained model accuracy and enables the creation of an accurate large dataset based on these predictions.[2]

Duplicate Removal In practice, models could generate multiple close predictions for a large object, yet only one or some of the predictions are sent for user inspection. Thus, if naively merging the user's labels with the remaining predictions, it can lead to overlapping labels for the same object. We fix this error by filtering out predictions overlapped with any human annotations over a score threshold ξ. Different from IOU scores, we use the the pairwise Overlap Coefficient, Overlap$(A, B) = |A \cap B|/\min(|A|, |B|)$, to better address scenarios where a predicted box is contained within a labeled box. The threshold ξ is set to 0.25 empirically.

Missing Annotation Recovery False-negatives occur when no prediction is generated for a given object. In typical object detection tasks, predictions are dropped when the confidence is under some threshold, which might lead to false negatives. It is an implicit signal from the model, requesting extra supervision from human annotators and is a key step for improving dataset accuracy (Section 4). It is implemented by highlighting the regions without model predictions, such that human annotators (or a simulated agent) can easily identify the mispredicted objects and add the annotations.

The implementation of this algorithm is different between real-world human annotation and simulated labeling experiments (with oracle beforehand). For human annotations, we carefully design a user interface which incorporates the three functions and augments human labeling, and we refer readers to Figure 6 in the supplementary material for more details. In simulations, we build a labeling agent that can automatically query the oracle for ground-truths under different scenarios (see Section 4).

[2]Self-training methods (*e.g.* (Wang et al., 2018)), usually discard the model predictions (pseudo labels) after training.

Algorithm 1: Object-level Active Learning Annotation

```
Input : Initial sets $\mathcal{U}_0$, $\mathcal{L}_0$; labeling budget $m$; object
        selection ratio $r$
Initialize $\mathcal{U} = \mathcal{U}_0$, $\mathcal{L} = \mathcal{L}_0$, and model weights $\Theta$;
for $t = 0$ to $T - 1$ do
    Calculate budget $m$ and selection ratio $r$ for at $t$
    Update the model $\Theta$ using $\mathcal{L}$
    Let $\bar{\mathcal{M}}$ = { }
    for $i = 0$ to $|\mathcal{U}|$ do
        Generate object predictions $\hat{Y}_i$ for $X_i \in \mathcal{U}$
        Let $m_i = \min\{r|\hat{Y}_i|, m\}$, $m = m - m_i$
        if $m \leq 0$ then break;
        Calculate object scores $f(\hat{y}_j)\ \forall \hat{y}_j \in \hat{Y}_i$
        Select $m_i$ objects of top scores and label $\bar{Y}_i$
        Correct errors in unselected predictions $\hat{Y}_i^-$
        Merge $\bar{Y}_i$ with $\hat{Y}_i$ for image annotations $\tilde{Y}_i$
        Remove $X_i$ from $\mathcal{U}$ and add $(X_i, \tilde{Y}_i)$ to $\bar{\mathcal{M}}$
    end
    Update $\mathcal{L} \leftarrow \mathcal{L} \cup \bar{\mathcal{M}}$
end
Update the model $\Theta$ using $\mathcal{L}$
```

2.4 Overview of the Proposed Algorithm

We now present the `OLALA` Algorithm 1. Given an initial labeled set $\mathcal{L}_0$, it aims to use the predictions from a model Θ to optimally label the remaining unlabeled set $\mathcal{U}_0$ given some labeling budget. Different from existing work, we define the labeling budget per round m as the number of *objects* - rather than images - that human annotators can label. The algorithm iteratively proposes the most informative objects to label for a total of T rounds. At each round t, it selects up to m objects. For each image X_i from the existing unlabeled set $\mathcal{U}$, r percent of predicted objects are selected for user labeling according to some object scoring function f. The rest of the labels are created by correcting errors in the unselected model prediction $\hat{Y}_i^-$ based on the semi-automatic prediction correction algorithm. The labeled image X_i will be removed from $\mathcal{U}$ and the annotated samples $(X_i, \tilde{Y}_i)$ will be added to $\mathcal{L}$. After each round, the selection ratio r decays as the model accuracy improves.

3 Perturbation-based Scoring Function

The scoring function f also plays an important role in the OLALA framework. It evaluates prediction ambiguity and determines which objects to select for labeling. We propose a perturbation scoring method based on both the bounding box and category predictions. Inspired by the self-diversity idea in Jiang et al. (2020) and Zhou et al. (2017), the proposed method hypothesizes that the adjacent image patches share similar features vectors, and the predicted object boxes and categories for them should be consistent. Therefore, any large disagreement between the original and perturbed predictions indicates that the model is insufficiently trained for this type of input, or there are anomalies in the sample. Both cases demand user attention.

Specifically, for each object prediction $\hat{y}_j = (\hat{b}_j, \hat{c}_j) \in \hat{Y}_i$, we take the bounding box prediction $\hat{b}_j = (x, y, w, h)$ and apply some small shifts to perturb the given box, where x, y are the coordinate of the top left corner, and w, h are the width and height of the box. The new boxes are created via horizontal and vertical translation by a ratio of α and β: $p_{jk} = (x \pm \alpha w, y \pm \beta h, w, h)$, where p_{jk} is the k-th perturbed box for box prediction $\hat{b}_j$, and a total of K perturbations will be generated. Based on the image features within each p_{jk}, the model generates new box and category predictions (q_{jk}, v_{jk}). We then measure the disagreement between the original prediction $(\hat{b}_j, \hat{c}_j)$ and the perturbed versions $\{(q_{jk}, v_{jk})\}_{k=1}^{K}$, and use it as a criterion for selecting objects for labeling.

In practice, we build this method upon a typical object detection architecture composed of two stages (Ren et al., 2015): 1) a region proposal network estimates possible bounding boxes, and 2) a region classification and improvement network (ROIHeads[3]) predicts the category and modifies the box prediction based on the input proposals. We use the perturbed boxes $\{p_{jk}\}_{k=1}^{K}$ as the new inputs for the ROIHeads, and obtain the new box and class predictions $\{(q_{jk}, v_{jk})\}_{k=1}^{K}$. For object regions of low confidence, the new predictions are unstable under such perturbation, and the predicted boxes and category distribution can change drastically from the original version. To this end, we formulate the position disagreement D_p and the category disagreement D_c for the j-th object prediction as

$$D_p(\hat{b}_j) = \frac{1}{K} \sum\nolimits_k \left(1 - \text{IOU}(\hat{b}_j, p_{jk})\right)$$
$$D_c(\hat{c}_j) = \frac{1}{K} \sum\nolimits_k L(\hat{c}_j || v_{jk}),$$

where IOU calculates the intersection over union scores for the inputs, and $L(\cdot||\cdot)$ is a measurement for distribution difference, e.g., cross entropy. The overall disagreement D is defined as $D(\hat{y}_j) = D_p(\hat{b}_j) + \lambda D_c(\hat{c}_j)$, with λ being a weighting constant. Objects of larger D will be prioritized for labeling, and users will create annotations $\bar{Y}_i$ for them in the i-th image.

The proposed method can effectively identify false-positive object predictions. Based on the self-diversity assumption, incorrect category prediction $\hat{c}_j$ will cause high D_c because of the divergence of the new class prediction v_{jk} for nearby patches. When the predicted box $\hat{b}_j$ is wrong, the perturbed box p_{jk} is less likely to be the appropriate proposal box. The generated predictions (q_{jk}, v_{jk}) are unreliable, causing higher overall disagreement D.

Applicability to Layout Datasets Compared to previous work, the perturbation-based scoring function aims to solve two challenges unique to layout analysis tasks. First, layout regions are boundary-sensitive: a small vertical shift of a text region box could cause the complete disappearance of a row of texts. However, existing methods designed for image-level selection usually focus on the categorical—rather than positional—information in outputs (i.e. Brust et al. (2018) considers the marginal score of the object category predictions and does not use the bounding boxes, and Aghdam et al. (2019) indirectly uses the positional information based on a pixel map for image-level aggregation). By contrast, our method identifies samples that lead to ambiguous boundary predictions via D_p.

Moreover, document images usually contain numerous objects per page and content regions are densely arranged. Hence, we cannot adapt the object-level scoring function in Wang et al. (2018), which requires cropping an object, randomly pasting it to another image, and evaluating the consistency between the original and the newly detected boxes for this object. The random pasting will introduce non-existing structures (*e.g.* , overlaying a figure over tables or texts), and the calculated score cannot reliably assess the prediction. With OLALA, the original document structures are untouched.

[3]It's a module name in Detectron2 (Wu et al., 2019).

4 Experimental Setup

Objective Several experiments are designed to study the validity of the proposed `OLALA` framework and evaluate how it can improve the efficiency of the labeling process. Methods are considered better if they achieve similar accuracy while using less labeling budget m than their counterparts, or obtain higher accuracy given the same m. In the experiments, we measure object detection accuracy using mean Average Precision (AP) scores (Lin et al., 2014), and the labeling budget refers to the number of objects to label by default.

Datasets To validate our approach, we run simulations on three representative layout analysis datasets: PubLayNet (Zhong et al., 2019), PRImA (Antonacopoulos et al., 2009), and HJDataset (Shen et al., 2020). PubLayNet is a large dataset of 360k images. The images and annotations are generated from noiseless digital PDF files of medical papers. As the original training set in PubLayNet is too large to conduct experiments efficiently, we use a downsampled version. PRImA is created by human annotators drawing bounding boxes for text regions in both scanned magazines and technical articles, resulting in greater heterogeneity in this dataset than in PubLayNet. HJDataset contains layout annotation for 2k historical Japanese documents. HJDataset was established using noisy image scans, and the creation method is a combination of rule-based layout parsing from images and human correction. Table 1 shows a thorough comparison among them.

Labeling Simulation When running simulations, we build two additional helper algorithms to imitate human labeling behavior. First, for the selected objects, the corresponding ground-truths is found via a best-matching algorithm. For each prediction, we calculate the IOU with all ground-truth objects and choose the top one to substitute the prediction. Duplicated ground-truths selected in an image will be removed by this process. In real-world labeling experiments, we also notice human annotators do not need to correct an object prediction if it is accurate (high IOU with the ground-truth and category is the same). To best simulate this phenomena, if a selected prediction has an IOU>0.925 (determined empirically) with a ground-truth object of the same category, we do not substitute it with the ground-truth and only use a discounted budget $\eta = 0.2$. Finally, to mimic annotators' search for false-negative regions, we compute the pairwise IOU between the ground truth Y_i and the combined labeling objects $\tilde{Y}_i$. Ground-truth objects whose maximum IOU with predicted objects is less than ζ are chosen to add to $\tilde{Y}_i$, and the remaining budget is reduced accordingly. ζ is set to 0.05 in the following experiments to allow minor overlapping caused by noise in the predictions.

Datasets	PubLayNet	HJDataset	PRImA
Data Source	Digital PDF	Image Scan	Image Scan
Annotation	Automatic	Combined	Manual
Dataset Size	360,000	2,048	453
Train Size	8,896*	1,433	363
Test Size	2,249	307	90
Avg / max O	10.72 / 59	73.48 / 98	21.63 / 79
Labeling budget m	21,140	51,436	5,623
Equivalent Images	2,000	700	240
Total rounds T	10	8	4
Initial / last r	0.9 / 0.4	0.9 / 0.5	0.9 / 0.75

* We used a downsampled version of PubLayNet in our experiments.

Table 1: Statistics and parameters for the PubLayNet, HJDatasets, and PRImA. O is the number of objects in each image.

Implementation The proposed algorithms are implemented based on Detectron2 (Wu et al., 2019). The same object detection model (Faster R-CNN (Ren et al., 2015) with ResNet-50 (He et al., 2016) backbone and FPN (Lin et al., 2017)) is used for all experiments. The optimizer is based on SGD with Momentum (Sutskever et al., 2013) and Multi-Step learning rate warmup (Goyal et al., 2017) with a 0.00025 base learning rate. We train each model on a single Tesla V100 GPU with a batch size of 6.

The total labeling budget m and the total rounds T are set per dataset to account for different dataset sizes, and the labeling budget is evenly distributed for each round. For the object selection ratio, we use a linear decay function with a given initial and last value. These hyperparameters are initialized as indicated in Table 1. When calculating the object scores, we set λ to 1 and L as the cross entropy function. In addition, unless otherwise mentioned, we use four pairs of (α, β)'s: $(0.08, 0.04)$, $(0.08, 0.16)$, $(0.12, 0.04)$, $(0.12, 0, 16)$, and for each pair, four boxes are created (moving towards top left, top right, bottom left, and bottom right). A total of $K = 16$ perturbed boxes are generated per object prediction for comprehensive analysis of prediction performance under small and large perturbations in different directions.

	PubLayNet		HJData		PRImA*	
Experiments	Final AP	Labeled I / O	Final AP	Labeled I / O	Final AP	Labeled I / O
Image-Random [a]	60.73	2,046 / 21,430	69.82	709 / 51,959	31.49	244 / 4,799
`OLALA`-Random [c]	**64.21(+3.48)**†	3,187 / 21,412	**72.16(+2.34)**	1,105 / 51,626	**32.08(+0.59)**	277 / 4,785
Image-Marginal [b]	67.91	2,465 / 21,574	73.25	709 / 51,937	30.99	243 / 4769
`OLALA`-Marginal [d]	**69.23(+1.31)**‡	3,661 / 21,467	71.48(-1.77)	1,075 / 51,804	32.85(+1.86)	306 / 4,721
`OLALA`-Pertubation [e]	69.13(+1.21)	3686 / 21430	**73.40(+0.15)**	1,159 / 51,656	**33.87(+2.88)**	286 / 4,764

* The results in PRImA are averaged from the 5-folds in cross validation to account for possible noise due to the small dataset size.
†,‡ The `OLALA`-Random percentages are compared against Image-Random, and others are compared against Image-Marginal.

Table 2: The final AP and number of total labeled images I and objects O given the same object budget m. `OLALA` achieves strong performance improvements in model accuracy in all experiments, and creates datasets with considerably more images given the same labeling budget.

5 Results and Discussion

5.1 Better AP with the Same Budget

`OLALA` based labeling settings are compared against image-level AL and other labeling baselines: [a] Image-Random: randomly select images in each round, [b] Image-Marginal: image-level Active Learning baselines (Brust et al., 2018) with marginal scoring and mean aggregation, [c] `OLALA`-Random: randomly select objects in each round, [d] `OLALA`-Marginal: select objects using marginal scoring for object category prediction, [e] `OLALA`-Perturbation: select objects using the proposed perturbation-based scoring function.

5.2 Similar AP with a Lower Budget

Table 3 shows that `OLALA`-based methods considerably reduce the object budget expense. We observe at most a 50% reduction in the number of labeled objects compared to random image labeling cases in the PubLayNet experiments (7496 vs. 15980). Moreover, with this level of reduction, `OLALA`-based models manage to maintain a comparable level of accuracy. Similarly, the marginal scoring baseline is less stable and the performance is worse compared to the perturbation-based scoring method in `OLALA` settings.

In Figure 3, we visualize the model validation accuracy (line plot) and the budget expense (bar plot) for the PubLayNet dataset labeling simulations. Given the same object budget (dashed horizontal line), image-AL methods can only label 5 rounds, and the model AP is around 45 (indicated by the vertical line), significantly lower than 58.9 in `OLALA` models.

5.3 Analysis of the `OLALA` framework

In the `OLALA` framework, there are three sources of objects in the created dataset, namely, human annotations, directly used model predictions (unselected in the AL step), and unchanged model predictions (they are selected for manual check, but remain unchanged as they are accurate). `OLALA` strategically chooses objects to label and thus optimizes the overall efficiency.

	PubLayNet		HJData	
Exps*	AP	Labeled I/O	AP	Labeled I/O
[a]†	59.89	1,503 / 15,980	63.42	603 / 44,156
[c]	57.96(-1.93)	1,503 / 10,228	65.72(+2.30)	603 / 29,191
[b]	59.21	1,503 / 11,848	69.04	603 / 44,251
[d]	53.33(-5.88)	1,503 / 6,829	65.84(-3.19)	603 / 30,251
[e]	**58.90(-0.31)**	1,503 / **7,496**	**67.68(-1.36)**	603 / **28,899**

* The parameters in these experiments are slightly different from those mentioned in Table 1, and we report the details in the supplementary materials.
† The indexing is the same as Table 2.

Table 3: The final AP and number of total labeled images I and objects O when labeling the same number of images. `OLALA` maintains a similar level of AP while labeling significantly fewer objects. Similar results are observed in PRImA and abbreviated to save space.

Figure 4 shows the proportion of object sources in the three `OLALA` settings in the PublayNet Labeling experiments. The *Object Selection Scheduling* (Section 2.2) sets a high selection ratio r when training begins and r decays during training. Thus, the averaged percentage of manually labeled objects (blue line) is initially high but gradually decreases while the portion of model predictions (orange line) steadily grows in the labeling process. As the models becomes more accurate as training progresses (reflected in Figure 3), "annotators" find more accurate objects in the model-selected predictions, and include them in the dataset without changing them (green line). Though more than 50% of objects are directly from model prediction, the created datasets maintain the same high level of accuracy[4], indicated by grey bar plots in the background.

[4]The dataset accuracy is measured in AP via comparing the created version with the oracle.

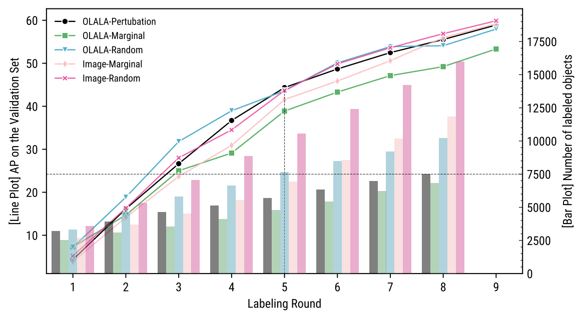


Figure 3: Model validation accuracy (line plot) and budget expenses (bar plot) at different rounds of PubLayNet labeling. `OLALA` methods (blue) require labeling fewer objects compared to image AL methods (red), while maintaining similar AP. If the same number of objects is allowed (horizontal dashed line), the image AL method stops at round 5, and the model AP is around 25% lower compared to `OLALA`.

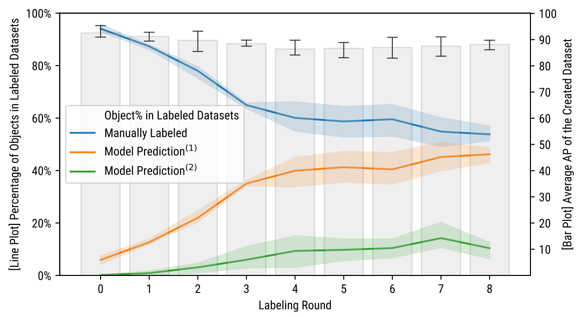


Figure 4: The created object sources (line plot) and dataset accuracy (bar plot) during the training process. The number of manually labeled objects (blue) decreases and directly used model predicted objects (orange) increases. As the model becomes more accurate, a higher portion of selected objects become accurate (green). Results shown are averaged from the three `OLALA` methods in the PubLayNet experiments.

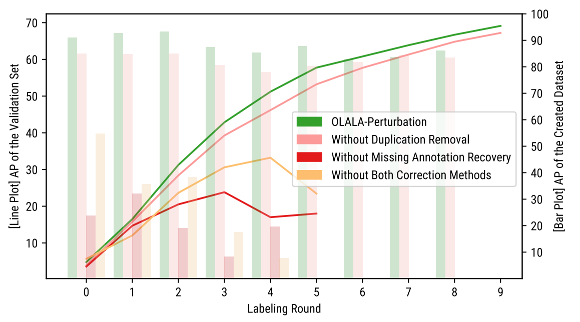


Figure 5: Influence of the prediction correction components on the model validation accuracy (line plot) and dataset accuracy (bar plot). Model performance suffers from removing the components, and dataset accuracy decreases accordingly. The *Missing Annotation Recovery* component, which corrects false negatives, is critical for model performance. Results shown are from experiments on PubLayNet.

We study how the semi-automatic prediction correction algorithm, mentioned in Section 2.3, contributes to the `OLALA` process. Shown in Figure 5, we compare the model validation AP (line plot) and accuracy of the created dataset (bar plot) with and without the *Duplication Removal* and *Missing Annotation Recovery* components in PubLayNet annotation. Without these components, models suffer from different levels of accuracy reduction compared to the `OLALA`-Perturbation baseline (green). We observe the most severe accuracy reduction when removing the missing annotation recovery components (red), indicating the necessity of extra supervision for correcting high ratios of false negatives. Interestingly, when removing both correction methods (orange), the model appears to perform better than when only discarding the missing annotation recovery component. Duplicated predictions add more instances per image for calculating the loss, thus reinforcing the signal to train the model and improve the initial performance. Unfortunately, without extra supervision, the models are trained on a dataset with many false negatives, and tend to generate fewer predictions. The error accumulates and finally both models collapse and stop improving. In both cases, the models exhaust all the training samples at round 5.

6 Conclusion

The objective of this paper is to develop rigorous methods that can increase the efficiency of extracting structured texts - required for downstream NLP applications - from social science documents. We propose the object-level active learning annotation framework, `OLALA`, for efficiently labeling document layouts. With a novel prediction correction algorithm and perturbation object scoring function, annotators only need to label a fraction of layout objects in each image. Through simulated labeling experiments on real-world data, we show that our proposed algorithms significantly improve dataset creation efficiency relative to image-level methods. Different components of `OLALA` are also carefully studied to demonstrate their validity and necessity. In summary, this work explores how to improve cooperation between human and machine intelligence, in order to unlock the structured text required for conducting modern NLP analyses at scale on historical documents.

Acknowledgements We thank the reviewers for their very helpful suggestions and feedback! This project is supported in part by NSF Grant #1823616 and funding from the Harvard Data Science Initiative and Harvard Catalyst, and we thank the computational resources from Computation Canada. We also thank the helpful discussions with Doug Downey, James Tompkin, and Ruochen Zhang.

References

Hamed H Aghdam, Abel Gonzalez-Garcia, Joost van de Weijer, and Antonio M López. 2019. Active learning for deep detection neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3672–3680.

Apostolos Antonacopoulos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. 2009. A realistic dataset for performance evaluation of document layout analysis. In *2009 10th International Conference on Document Analysis and Recognition*, pages 296–300. IEEE.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Clemens-Alexander Brust, Christoph Käding, and Joachim Denzler. 2018. Active learning for deep object detection. *arXiv preprint arXiv:1809.09875*.

Sai Vikas Desai, Akshay Chandra Lagandula, Wei Guo, Seishi Ninomiya, and Vineeth N Balasubramanian. 2019. An adaptive supervision framework for active learning in object detection. *arXiv preprint arXiv:1908.02454*.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

W Walker Hanlon and Brian Beach. 2022. Historical newspaper data: A researcher's guide and toolkit.

Elmar Haussmann, Michele Fenzi, Kashyap Chitta, Jan Ivanecky, Hanson Xu, Donna Roy, Akshita Mittel, Nicolas Koumchatzky, Clement Farabet, and Jose M Alvarez. 2020. Scalable active learning for object detection. *arXiv preprint arXiv:2004.04699*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Zhuoren Jiang, Zhe Gao, Yuguang Duan, Yangyang Kang, Changlong Sun, Qiong Zhang, and Xiaozhong Liu. 2020. Camouflaged chinese spam content detection with semi-supervised generative active learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3080–3085.

Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. 2019. Tablebank: Table benchmark for image-based table detection and recognition. *arXiv preprint arXiv:1903.01949*.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models.

Soumya Roy, Asim Unmesh, and Vinay P Namboodiri. 2018. Deep active learning for object detection. In *BMVC*, page 91.

Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. 2017. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 1162–1167. IEEE.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294.

Zejiang Shen, Kaixuan Zhang, and Melissa Dell. 2020. A large dataset of historical japanese documents with complex layouts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 548–549.

Zejiang Shen, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson, and Weining Li. 2021. Layoutparser: A unified toolkit for deep learning based document image analysis. In *International Conference on Document Analysis and Recognition*, pages 131–146. Springer.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147.

Keze Wang, Xiaopeng Yan, Dongyu Zhang, Lei Zhang, and Liang Lin. 2018. Towards human-machine cooperation: Self-supervised sample mining for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1605–1613.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. `https://github.com/facebookresearch/detectron2`.

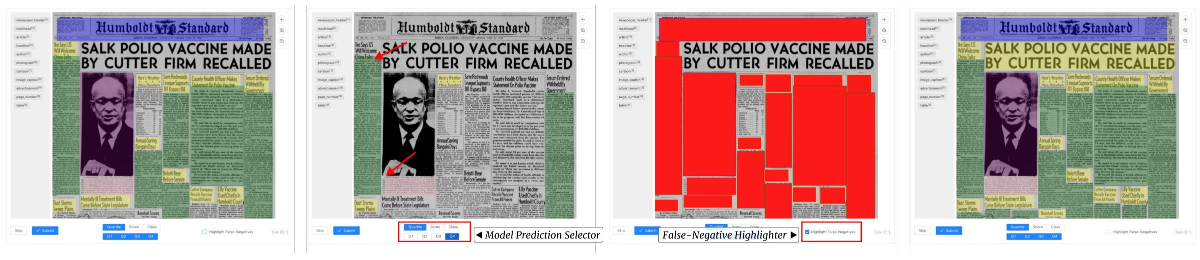
Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2020. Label Studio: Data labeling software. `https://github.com/heartexlabs/label-studio`.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698.

Angela Yao, Juergen Gall, Christian Leistner, and Luc Van Gool. 2012. Interactive object detection. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3242–3249. IEEE.

Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. 2019. Publaynet: largest dataset ever for document layout analysis. *arXiv preprint arXiv:1908.07836*.

Zongwei Zhou, Jae Shin, Lei Zhang, Suryakanth Gurudu, Michael Gotway, and Jianming Liang. 2017. Fine-tuning convolutional neural networks for biomedical image analysis: actively and incrementally. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7340–7351.



(a) Full Model Predictions (b) Selected Model Predictions (c) Highlight False-Negative Regions (d) Final Created Annotation

Figure 6: Illustration of the annotation interface with OLALA features. (a) Given an input scan, a pre-trained model generates object predictions, and they are highlighted as rectangular boxes on the original image. The color denotes the category of the given object. (b) The *Model Prediction Selector* enables hiding predictions of low object scores. In this case, objects of top 25% (the 4th Quartile, Q4) scores are presented. Two of the objects (pointed to by red arrows) have minor errors in object location predictions. Human annotators check only the displayed objects and modify inaccuracies. (c) The *False-Negative Highlighter* helps recognize mis-identified objects from the model predictions. When enabled, it converts all predicted regions to a dummy color, and regions without predictions are highlighted. Annotators can easily spot false-negatives regions and have them labeled. (d) After these steps, the full image annotation is created with less effort.

Appendix

A OLALA Implementation Details

Different from image-level labeling, annotating objects within images is fundamentally a search task: "annotators"[5] need to scan through the image and find objects matching specific criteria. The nature of object-based labeling leads to different objectives in simulated labeling experiments and real-world human annotation. In labeling simulations, the ground-truth objects are known ex-ante. The labeling agent only needs to query the oracle and choose objects that meet certain conditions. As the search space is pre-defined, the core challenge is to construct such query conditions for finding ground-truths. By contrast, when humans annotate objects, there is no ground-truth known beforehand, and the object search space is yet undefined. Their vision systems are capable of efficiently identifying correct objects within the space. Hence, the objective for human annotation is to reduce the object search space, and annotators will select valid objects within the space. To this end, as mentioned in the main paper, the OLALA framework is implemented differently for real-world human annotation (Section A.1) and simulated labeling experiments (See Section 4 in the main paper).

A.1 OLALA Annotation User Interface

To help with human annotation, we build a labeling interface incorporated with OLALA functionalities based on label-studio (Wu et al., 2020). Figure 6 shows an example of annotating newspaper layouts using this tool[6].

a Given an input scan, a pre-trained model generates object predictions $\{(b_j, c_j)\}_{j=1}^n$, which are highlighted as rectangular boxes on the original image. The color denotes the category c_j of an object. Within the outputs, duplicated object detections are precluded using *Duplication Removal*.

b A *Model Prediction Selector* is implemented for hiding objects with low scores generated by the object scoring function f. In this case, objects of top 25% (the 4th Quartile, Q4) scores are presented. Two selected objects (pointed by red arrows) have minor errors in object location predictions by missing one line or one column of text (see Section 3 "Applicability to Layout Datasets" in the main paper), while others being correct. Human annotators can focus on checking the displayed objects and only need to modify the two incorrect predictions while other accurate ones are kept untouched.

c We also develop a *False-Negative Highlighter* to help annotators find mis-identified objects from the model predictions. After enabled, it

[5]We use the general term annotator to refer to a human annotator or a simulated labeling agent.

[6]In this example, the used model has been trained on 200 hundred images. For illustration purpose, we reduce the number of objects generated by models to emphasize the false-negative selection process. But in practice, the false-negative rate is lower.

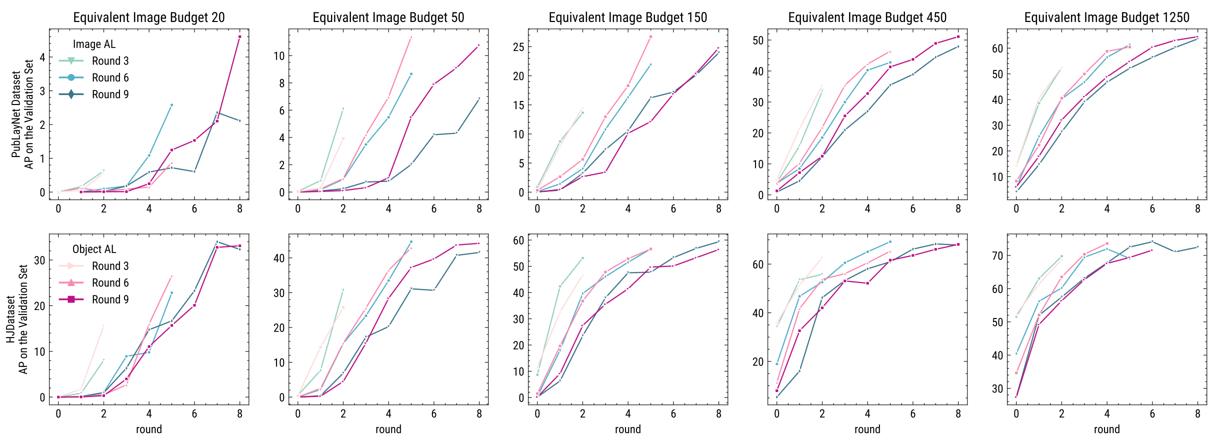


Figure 7: The model validation AP during the labeling process under different total rounds T and labeling budget m. The plots in row one and two are for experiments on the PubLayNet and HJDataset, respectively. Within each plot, image-level AL results are colored in blue while `OLALA` results are in red. To best show results at different stage of training, the ranges for the y-axis are set differently. Under the same budget, an increase in T can generally lead to better model performance. For different datasets, the optimal budget and total round settings are different. As the budget increases, image-level methods narrow the performance gap (in PubLayNet experiments) or perform better than `OLALA` methods (in HJDataset experiments).

Configuration	**Configuration A**		**Configuration B**	
Datasets	PubLayNet	HJDataset	PubLayNet	HJDataset
Labeling budget m	21,140	51,436	15,855	44,088
Equivalent image budget	2,000[1]	700	1,500	600
Total rounds T	10	8	9	9
Initial / last r	0.9/0.4	0.9/0.5	0.9/0.5	0.9/0.5

[1] To get the number of equivalent image budget, we simply divide m by the average number of objects per page for the given dataset.

Table 4: Different parameter configurations for labeling settings (1) and (2). Configuration A is used for labeling setting (1) where the same number of objects are labeled and B for labeling setting (2) where the number of labeled images is fixed.

will assign a dummy color overlay to object predictions, thus regions without predictions will be highlighted. Annotators can easily spot false-negatives regions and have them labeled. And this is the *Missing Annotation Recovery* step in the `OLALA` algorithm.

d Finally, the full image annotation will be created with significantly less effort.

Through the interface, annotators' labeling effort is saved via a reduced object search space: one only needs to check the selected model predictions and the highlighted false-negative regions.

B Additional Experiments

B.1 Different model configurations

In the main paper, we report results under two different settings, namely, (1) labeling the same number of objects and (2) labeling the same number of images. During these experiments, the model configurations for labeling settings (2) is slightly different than those in (1), and we include the details in Table 4. Labeling setting (1) is only experimented under configuration A while (2) under configuration B. For fair comparison, we complete another set of experiments for labeling setting (1) using configuration B. The results are reported in Table 5, and similar conclusion could be made based on this set of experiments.

	PubLayNet		**HJData**	
Exps	AP	Labeled I/O	AP	Labeled I/O
[a]	61.65	1,558/16,123	62.73	605/44,505
[c]	63.73(+2.07)	2,501/16,122	65.75(+3.02)	980/44,260
[b]	65.52	1,961/16,108	68.16	607/44,344
[d]	69.36(+3.83)	2,995/16,104	69.13(+0.97)	956/44,398
[e]	65.53(+0.01)	2,996/16,142	69.15(+0.99)	1,041/44,398

Table 5: The final AP and number of total labeled images I and objects O when labeling the same number of objects under model configuration B.

B.2 Analysis of labeling budget and total training rounds

We run additional labeling simulations to find the optimal configurations for the labeling budget and the total training rounds. Given the same budget, we could perform multiple rounds of labeling and re-training, with the optimal total round yet to be determined. Similarly, for a given dataset, it is important to allocate appropriate labeling budget such that the labeled samples can most effectively boost the model performance. This study could also shed light on the applicability of `OLALA` to labeling scenarios where only small labeling budget is allowed. To this end, we experiment with object budget m equivalent to labeling 20, 50, 150, 450, and 1250 images (equivalent image budget [7]) for a given dataset. For each m, we also experiment with three different total labeling rounds T of 3, 6, and 9. The model validation accuracy during the labeling process is visualized in Figure 7.

Given the same labeling budget, we find that increasing the total labeling rounds T tends to improve the model accuracy, especially for scenarios where small labeling budget is available. Under such small budget, `OLALA`-based annotation usually leads to models of higher accuracy than those from image-level AL settings. However, as labeling budget increases, the performance gap between `OLALA` and image AL models narrows. With sufficient labeling budget, image AL models even performs better than `OLALA` models in HJDataset. It reveals that `OLALA` is more helpful in the initial stage of labeling, as it exposes more images samples to the model and thus boosts the performance. For different datasets, the optimal combination of total labeling rounds and budget is different: $T = 9$ with the equivalent image budget of 450 for PubLayNet, and $T = 9$ with 50 equivalent image budget for HJDataset. Based on our observation, this is largely determined by the diversity of samples in the dataset. `OLALA` helps to explore unique object instances in the early training stage, and requires more labeling steps to achieve optimal performance boost for datasets of diverse examples like PubLayNet.

[7]Directly setting thresholds for m does not account for the variances of objects per image for different datasets.