GraphDetection

Omer Taub, Guy Or, Sean Metlitski. Supervisor: Natan Kaminsky

June 24, 2024

1 Introduction

In various scientific fields, the ability to automatically detect graphs in scientific papers can significantly streamline data extraction and analysis processes. While current language models excel at textual analysis, they do not inherently possess the capability to identify visual data formats such as graphs[3][2]. This project aims to fill this gap by developing a tool specifically designed for the detection of graphs, distinguishing them from general illustrations, images, or objects typically found in scientific literature.

The focus of our research is to construct a robust model capable of accurately detecting graphs without interpreting their content. This is crucial for subsequent applications where foundation models may need to interpret or analyze these graphs. To train and test our detection models, we have created a diverse dataset that includes both simple layouts that mimic clean scientific papers and complex images that replicate the cluttered, multi-element backgrounds common in actual research articles.

Our approach utilizes advanced machine learning techniques to develop detection models that can operate effectively in a variety of visual contexts. The performance of these models is evaluated primarily through the mean Average Precision (mAP) at an Intersection over Union (IoU) threshold of 0.50, which provides a balanced measure of precision and recall, essential for assessing detection accuracy.

2 Dataset Creation

2.1 Regular Dataset

Our Regular Dataset comprises approximately 10,000 images sourced from [5]. These images, each containing between zero to ten graphs, were arranged on blank white images(640X640 of zeros) to simulate images in scientific papers. The graphs were automatically labeled based on their predetermined positions.

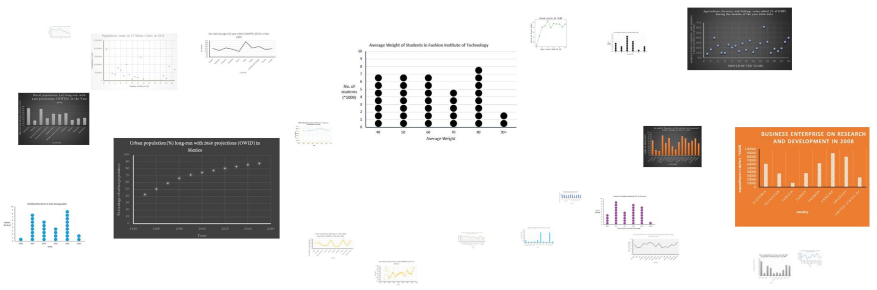

Figure 1: 3 images from the "Regular" dataset

2.2 Complex Dataset

To mimic the complexity of real scientific papers, we introduced the Complex Dataset. This involved using an inpainting diffusion model[6] to enhance the realism of the images. By providing the model with textual context, masks delineating areas to be painted or left untouched, and the original images, we were able to generate varied backgrounds interspersed among the graphs. This approach produced 10,000 complex images that more closely mimic the cluttered, multi-element environment typical of scientific literature.

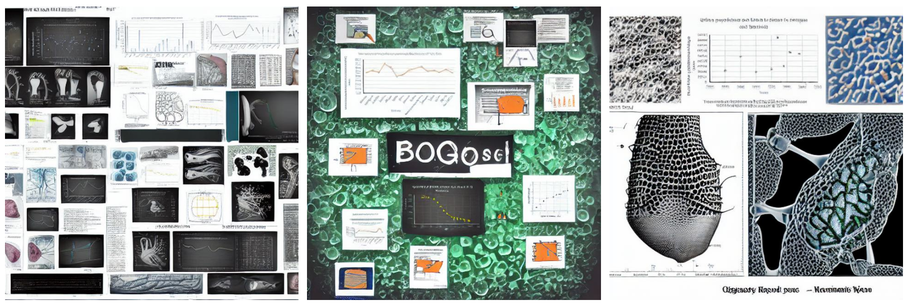

Figure 2: 3 images from the "Complex" dataset

3 Models

In our study, we employed two advanced object detection models known for their robustness and efficiency in handling complex visual data: YOLOv8 and Faster R-CNN . These models were chosen to benchmark their performance on both synthetic and real-world datasets. In addition we implemented our own simple model.

3.1 YOLOv8

YOLOv8, an evolution of the YOLO [4] (You Only Look Once) family, is renowned for its exceptional speed and accuracy in detecting objects in real-time. This model processes images in a single evaluation step, predicting both bounding boxes and class probabilities directly from full images. This direct approach minimizes processing time and is highly beneficial for applications requiring real-time analysis. For our project, YOLOv8 was adapted to detect 'Graph' objects within scientific papers, with the model being trained from scratch and fine-tuned from pre-trained weights on our synthetic dataset.

3.2 Faster R-CNN

Faster R-CNN stands as a pivotal model in the field of deep learning for object detection[1], known for its accuracy and efficiency. It is a two-stage process where a region proposal network (RPN) first proposes candidate object bounding boxes, and a deep convolutional network then refines these suggestions, classifying object types and adjusting their positions simultaneously. This model is particularly suited for scenarios where accuracy is paramount over speed. In our context, Faster R-CNN was employed to meticulously identify and classify graphs within complex images, leveraging both its high precision capabilities and the availability of pre-trained models to enhance initial learning on our diverse datasets.

3.3 Our Object Detection Model

In addition to the two state-of-the-art models we described, we also wanted to create our own simple object detection model. Our model consists of several CNN and pooling layers. It predicts the coordinates of each box in the YOLO format (X center, Y center, height, width). The number of boxes is determined by the maximum number of boxes in the training set.

We created a network with 3 convolutional layers all with a kernel size of 3x3 and padding of 1 to preserve the spatial dimensions. After the convolutional and pooling layers, the feature maps are flattened into a single vector. This vector is passed through a fully connected layer and a ReLU activation function. The output layer is another fully connected layer that uses a sigmoid activation function to predict the bounding box coordinates, ensuring they fall within the range [0, 1].

4 Evaluation

4.1 Overview of Evaluation Metrics

To assess the performance of our graph detection models, we primarily use the mean Average Precision at an Intersection over Union (IoU) threshold of 0.50 (mAP50). This metric is widely recognized in object detection tasks for its ability to balance precision and recall, providing a comprehensive measure of model accuracy.

4.1.1 Intersection over Union (IoU)

Intersection over Union (IoU) is a critical metric in object detection that quantifies the accuracy of a predicted bounding box. It measures the overlap between the predicted bounding box and the ground truth, defined as:

$$\text{IoU} = \frac{\text{Area of Overlap between the predicted box and ground truth box}}{\text{Area of Union of the predicted box and ground truth box}}$$

For the purposes of mAP50, we consider a detection to be correct if the IoU ≥ 0.50.

4.1.2 Precision and Recall

The calculation of mAP involves understanding precision and recall at various threshold levels:

- **Precision** measures the proportion of correct positive predictions:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall** indicates the ability of the model to find all relevant instances:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

4.1.3 Mean Average Precision (mAP) at IoU=0.50

Mean Average Precision (mAP) at an IoU threshold of 0.50 (mAP50) is used as the main performance metric. mAP50 is calculated by averaging the AP values across all classes at the specified IoU threshold. AP for each class is determined from the area under the precision-recall curve, calculated as:

$$\text{AP} = \int_0^1 p(r)\,dr$$

where $p(r)$ is the precision at recall r. Thus, mAP50 is given by:

$$\text{mAP50} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \quad \text{for IoU} \geq 0.50$$

where N is the number of classes. In this project we have only 1 class - "Graph" class.

5 Experiments

Our experiments were meticulously designed to evaluate the performance of YOLO models and Faster R-CNN on both our Regular and Complex datasets, considering variations in pre-training. Each model underwent training and validation on the respective dataset, with the aim of optimizing mean Average Precision (mAP) at an Intersection over Union (IoU) threshold of 0.50 (mAP50).

5.1 YOLO Models

For the YOLO architecture, we conducted a series of experiments to assess its performance under different conditions. We trained YOLO models on both the Regular and Complex datasets, varying the presence of pre-training on ImageNet10K. Specifically, we trained YOLO models from scratch and fine-tuned pre-trained models on our datasets. Each experiment involved training and validation phases, with hyperparameters carefully adjusted to maximize mAP50. hyperparameters:

epochs: 100
batch: 32
imgsz: 640
optimizer: auto
dropout: 0.0

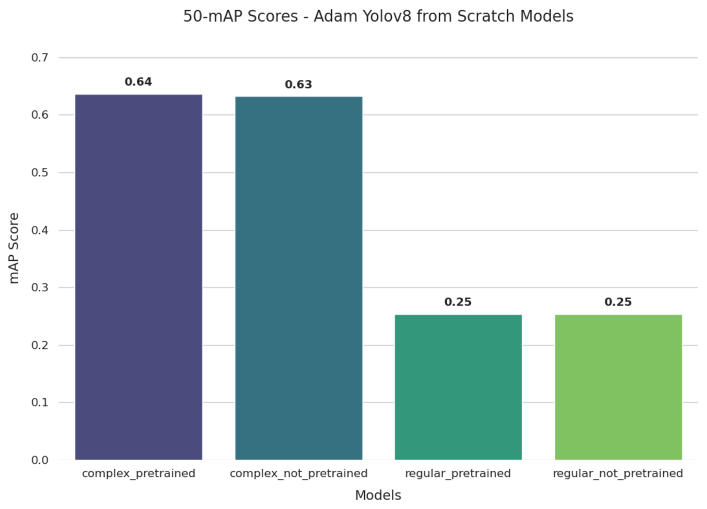

Figure 3: mAP50 Comparison between the 4 Yolo Models on the validation set

5.2 Faster R-CNN

Similarly, experiments with Faster R-CNN were carried out to gauge its effectiveness in detecting graphs within scientific papers. We followed a similar experimental setup as with YOLO models, training and validating Faster R-CNN on both Regular and Complex datasets with and without pre-training on ImageNet10K. Hyperparameters were optimized iteratively to achieve the highest mAP50. hyperparameters:

epochs: 75
batch: 16
imgsz: 640
optimizer: sgd
dropout: 0.0

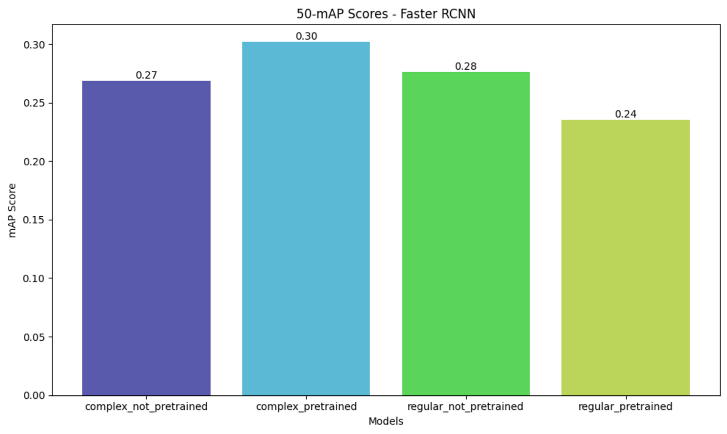

Figure 4: mAP50 Comparison between the 4 Faster Rcnn Models on the validation set

5.3 Our CNN Model

As for our model, we followed a similar experimental setup, but without any pretrained weights (as there are none). We trained the model on both the regular and the complex datasets.

hyperparameters:
epochs: 75
batch: 16
imgsz: 640
optimizer: adam
dropout: 0.0

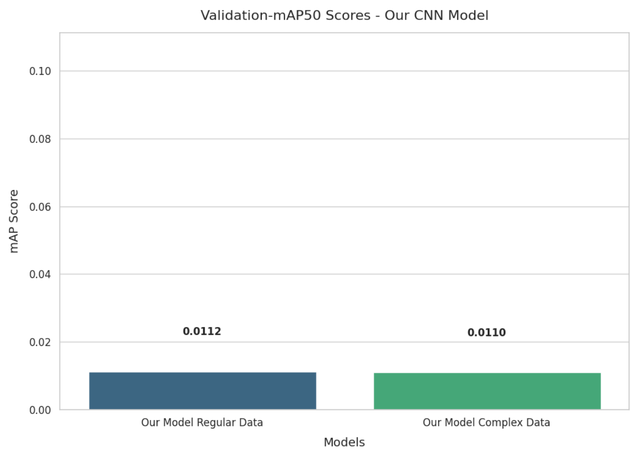

Figure 5: mAP50 Comparison between the 2 CNN-based object detection Models we implemented on the validation set

6 Results

We chose to test the most promising models in our evaluation metric and test them on our un-touched real-world scietific paper images. We saved the best models architecture and weights of each experiment so we can load them on demand and test them. The model we chose to test on the test set are:

yolov8n - pretrained weights on the complex dataset

yolov8n - trained from scratch on the complex dataset.

FasterRcnn - pretrained weights on complex dataset.

FasterRcnn - trained from scratc on regular dataset.

The results are:

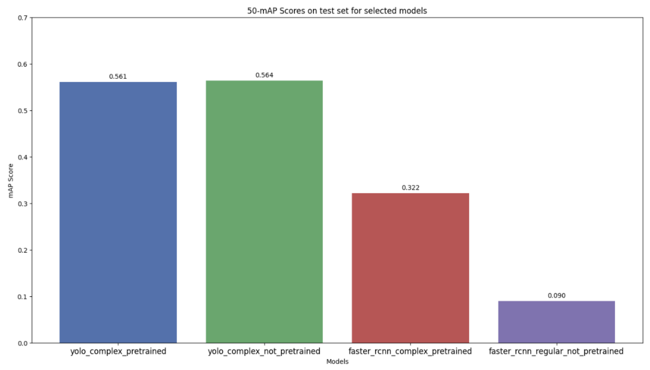

Figure 6: mAP50 Comparison on test set between 4 best performing models

Below are some examples of the selected models predictions:

YOLOv8:

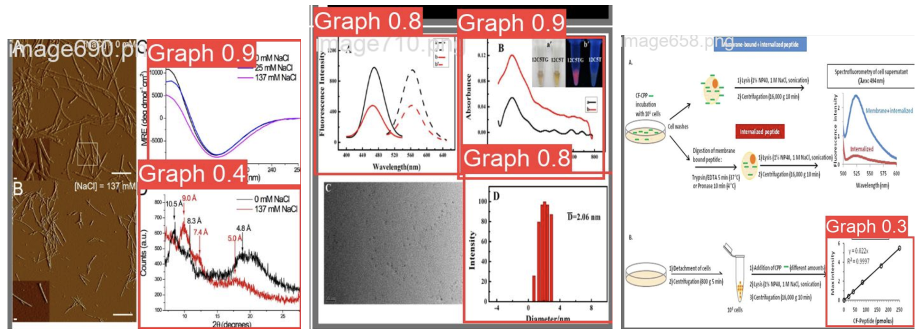

Figure 7: 3 predicitions on real world data using pretrained YOLOv8 model trained on the complex dataset

FasterRcnn:

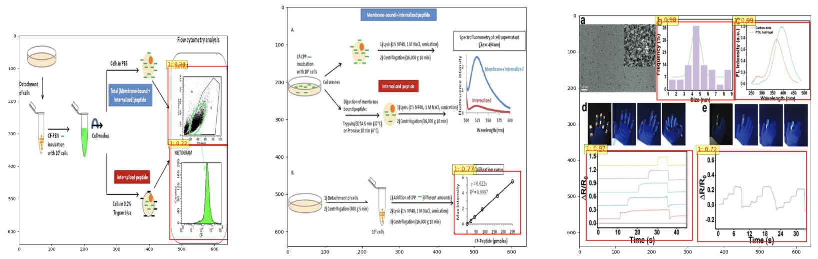

Figure 8: 3 predicitions on real world data pretrained FasterRcnn model trained on the complex dataset

7 Conclusion

The results of our study highlight the significant impact of the complex dataset on the performance of all models. The YOLOv8 model, in particular, demonstrated superior performance, achieving the highest mAP scores. The Faster R-CNN model also performed well, achieving commendable mAP scores and validating its effectiveness for complex detection scenarios. Our simple model, however, fell short in comparison, producing suboptimal results.

A key finding from our experiments is that the complex dataset substantially enhanced the learning capability of the models. By mimicking the cluttered, multi-element environments typical of real-world scientific literature, the complex dataset provided better training conditions, resulting in more accurate graph detection. The superior performance of models trained on the complex dataset underscores its closer resemblance to real-world graph images, which is crucial for effective detection.

For future research or applications in this domain, we strongly recommend the use of generative models or other tools to synthesize training images that better capture the complexity and diversity of real-world scenarios. This approach will likely yield more expressive and effective detection models.

References

[1] Fasterrcnn model - object detection. `https://arxiv.org/pdf/1506.01497`. Accessed: 2024-04-30.

[2] Datatone: Managing ambiguity in natural language interfaces for data visualization. `https://dl.acm.org/doi/abs/10.1145/2807442.2807478`, .

[3] Bridge the modality and capacity gaps in vision-language model selection. `https://arxiv.org/pdf/2403.13797`, .

[4] Yolo - you only look once model - object detection. `https://github.com/ultralytics/ultralytics`. Accessed: 2024-04-30.

[5] Benetech making graphs accessible competition. `https://www.kaggle.com/competitions/benetech-making-graphs-accessible`. Accessed: 2024-04-30.

[6] Inpainting with stable diffusion. `https://github.com/runwayml/stable-diffusioninpainting-with-st`
Accessed: 2024-04-30.