Automatic Scoring Feature

Guide: Prof. Raghunathan Rengaswamy

Shashank M Patil (CH18B022)

Acknowledgements

GITAA Team

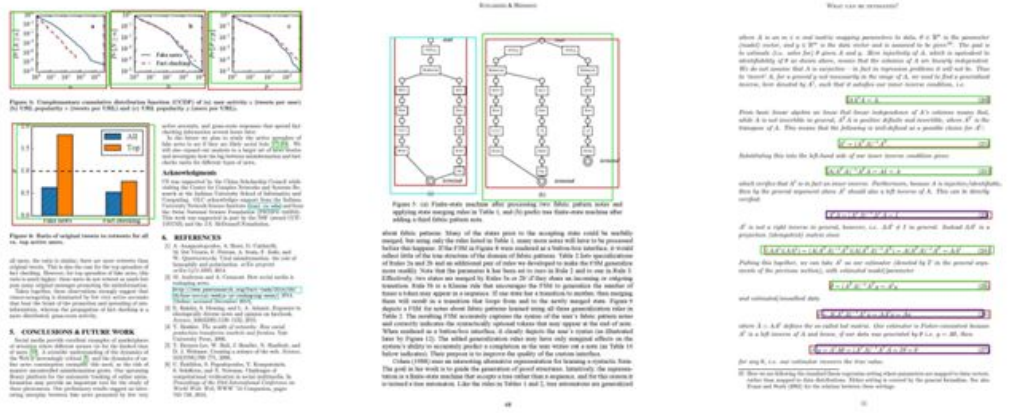
- Suchitra Veeravalli Ma’am
- Navin Kumar Sir
- Baby Aarthy Ganesh Kumar Ma’am
- Selvamani R Sir
- Anilkumar Vempadapu Sir
- Sangeetha Jayasankar Ma’am
- Bhuvaneswari Ramesh Ma’am

- Prof. Arun K Tangirala

Problem Statement

Why automatic scoring?

- Efficiency
- Objectivity
- Scalability
- Immediate Feedback
- Customized Learning



Sample Output from FFD Algorithm (taken from paper)

Note: Evaluation of the score based on figures, formulas and tables

Method Overview

UX: Prof uploads zip file with students' assignments (integrated with Moodle) with template answer key → Scores generated for each student

3 Step Process

Detect: FFT (figure, formula, table) in the answer key and student assignment.

Analyse: Compare each figure from answer key with all the figures detected in assignment

Evaluate: Once the we have the similarity scores, give relevant weightages to the figures in answer key to arrive at the final mark

Detection

Literature Survey

Methods

- based on heuristics, which includes colour-based features, shape-based features, and/or geometric features, etc.
- deep learning approaches include conventional neural networks, region proposal networks and/or deformable neural networks

Papers (deep learning approaches, best results so far)

- Vo et al – based on deep neural networks which is an ensemble of fast-RCNN, and faster-RCNN
- Li et al - combination of conventional computer vision techniques, deep neural networks, and statistical models.

Literature Survey

Datasets

1. ICDAR 2017 POD
    a. 2000 English document page images with annotations for figure, table and formulae
    b. Labeling - includes missing labels, wrong labels, non-uniform labeling conventions
    c. Annotations - at one instance, figures were annotated considering outer boundary, whereas on other pages outer boundary was totally neglected
2. FFD Dataset
    a. 680 document images with annotations for only formulae and figure in the standard format
    b. Manually annotated by a single person

Custom Dataset

Building the feature for tackling hand written answer sheets as well.

- ICDAR Dataset : ~2500 images (tables, formula, figure)
- FFD Dataset: ~700 images (formula, figure)
- Handwritten answer sheets: ~3000 images (formula, figure, table)

Model

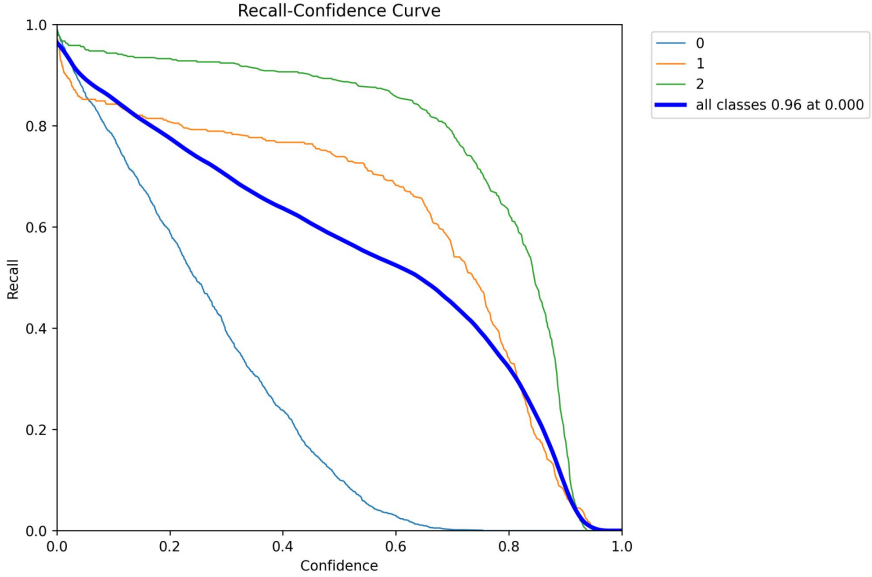
- Tuned the YOLOv8 model on the dataset for object detection
- Open source, state of the art model in terms of accuracy and precision

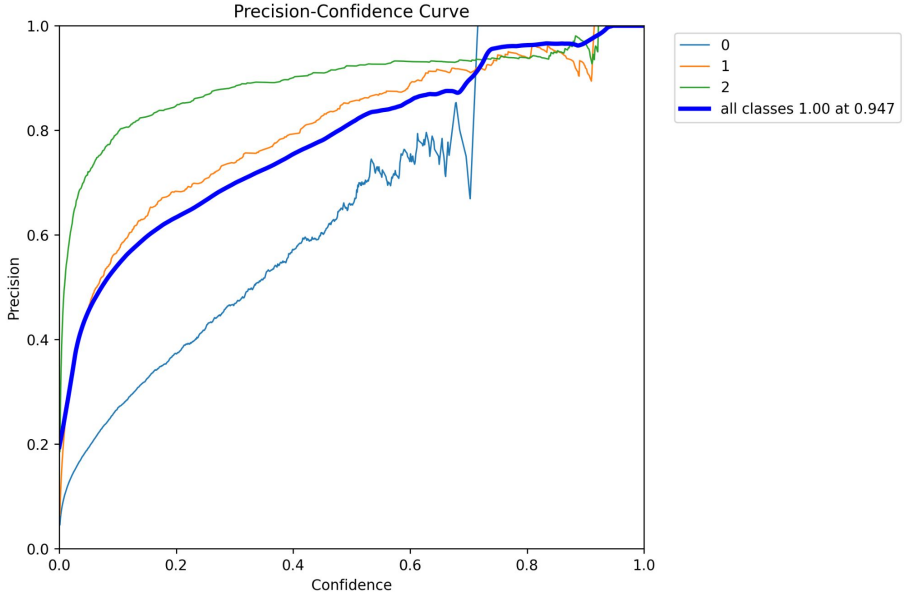
Yolov8

Loss functions included the model:

- Localization loss - predict the bounding box coordinates accurately, priority in our case (mse)
- Confidence loss - differentiate between the object and background binary cross entropy
- Classification loss - predicting the classes, cross entropy

Results


Recall-Confidence Curve
Recall
Confidence
0
1
2
all classes 0.96 at 0.000

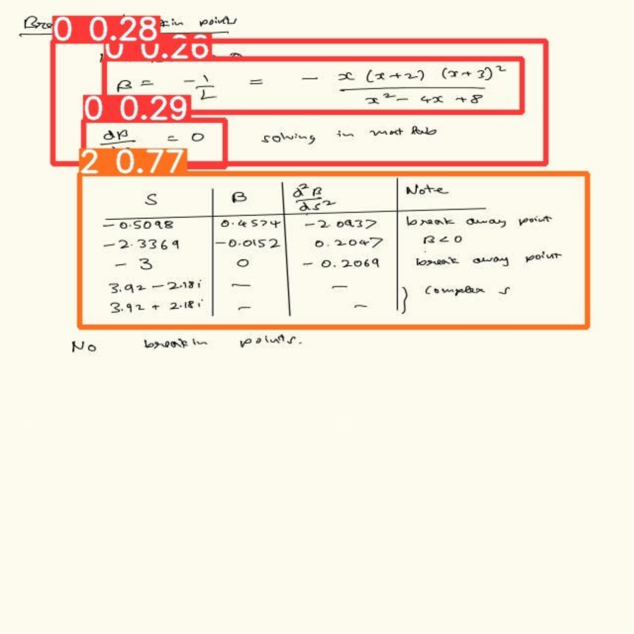

Precision-Confidence Curve
Precision
Confidence
0
1
2
all classes 1.00 at 0.947

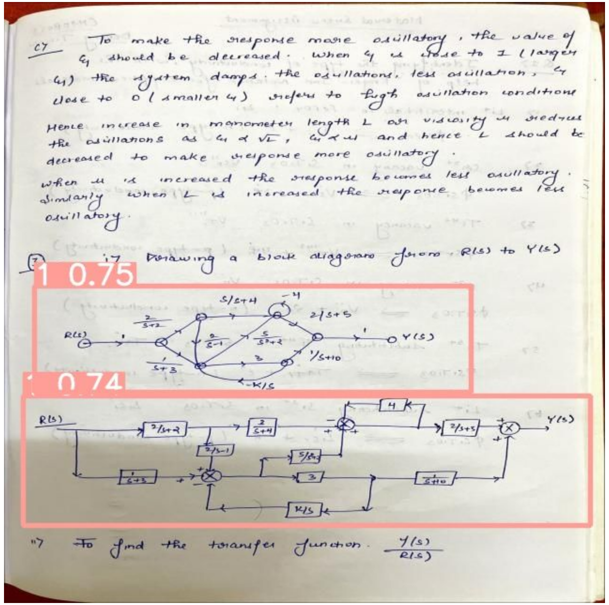
Results

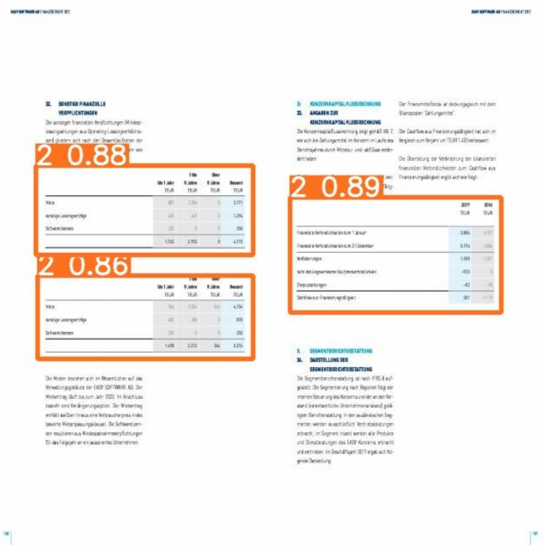
Based on 729 instances (10% of the data)

Class	R	mAP50
Overall	0.729	0.719
Formula	0.468	0.421
Figure	0.792	0.811
Table	0.926	0.924

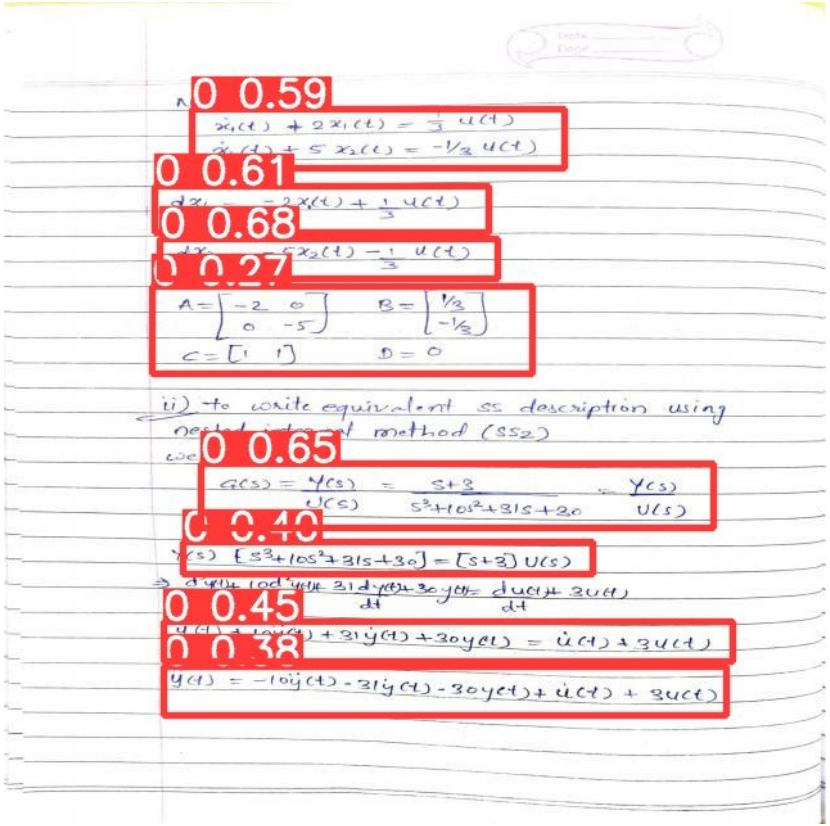
Results (Egs)

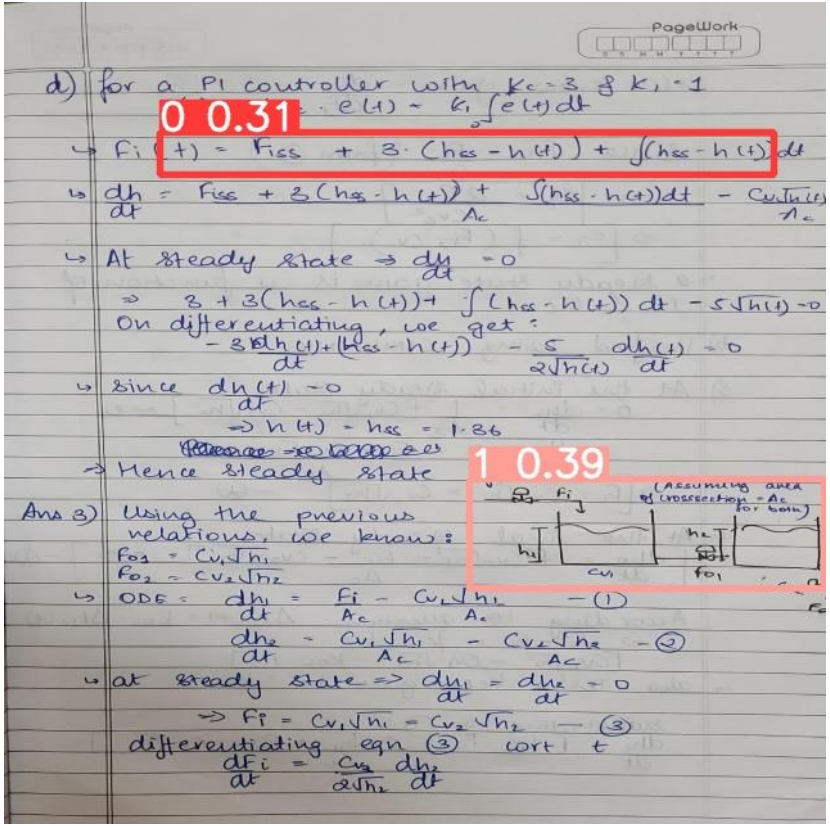




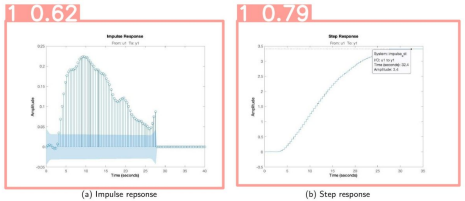


Results





What do we have now?

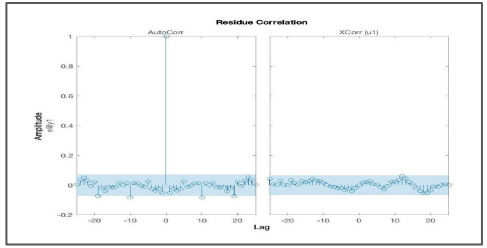


(a) Impulse repsonse (b) Step response

Figure 7

page_5.txt

```
1  1 0.494714 0.686978 0.711204 0.252205
2  1 0.711892 0.254658 0.392535 0.205831
3  1 0.29804 0.256325 0.38476 0.209272
```



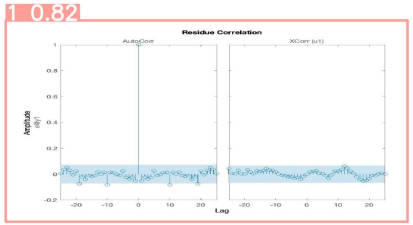
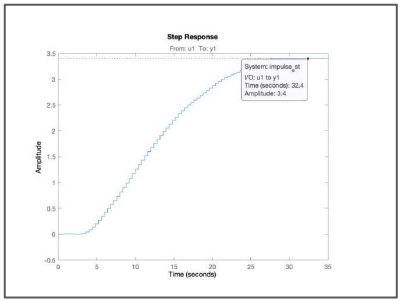
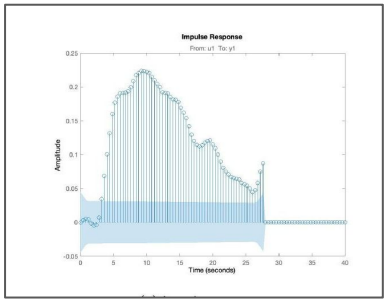


Figure 8: Autocorrelation of residuals and crosscorrelations of residuals with the input





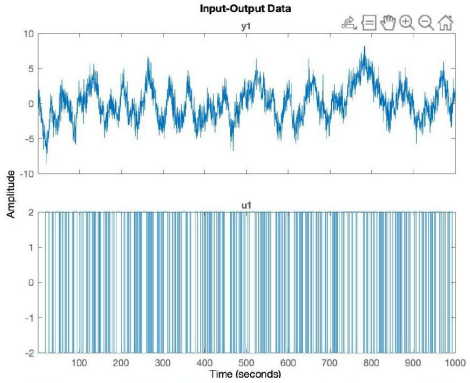
Analyse

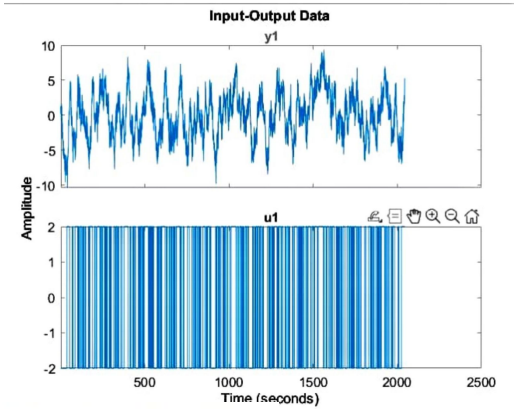
Literature Survey

- Traditional methods - histogram comparison, cosine similarity, euclidean distance
- However they don’t capture the complex relationships between the images
- Methods in use
  - Siamese networks
  - Vision Transformer
  - Deep Image Retrieval Models

Keeping the following factors in mind - Data availability, Dataset size and Image characteristics, chose Vision Transformer model

Results





$$G(S) = \frac{-0.99977e^{-2.1078s}}{(16.508s+1)(16.508s+1)}$$

Image: page_4.jpg, Max Similarity Score: 0.8724057078361511

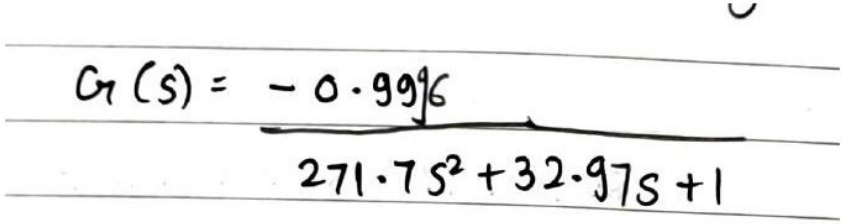


Image: page_1.jpg, Max Similarity Score: 0.8556966781616211

Results

Approximation	Observation
KS FOPTD	Delay is higher than the true value, thus, the initial response does not match. The remaining process is seen to match well
Skogestad's FOPTD	Delay is higher than the true value, thus, the initial response does not match. The remaining process is seen to match well, but not as well as KS FOPTD
Skogestad's SOPTD	Matches best among all approximations and fits the actual step response. SOPTD models do better in capturing the process characteristics, when compared to FOPTD models, due to more parameters
Least Squares SOPTD	Matches least among all the approximations. Due to lower delay, initial response does not match. The step responses match only at steady state, but good approximation for frequency response.

Table 1: Observations for different approximations

All the models give the same gain as the given transfer function.

Approximation	Observations	Parameters			
		K_p	τ_1	τ_1	D
KS FOPTD	Highest delay compared to the process, so it does not match well in the initial response. But it is a good approximation for the rest of the response	-1	23.785	-	18.49
Skogestad's FOPTD	Similar to the above for this FOPTD approximation also the delay is higher. The KS FOPTD is a better fit than this model for the rest of the step response	-1	26	-	15
Skogestad's SOPTD	Skogestad's SOPTD is the best approximation among the four, it matches with the actual step response. Compared to FOPTD models, the SOPTD models capture the characteristics of the response better.	-1	20	14	7
Least squares SOPTD	This is the worst approximation of the 4. The delay found from least squares is lower than the actual process which is why it does not match the initial response and is faster than the actual process. It gives a good approximation for the step response only near the steady state. But this approximation gives better for the frequency response	-0.99977	16.508	16.508	2.1078

Image: page_2.jpg, Max Similarity Score: 0.8150273561477661

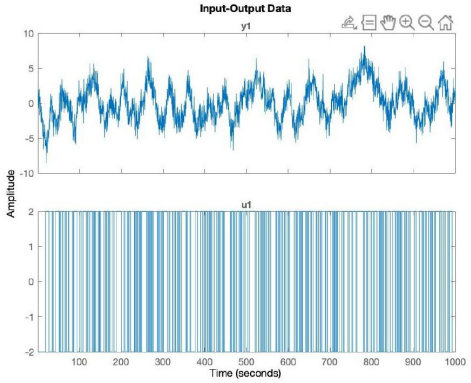
$$\therefore G(s) = -\frac{e^{-7s}}{(20s+1)(14s+1)}$$

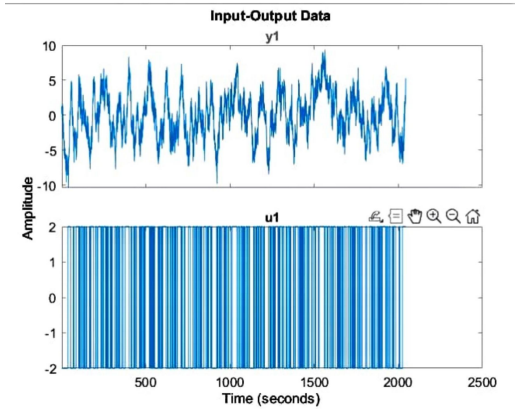
$$\therefore G(s) = -\frac{e^{-7s}}{(20s+1)(14s+1)}$$

Image: page_12.jpg, Max Similarity Score: 0.8818373680114746

What do we have now?

- Similarity scores of the template answer sheet extracted images against all the extracted images of the students answer sheet
- We consider the image corresponding to the max score as the best match for the given template answer sheet image






```
Image: page_4.jpg, Max Similarity Score: 0.8724057078361511
```

Evaluate

Method followed for evaluation

- We need to arrive at a final score
- Usually, marks are allotted for each step a student has written correctly
- Analogously, in this case, professor will manually input the marks for each image (figures/formulas/tables) in the template answer sheet
- We multiply the step marks with the similarity score to arrive at the score for that step
- Summing up for all the images in template answer sheet, we get the final score of the student

Future Scope

- Dataset for object detection can be made robust and trained before deployment
- Different models/ensemble methods can be used for similarity score calculations. Labelled data can be prepared for this purpose.
- Need for quantifying how well the algorithm is performing in evaluating the score
- Converting the handwritten formula to latex format before calculating the similarity
- Can be used at places where there is a designated place to answer a question unlike handwritten assignments.

References

Papers

1. X. Li, F. Yin, and C. Liu. Page object detection from pdf document images by deep structured prediction and supervised clustering. [Link]

2. N. Vo, K. Nguyen, T. Nguyen, and K Nguyen. Ensemble of deep object detectors for page object detection. [Link]

3. Younas, Junaid & Rizvi, Syed Tahseen Raza & Malik, Muhammad Imran & Shafait, Faisal & Lukowicz, Paul & Ahmed, Sheraz. (2019). FFD: Figure and Formula Detection from Document Images. [Link]

Datasets

1. ICDAR 2017 POD [Link]
2. FFD Dataset [Link]

Thank You

Appendix - Yolov8

Loss functions

- Bbox - how tight is the bounding box
- Cls - classification (cross entropy)

Lce = - sum(i,n) t_i * log(p_i) for n classes

p_i softmax probability

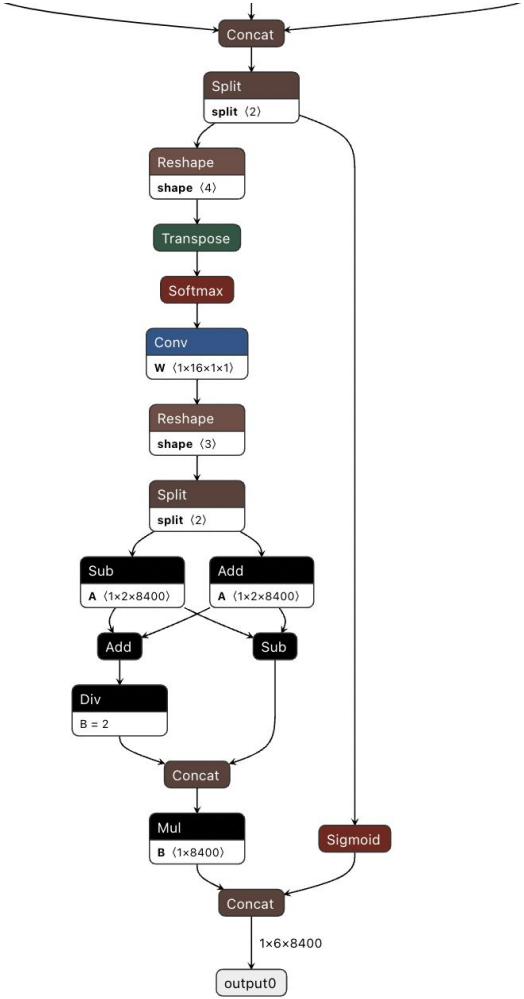
$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

$$L_{\text{CE}} = -\sum_{i=1}^{n} t_i \log(p_i), \text{ for n classes,}$$

where t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

Appendix - Yolov8

Model Architecture



Appendix - VIT B 32

Visual Transformer

The Vision Transformer (ViT) is a model that applies the transformer architecture, originally developed for natural language processing, to computer vision tasks such as image classification.

The "B/32" notation in ViT-B/32 represents the specific configuration of the ViT model. It consists of the following details:

"B" indicates the model size or scale. In the case of ViT, larger model sizes generally indicate higher capacity and potential for improved performance. The specific naming conventions for model sizes may vary across different implementations or papers.

"32" refers to the patch size used in the model. In ViT, the input image is divided into smaller patches, and each patch is treated as a token for processing within the transformer. The patch size defines the dimensions of these patches, with "32" indicating that each patch is a 32x32 pixel square.

The ViT-B/32 model, like other ViT variants, learns representations for images by leveraging the self-attention mechanism of the transformer architecture. This allows the model to capture global relationships between different patches, enabling it to understand the context of the entire image.

VIT B 32

The self-attention mechanism is a key component of the transformer architecture, originally introduced for natural language processing (NLP) tasks, and later adapted for computer vision tasks such as image classification in models like the Vision Transformer (ViT).

In a transformer, self-attention allows the model to weigh the importance of different elements (words in NLP or image patches in computer vision) within a sequence in order to capture dependencies and relationships between them. It enables the model to attend to different parts of the input sequence while processing each element.

The self-attention mechanism computes the attention scores between all pairs of elements in the input sequence. These scores indicate the importance or relevance of one element with respect to the others. The attention scores are used to calculate weighted sums of the values associated with each element, creating context-aware representations.

Here's a high-level overview of the self-attention mechanism:

1. Inputs:
   - Query: Represents the element for which attention is being computed.
   - Key: Represents the elements against which the query is compared.
   - Value: Represents the information associated with each element.

2. Calculating Attention Scores:
   - For each query element, the similarity between the query and all key elements is computed using dot product or other similarity measures.
   - The similarity scores are scaled using a softmax function to obtain attention weights that sum to 1.

3. Weighted Sum:
   - Each value element is multiplied by its corresponding attention weight.
   - The resulting weighted values are summed together to create the context-aware representation for the query element.

4. Multi-Head Attention:
   - In practice, the self-attention mechanism is often performed multiple times in parallel, with different learned linear projections called attention heads.
   - This allows the model to capture different patterns or relationships at different levels of granularity.

The self-attention mechanism enables the transformer to capture long-range dependencies, as each element can attend to any other element in the sequence. It has been shown to be effective in modeling both spatial and temporal relationships, making it well-suited for tasks that require capturing context and relationships between elements.

In computer vision, self-attention has been applied to image classification tasks, where the input sequence consists of image patches or flattened features. By allowing the model to attend to different patches while processing each patch, self-attention helps capture global relationships and long-range dependencies in the image.