Light OCR Engine with YOLO

Kun Xiao
kx2090@nyu.edu

Haotian Zheng
hz2687@nyu.edu

Luo Chengtao
cl6418@nyu.edu

Abstract

Nowadays, OCR (Optical Character Recognition), a highly successful model, is widely used for automating data extraction from any written text from any images and turning these written text into an accessible form for the machine to do the data processing involving searching the files. With the development of machine learning, OCR (Optical Character Recognition) has turned out to be one of the most crucial methods in the past few years. When our team members were starting to work on this project related to OCR (Optical Character Recognition), the very first thing coming out in our minds was to make an OCR recognition system with YOLO, which was a light weight model. To finalize this light weight model, we would likely apply a large variety of trainable deep neural networks for the analysis. A Yolo detection model might be introduced by ourselves, which was used to acquire any paragraphs or words from certain images or selected files as well. An image morphology method was also an option that would be taken into our consideration for extracting any paragraphs. Eventually, our goal would be to develop a CNN (Convolutional Neural Network) model with the capability to extract and classify any characters from certain text files. The codes can be seen in the Github link: https://github.com/RickyLCT/DL_project2

1 Introduction:

For our dataset, it was set to tackle different tasks and cases. From the original paper, we could observe that this dataset was crucial. [1] For this dataset, we were going to develop a OCR (Optical Character Recognition) Model with the functions including text recognition, semantic entity labeling and entity linking.

OCR (Optical Character Recognition) engine, referred to as text recognition, extracted and classified data from various scanned documents. It served as a reliable tool for users, developers and engineers to complete tasks effortlessly and efficiently. In particular, OCR was commonly used in a large variety of industries. The use of OCR primarily included:

- Scanning driver's license to obtain driver's full name, ID number, and date of expiration/issuance
- Scanning handwritten or printed documents to extract paragraphs, a collection of words, or numbers.

In this project, the primary aim involved constructing a light OCR model. The model targeted at extracting textual data from an image document represented as a PNG/JPG file. The model consisted of a sequence of operations, which can be separated as two roles. The model was trained using the YOLO [2] network to detect areas which contain the texts. After the detection was finished, the model scanned each detected area and performed classification on the texts perceived inside the area to output the corresponding category and built relationships between words, for example, a word labeled question and a word labeled answer might be connected when the answer was corresponding to the question. The model had a maximum 20M of trainable parameters. Overall, the Light OCR model was able to reach an accuracy of 70%. Deep learning models such as FCNN (Fully Convolutional Neural Network) [3] and YOLO (You Only Look Once) might be utilized to develop the model. The performance for our model was evaluated using two different accuracies: precision and recall.

2 Related work:

2.1. Object detection

As we knew, relying on any hand-crafted characteristics, Text detection as an Object detection was used to recognize each character literally. With the development and implementation of Deep learning, it has made a great contribution in object detection. For instance, we implemented some methods that help us to tackle the problems related to text detection. When developers were working on the project and trying to develop some efficient text detectors at the time, they might find out that using Faster R-CNN models [4], well-performing object detectors from traditional literature that was related to computer vision could be a better option. In order to enable object detectors to detect and extract text much more efficiently, lengthy default boxes were introduced. A Rotation-Sensitive Regression Detector was introduced to improve the performance as well.

2.2 End-to-end model

For the end-to-end model, both text detection and transcription were used to improve. Once the text prediction has a low likelihood, it means that the detected box might not catch the complete word. Under this circumstance, an end-to-end strategy might be preferable. For instance, TextDragon, an end-to-end model developed by Feng et al. performed well on distorted text.

2.3 Dataset

The dataset, named as FUNSD, was conducted by Guillaume, Hazom and Jean. It was a subset of the RVL-CDIP dataset [5], and it was created for form understanding. The documents were noisy and widely different in appearance since RVL-CDIP contained 15 classes of data. The FUNSD includes:

- 199 fully annotated images that were scanned form documents and contained a wide distribution of texts. The training set contained 149 forms, 22512 words, 7411 entities, 4236 relations, and the testing set included 50 forms, 8971 words, 2332 entities, 1076 relations. The classes of entities were header, question, answer and other.
- 199 jason files that were corresponding to 199 form documents. They stored arrays of data for the scanned documents. Each entry contained a list of structures that contain data “box”, “text”, “label”, “words”, “linking”, and “id”. “Box” was used to describe the bounding box of the entire structure. The data “text” stored a string of words and/or digits, and their tokenized substrings were described inside the data “words”. The data “word” contained each substring tokenized in “text” and a set of four bounding box vertices to cover the corresponding tokenized string. The data “label” represents the category of the text. The data “linking” simply built the connection between two different structures in the Json file, and the data “id” was used to name the structure.

3 Method:

3.1 YOLOv5

Based on its older version, YOLOv4 [6], YOLOv5 that standed for You Only Look Once Version 5, was introduced, which utilized self-adapted anchor computation, leading to great progress on the detection functionality. In terms of different datasets, YOLOv5 algorithms set different initial value of the length, and the width of the anchor box. And then, in this project, the value was initialized to be [10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]. During the training, the output of this model predicted the box based on the initial box, compared it to the ground-truth of the actual box, computed the difference, and implemented the backward upgrade to iterate hyperparameters.. The figure shown below describes the architecture of YOLOv5s. The 4 red marked sections were the 4 common modules in a target detection algorithm. Corresponding to the 4 marked red sections, YOLOv5 algorithm contained 4 versions, including YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, which were used for training in this project. Among these 4 models, YOLOv5s had the smallest depth, and feature map with the smallest width. Compared with the YOLOv5s, other 3 models turned out to be deeper and wider based on YOLOv5s architecture.

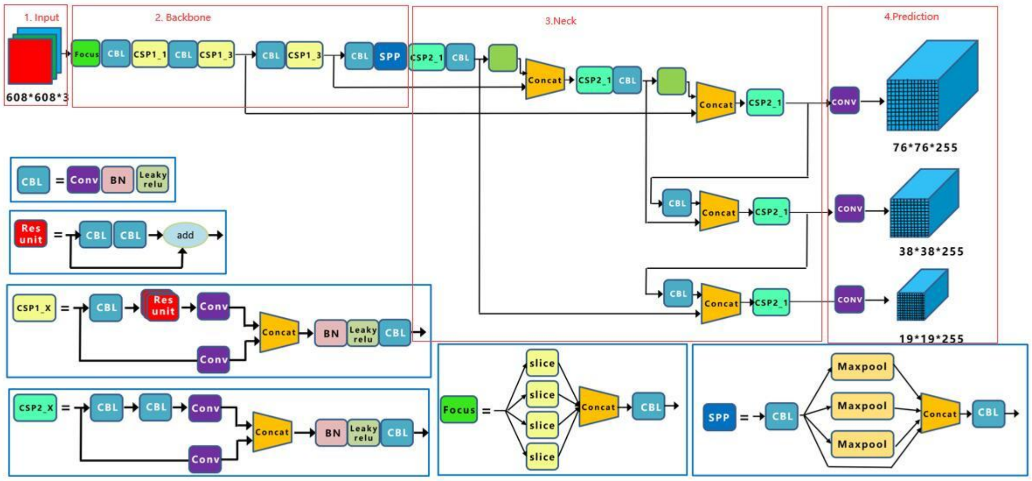


Figure 1

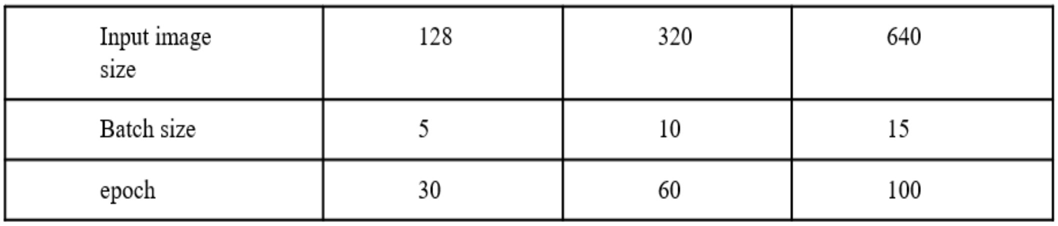
3.2 Loss function

YOLOv5 adopted BECLogits loss function to compute the loss of objectness score. Class probability score utilized BCEclsloss and bounding box uses GIOU Loss. GIOU set IoU as a regression loss. The formula below took any box A, B to be the subset of C, which was the smallest bounding box to be found, and then calculated the ratio of the area of C that did not cover A and B to the total area of C. And the formula was shown below.

$$GIoU = IoU - \frac{|C \backslash (A \cup B)|}{|C|}$$

3.3 Hyperparameter value adjustment

The hyperparameters adjusted were input image size, batch size, epoch. The initial values for each hyperparameter were set to be 640, 10 and 60 respectively.



Input image size	128	320	640
Batch size	5	10	15
epoch	30	60	100

And the table below described some important parameter values.

Learning rate	momentum	Weight decay
0.01	0.937	0.005

4 Preliminary results:

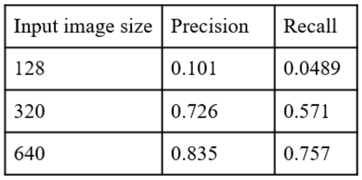
Using the dataset from FUNSD, the Model ‘YOLO’ gave the accuracy curve with respect to the number of samples from which the model has learned. The accuracy was obtained by applying the cross-entropy loss function.

4.1 Results on different models

Model	Precision	Recall
YOLOv5s	0.762	0.612
YOLOv5l	0.835	0.757
YOLOv5m	0.808	0.712
YOLOv5x	0.814	0.766

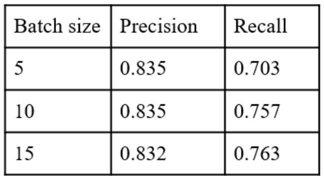
In consideration of both the precision and recall, the architecture YOLOv5l was more qualified to be the text extraction model since it had almost the highest precision and recall; however, as the depth and width of the model increased, the training time increased and the size of model increased. Since the aim of this project was developing a light OCR, the model selected now might be changed in the future.

4.2 Results on different hyperparameter value



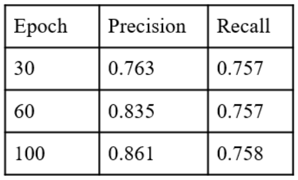
Input image size	Precision	Recall
128	0.101	0.0489
320	0.726	0.571
640	0.835	0.757

Table 1



Batch size	Precision	Recall
5	0.835	0.703
10	0.835	0.757
15	0.832	0.763

Table 2



Epoch	Precision	Recall
30	0.763	0.757
60	0.835	0.757
100	0.861	0.758

Table 3

The table 1 indicated that as the input image size increased, both the precision and recall increased, because if the input image size was too small, the text would be hard to detect and extract. Therefore, 640 as the value of input image size was optimal.

The batch size was the number of images that were put into training concurrently. The table 2 described that both the precisions and recall among three values were similar, therefore, we would choose the initial value.

As we can see from the table3, 60 to be the value of epoch was optimal since compared to 30, value 60 had a greater precision, and compared to value 100, the precision and recall did not increase much but it cost much training time, which was not worthy. Therefore, for the 3 hyperparameters, input image size, batch size, epoch were chosen to be 60, 10, 60 respectively.

4.3 Visible traits and future improvements

Based on the results, we observed that the precision score and recall score were below 90% by executing four different versions of 'YOLOv5'. The model 'YOLOv5l' that obtained the highest scores of precision and recall equivalently underperforms any text recognition tasks. As a result of such low accuracy and stability, we need to tailor the model by searching for the optimal parameters for gradient descent through cross validation.

5 Future work:

For the model 'YOLO', it was constructed and adjusted to meet the reliability and stability towards text extraction. However, the output from the model was only represented as a collection of multiple output data. The output data stored a number of scanned words, sentences, numbers, or some mixed interpretable strings. Besides, the majority of output datas were connected to each other as specified in their structures. In order to optimize the model, there is a list of approachable methods to enrich the features of the existing model.

5.1 Semantic entity labeling

Given that the model 'YOLOv5' has been tested to recognize text, the next step is to implement another possible model to classify the text. Specifically, the texts extracted from 'YOLOv5' can be represented as a semantic entity with no significant meanings. Therefore, it is of great necessity to have another model connected to 'YOLOv5', where the output texts can be associated with one of many categories. For simplicity, the categories will be based on the same training and testing data in 'YOLOv5'. The loss function 'softmax' would be applied to evaluate the performance of the classification model. The details of how this model is constructed will be shared in the final report.

5.2 Entity linking

After all extracted texts are accurately labeled, each text already has its own predicted category. The next step is to design and test the model with the ideas of multiple layer perceptrons to predict the connections among multiple labeled texts. The connection can be treated as a relationship between two labeled texts in different categories (ex., two different texts categorized as "question" and "answer"). The purpose of having the feature of entity linking as a portion of the future work is to try every possible way towards the maximum level of interpretation of each scanned image document. Likewise, the details and experimentations will be equally shared in the final report.

6 Citation:

[1] Jaume, G., Ekenel, H. K., & Thiran, J. P. (2019, September). Funsd: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)* (Vol. 2, pp. 1-6). IEEE.

[2] Jocher, G. et al., 2021. Ultralytics/yolov5: V6.0 - yolov5n 'nano' models, Roboflow integration, TensorFlow Export, opencv DNN support. *Zenodo*. Available at: https://zenodo.org/record/5563715#.YmTMo9pKj-g [Accessed April 24, 2022].

[3] Pratt, H. et al., 1970. FCNN: Fourier Convolutional Neural Networks. *SpringerLink*. Available at: https://link.springer.com/chapter/10.1007/978-3-319-71249-9_47 [Accessed April 24, 2022].

[4] Facebookresearch, Facebookresearch/MASKRCNN-benchmark: Fast, modular reference implementation of instance segmentation and object detection algorithms in pytorch. *GitHub*. Available at: https://github.com/facebookresearch/maskrcnn-benchmark [Accessed April 24, 2022].

[5] Anon, Evaluation of deep convolutional nets for document image classification and retrieval. *IEEE Xplore*. Available at: https://ieeexplore.ieee.org/abstract/document/7333910 [Accessed April 24, 2022].

[6] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.