Gesellschaft für Informatik e.V. (GI) (Hrsg.): INFORMATIK 2023,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 11

Classifying figures and illustrations in electronics datasheets: A comparative evaluation of recent computer vision models on a custom collection of 4000 technical documents

Lymperis Perakis[1], Julian Balling[2], Frank Binder[2], Gerhard Heyer[2], Franz Kreupl[3]

Abstract: We report findings from a comparative evaluation of several recent object detection models applied to a domain-specific use case in technical document analysis and graphics recognition. More specifically, we apply models from the EfficientDet and YOLO model families to detect and classify figures in electronics datasheets according to a custom classification scheme. We identify YOLOv7-D6 as the most accurate model in our study and show that it can successfully solve this task. We highlight an iterative approach to figure annotation in document page images for creating a comprehensive and balanced custom dataset for our use case. In our experiments, the object detection models show impressive performance levels on par with state-of-the-art results from the literature and related studies.

Keywords: Computer Vision; Object Detection; Document Analysis; Graphics Recognition; Electronic Design Automation; Machine Learning

1 Introduction

The design and development process of electronics components, for example, for automotive or IoT markets, is complex and time-consuming. The market need for embedded software and hardware engineers is continuously growing and cannot be covered by the number of professionals in the market. Using intelligent software algorithms and AI can automate some development tasks, enabling a broader spectrum of companies, engineers, and technicians to contribute to the development of embedded systems. In this context, many challenges must be addressed to better assist engineers and to automate certain process steps in hardware and software development.

The KIRESys project aims to contribute to the automation in hardware development towards an automatic generation of PCB layouts. A prerequisite for such automatic generation is the availability of machine-processable knowledge and information on electronic components. However, to date such knowledge is primarily represented texts, tables, and figures in PDF

[1] CELUS GmbH, Ridlerstraße 57, 80339 Munich, Germany lymperis.perakis@celus.io
[2] Institute for Applied Informatics (InfAI) e.V. at Leipzig University, Goerdelerring 9, 04109 Leipzig, Germany {balling,binder,heyer}@infai.org
[3] Department of Hybrid Electronic Systems, TUM School of Computation, Information and Technology, Technical University of Munich, Arcisstraße 21, 80333 München, Germany franz.kreupl@tum.de

documents, i.e., datasheets and reference designs, intended to be consumed by humans. This paper targets one particular challenge in information extraction from these richly structured documents: Extracting and classifying illustrations and figures from electronic datasheets and reference designs.[4]

This paper reports the results of our study on the following questions: How well can current machine learning approaches recognize, extract, and classify illustrations and figures from electronic component datasheets. Which results can currently be achieved with different state-of-the-art and efficient object detection models? To answer those questions, we developed and curated a custom dataset from datasheets from the design automation domain and compared the performance of several state-of-the-art object detection models from computer vision when applied to detecting and classifying figures and illustrations in these documents.

2 Background and related work

2.1 Information extraction from technical documents

Information extraction from documents is a long-standing research area with a multitude of subfields, as documented, for instance, by the International Association for Pattern Recognition (IAPR)'s established series of conferences, workshops, and journals in the field.[5] Information extraction from non-textual contents in document images had started to thrive as early as the 1980s, driven by industry needs for establishing computer-assisted design processes (CAD) and geographical information systems (GIS), among others [LR14]. An early example of aligning visual and textual information for product and component data, such as interpreting electronic diagrams and engineering drawings for conversion into operational CAD data, was presented in 1997 by Boeing [BBK98], as reported in [LO14]. It is also noted there, that as of 10 years ago (and still today) no generalized and comprehensive solution has been found, but that many approaches focus on information spotting and partial interpretation [LO14].

However, in recent years, significant developments in machine learning and computer vision, as well as the availability of large curated datasets for training deep learning models have brought new potential into the field of information extraction from technical documents. For example, state-of-the-art computer vision models, such as Mask R-CNN [He20], can be successfully leveraged for document layout analysis, which also entails the detection

[4] *Reference designs* are manufacturer recommendations for the correct and specification-compliant implementation of electronic components in an overall system. They are usually available as PDF documents.

[5] Consider, for example, the International Conference on Document Analysis and Recognition (ICDAR), the International Journal on Document Analysis and Recognition (IJDAR), the International Workshop on Document Analyis Systems (DAS), the International Workshop on Graphics Recognition (GREC), some of which host serial competitions, such as the *Robust Reading Competition* linking the document analysis and computer vision communities [Ka13; Ya17], cf. `https://rrc.cvc.uab.es/`.

of figures and illustrations on document page images [Sh21]. Such improvements are fueled by the availability of large public training data sets as PubLayNet [ZTY19] and DocLayNet[Pf22]. In addition, recent systems for information extraction from documents "born digital"[6] successfully integrate data from multiple modalities into coherent knowledge bases by incorporating textual, structural, tabular and visual cues [Wu18]. One common aspect of most comprehensive systems and related approaches, as seen from 1998 [BBK98] through 2018 [Wu18] and beyond [OSP21], has been the effective integration of user feedback [Bi22] and the design for efficient human-machine-cooperation [Op22]. To achieve that, the automatic information extraction steps have to be performed accurately and fast. For that reason, closely related studies, such as recent work on figure extraction from electronics datasheets by Chen et al. [Ch21], have relied and improved upon (earlier) state-of-the-art object detection models from computer vision.

2.2 Object detection with EfficientDet and the YOLO model families

Object detection has seen an increasing interest over the past years, mainly due to the general advance of technology in computer vision with the help of deep learning. While image classification refers to categorizing the whole image into classes, object detection involves classification and localization tasks where multiple objects may be present in one image. More precisely, object recognition discerns instances of objects of a specific class in an image and then highlights them, usually delimiting them with bounding boxes [SSD19]. Historically we can divide the progress of object detection into two periods: the "traditional object detection period (before 2014)" and the "deep learning-based detection period (after 2014)" [Zo23].

Modern deep learning object detection models can be divided into two main types: one-stage and two-stage detectors. Two-stage detectors have an extra step that is called region proposal. One-stage object detection models skip the region proposal and run detection directly over a dense sampling of locations. These models usually have faster inference (possibly at the cost of accuracy). They are more suitable for real-time applications, and sample models include YOLO [Re16], SSD [Li16], RetinaNet [Li17b], and EfficientDet [TPL20]. Two-stage methods prioritize detection accuracy, and sample models include FPN [Li17a], Faster R-CNN [Re17], and Mask R-CNN [He20].

Two families of object detectors were used for the present study: EfficientDet and YOLO. EfficientDet [TPL20] is a recent object detection family that is based on EfficientNet [TL19]

[6] For a definition of "documents born digital" versus printed media and scanned documents, we refer to [HL14]. Documents born digital do not require the application of optical character recognition and can be processed with specific software libraries, such as *Pdfminer.six* or *pdfplumber*. However, extracting content in a meaningful and robustly automated way still poses challenges, as the PDF format by design is not intended to be machine-readable but rather encodes a visual document representation for human consumption.

as a backbone[7], and adds a new neck layer to extend the functionality of the *Path Aggregation Network* (PAN). Similarly to EfficientNet, it applies compound scaling to optimize both accuracy and efficiency.

YOLO ("You only look once") is a series of prevalent one-stage object detection model families that has seen several iterations with industry-leading performance when considering speed and accuracy in object detection. The initial release was based on its creator's *Darknet* framework [Re13; Re16], followed by to iterative improvements, YOLOv2[RF17] and YOLOv3 [RF18]. The YOLOv4 architecture was then proposed by the maintainer of the Darknet framework at that time [BWL20]. Its backbone utilizes a variant of the *Cross Stage Partial* (CSP) network [Wa20]. It also included techniques, such as mosaic data augmentation and hyper-parameter optimization using genetic algorithms. This made YOLOv4 the state-of-art detector based on its speed and accuracy compared to available alternative detectors.

The YOLO versions used in this study are YOLOv5 [Jo22], YOLOv6 [Li22], and YOLOv7 [WBL22]. An overview of the distinct EfficientDet and YOLO architectural choices can be found in Table 1.

Models	**Backbone**	**Neck**	**Head**	**Loss Function**	**Anchor**	**Augm.**
EfficientDet	EfficentNet	Bi-FPN	Coupled	Focal	Yes	No
YOLOv5	CSPDarknet53	PAN	YOLOv3	BCE with Logits Loss	Yes	Yes
YOLOv6	RepVGG or CSPRepStack	Rep-PAN	Efficient Decoupled	VFL + IoU Loss	No	Yes
YOLOv7	E-ELAN	PAN	Lead + Auxiliary	VFL + DFL	Yes	Yes

Tab. 1: Overview of the architectures of EfficientDet and different YOLO versions. The models are compared based on their backbone, neck, head, loss function, anchor method choices, and whether they use augmentation in their implementation. This table was developed based on the following sources: [Li22; TPL20; Ul21; WBL22].

YOLOv5 is a model developed and published by Ultralytics shortly after YOLOv4 was introduced. The implementation of the network changed from Darknet to Pytorch. Similarly to YOLOv4, the model's improvements include data augmentation and hyper-parameter optimization. Since its initial release, Ultralytics have continued to modify YOLOv5's architecture to further improve exportability and speed [Ul21]. Chronologically, YOLOv7 was published next [WBL22], which was developed in cooperation with the creator of YOLOv4. YOLOv7 achieved state-of-the-art for real-time object detectors. Compared to

[7] The basic architectural pattern of one-stage object detectors comprises a *backbone* network for feature extraction from input images, a *neck* for multi-resolution feature aggregation, and a *head* for predicting the objects with their bounding boxes and classes.

YOLOv5 on a model with a similar *mean Average Precision* (mAP), YOLOv7 is 120% faster on inference time in a batch size of 1. Shortly after, YOLOv6 was released [Li22]. The authors propose two scaled re-parameterizable backbones and necks to serve efficient and hardware-friendly models of different sizes.

3 Methods

3.1 Data collection, preprocessing, and pre-labeling

Supervised learning techniques require labeled data to train the models. For our specific use case we collected and developed a custom dataset. We manually downloaded over 4000 electronic component datasheets from the website of a large international electronic components distributor. Only documents from the *Interface ICs* category and its subcategories were considered, since that already yielded a fairly large collection of born-digital PDF documents with a sufficient number and variability of figures and illustrations.

Since object detection models need a visual image as input, we converted the PDFs to document page images in PNG format. We then used LayoutParser [Sh21] with an existing object detection model to generate pre-labeled bounding boxes for figures on the document pages, and to remove all page images without figures from the dataset.[8]

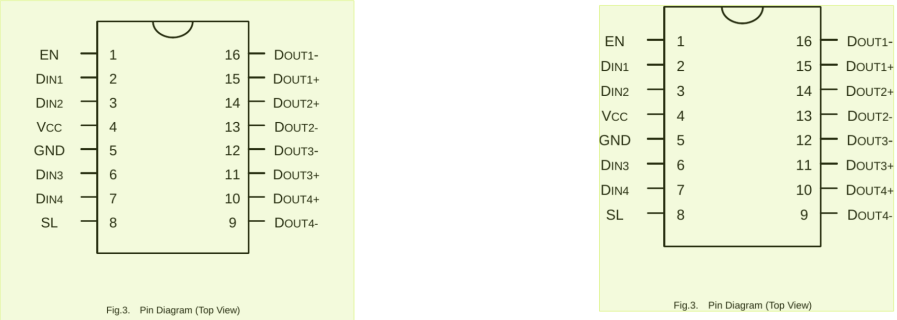
3.2 Data labeling and bounding box optimization

Given our use case, the following classes of figures in electronics datasheets were identified as relevant by CELUS' electronics engineering experts: *Package General*, *Schematic*, *Block Diagram*, *Timing*, *Pin Assignment*, *Memory*, *Plot*, *Package Drawing*, *Footprint*, *Other*. After manually labeling more than 1.000 figures on more than 600 pages, we outsourced the remaining labeling workload to a data annotation and labeling service provider, for which we composed and piloted a comprehensive labeling guide. The resulting bounding box labels were optimized using a technique inspired by [Ch21] to improve the precision and consistency of the ground truth annotations. We implemented their *Bounding Box Refinement* method to shrink and fit the bounding boxes to their respective visual objects. Figure 1 shows an example of the technique used on our dataset.

3.3 Model training with iterative pre-labeling of data piles

To prepare for model training and evaluation, we randomly split our labeled dataset into training, validation, and test subsets of 80%, 10%, and 10% respectively. The test set

[8] From LayoutParser's model zoo we used an available Mask R-CNN model, which had been trained on the PubLayNet dataset [ZTY19] and which could identify (but not yet classify) figures in our dataset.



(a) (b)

Fig. 1: Comparison of the bounding box before (a) and after (b) applying the *Bounding Box Refinement* on a *Pin Assignment* label. The excess white space around the object is removed. Note that we include figure captions inside our labels whenever possible.

underwent extended quality assurance, to ensure that the evaluation mirrors real-life results. Finally, we exported and transformed the respective data to the different data formats required by the various models.

We then trained our models using the following Python libraries: For *EfficientDet*, we used Ross Wightman's PyTorch implementation [Wi], whose performance on the COCO dataset is on par with the official results published in [TPL20]. We trained 4 model variants for EfficientDet.[9] For *YOLOv5*, we used version 6.0 from the official repository [Ul21]. Here we trained 6 models. As an ablation study, we trained the YOLOv5m model a second time without augmentations to observe their effect on the model's performance. For *YOLOv6*, we trained 5 models using the official repository [Me]. For *YOLOv7*, we trained 4 models using the official repository of the authors [Ki].

Table 2 lists the models and their characteristics from literature. For all model trainings, we used the default parameters and hyperparameters. All models were trained for 100 epochs with maximum batch sizes ranging from 4 for larger models to 64 for smaller ones. We utilized the default weights of the models, gained through pre-training on the COCO dataset, as provided by the authors' libraries. Note that the implementation of EfficientDet does not include any augmentations, contrary to the YOLO families. We expect that the augmentations will improve the YOLO models' performance without sacrificing speed. Performance of the models was evaluated on the COCO dataset.

In order to ensure a sufficient number of training items per class, we iterated over the steps from pre-labeling to model training as follows: As noted above, we first used Mask-RCNN, trained on PubLayNet, to filter out the pages without figures and to pre-annotate a first pile

[9] The plan was to train the larger models as well, but our cluster could not provide the required amount of RAM.

of data. We then had the figures labeled for this first pile, and trained the EfficientDet model with those labels. Thus, we then had a model that could both detect and classify figures in our datasheets. This allowed us to estimate the distribution of figure classes, which we used for the subsequent iteration to *roughly balance* the dataset. So, we used the predictions of EfficientDet for pre-labeling and filtering pages when creating the second pile. We then had the second pile labeled, and trained YOLOv5 with it, which we used for pre-labeling and filtering of the third pile, and so on. After training the various models we evaluated their performance on the held-out test set. Results are reported in section 4.2.

Model	Image Size	mAP %	mAP^{50} %	Speed V100 b1 (ms)	Speed V100 b32 (ms)	Params (M)	FLOPS (G)
EfficientDet-D0	**512**	34.2	53.0	-	-	3.88	**2.5**
EfficientDet-D1	640	39.4	59.1	-	-	6.62	6.1
EfficientDet-D2	768	43.4	62.7	-	-	8.1	11
EfficientDet-D3	896	47.1	65.9	-	-	12.0	25
YOLOv5n	640	28.0	45.7	6.3	**0.6**	**1.9**	4.5
YOLOv5s	640	37.4	56.8	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	12.1	4.8	86.7	205.7
YOLOv5l6	1280	53.7	71.3	15.8	10.5	76.8	445.6
YOLOv7-tiny	640	30.8	47.3	-	-	6.2	5.8
YOLOv7	640	51.4	69.7	**6.2**	2.8	36.9	104.7
YOLOv7-X	640	53.1	71.2	8.8	4.3	71.3	189.9
YOLOv7-D6	1280	**56.6**	**74**	22.7	15.0	154.7	806.8
YOLOv6-N	640	36.3	51.2	-	-	4.3	11.1
YOLOv6-T	640	41.1	56.6	-	-	15.0	36.7
YOLOv6-S	640	43.8	60.4	-	-	17.2	44.2
YOLOv6-M	640	49.5	66.8	-	-	34.3	82.2
YOLOv6-L	640	52.5	70.0	-	-	58.5	144.0

Tab. 2: Characteristics of the models used in this study as reported in [Ki; Li22; Me; TPL20; Ul21; WBL22; Wi] prior to conducting our own experiments (cf. Table 4). The primary challenge metric on the COCO dataset is mAP calculated at IoU=.50:.05:.95. Speed metrics for YOLOv5 and YOLOv7 are reported for batch sizes of 1 and 32 in ms on NVIDIA Tesla V100 GPUs. As other models were tested on different setups, their speed metrics are omitted here, but findings from our own experiments are listed in Table 4. The number of model parameters is given in millions. FLOPS are largely determined by input resolution, i.e. 640x640 or 1280x1280.

4 Results and discussion

4.1 Data set

For our use case in supporting electronics design automation with information extraction from datasheets and reference designs, we built a custom dataset of 17019 page images with figures from over 4000 born-digital PDF documents of electronic component datasheets. This page image dataset contains 28093 high-quality labels for a custom set of 10 categories of figures. The diverse dataset contains labeled figures for electronic components from 26 subcategories of *Interface ICs*. We used datasheets from over 66 manufacturers with more than 2300 components to achieve that. Around 3% of the pages in our dataset include no figures. As reported in Table 3, the dataset is fairly balanced.[10]

	Number of Figures				Number of Pages
Figure Type	All	Train	Val	Test	All
Block Diagram	2941	2347	296	298	2581
Footprint	1306	1015	143	148	1220
Memory	2796	2243	282	271	1860
Other	2843	2357	273	213	2190
Package Drawing	3488	2818	340	330	3207
Package General	1749	1406	161	182	1458
Pin Assignment	2789	2286	252	251	2289
Plot	3436	2901	255	280	1038
Schematic	3017	2432	281	304	2093
Timing	3728	2945	394	389	2469
Total	28093	22750	2677	2666	-

Tab. 3: Overview of the number of figures in our self-created electronic components datasheet collection. We display the number of samples in each figure category and the number of pages containing figures from each category.

4.2 Model performance scores after fine-tuning

Using our custom dataset, we fine-tuned and comparatively evaluated object detection models of various sizes from four recent model families: *EfficientDet*, *YOLOv5*, *YOLOv7*, and *YOLOv6*. We ran all validation experiments on the same hardware setup using an NVIDIA GTX 1080 Ti GPU, with 11GB RAM. Table 4 lists the inference performance scores of the fine-tuned models regarding their *mAP*, *precision*, *recall*, and *speed*.

[10] The categories *Footprint* and *Package General* occur only limited times in our datasheet collection. Fortunately, they appear very homogeneous between datasheets, which will still allow to successfully train the models.

Model	mAP %	mAP^{50} %	P^{50} %	R^{50} %	Speed b1 (ms)	Speed b16 (ms)	Speed b32 (ms)
EfficientDet-D0	82.0	88.4	-	-	**23.0**	**10.9**	**10.9**
EfficientDet-D1	83.5	88.7	-	-	35.0	21.6	21.3
EfficientDet-D2	84.5	89.6	-	-	49.0	33.8	33.3
EfficientDet-D3	**85.6**	**90.4**	-	-	84.0	64.18	-
YOLOv5n	83.37	91.12	85.69	86.46	**6.6**	**2.3**	**2.2**
YOLOv5s	85.62	90.73	85.5	88.64	7.7	4.0	4.2
YOLOv5m	87.22	91.51	85.58	88.65	13.0	8.3	8.0
YOLOv5m*	80.09	86.38	84.36	82.04	14.0	8.8	8.4
YOLOv5l	87.88	**91.87**	85.65	**88.97**	19.8	13.5	13.4
YOLOv5x	87.93	91.32	**87.1**	87.52	34.6	23.2	22.9
YOLOv5l6	**88.37**	91.75	86.46	88.26	62.9	52.8	-
YOLOv7-tiny	85.68	91.58	85.64	87.84	**6.9**	**4.7**	**4.4**
YOLOv7	87.75	91.98	87.32	88.38	19.8	14.9	15.7
YOLOv7-X	87.75	91.87	85.34	88.54	31.0	24.4	26.6
YOLOv7-D6	**88.75**	**93.08**	**87.64**	**89.18**	101.1	85.5	-
YOLOv6-N	85.6	90.4	-	-	7.1	**2.4**	**2.3**
YOLOv6-T	86.7	90.7	-	-	**7.1**	4.3	3.4
YOLOv6-S	86.3	90.3	-	-	8.1	5.4	4.6
YOLOv6-M	**87.3**	**90.8**	-	-	12.9	10.2	9.3
YOLOv6-L	87.3	90.6	-	-	19.8	14.5	14.4

Tab. 4: Performance comparison after finetuning: Overall mean Average Precision (mAP); mAP, precision and recall at an IoU=0.5; inference speed with batch sizes of 1, 16, and 32. EfficientDet and YOLOv6 libraries did not provide recall and precision metrics. *YOLOv5m** is the same model as the YOLOv5m without any augmentation during training. Best scores in each family are shown in bold.

4.3 Comparison of model performances

As expected from the literature, the *EfficientDet* models fall behind in performance and speed compared to the YOLO variants. Additionally, the library used for training EfficientDet posed problematic RAM requirements,it was impossible to complete the training for models larger than EfficientDet-D3. In general, we observe that the bigger the model the better its performance on all metrics, except for inference speed, which was slower for larger models.

The *YOLOv5* family' larger models showed the best trade-off between mAP and speed, as seen in Figure 2. Regarding the effect of augmentations, we find that the model fine-tuned without any augmentation (YOLOv5m*) delivers the worst performance among all experiments. Specifically, its mAP is 7% worse than the equivalent YOLOv5m model. This underlines the importance of using augmentations during fine-tuning. Figure 2 also shows that the mAP stagnates for those large models of YOLOv5 and YOLOv7 that still use an

image size of 640. YOLOv5l6 and YOLOv7-D6 use a larger input image size of 1280 and show a steeper improvement compared to their smaller counterparts.

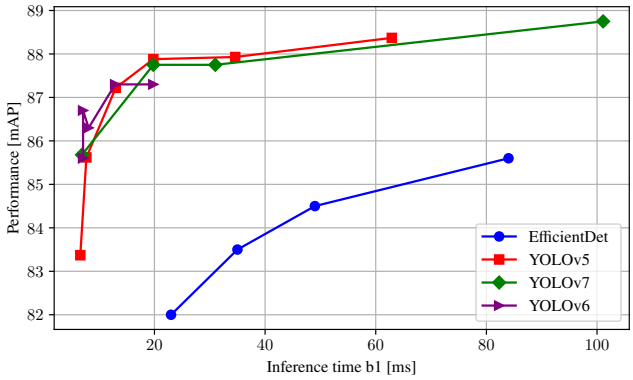


Fig. 2: Performance (mAP) of the trained models regarding their inference time in ms with a batch size of 1.

YOLOv7-D6 of the *YOLOv7* family has the best accuracy metrics of all models in our study, but it is also the slowest. We will analyze it further in Section 4.4. The smaller models of the newest *YOLOv6* family have the best trade-off between mAP and inference time. Specifically, the YOLOv6-T achieves a mAP of 0.87 with a speed of almost 300fps when using a batch size of 32. Its bigger siblings, YOLOv6-M and YOLOv6-L, are not improving much compared to equivalent ones from the other YOLO families. This may be because the larger models have different architectural choices, as listed in table 1. Another essential aspect is that the YOLOv6 models were faster to train and would converge sooner, thus potentially saving computing cost.

4.4 Best model results and related SOTA

The best model in terms of accuracy in our study is YOLOv7-D6. In Table 5, we can see its performance across the different classes. Notice that the recall is relatively high in all classes except the *Other*. Hence, the model can identify most figures in the dataset. The precision is also relatively high, showing that the model primarily identifies relevant objects.

The confusion matrix in figure 3 shows that some classes are interchanged more than others. Specifically, as expected, the model confuses the categories *Block Diagram* and *Schematics*. They have many similarities, and even electronic experts have difficulties categorizing some examples in our dataset. Another occasion that the confusion matrix shows improvement potential for our dataset is the *Timing* vs. *Plot*. Timing is a specific category of plots, and probably not perfectly consistent across our dataset.

Figure Type	mAP	mAP^{50}	P^{50}	R^{50}
all	0.888	0.931	0.876	0.892
Block Diagram	0.883	0.904	0.822	0.873
Footprint	0.878	0.913	0.831	0.918
Memory	0.826	0.891	0.8	0.886
Other	0.802	0.836	0.794	0.715
Package Drawing	0.88	0.977	0.951	0.939
Package General	0.938	0.977	0.94	0.951
Pin Assignment	0.929	0.98	0.937	0.96
Plot	0.952	0.971	0.964	0.909
Schematic	0.872	0.916	0.858	0.851
Timing	0.915	0.944	0.867	0.916

Tab. 5: Metrics over classes for our best model YOLOv7-D6. The table gives the values of mAP as well as mAP, precision, and recall at an IoU=0.5 over each class.

Although comparing experimental results obtained on different datasets is debatable, we will try to analyze the similarities and differences to the study by Chen et al. [Ch21]. They use the following figure types: *Schematic Diagram*, *Package Diagram*, *Timing Diagram*, *Footprint Diagram* and *Characteristic Diagram* and also include the categories *Header*, *Footer*, and *Table* in their data set. When considering only the categories *Block Diagram*, *Schematic*, *Plot*, *Pin Assignment*, *Package Drawing*, and *Timing*, which appear to be very similar between our study and theirs, the performance scores of our model are: $mAP^{50} = 0.949$, $P^{50} = 0.9$, and $R^{50} = 0.908$ (cf. table 5). Their model has the following scores: $mAP^{50} = 0.907$, $P^{50} = 0.933$, and $R^{50} = 0.903$. Hence, it appears that the model in our experiments outperforms theirs regarding mAP and recall, but falls short in precision. We assume that, since we have defined more specific and mutually similar categories, e.g., *Block Diagram*, and *Schematic*, the precision of our model regresses as it confuses these categories, while Chen et al. [Ch21] group them into one figure type, which avoids confusion.

4.5 Results discussion and potential improvements

To gain further improvements, our approach could be improved in several ways. Additional rigorous data QA with domain experts and data scientists, possibly guided by confusion matrices such as in figure 3, would most certainly identify some remaining labeling inconsistencies, and could further improve data quality and model performance. During preprocessing, we used a pretrained model to filter out document pages without any figures. False negatives from that step are missing in our dataset. To determine the impact of this method, we should evaluate our models in real-world scenarios where all the pages of the datasheets are assessed [AF18]. Using layout analysis models trained on DocLayNet [Pf22] (instead of the earlier PubLayNet [ZTY19]) during prelabeling could also have a positive effect on the results. Furthermore, we could extend our dataset to contain more images from

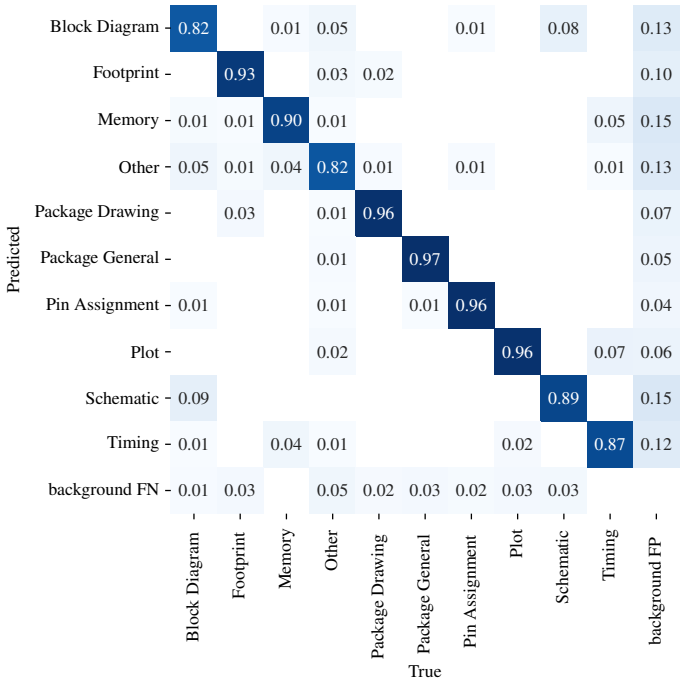


Fig. 3: Confusion matrix of the best model (YOLOv7-D6). The model's predictions are on the vertical axis, while the ground truth labels are aligned horizontally. For instance, 97% of the *Package General* were correctly classified. The *background FP* are the false positives, and the *background FN* are figures that the model failed to recognize.

an even broader range of datasheets. Lastly, we can define further relevant categories, such as *Decision tree*, that our models would have to identify.

Albeit their impressive performance, there are areas where the models can be improved. The hyperparameters, model architectures, and augmentation techniques we utilized are mainly optimized for the COCO dataset. We could adapt those to our dataset that has very different characteristics compared to COCO. To adjust those parameters automatically, *hyperparameter evolution* could be used [Ul]. However, that would require many more model training runs and considerably increase computing costs. We could also attempt to optimize the feature extraction process to more closely adapt it to our dataset of document page images (instead of the COCO data that was used for pretraining). Chen et al. [Ch21] achieve significant performance improvements through such adaptations. Finally, as object detection models continue to evolve, it would be interesting to update our research with the recent YOLOv8 family that was published after we finished our experiments [Ul23].

5 Conclusion

With this study on extracting and classifying figures from electronics datasheets we show that recent object detection models from the computer vision domain can be successfully leveraged for specific tasks in document analysis and graphics recognition. We add to prior studies by considering newer generations of the YOLO model families, and compare against the also established EfficientDet. We highlight an iterative approach to outsourced data labeling, as well as the importance of using augmentations during model training. Our Experiments on our comprehensive custom dataset of page images from technical documents show impressive performance levels that are on par with SOTA results from the literature and related studies.

Acknowledgements

This research was funded by the German Federal Ministry of Education and Research (BMBF) under grants no. 01IS20091A, 01IS20091B (Project KIRESys). Computations for this work were done (in part) using resources of the Leipzig University Computing Center.

References

[AF18] Auer, F.; Felderer, M.: Shifting Quality Assurance of Machine Learning Algorithms to Live Systems. In (Tichy, M.; Bodden, E.; Kuhrmann, M.; Wagner, S.; Steghöfer, J.-P., eds.): Software Engineering und Software Management 2018. Gesellschaft für Informatik, Bonn, pp. 211–212, 2018, URL: https://dl.gi.de/handle/20.500.12116/21162.

[BBK98] Baum, L. S.; Boose, J. H.; Kelley, R. J.: Graphics recognition for a large-scale airplane information system. In: Graphics Recognition Algorithms and Systems. GREC 1997. Springer Berlin Heidelberg, pp. 291–301, 1998.

[Bi22] Binder, F.; Diels, J.; Balling, J.; Albrecht, O.; Sachunsky, R.; Philipp, J. N.; Scheurer, Y.; Münsch, M.; Otto, M.; Niekler, A.; Heyer, G.; Thorun, C.: Putting Users in the Loop: How User Research Can Guide AI Development for a Consumer-Oriented Self-service Portal. In (Rauterberg, M., ed.): Culture and Computing. HCII 2022. Vol. 13324. Lecture Notes in Computer Science, Springer, pp. 3–19, 2022.

[BWL20] Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y. M.: Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934/, 2020.

[Ch21] Chen, K.; Lee, C.; Lin, M. P.; Wang, Y.; Chen, Y.: Massive Figure Extraction and Classification in Electronic Component Datasheets for Accelerating PCB Design Preparation. In: 3rd ACM/IEEE Workshop on Machine Learning for CAD, MLCAD 2021. IEEE, pp. 1–6, 2021, URL: https://doi.org/10.1109/MLCAD52597.2021.9531275.

[He20] He, K.; Gkioxari, G.; Dollár, P.; Girshick, R.: Mask R-CNN. IEEE Transactions on Pattern Analysis and Machine Intelligence 42/2, pp. 2961–2969, Feb. 2020.

[HL14] Hu, J.; Liu, Y.: Analysis of Documents Born Digital. In: Handbook of Document Image Processing and Recognition. Springer London, pp. 775–804, 2014.

[Jo22] Jocher, G.; Chaurasia, A.; Stoken, A.; Borovec, J.; NanoCode012; Kwon, Y.; TaoXie; Michael, K.; Fang, J.; imyhxy; Lorna; Wong, C.; Yifu, Z.; V, A.; Montes, D.; Wang, Z.; Fati, C.; Nadar, J.; Laughing; UnglvKitDe; tkianai; yxNONG; Skalski, P.; Hogan, A.; Strobel, M.; Jain, M.; Mammana, L.; xylieong: ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations, version v6.2, Aug. 2022, URL: https://doi.org/10.5281/zenodo.7002879, visited on: 11/05/2022.

[Ka13] Karatzas, D.; Shafait, F.; Uchida, S.; Iwamura, M.; i Bigorda, L. G.; Mestre, S. R.; Mas, J.; Mota, D. F.; Almazan, J. A.; de las Heras, L. P.: ICDAR 2013 Robust Reading Competition. In: 2013 12th International Conference on Document Analysis and Recognition. IEEE, Aug. 2013.

[Ki] Kin-Yiu, W.: Official YOLOv7, URL: https://github.com/WongKinYiu/yolov7, visited on: 11/05/2022.

[Li16] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A. C.: SSD: Single Shot MultiBox Detector. In: Computer Vision – ECCV 2016. Springer, pp. 21–37, 2016.

[Li17a] Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S.: Feature pyramid networks for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017). IEEE, pp. 2117–2125, July 2017.

[Li17b] Lin, T.; Goyal, P.; Girshick, R. B.; He, K.; Dollár, P.: Focal Loss for Dense Object Detection. In: IEEE International Conference on Computer Vision, ICCV 2017. IEEE Computer Society, pp. 2999–3007, 2017, URL: https://doi.org/10.1109/ICCV.2017.324.

[Li22] Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W., et al.: YOLOv6: a single-stage object detection framework for industrial applications. arXiv preprint arXiv:2209.02976/, 2022.

[LO14] Lamiroy, B.; Ogier, J.-M.: Analysis and Interpretation of Graphical Documents. In: Handbook of Document Image Processing and Recognition. Springer London, pp. 553–590, 2014.

[LR14] Lladós, J.; Rusiñol, M.: Graphics Recognition Techniques. In: Handbook of Document Image Processing and Recognition. Springer London, pp. 489–521, 2014.

[Me] Meituan: YOLOv6, URL: https://github.com/meituan/YOLOv6, visited on: 11/05/2022.

[Op22] Opasjumruskit, K.; Böning, S.; Schindler, S.; Peters, D.: OntoHuman: Ontology-Based Information Extraction Tools with Human-in-the-Loop Interaction. In (Luo, Y., ed.): Cooperative Design, Visualization, and Engineering - 19th International Conference, CDVE 2022. Vol. 13492. Lecture Notes in Computer Science, Springer, pp. 68–74, 2022.

[OSP21] Opasjumruskit, K.; Schindler, S.; Peters, D.: Automatic Data Sheet Information Extraction for Supporting Model-Based Systems Engineering. In (Luo, Y., ed.): Cooperative Design, Visualization, and Engineering - 18th International Conference, CDVE 2021. Vol. 12983. Lecture Notes in Computer Science, Springer, pp. 97–102, 2021.

[Pf22] Pfitzmann, B.; Auer, C.; Dolfi, M.; Nassar, A. S.; Staar, P.: DocLayNet: A Large Human-Annotated Dataset for Document-Layout Segmentation. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. KDD '22, Association for Computing Machinery, Washington DC, USA, pp. 3743–3751, Aug. 2022, ISBN: 9781450393850, URL: https://doi.org/10.1145/3534678.3539043.

[Re13] Redmon, J.: Darknet: Open Source Neural Networks in C, 2013, URL: http://pjreddie.com/darknet/.

[Re16] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A.: You only look once: Unified, real-time object detection. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016. IEEE Computer Society, pp. 779–788, 2016.

[Re17] Ren, S.; He, K.; Girshick, R.; Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 39/6, pp. 1137–1149, June 2017.

[RF17] Redmon, J.; Farhadi, A.: YOLO9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition. Pp. 7263–7271, 2017.

[RF18] Redmon, J.; Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767/, 2018.

[Sh21] Shen, Z.; Zhang, R.; Dell, M.; Lee, B. C. G.; Carlson, J.; Li, W.: LayoutParser: A Unified Toolkit for Deep Learning Based Document Image Analysis. In (Lladós, J.; Lopresti, D.; Uchida, S., eds.): Document Analysis and Recognition – ICDAR 2021. Springer International Publishing, Cham, pp. 131–146, 2021, ISBN: 978-3-030-86549-8.

[SSD19] Sultana, F.; Sufian, A.; Dutta, P.: A Review of Object Detection Models based on Convolutional Neural Network. CoRR abs/1905.01614/, 2019, arXiv: 1905.01614.

[TL19] Tan, M.; Le, Q.: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In (Chaudhuri, K.; Salakhutdinov, R., eds.): Proceedings of the 36th International Conference on Machine Learning. Vol. 97. Proceedings of Machine Learning Research, PMLR, pp. 6105–6114, June 2019.

[TPL20] Tan, M.; Pang, R.; Le, Q. V.: EfficientDet: Scalable and efficient object detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2020. IEEE, pp. 10778–10787, 2020.

[Ul] Ultralytics: Hyperparameter Evolution, URL: https://docs.ultralytics.com/tutorials/hyperparameter-evolution/, visited on: 11/25/2022.

[Ul21] Ultralytics: Release V6.0 - yolov5n 'nano' models, Roboflow integration, tensorflow export, opencv DNN support · ultralytics/yolov5, Oct. 2021, URL: https://github.com/ultralytics/yolov5/releases/tag/v6.0, visited on: 11/05/2022.

[Ul23] Ultralytics: ultralytics/ultralytics: NEW - YOLOv8 in PyTorch > ONNX > CoreML > TFLite, 2023, URL: https://github.com/ultralytics/ultralytics, visited on: 01/12/2023.

[Wa20] Wang, C.-Y.; Liao, H.-Y. M.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W.; Yeh, I.-H.: CSPNet: A new backbone that can enhance learning capability of CNN. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. Pp. 390–391, 2020.

[WBL22] Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y. M.: YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696/, 2022.

[Wi] Wightman, R.: EfficientDet (PyTorch), URL: https://github.com/rwightman/efficientdet-pytorch, visited on: 11/05/2022.

[Wu18] Wu, S.; Hsiao, L.; Cheng, X.; Hancock, B.; Rekatsinas, T.; Levis, P.; Ré, C.: Fonduer: Knowledge Base Construction from Richly Formatted Data. In: Proceedings of the 2018 International Conference on Management of Data (SIGMOD 2018). SIGMOD '18, ACM, Houston, TX, USA, pp. 1301–1316, May 2018.

[Ya17] Yang, C.; Yin, X.-C.; Yu, H.; Karatzas, D.; Cao, Y.: ICDAR2017 Robust Reading Challenge on Text Extraction from Biomedical Literature Figures (DeTEXT). In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). IEEE, Nov. 2017.

[Zo23] Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J.: Object Detection in 20 Years: A Survey. Proc. IEEE 111/3, pp. 257–276, Mar. 2023.

[ZTY19] Zhong, X.; Tang, J.; Yepes, A. J.: Publaynet: largest dataset ever for document layout analysis. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, pp. 1015–1022, 2019.