ScanBank: A Benchmark Dataset for Figure Extraction from Scanned Electronic Theses and Dissertations

Sampanna Yashwant Kahu
Virginia Polytechnic Institute and State University
Blacksburg, VA, USA
sampanna@vt.edu

William A. Ingram
Edward A. Fox
Virginia Polytechnic Institute and State University
Blacksburg, VA, USA
{waingram,fox}@vt.edu

Jian Wu
Old Dominion University
Norfolk, VA, USA
jwu@cs.odu.edu

Abstract

We focus on electronic theses and dissertations (ETDs), aiming to improve access and expand their utility, since more than 6 million are publicly available, and they constitute an important corpus to aid research and education across disciplines. The corpus is growing as new born-digital documents are included, and since millions of older theses and dissertations have been converted to digital form to be disseminated electronically in institutional repositories. In ETDs, as with other scholarly works, figures and tables can communicate a large amount of information in a concise way. Although methods have been proposed for extracting figures and tables from born-digital PDFs, they do not work well with scanned ETDs. Considering this problem, our assessment of state-of-the-art figure extraction systems is that the reason they do not function well on scanned PDFs is that they have only been trained on born-digital documents. To address this limitation, we present ScanBank, a new dataset containing 10 thousand scanned page images, manually labeled by humans as to the presence of the 3.3 thousand figures or tables found therein. We use this dataset to train a deep neural network model based on YOLOv5 to accurately extract figures and tables from scanned ETDs. We pose and answer important research questions aimed at finding better methods for figure extraction from scanned documents. One of those concerns the value for training, of data augmentation techniques applied to born-digital documents which are used to train models better suited for figure extraction from scanned documents. To the best of our knowledge, ScanBank is the first manually annotated dataset for figure and table extraction for scanned ETDs. A YOLOv5-based model, trained on ScanBank, outperforms existing comparable open-source and freely available baseline methods by a considerable margin.

CCS Concepts

• **Information systems** → **Digital libraries and archives**; • **Computing methodologies** → *Object detection*; • **Computer systems organization** → *Neural networks.*

JCDL '21, September 27–30, 2021, Virtual

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

Keywords

Dataset, Digital Libraries, Deep Neural Networks, YOLOv5, Figure Extraction, Electronic Theses and Dissertations

ACM Reference Format:
Sampanna Yashwant Kahu, William A. Ingram, Edward A. Fox, and Jian Wu. 2021. ScanBank: A Benchmark Dataset for Figure Extraction from Scanned Electronic Theses and Dissertations. In *JCDL '21: ACM/IEEE Joint Conferences on Digital Libraries, September 27–30, 2021, Virtual.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 Introduction

Over the past decade, deep learning techniques have significantly boosted the accuracy of object detection and classification in natural images [16, 18]. Document objects like figures and tables contain important information. Their automatic identification and extraction from PDF files is key to enhancing computational access to scholarly works. This facilitates important operations such as semantic parsing, searching, and summarizing. Our research focuses on Electronic Theses and Dissertations (ETDs), aiming to improve access and expand their utility. Since more than 6 million ETDs are publicly available, they constitute an important corpus to aid research and education across disciplines. Beginning with Virginia Tech in 1997, graduate programs all over the world allow (or often mandate) electronic submission of an ETD as a requirement for graduation. University libraries often provide public access to digital libraries of ETDs. Some universities scan older theses and dissertations to provide electronic access to these older works. For instance, Virginia Tech's ETD collection dates back to the year 1903. Most ETDs before the late 1990s are scanned versions of physical copies. Our work aims at identifying and extracting figures and tables from scanned ETDs. For brevity, we use *figure extraction* to refer to the extraction of both figures and tables.

There are many challenges to accurately identifying figures in scanned ETDs. The image resolution and scanning quality may vary across the collection. OCR output is often error-ridden. Most older ETDs were typewritten. In very old documents, figures and tables may have been hand-drawn or rendered in a separate process and literally cut-and-pasted into typewritten documents. Further, since ETD collections are cross-disciplinary, the documents in them present a variety of layout styles.

In the past decade, there has been a significant amount of effort on mining scholarly big data, represented by hundreds of millions of scholarly papers [15]. Comprehensive frameworks were developed to segment scholarly papers into different levels of elements (e.g.,

GROBID [23] and CERMINE [31]). Table and figure extraction software also was developed [3] (e.g., PDFFigures [5], PDFFigures2 [4], TableBank [20], DocBank [21], and DeepFigures [28]). However, these either rely on the underlying document structure of a PDF file [3–5, 31], or exclusively cater to the analysis of born-digital documents [20, 21, 28].

One of the models that inspired our work is DeepFigures [28], which generated high-quality labels for figures extracted in scientific documents. DeepFigures was trained on data derived from arXiv and PubMed datasets with a reported average precision of 96.8%. However, it performed poorly on scanned ETDs, as we demonstrate in the experiments below.

Following are some of the contributions of our paper:

(1) We curate and release ScanBank, a new manually labeled dataset for figure extraction from scanned ETDs with 10K candidate page images labeled by humans as to the presence of the 3.3K figures or tables found therein [14].
(2) We pose and answer important research questions aimed at finding better methods for figure extraction from scanned documents.
(3) We train and evaluate the performance of the YOLOv5 model to extract figures and tables from scanned ETDs. To the best of our knowledge, no existing baseline achieves comparable performance.
(4) We propose novel data augmentation techniques to make born-digital documents look like scanned ETDs.
(5) We also release the source code used for producing the results in this paper, along with the trained models [13].

2 Related Work

In PDFFigures2 [4], Clark and Divvala proposed a new approach that analyzes the structure of individual pages by detecting chunks of body text, graphical elements, and captions – and then locates figures and tables by reasoning about the empty regions in the pages. Their results were used as a baseline in later work [28]. PDFFigures2 was designed for born-digital PDF documents since it used a rule-based approach to extract figures by leveraging the underlying document structure of the PDF document, which is not explicit for scanned PDF documents.

Siegel et al. proposed DeepFigures [28], a method for extracting figures, and tables from scholarly PDFs. Data from arXiv[1] and PubMed[2] were used for locating figures in scientific papers, which were then used for training a deep learning model, consisting of ResNet [11] in conjunction with the Overfeat architecture [27], to predict coordinates of the bounding boxes around figures.

Li et al. [20] proposed TableBank, an image-based table detection and recognition framework. Their contribution was a weakly supervised machine learning (ML) model trained on a dataset of Microsoft Word and LaTex documents crawled from the Web. The authors included documents of different languages (e.g., Chinese, English, Japanese, etc.) in TableBank, thereby making it more general. The method also modified the document source code, allowing them to generate a ground truth dataset of figures and tables, with known bounding boxes. The authors took a step further and built a recurrent neural network that converted a detected table (in an image format) into a table markup format (i.e., a table parsed into text). In other words, it converted an image of a table into a structured machine readable format; this was called table structure recognition.

Hansen et al. [9] employed an object detection model called Faster R-CNN, allowing them to achieve better region assignments for tables in a PDF document than DeepFigures. They introduced a dataset with 31,639 manually labeled PDF pages with image bounding boxes. Like DeepFigures, Faster R-CNN and TableBank were trained on born-digital documents.

Recently, Lee et al. [19] proposed the Newspaper Navigator dataset, used for extracting and analyzing visual content from 16 million historic newspaper pages from "Chronicling America." This work used a manually labeled dataset of historic newspaper pages containing labels for seven classes (headlines, photographs, illustrations, maps, comics, editorial cartoons, and advertisements). A pre-trained Faster R-CNN model was fine-tuned on this dataset to enable the extraction of the targeted visual content. The documents used by this work were archived newspapers, which have different visual structures from scholarly documents, such as ETDs.

YOLO is a popular deep learning framework, designed to detect multiple objects in an input image in a single inference pass. It is also well-known for its low space and time complexity during inference, which makes it an ideal alternative for deployment on devices where low resource consumption is vital. The initial version of YOLO (YOLOv1) was proposed by Redmon et al. in 2016 [25]. This was the first work which, instead of repurposing classifiers as object detectors, framed object detection as a regression problem. YOLOv1 detects multiple bounding boxes in a single forward pass, so it can be trained end-to-end directly for detection. Many subsequent versions of YOLO were proposed by various authors in the following years. In 2020, the fifth version of YOLO (YOLOv5) was proposed [32], which achieved the best detection performance among the versions. It has four different network sizes (small, medium, large, and extra-large), which allows users to make trade-offs between the time and space complexity. We adopt the extra-large version of YOLOv5 containing about 89 million trainable parameters.

Data augmentation [24] is a popular technique in deep learning that helps to train a model better without collecting new data. Some of the common data augmentation methods are affine transformations, random rotations, additive noise (e.g., salt-and-pepper, Gaussian), perspective transformations, and random cropping [30]. We use the popular ImgAug [3] open-source software, which provides the capability to augment not only the image but also the corresponding bounding boxes around images.

3 Research Questions

We address the following research questions (RQs) in this paper.

RQ1: How well can existing methods (e.g., DeepFigures*), extract figures from scanned ETDs?* Existing methods, like DeepFigures, have been trained and tested exclusively on born-digital documents. Since scanned and born-digital documents differ considerably in visual appearance, it is necessary to investigate how well the existing methods for figure extraction perform on scanned ETDs.

[1] https://arxiv.org/help/bulk_data
[2] https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist
[3] https://github.com/aleju/imgaug

RQ2: Can the performance of existing methods (e.g., DeepFigures) be improved by training them using an augmented version of their original data or by weight initialization? Since the visual appearance of scanned and born-digital ETDs differ significantly, we apply different data augmentation techniques on born-digital documents to make them visually look like scanned documents which are then used to train existing methods and evaluate their performance. Answers to RQ1 and RQ2 can help clarify if there is value in creating a dataset like ScanBank.

RQ3: Can figure-extraction performance for scanned ETDs be improved by training the DeepFigures or YOLOv5 model on our ScanBank dataset? The original DeepFigures model was trained using the born-digital arXiv dataset. The model in RQ2 was also trained using the born-digital arXiv dataset, albeit with augmentations. However, the source of the data in both cases is still born-digital and labels are automatically generated (Section 4.1.1). Therefore, to answer this question, we train a state-of-the-art model (YOLOv5) using the ScanBank dataset.

RQ4: Can transfer learning improve performance, training only some of the layers of the deep neural network used in DeepFigures? We freeze some of the layers of the pre-trained deep neural network in DeepFigures and re-train the model.

4 Existing Frameworks

4.1 DeepFigures

Since our work builds upon DeepFigures [28], we review the general strategy employed in that work. The overall approach of DeepFigures is to generate high-quality labels for figure extraction in a large number of scientific documents for training a deep learning model (ResNet-101 [11] + Overfeat [27]).

4.1.1 Label induction This step of the DeepFigures pipeline deals with two types of datasets. The first is the arXiv[4] dataset which can be obtained using AWS's S3 API[5] as mentioned on their bulk access website[6]. This dataset contains the LaTeX source code for each research paper in the dataset which is first used to compile the PDF which is converted to a list of page images. Then the LaTeX source code is modified to add bounding boxes around figures. This is achieved by adding a few lines of LaTeX code at the beginning of the source code. These modified LaTeX files are compiled to generate PDFs that are converted to images.

Using this method, each scholarly paper in the dataset can be transformed into two lists of images. The first does not have any bounding boxes around its figures but the second list does. Each image in these two lists represents a single page from the PDF document. Subtraction on each corresponding pair of images from these two lists in a pixel-by-pixel fashion yields a subset of the images, each of which has a bounding box around the figures. The images resulting from the subtraction contain only the difference between the subtrahend and minuend, which are the bounding boxes around the figures. The top-left and the bottom right of each of the bounding boxes is found using simple image processing operations and used as the co-ordinates of the bounding boxes.

[4]https://arxiv.org
[5]https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html
[6]https://arxiv.org/help/bulk_data

Labels are also generated for the PubMed dataset using its auxiliary data which includes image files for all graphics and XML markup indicating the locations of captions. These image files are used for multi-scale template matching on the page images of the original paper to obtain the figure locations which are then used as labels for training the DeepFigures model. The quality of a model is evaluated by manually labeling randomly sampled images and comparing against predicted labels.

4.2 YOLOv5

Our proposed approach is based on YOLOv5 [32], which offers better usability with superior performance on the MS COCO benchmark [22] compared with YoloV4. The backbone network (for object detection) of YOLOv5 implements BottleneckCSP [33]. YOLOv5 chooses PANet [34] for feature aggregation and adds an SPP [10] block after BottleneckCSP to increase the receptive field and separate out the most important features from the backbone [2]. In general, the models in the YOLO family have better performance and are more compact than models of similar or larger size.

4.2.1 Data augmentation Before passing the input images to the model, YOLOv5 uses three methods to augment the data: scaling, color space adjustment, and mosaic augmentation. The mosaic augmentation was a novel augmentation technique when YOLOv5 was developed, which works by combining four images into four tiles with random ratios [29].

4.2.2 Anchor boxes The YOLOv5 model predicts bounding boxes as deviations from a list of anchor boxes. Using K-means and a genetic algorithm, an initial set of anchor bounding boxes is learned from the training set. These anchor boxes are then used as references for learning the deviation to get the predicted bounding boxes [29].

4.2.3 Architecture Cross Stage Partial (CSP) networks [33] used in YOLOv5 have significantly lower number of trainable parameters and use fewer flops since they address the problem of duplicate gradients from larger convolutional networks [29] resulting in a faster inference time for YOLOv5.

4.3 Comparison of YOLOv5 and DeepFigures

4.3.1 Parameters vs. inference times Table 1 compares the DeepFigures and YOLOv5 models on the number of trainable parameters, the average inference time across 500 images processed on an Nvidia Tesla P100 GPU, and the year the model was proposed. The inference times in Table 1 do not include any time needed for pre-processing the image or post-processing the predictions. We measure only the time to make a single forward pass for a single pre-processed image on the GPU.

Table 1: Comparison between DeepFigures and YOLOv5.

Model	# params	Inference time	Year published
DeepFigures	45 million	33.514 ms	2018
YOLOv5 (XL)	89 million	35.048 ms	2020

The inference times are almost the same even though the number of parameters in YOLOv5 is almost double that of DeepFigures, which is because of the CSP network.

4.3.2 Architectural comparison The backbone of the DeepFigures model is ResNet-101, and the head uses the Overfeat architecture. The ResNet backbone consists of several CNN layers stacked on top of each other with skip connections between them. With these skip connections, an entire copy of the previous layer's outputs is bypassed and added to the current layer's outputs. This makes the back-propagation of gradients much easier for deep layers, and significantly reduces learning parameters and convergence time for deep neural networks.

The backbone and the head of YOLOv5 consist of a number of stacked 1D and 2D convolution, up-sampling, and BottleNeckCSP layers. BottleNeckCSP layers are BottleNeck layers that incorporate the CSP architecture. In this architecture, a part of the outputs of the base layers is split into two parts. The first part is directly linked to the end of the stage, thereby skipping the model stage. The second part goes through the model stage and is then concatenated together with the first part.

Apart from these architectural differences, YOLOv5 also performs mosaic data augmentation, which is absent in DeepFigures. Furthermore, YOLOv5 has adaptive bounding box anchors which are absent in DeepFigures. These anchors help with better prediction in case the classes to be detected have a bias for shapes.

5 Data

5.1 arXiv dataset

The arXiv dataset consists of the LaTeX source code of research papers. The authors of DeepFigures [28] used these LaTeX files to induce labels for figures and then trained their models based on these labels. We augment the data to make it visually similar to scanned data, used for training models. Our proposed augmentation techniques are elaborated in Section 6.2.

The arXiv dataset is born-digital by nature, which is still different from scanned (non-born-digital) ETDs. This is why the MIT dataset is introduced.

5.2 MIT dataset

As opposed to the arXiv dataset, PDFs in the MIT dataset are not compiled from LaTeX source but by scanning physical hard-copies of ETDs. However, because of a lack of bounding box information, we need to manually label PDFs in this dataset. The result is ScanBank, a manually labeled subset of the MIT dataset, described in Section 6.1. We use it for training and evaluation.

5.3 Characteristics of scanned PDFs

As opposed to born-digital PDFs, scanned PDFs originally existed as hard-copies and were later digitized into PDF using scanning tools such as flat-bed scanners. The process of scanning introduces certain artifacts in PDFs. For example, the content for some pages might be slightly rotated or tilted because of errors in the placement of paper in the scanner. Other kinds of noise such as salt-and-pepper noise, blurring, and perspective transformations are also possible. The content of the PDFs could have been typed using a typewriter, or even hand-written. Therefore, the overall appearance of a scanned PDF can vary significantly from a born-digital PDF. As a result, the feature distribution of the data on which DeepFigures was trained is significantly different from that of scanned PDFs. This leads to worse performance for DeepFigures for scanned ETDs, according to our experiments.

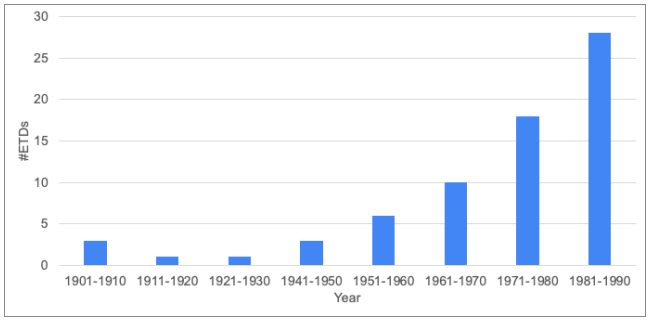

Figure 1: Year-wise distribution of ETDs sampled for the ScanBank dataset.

6 Proposed Methods

6.1 The ScanBank dataset

To overcome the limitation of this born-digital dataset, we create a new non-born-digital benchmark standard dataset for evaluation.

6.1.1 Collecting ETDs We download the PDFs and metadata of all ETDs from MIT's DSpace repository[7]. We choose this source for three reasons. First, all of these ETDs were initially submitted as paper copies and scanned into PDF. Second, the ETDs are organized by department[8] which facilitates sampling over different fields of study. Third, each ETD has associated metadata that can be used for sampling across years.

From the downloaded ETDs, we randomly sampled ETDs with the following constraints. The publication date needs to be prior to 1990. At most one ETD comes from each sub-community (doctoral, master's, and bachelor's) within each department. After accounting for empty sub-communities, our sample contained a total of 70 ETDs (see Figure 1).

6.1.2 Labeling ETDs Next, we converted each page from PDF to an image. Thus, if an ETD had 100 pages, we would have a total of 100 images. Then, we scaled the resolution to 100 dots per inch, so an image resembles a "screenshot" of a page from the PDF. We did not change the original aspect ratio of the pages when converting them to images.

A total of 10,182 images of pages were obtained across the 70 sampled ETDs. We used the VGG Image Annotator (VIA) [7, 8] to manually label these images with bounding boxes around figures. The VIA tool provides a graphical user interface for manual labeling of images. We used rectangular bounding boxes whose coordinates can be recorded using mouse click-and-drag in the VIA tool. Each bounding box contains the coordinates of the top left corner of the bounding box and its actual width and height in pixels.

The following labeling guidelines were used:

(1) Some ETDs contained source code snippets. These code snippets were not labeled.

[7] https://hdl.handle.net/1721.1/7582/
[8] https://dspace.mit.edu/

(2) "Table of Contents," "List of Figures," and "List of Tables" sections were labeled since they are visually similar to tables.
(3) Captions for both figures and tables were labeled.
(4) The bibliography was not labeled since it can neither be classified as a figure nor a table.
(5) Math equations (including matrices) were not labeled.
(6) For screen-captures (including newspaper clippings that contain figures), the individual figures within the figures were labeled. The encompassing figure was not labeled. No nested or overlapping labelling was done.

When manually labeling the ScanBank dataset, the captions for respective figures were included in their respective bounding boxes to be consistent with the choice of DeepFigures. In total, 3,375 figures were labeled across the entire dataset.

6.1.3 Validation and test splits We split the ScanBank dataset into two equal halves after shuffling it randomly. The first half will be used as the validation set for fine-tuning or choosing the best model during training. The second half will be used as the test set for evaluating the model.

6.1.4 Accessing ScanBank The ScanBank dataset is freely available online [14]. This contains a *.json* file which contains the coordinates of the 3.3K bounding boxes that represent the figures in the 10K images. Further, to limit the size of the downloaded file, this dataset only contains the URLs of the ETDs which were used to create the 10K images in this dataset. The Python source code and instructions, which are included in the dataset, can be executed to download the ETDs and convert them into the individual 10K page images.

6.2 Data augmentation

We propose to improve the performance of DeepFigures by applying data augmentation strategies to the training data. The purpose is to apply data augmentation techniques on born-digital documents with the purpose of making them resemble scanned documents.. We use image-based and LaTeX-based transformations to modify the original documents.

6.2.1 Image-based transformations We use a software library called ImgAug[9] to apply image-based transformations on each page. Each transformation below is available as a function in ImgAug.

Random affine rotation While scanning the hard-copy of a document, pages may be slightly rotated, and hence might not be perfectly aligned.. Therefore, we rotate each page of a PDF file by n degrees, where n is a float sampled from a standard uniform distribution.

Additive Gaussian noise A flatbed scanner works by reflecting the light from paper and creating an image of the paper based on the naturally reflected light. Hence, we use Additive Gaussian Noise to mimic this effect. The parameters of this noise are heuristically chosen using trial-and-error.

Salt-and-pepper noise Salt-and-pepper noise is often seen on images caused by sharp and sudden disturbances in the image signal [26]. We heuristically chose 0.1 as the probability of replacing a pixel by noise.

Gaussian blur Unlike natural (analog) images, digital images must be encoded with a specified resolution resulting in a pre-determined number of bytes, and some loss of sharpness. Therefore, we apply Gaussian blurring to smoothen the images using a Gaussian Kernel ($\sigma = 0.5$).

Linear contrast Although today's scanners are built using modern technology, they are incapable of capturing all colors of a natural object. To incorporate this scanning effect, we add Linear Contrast (parameterized by $\alpha = 1$).

Perspective transform Since scanned pages can sometimes look stretched, we implement Perspective Transform[10] in which a part of the image (formed by randomly selecting 4 points) is stretched.

Random affine rotation and perspective transform might cause geometric changes in the images which could lead to a mismatch between the locations of bounding boxes and locations of the figures. To correct this mismatch, we use a feature in ImgAug that transforms the bounding boxes according to the transformations being applied.

Figure 2 (middle) shows the image of a page from a research paper from arXiv [1] after applying the image-based data augmentation operations mentioned above.

6.2.2 LaTeX-based data augmentations Although image-based transformations change the overall appearance of a page, they do not change the inherent structure of the text within the page. The text still is well-formed and does not quite resemble the text from a PDF scanned from a typewritten document. To achieve such effects, we impose modifications in the LaTeX source code in addition to the image-based transformations.

Change font size We incorporate this effect by modifying the following command at the beginning of each document.

```
\documentclass[sigconf]{acmart}
```

We replace it with the following command:

```
\documentclass[sigconf,12pt]{acmart}
```

Modify LaTex macros We add the following LaTeX source code before the beginning of the document to change the font type of the document to look more like a typewriter font and also increases the line spacing to 1.5:

```
\renewcommand\ttdefault{cmvtt}
\renewcommand{\familydefault}{\ttdefault}
\linespread{1.5}
```

Figure 2 (right) shows the image after applying LaTeX-based transformations. The change in the font type and text layout as compared to the original page is shown in Figure 2 (left). The overall processes for obtaining the labels from these augmented images are illustrated in Figure 3; they occur after incorporating the proposed modifications.

7 Results

7.1 Experiment 1: Evaluating the pre-trained DeepFigures model using ScanBank

7.1.1 Experimental setup Siegel et al. [28] released their source code and model weights used for training DeepFigures. In this

[9] https://github.com/aleju/imgaug

[10] https://imgaug.readthedocs.io/en/latest/source/overview/geometric.html

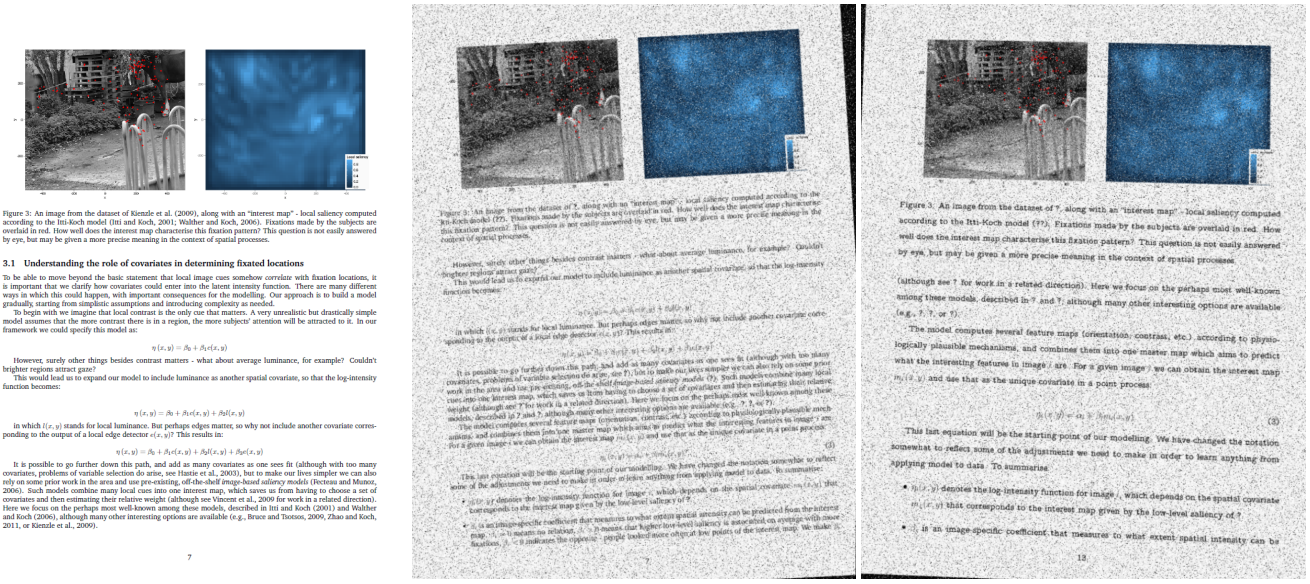

Figure 2: (a) The left image shows the original page from [1] (page number 7). (b) The middle image shows the same page after applying only image-based transformations. (c) The right image shows the page after applying both image-based and LaTeX-based transformations. Note the change in font-size and font layout.

experiment, we evaluate its performance on our proposed ScanBank dataset consisting of scanned ETDs. We run inference for DeepFigures (using the trained model weights released in [28]) on the validation and test splits of ScanBank.

Table 2: Performance of DeepFigures on 4 Datasets.

Model	Precision	Recall	F1
CS-Large	-	-	0.849
PubMed	-	-	0.806
Validation	0.461	0.491	0.475
Testing	0.439	0.445	0.442

7.1.2 Evaluation metrics The DeepFigures model outputs a set of bounding boxes for each figure it detects in the page image. We filter out the bounding boxes whose confidence scores are lower than 0.5. We match the predicted bounding boxes with the true bounding boxes to minimize the total Euclidean distance between the centers of paired bounding boxes. This is an instance of the linear assignment problem [4, 28] solved using the Hungarian algorithm [17]. Once the predicted boxes have been matched with the true boxes, we deem a predicted box as correct if its intersection over union (IOU) with the true box is greater than or equal to 0.8 (true positive), incorrect if less than 0.8 (false positive) [28]. When a ground truth is present in the image and the model fails to detect it, we deem it a false negative. Using these metrics, we calculate the precision, recall, and F1 scores. The choices of the thresholds described in Section 7.1.2 are made to be consistent with choices made in DeepFigures [28].

7.1.3 Results Table 2 shows the performance of DeepFigures evaluated on four datasets. CS-Large and PubMed datasets are born-digital scholarly documents. (Only the F1 scores were reported.) The "Validation" and "Testing" are scanned ETDs. The performance of DeepFigures with scanned ETDs is considerably lower than its performance on CS-Large and PubMed datasets, indicating that it is not suitable for accurately extracting their figures and tables.

7.2 Experiment 2: Ablation studies

7.2.1 Motivation In Section 6.2, we described the different proposed data augmentation transformations, which were chosen heuristically. To know the effectiveness of these transformations we conduct ablation studies in two parts — first for LaTeX-based transformations and then for image-based transformations.

In our implementation, each transformation in the list mentioned in Section 6.2.1 and Section 6.2.2 can easily be disabled or enabled. The total number of possible combinations for this is 2^n, where n is the number of transformations possible. Training 2^n models for a significantly large number of transformations may not always be computationally feasible. Therefore, we propose a leave-one-out ablation study to get a better idea about which transformations are actually helpful.

7.2.2 Leave-one-out ablation study We train n different models in parallel (n is the total number of possible transformations). For each model, we keep all of the hyper-parameters constant, except the list of transformations to apply. For the i-th model, we disable the i-th transformation and leave the remaining enabled. Based on the results, if the i-th model performs poorly, we can say that disabling the i-th transformation has worsened the performance and therefore, enabling that transformation contributes positively towards the model performance.

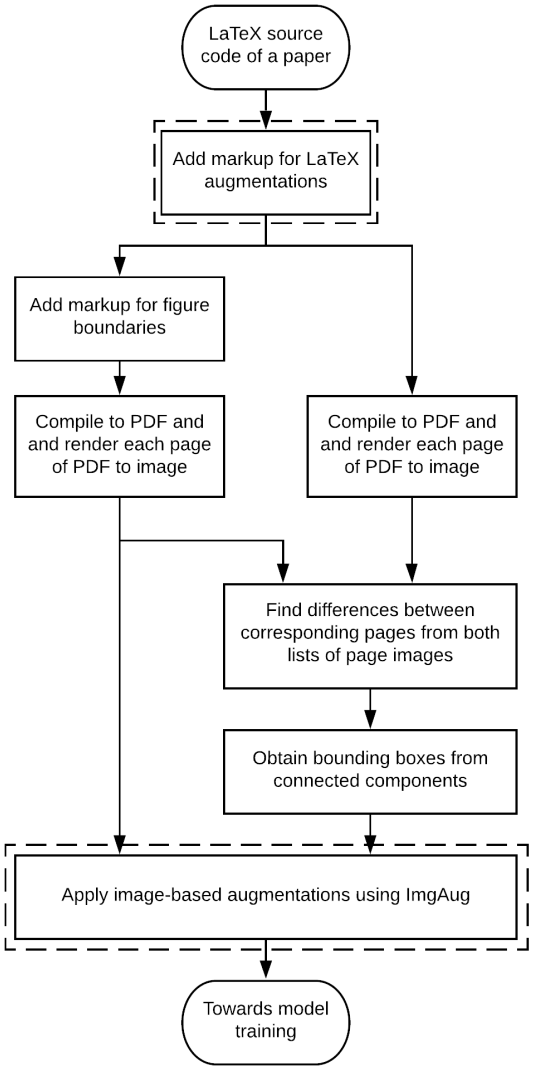

Figure 3: Our proposed pipeline to obtain training labels using data augmentation. Our proposed modifications on top of DeepFigures [28] are highlighted using dotted lines.

Although the leave-one-out ablation study gives us some idea about the performance of each transformation, it does not give the entire picture. For instance, two transformations performing well on their individual ablation studies could perform worse when used together. This can be further extended for combinations of more than two transformations. Therefore, we do not claim that the leave-one-out strategy gives us the best combination of transformations. However, it is helpful for weeding out the transformations that led to a poor performance.

7.2.3 Results We trained 11 deep learning models described in Section 7.2.2 for 24 hours. For each model, we choose the checkpoint that performed the best on the validation set of ScanBank. Then, we reported its performance on the test set (Table 3). The performance of the original pre-trained DeepFigures model was computed by directly running inference on our testing dataset.

Table 3: Performance of the original DeepFigures model on the ScanBank dataset compared to the models trained as part of our leave-one-out ablation study. Inside parentheses are transformations disabled and the training time in hours.

Model	Precision	Recall	F1
DeepFigures	0.450	0.468	0.459
Ours (All enabled, 24)	0.550	0.521	0.535
Ours (All enabled, 72)	0.556	0.498	0.526
Ours (Gaussian Noise, 24)	0.547	0.456	0.498
Ours (Gaussian Noise, 72)	0.578	0.492	0.531
Ours (Affine, 24)	0.521	0.496	0.508
Ours (Affine, 72)	0.591	0.457	0.516
Ours (Gaussian Blur, 24)	0.450	0.471	0.460
Ours (Gaussian Blur, 72)	0.557	0.542	0.550
Ours (Linear Contrast, 24)	0.559	0.527	0.542
Ours (Linear Contrast, 72)	0.567	0.488	0.524
Ours (Perspective Transform, 24)	0.526	0.517	0.521
Ours (Perspective Transform, 72)	0.431	0.586	0.497
Ours (Salt and Pepper, 24)	0.574	0.579	0.576
Ours (Salt and Pepper, 72)	0.554	0.525	0.539
Ours (Line spacing 1.5, 24)	0.454	0.642	0.532
Ours (Line spacing 1.5, 72)	0.542	0.641	0.588
Ours (Typewriter font, 24)	0.543	0.465	0.501
Ours (Typewriter font, 72)	0.624	0.490	0.549

We make the following observations from Table 3. (1) The F1-scores of almost all of our models surpass the F1-score of the original DeepFigures model, indicating that the performance of figure extraction for scanned documents can be improved using augmented data. (2) The model in which Gaussian Blur was disabled has a score of 0.460. This is close to the F1-score of the original DeepFigures model (0.459). In all of the other models, Gaussian Blur was enabled, and their performance is significantly higher than the original DeepFigures model. Apparently, Gaussian Blur is the most helpful transform.

To find out how the model performance changes with training time, we re-run this experiment by re-training the eleven models for 72 hours. Again, we observe that almost all of our models significantly out-perform the original DeepFigures model, which is consistent with the observations when models were trained for 24 hours (Table 3). However, the new experiments show contrastive results when the models are trained for 72 hours. For example, the F1-score of the model in which Gaussian Blur is disabled is no longer the model with the lowest performance. In contrast, it significantly outperforms the model when the training time is 24 hours. On the other hand, the F1-score of the model in which Additive Gaussian Noise is disabled (0.531) is higher than the F1-score

of the model in which Affine transform is disabled (0.516). The F1 decreases the most when Perspective Transform was disabled, indicating that Perspective Transform seemed the most helpful. Because of these contrasting observations, the current experiments do not prove which type of transformation is the most helpful. The significant change of F1 when the training time is 72 hours indicated the model at 24 hours may not be sufficiently trained. Training for a longer time is needed to observe the decrease of F1 for certain features before conclusively assessing their importance in data augmentation.

7.3 Experiment 3: Training YOLOv5

7.3.1 Experimental setup Experiments 1 and 2 used the DeepFigures model. However, the best F1-score we have obtained is still less than 0.588. In this experiment, we use YOLOv5, a state-of-the-art object detection framework. We choose the extra-large version of YOLOv5 with about 89 million trainable parameters (Section 2). We train on the ScanBank dataset with batch size of eight, and use eight-fold cross-validation to report its performance.

7.3.2 Results We observe that the mean F1-score of all the cross validation folds is 0.860 with a standard deviation of 0.073 (Table 4). This F1-score is significantly higher than the F1-score of the original DeepFigures model evaluated on our ScanBank standard dataset (Table 2). This F1-score is also significantly higher than the F1-score obtained in any of our previous experiments.

In the previous experiments, we used the DeepFigures model architecture which uses a combination of ResNet-101 and Overfeat. The total number of parameters in DeepFigures is about 45 million. YOLOv5's model contains about 89 million parameters. This means that the number of trainable parameters in YOLOv5 is almost double that of DeepFigures (Table 1). This result is consistent with previous observation that empirically, a model's performance increases with the model size [12].

Table 4: Performance of the original DeepFigures model on the ScanBank dataset compared with the 8-fold cross validation of YOLOv5.

Model	Precision	Recall	F1
DeepFigures	0.450	0.468	0.459
YOLOv5 (K=0)	0.749	0.869	0.804
YOLOv5 (K=1)	0.870	0.821	0.845
YOLOv5 (K=2)	0.75	0.691	0.720
YOLOv5 (K=3)	0.928	0.972	0.949
YOLOv5 (K=4)	0.886	0.937	0.911
YOLOv5 (K=5)	0.887	0.935	0.910
YOLOv5 (K=6)	0.804	0.889	0.844
YOLOv5 (K=7)	0.859	0.932	0.894
YOLOv5 (Mean)	0.842	0.881	0.860
YOLOv5 (Std. dev.)	0.066	0.090	0.073

7.4 Experiment 4: Comparison with other models

7.4.1 Experimental setup In this section, we compare our results with other models on this problem. In our work, we try to extract figures from scanned ETDs. Other similar models were not exactly designed for this task. Therefore, we compare our results with other research in this field which aims to achieve similar goals. For example, Google Cloud's commercial machine learning offering AutoML, and Microsoft Azure's Custom Vision, allow users to upload a labelled dataset and train it on the cloud without the need to select any model architecture or hyper-parameters.

We uploaded our labeled ScanBank dataset to Google Cloud and Microsoft Azure and trained our model on these platforms. For Google Cloud AutoML, we let the platform choose how to split the dataset into different classes for training. Such an option is not available for Microsoft Custom Vision, so we used 80% of the dataset for training and validation, and the rest for testing. Amazon AWS's SageMaker (AutoPilot) supports only Regression, Binary Classification, and Multiclass Classification[11]. Since it does not yet support Object Detection, we exclude it from this experiment. Another baseline we compare is [19] proposed by Lee et al. In their work, a Detectron2 model is trained to extract figures from a manually labelled dataset of historic scanned American newspapers, the visual appearance of which significantly differs from scanned ETDs. We used the pre-trained model released in [19] to run inference on our ScanBank dataset. Each predicted bounding box is labelled with one of the following seven classes: Photograph, Illustration, Map, Comics/Cartoon, Editorial Cartoon, Headline, and Advertisement. The last two classes (i.e., Headline, and Advertisement) are not figure-like. Therefore, for the purpose of this experiment, we only consider the predictions for the first five classes, and apply non-maximal suppression to eliminate duplicate predictions (confidence/objectness threshold = 0.5, IOU threshold = 0.8).

For all of these models, we used a confidence threshold of 0.5 to filter out less confident predictions. Further, to compare the predicted labels with ScanBank's labels, we used an IOU threshold of 0.8 to maintain parity with our previous experiments and with DeepFigures.

7.4.2 Results The performance of the Newspaper Navigator model [19] is significantly lower than the YOLOv5 model trained on our ScanBank dataset (Table 5). This is likely because the Newspaper Navigator model was originally trained on a different dataset, while AutoML and Custom Vision models were trained on the ScanBank dataset. Moreover, our ScanBank dataset includes the labels for tables too, which the Newspaper Navigator model was not explicitly trained to extract. This drop in performance further highlights the novelty and distinct use-case satisfied by our ScanBank dataset. The drop in performance could also be potentially explained by the different model architectures used (i.e., YOLOv5 vs. Detectron2). However, such a significant drop in performance for similar tasks is highly unlikely in two state-of-the-art models from a similar time-period.

[11] https://docs.aws.amazon.com/sagemaker/latest/dg/autopilot-automate-model-development-problem-types.html

Since Custom Vision does not disclose the architecture of its model(s), it is difficult to investigate its performance trends. However, it was surprising that even when trained on our ScanBank dataset, Custom Vision performs significantly lower than the YOLOv5 model trained in Section 7.3. Similar to Custom Vision, AutoML does not disclose the specifics of its models. However, it is mentioned that AutoML[12] uses Neural Architecture Search (NAS) to automatically find the best model architecture for the given task, which could be one of the reasons for higher performance of AutoML on ScanBank (Table 5). However, NAS usually is more computationally expensive than a regular fixed-architecture neural network.

Table 5: Performance comparison of various models with 8-fold cross validation of YOLOv5 trained in Section 7.3.

Model	Precision	Recall	F1
DeepFigures	0.450	0.468	0.459
Newspaper Navigator (LOC)	0.328	0.311	0.320
Azure Custom Vision	0.468	0.564	0.511
Google AutoML	0.908	0.878	0.893
YOLOv5 (trained on ScanBank)	0.842	0.881	0.860

8 Discussion

In Section 7.1, we evaluated the performance of the pre-trained DeepFigures model on ScanBank (Table 2) which served as the baseline for subsequent experiments. We observed that the F1-score of the pre-trained DeepFigures model was substantially lower for extracting figures from scanned scholarly documents than that of born-digital ones. This is likely due to the different visual characteristics of a scanned document and a born-digital document introduced during scanning hard copies.

Answer to RQ1: The performance of the original DeepFigures is significantly lower for figure extraction from scanned ETDs as compared to its performance for figure extraction from born-digital documents.

In Section 7.2, we use the various data augmentation techniques described in Section 6.2.1 and Section 6.2.2 to improve the performance of DeepFigures. The goal of these data augmentation techniques is to leverage the LaTeX source code of the training data to make the compiled PDFs look more like scanned PDFs. In Table 3, we observe that models trained using augmented data almost always produced a higher F1-score than the original pre-trained DeepFigures model, indicating the effectiveness of our data augmentation techniques.

Answer to RQ2: The original DeepFigures model can be improved by retraining it on augmented data using weight initialization from the pre-trained model.

In Section 7.3, we train YOLOv5 to improve the F1-score further. We initialized the weights randomly since we did not have any pre-trained set of YOLOv5 weights for a similar task. When we trained the YOLOv5 (extra-large) on the ScanBank dataset using 8-fold cross-validation, we obtained a mean F1-score of 0.86, indicating the advantages introduced by YOLOv5 and its relatively big size compared to DeepFigures (see Table 1). A similar trend was seen in improvement of image classification tasks with larger models [12].

The better performance of YOLOv5 compared with DeepFigures could also be attributed to the mosaic data augmentation in YOLOv5. Although our data augmentation techniques try to make the pages look more like scanned pages, the mosaic data augmentation has been shown to provide a stronger regularization effect [2]. Furthermore, the backbones used in these two networks are different which contribute to the difference in their performance. YOLOv5's backbone heavily borrows from CSPNet while DeepFigures uses the ResNet-101.

Answer to RQ3: The performance of the original DeepFigures model was not improved by training on manually labeled data. However, by using YOLOv5, we were able to achieve an F1-score much higher than any of the trained DeepFigures models.

All models in Section 7.2 were trained on the augmented born-digital arXiv dataset. To check whether the performance of DeepFigures can be further improved, we conducted two more experiments. In both of these experiments, we initialized the weights of the DeepFigures model with the weights released in the original DeepFigures paper [28] and trained the models on the ScanBank dataset with an 80-20 split. In the first model, we allowed all layers to train, while in the second model, we allowed only the final fully-connected layers to train (a.k.a., the Overfeat layers). In both experiments, the F1-score decreased and never surpassed the original score.

Answer to RQ4: The performance of the original DeepFigures model was not improved by using transfer learning.

9 Conclusion and Future Work

This work focuses on extracting figures from scanned ETDs. We introduce our ScanBank dataset, which, to the best of our knowledge, is the first manually annotated dataset for figure and table extraction from scanned ETDs. In our ablation study, some of our augmentation methods did not help (e.g. Salt-and-pepper and line spacing 1.5), others resulted in F1-scores even higher than the pre-trained DeepFigures model (e.g. Gaussian Blur). Table 3 shows the results of our leave-one-out ablation study. Finally, the YOLOv5 model trained on the ScanBank dataset beats all of the previous models by a significant margin.

One of the real-world applications of this type of research would be to enhance the search engines for academic publications. This was demonstrated in DeepFigures by deploying their system at scale on the Semantic Scholar website. Another potential application is for visual question answering from the extracted figures. Examples include finding the number of bar charts from the extracted figures, or answering questions from the extracted figures, such as "What is the peak value in a given plot?" Our work can be potentially used for building search interfaces of archived periodicals in digital libraries such as HathiTrust [6].

We plan to boost the performance by doubling the size of the ground truth dataset and combining heuristic methods (as in PDFFigures2) and learning based methods. An alternative and promising approach to generate scanned ETDs out of born-digital ETDs is to

[12] https://www.fast.ai/2018/07/23/auto-ml-3/

leverage the recent advances in style transfer using CycleGANs [35]. We will also investigate differential performance of the proposed model on figures and tables, respectively, given separate labels for these two content types. Figures and tables extracted from our model can be used for building figure search functionalities for large scale digital library search engines for ETDs.

Acknowledgments

Support was made in part by the Institute of Museum and Library Services for grant LG-37-19-0078-198. We thank Dr. Ammar at Allen Institute of Artificial Intelligence for helpful suggestions.

References

[1] Simon Barthelmé, Hans Trukenbrod, Ralf Engbert, and Felix Wichmann. 2012. Modelling fixation locations using spatial point processes. (2012). arXiv:1207.2370 [stat.AP] http://arxiv.org/abs/1207.23701

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. *CoRR* abs/2004.10934 (2020). arXiv:2004.10934 https://arxiv.org/abs/2004.10934

[3] Sagnik Ray Choudhury, Suppawong Tuarob, Prasenjit Mitra, Lior Rokach, Andi Kirk, Silvia Szep, Donald Pellegrino, Sue Jones, and Clyde Lee Giles. 2013. A figure search engine architecture for a chemistry digital library. In *13th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '13, Indianapolis, IN, USA, July 22 - 26, 2013*. 369–370. https://doi.org/10.1145/2467696.2467757

[4] Christopher Clark and Santosh Divvala. 2016. PDFFigures 2.0: Mining Figures from Research Papers. In *Proceedings of the 16th ACM/IEEE-CS Joint Conference on Digital Libraries* (Newark, New Jersey, USA) *(JCDL '16)*. ACM, New York, NY, USA, 143–152. https://doi.org/10.1145/2910896.2910904

[5] Christopher Andreas Clark and Santosh Kumar Divvala. 2015. Looking Beyond Text: Extracting Figures, Tables and Captions from Computer Science Papers. In *Scholarly Big Data: AI Perspectives, Challenges, and Ideas, Papers from the 2015 AAAI Workshop, Austin, Texas, USA, January, 2015*. http://aaai.org/ocs/index.php/WS/AAAIW15/paper/view/10092

[6] J. S. Downie, Sayantani Bhattacharyya, Francesca Giannetti, Eleanor Koehl, and Peter Organisciak. 2020. The HathiTrust Digital Library's potential for musicology research. *International Journal on Digital Libraries* (2020), 1–16.

[7] A. Dutta, A. Gupta, and A. Zissermann. 2016. VGG Image Annotator (VIA). http://www.robots.ox.ac.uk/ vgg/software/via/. Version: 2.0.9, Accessed: March 2, 2020.

[8] Abhishek Dutta and Andrew Zisserman. 2019. The VIA Annotation Software for Images, Audio and Video. In *Proceedings of the 27th ACM International Conference on Multimedia* (Nice, France) *(MM '19)*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3343031.3350535

[9] Matthias Hansen, André Pomp, Kemal Erki, and Tobias Meisen. 2019. Data-Driven Recognition and Extraction of PDF Document Elements. *Technologies* 7, 3 (2019), 65.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2014. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 8691)*, David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer, 346–361. https://doi.org/10.1007/978-3-319-10578-9_23

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385. arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[12] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Xu Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. 2019. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 103–112. http://papers.nips.cc/paper/8305-gpipe-efficient-training-of-giant-neural-networks-using-pipeline-parallelism

[13] Sampanna Kahu, William A. Ingram, Edward A. Fox, and Jian Wu. 2021. *SampannaKahu/ScanBank: v0.2*. https://doi.org/10.5281/zenodo.4663540

[14] Sampanna Kahu, William A. Ingram, Edward A. Fox, and Jian Wu. 2021. The ScanBank Dataset. https://doi.org/10.5281/zenodo.4663578

[15] Madian Khabsa and C. Lee Giles. 2014. The number of scholarly documents on the public web. *PLoS ONE* 9, 5 (May 2014), e93949.

[16] Nick Koudas, Raymond Li, and Ioannis Xarchakos. 2020. Video Monitoring Queries. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 1285–1296. https://doi.org/10.1109/ICDE48307.2020.00115

[17] H. W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1-2 (1955), 83–97. https://doi.org/10.1002/nav.3800020109

[18] Rayson Laroca, Evair Severo, Luiz A. Zanlorensi, Luiz S. Oliveira, Gabriel Resende Gonçalves, William Robson Schwartz, and David Menotti. 2018. A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector. In *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*. IEEE, 1–10. https://doi.org/10.1109/IJCNN.2018.8489629

[19] Benjamin Charles Germain Lee, Jaime Mears, Eileen Jakeway, Meghan Ferriter, Chris Adams, Nathan Yarasavage, Deborah Thomas, Kate Zwaard, and Daniel S. Weld. 2020. The Newspaper Navigator Dataset: Extracting and Analyzing Visual Content from 16 Million Historic Newspaper Pages in Chronicling America. *CoRR* abs/2005.01583 (2020). arXiv:2005.01583 https://arxiv.org/abs/2005.01583

[20] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. 2019. TableBank: Table Benchmark for Image-based Table Detection and Recognition. *CoRR* abs/1903.01949 (2019). arXiv:1903.01949 Retrieved October 9, 2019 from http://arxiv.org/abs/1903.01949

[21] Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. 2020. DocBank: A Benchmark Dataset for Document Layout Analysis. arXiv:2006.01038 [cs.CL]

[22] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V (Lecture Notes in Computer Science, Vol. 8693)*, David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

[23] Patrice Lopez. 2009. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. In *Proceedings of the 13th European Conference on Research and Advanced Technology for Digital Libraries* (Corfu, Greece) *(ECDL'09)*. Springer-Verlag, Berlin, Heidelberg, 473–474. https://dl.acm.org/doi/10.5555/1812799.1812875

[24] Luis Perez and Jason Wang. 2017. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *CoRR* abs/1712.04621 (2017). arXiv:1712.04621 http://arxiv.org/abs/1712.04621

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.

[26] Zhibin Ren, Xingyuan He, Haifeng Zheng, and Hongxu Wei. 2018. Spatio-Temporal Patterns of Urban Forest Basal Area under China's Rapid Urban Expansion and Greening: Implications for Urban Green Infrastructure Management. *Forests* 9, 5 (May 2018), 272. https://doi.org/10.3390/f9050272

[27] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. 2013. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *CoRR* abs/1312.6229 (2013). arXiv:1312.6229 Retrieved October 9, 2019 from http://arxiv.org/abs/1312.6229

[28] Noah Siegel, Nicholas Lourie, Russell Power, and Waleed Ammar. 2018. Extracting Scientific Figures with Distantly Supervised Neural Networks. *CoRR* abs/1804.02445 (2018). arXiv:1804.02445 Retrieved October 9, 2019 from http://arxiv.org/abs/1804.02445

[29] Jacob Solawetz. 2020. YOLOv5 New Version - Improvements And Evaluation. Retrieved March 27, 2021 from https://blog.roboflow.com/yolov5-improvements-and-evaluation/

[30] Martin A Tanner. 2012. *Tools for statistical inference: observed data and data augmentation methods*. Vol. 67. Springer Science & Business Media. https://doi.org/10.1007/978-1-4684-0510-1

[31] Dominika Tkaczyk, Paweł Szostek, Mateusz Fedoryszak, Piotr Jan Dendek, and Łukasz Bolikowski. 2015. CERMINE: Automatic Extraction of Structured Metadata from Scientific Literature. *International Journal on Document Analysis and Recognition (IJDAR)* 18, 4 (2015), 317–335. https://doi.org/10.1007/s10032-015-0249-8

[32] Ultralytics. 2020. YOLOv5. Retrieved October 10, 2020 from https://github.com/ultralytics/yolov5

[33] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. 2020. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*. IEEE, 1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203

[34] Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou, and Jiashi Feng. 2019. PANet: Few-Shot Image Semantic Segmentation With Prototype Alignment. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 9196–9205. https://doi.org/10.1109/ICCV.2019.00929

[35] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *CoRR* abs/1703.10593 (2017). arXiv:1703.10593 http://arxiv.org/abs/1703.10593