Table Detection from scanned images

EE 691- R&D Project
Ayush Munjal, 19D070014
Guided By Prof. Rajbabu Velmurugan & Prof. Biplab Banerjee

Department of Electrical Engineering,
IIT Bombay

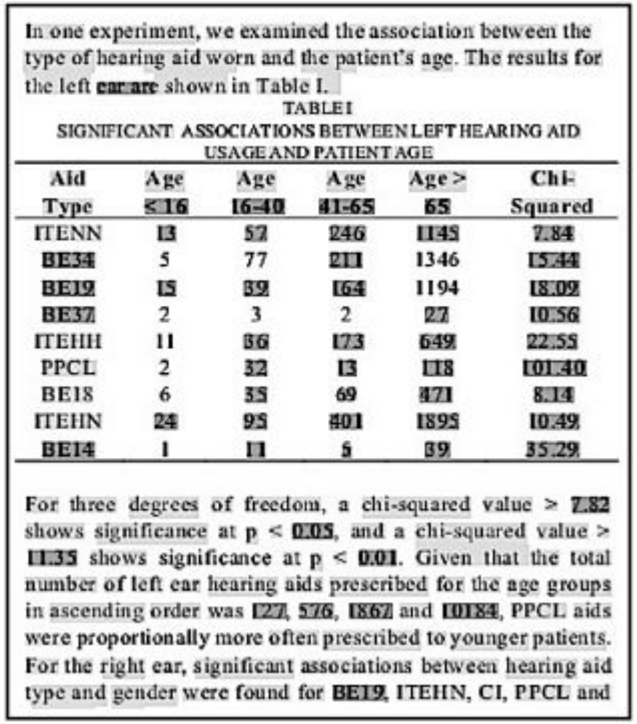
Contents

Introduction

- Important to extract structured data from a document
- Extracting data from images of table is still a big challenge
- We use TableNet, a DL model for both table detection and structure recognition
- Observe the change in accuracy by changing the pre-trained models
- A post processing technique to increase accuracy

TableNet

- Table extraction can be divided into two parts:
  - Table detection: detecting coordinates of table boundary
  - Structure recognition: segmentation of individual rows & columns
- TableNet exploits interdependence of these two problems
- Encoder-Decoder model used
- Two decoder models one each for table and column mask
- Model uses a pre-trained base model (VGG19)



In one experiment, we examined the association between the type of hearing aid worn and the patient's age. The results for the left ear are shown in Table I.

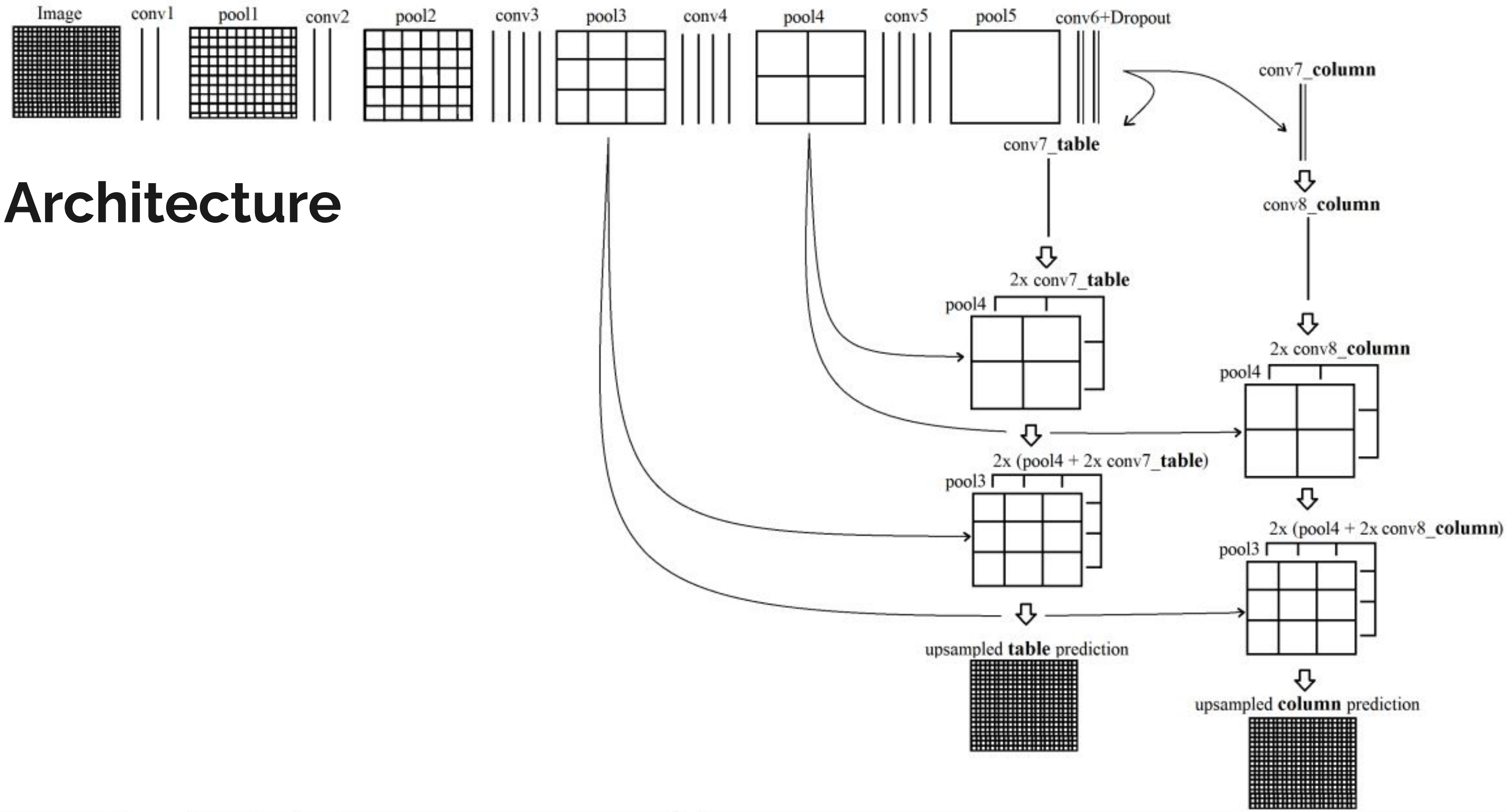
TABLE I
SIGNIFICANT ASSOCIATIONS BETWEEN LEFT HEARING AID USAGE AND PATIENT AGE

Aid Type	Age ≤16	Age 16-40	Age 41-65	Age > 65	Chi-Squared
ITENN	13	57	246	1145	7.84
BE34	5	77	211	1346	15.44
BE19	15	39	164	1194	18.09
BE37	2	3	2	27	10.56
ITEHH	11	36	173	649	22.55
PPCL	2	32	13	118	101.40
BE18	6	35	69	471	8.14
ITEHN	24	95	401	1895	10.49
BE14	1	11	5	39	35.29

For three degrees of freedom, a chi-squared value > 7.82 shows significance at $p < 0.05$, and a chi-squared value > 11.35 shows significance at $p < 0.01$. Given that the total number of left ear hearing aids prescribed for the age groups in ascending order was 127, 576, 1867 and 10184, PPCL aids were proportionally more often prescribed to younger patients. For the right ear, significant associations between hearing aid type and gender were found for BE19, ITEHN, CI, PPCL and

References: Shubham Paliwal, Vishwanath D, Rohit Rahul, Monika Sharma, & Lovekesh Vig. (2020). TableNet: Deep Learning model for end-to-end Table detection and Tabular data extraction from Scanned Document Images.

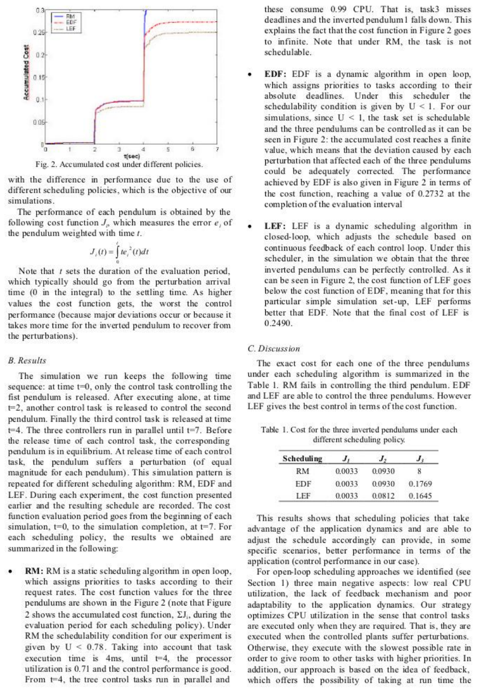
Architecture



References: Shubham Paliwal, Vishwanath D, Rohit Rahul, Monika Sharma, & Lovekesh Vig. (2020). TableNet: Deep Learning model for end-to-end Table detection and Tabular data extraction from Scanned Document Images.

Dataset

- **Marmot dataset** used
- Consists of 1016 documents in Chinese and English, containing tables
- Consists of conference and journal papers, having great variety in page layout and table styles
- 509 English annotated documents used for training
- Image data in *.bmo* file
- Bounding box coordinates in *.xml* file in



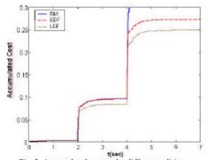


Fig. 2. Accumulated cost under different policies.

with the difference in performance due to the use of different scheduling policies, which is the objective of our simulations.

The performance of each pendulum is obtained by the following cost function J_i, which measures the error e_i of the pendulum weighted with time t.

$$J_i(t) = \int_0^t t e_i^2(t)dt$$

Note that t sets the duration of the evaluation period, which typically should go from the perturbation arrival time (0 in the integral) to the settling time. As higher values the cost function gets, the worst the control performance (because major deviations occur or because it takes more time for the inverted pendulum to recover from the perturbations).

B. Results

The simulation we run keeps the following time sequence: at time t=0, only the control task controlling the fist pendulum is released. After executing alone, at time t=2, another control task is released to control the second pendulum. Finally the third control task is released at time t=4. The three controllers run in parallel until t=7. Before the release time of each control task, the corresponding pendulum is in equilibrium. At release time of each control task, the pendulum suffers a perturbation (of equal magnitude for each pendulum). This simulation pattern is repeated for different scheduling algorithm: RM, EDF and LEF. During each experiment, the cost function presented earlier and the resulting schedule are recorded. The cost function evaluation period goes from the beginning of each simulation, t=0, to the simulation completion, at t=7. For each scheduling policy, the results we obtained are summarized in the following:

- **RM:** RM is a static scheduling algorithm in open loop, which assigns priorities to tasks according to their request rates. The cost function values for the three pendulums are shown in the Figure 2 (note that Figure 2 shows the accumulated cost function, ΣJ_i, during the evaluation period for each scheduling policy). Under RM the schedulability condition for our experiment is given by U < 0.78. Taking into account that task execution time is 4ms, until t=4, the processor utilization is 0.71 and the control performance is good. From t=4, the tree control tasks run in parallel and these consume 0.99 CPU. That is, task3 misses deadlines and the inverted pendulum1 falls down. This explains the fact that the cost function in Figure 2 goes to infinite. Note that under RM, the task is not schedulable.

- **EDF:** EDF is a dynamic algorithm in open loop, which assigns priorities to tasks according to their absolute deadlines. Under this scheduler the schedulability condition is given by U < 1. For our simulations, since U < 1, the task set is schedulable and the three pendulums can be controlled as it can be seen in Figure 2: the accumulated cost reaches a finite value, which means that the deviation caused by each perturbation that affected each of the three pendulums could be adequately corrected. The performance achieved by EDF is also given in Figure 2 in terms of the cost function, reaching a value of 0.2732 at the completion of the evaluation interval

- **LEF:** LEF is a dynamic scheduling algorithm in closed-loop, which adjusts the schedule based on continuous feedback of each control loop. Under this scheduler, in the simulation we obtain that the three inverted pendulums can be perfectly controlled. As it can be seen in Figure 2, the cost function of LEF goes below the cost function of EDF, meaning that for this particular simple simulation set-up, LEF performs better that EDF. Note that the final cost of LEF is 0.2490.

C. Discussion

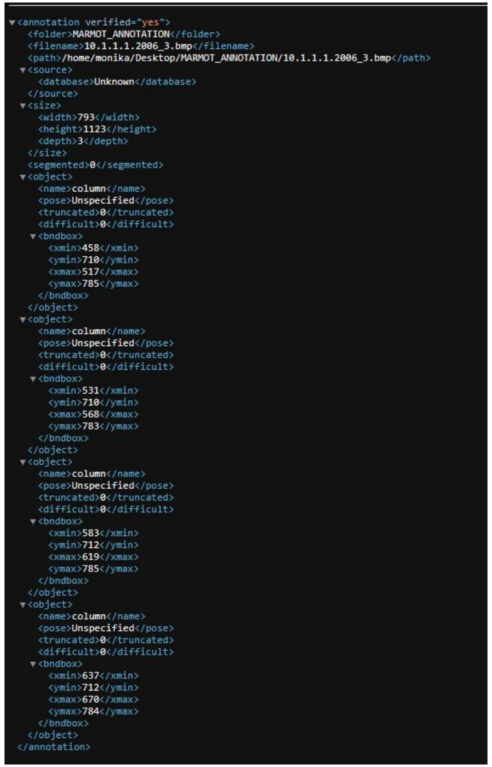
The exact cost for each one of the three pendulums under each scheduling algorithm is summarized in the Table 1. RM fails in controlling the third pendulum. EDF and LEF are able to control the three pendulums. However LEF gives the best control in terms of the cost function.

Table 1. Cost for the three inverted pendulums under each different scheduling policy.

Scheduling	J_1	J_2	J_3
RM	0.0033	0.0930	8
EDF	0.0033	0.0930	0.1769
LEF	0.0033	0.0812	0.1645

This results shows that scheduling policies that take advantage of the application dynamics and are able to adjust the schedule accordingly can provide, in some specific scenarios, better performance in terms of the application (control performance in our case).

For open-loop scheduling approaches we identified (see Section 1) three main negative aspects: low real CPU utilization, the lack of feedback mechanism and poor adaptability to the application dynamics. Our strategy optimizes CPU utilization in the sense that control tasks are executed only when they are required. That is, they are executed when the controlled plants suffer perturbations. Otherwise, they execute with the slowest possible rate in order to give room to other tasks with higher priorities. In addition, our approach is based on the idea of feedback, which offers the possibility of taking at run time the



```
<annotation verified="yes">
  <folder>MARMOT_ANNOTATION</folder>
  <filename>10.1.1.1.2006_3.bmp</filename>
  <path>/home/monika/Desktop/MARMOT_ANNOTATION/10.1.1.1.2006_3.bmp</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>793</width>
    <height>1123</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>column</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>458</xmin>
      <ymin>710</ymin>
      <xmax>517</xmax>
      <ymax>785</ymax>
    </bndbox>
  </object>
  <object>
    <name>column</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>531</xmin>
      <ymin>710</ymin>
      <xmax>568</xmax>
      <ymax>783</ymax>
    </bndbox>
  </object>
  <object>
    <name>column</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>583</xmin>
      <ymin>712</ymin>
      <xmax>619</xmax>
      <ymax>785</ymax>
    </bndbox>
  </object>
  <object>
    <name>column</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>637</xmin>
      <ymin>712</ymin>
      <xmax>670</xmax>
      <ymax>784</ymax>
    </bndbox>
  </object>
</annotation>
```

Document Image

XML file

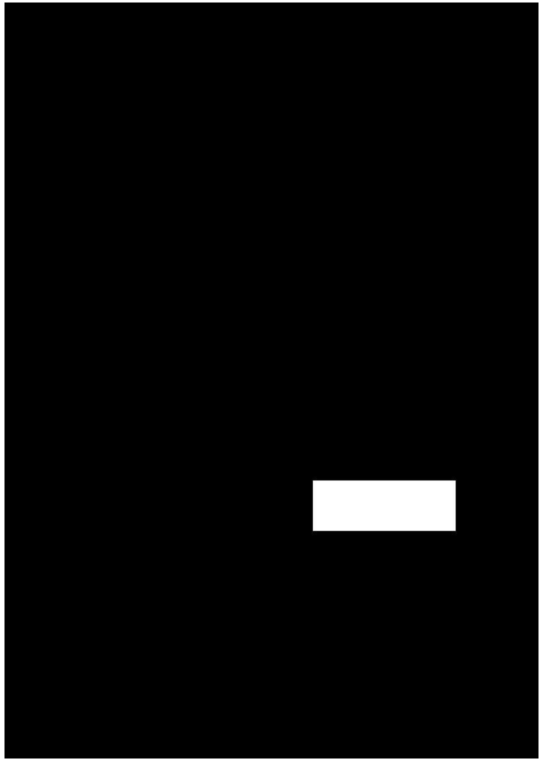
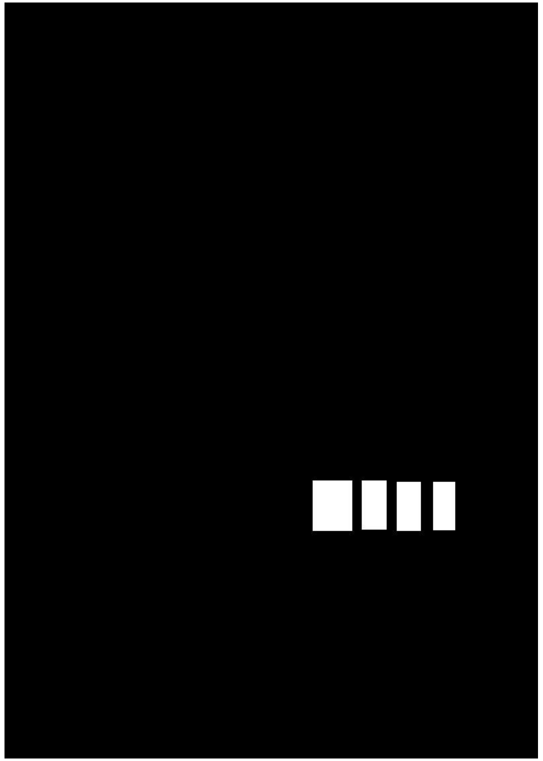


Table mask


Column mask

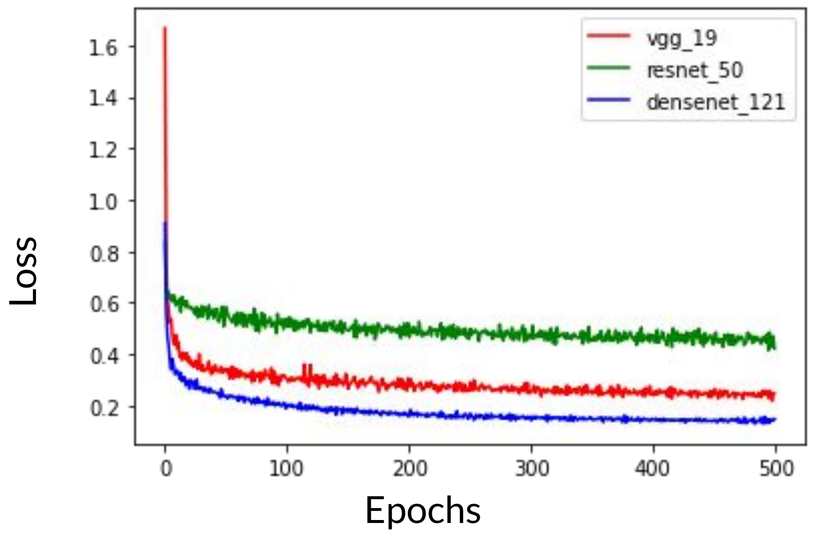
Model architecture

- TableNet architecture used
- Consists of:
    - Base pre-trained model
    - Common encoder-decoder for table and column graph
    - Separate decoder layers for table and column mask
- Both decoders consists of two alternate convolution and pooling layers
- Any image classifier model can be used as pre-trained model
- This pre-trained model used as common encoder

Experiments

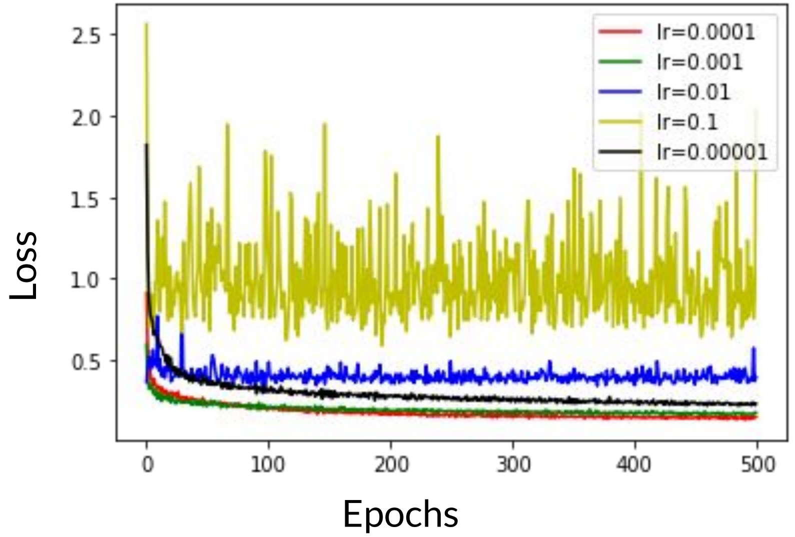
- Epochs = 500
- Training data = 444 samples
- Testing data = 50 samples
- Validation subsplits = 5
- Sparse Categorical Cross Entropy loss used for both column and table masks
- Adam optimizer used
- Three based models used: VGG19, ReseNet51, DenseNet121

Comparison of these models

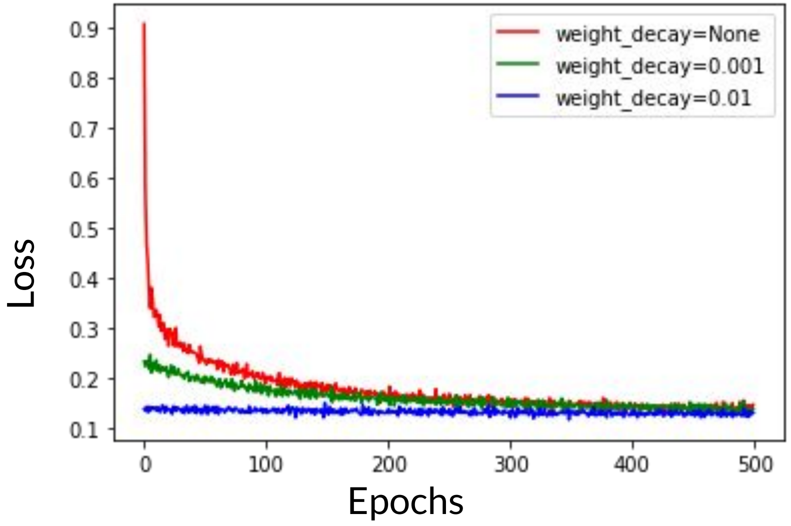


Comparison of loss of VGG19, ResNet50, DenseNet121 models for 500 epochs

Hyperparameter tuning for DenseNet121



DenseNet121 loss values for
different learning rate for 500 epochs



DenseNet121 loss values for different
weight decay for 500 epochs

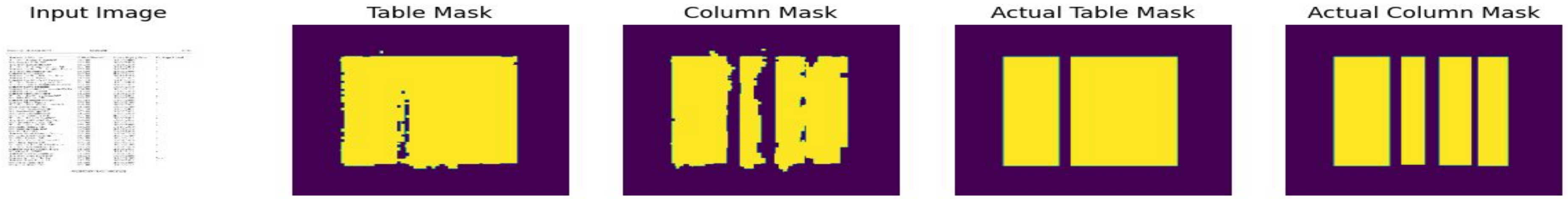
Results for different learning rates for DenseNet121

Learning rate	Training table acc	Training column acc	Validation table acc	Validation column acc
0.0001	0.9634	0.9367	0.9738	0.9375
0.001	0.9567	0.9316	0.9545	0.9288
0.00001	0.9472	0.9216	0.9550	0.9280

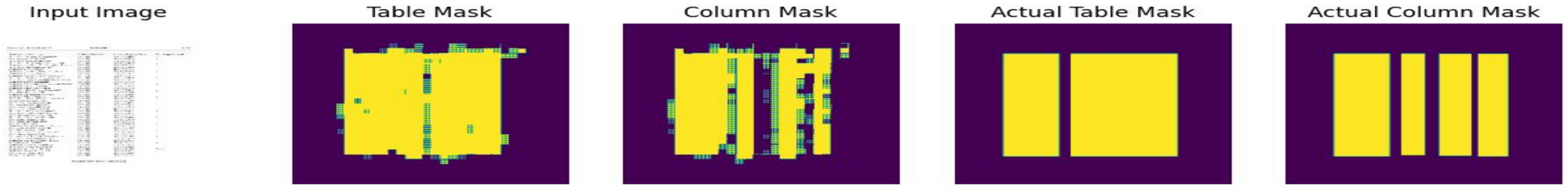
Comparison of three models

Average IoU scores for test dataset

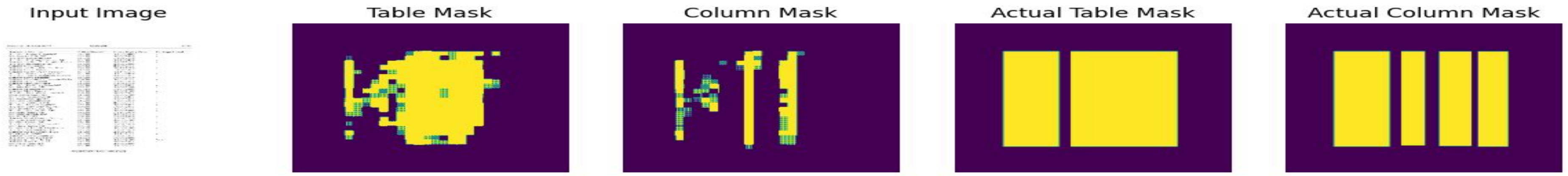
Model	IoU score for table mask	IoU score for column mask
DenseNet121	0.84	0.72
VGG19	0.67	0.50
ResNet50	0.36	0.18



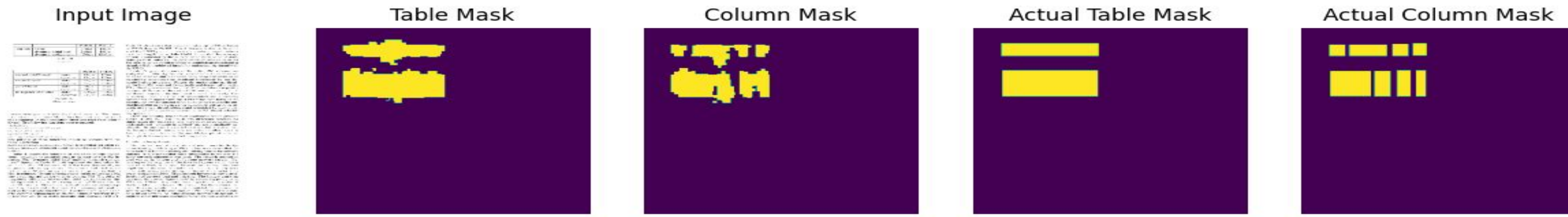
Results for DenseNet121 - IoU score for table mask = 0.92, column mask = 0.86



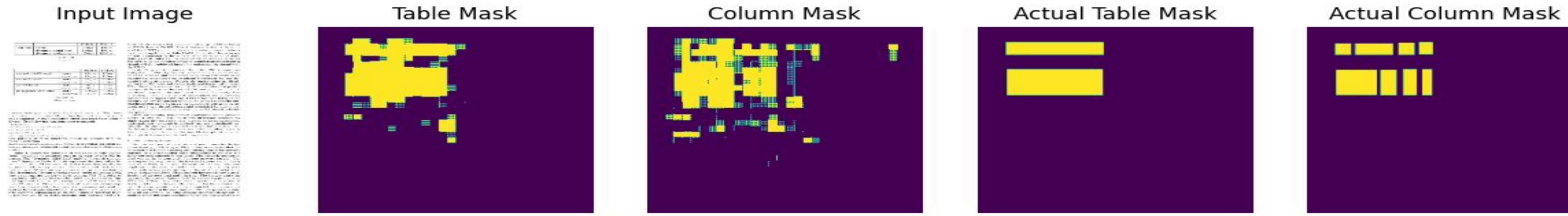
Results for VGG19 - IoU score for table mask = 0.79, column mask = 0.65



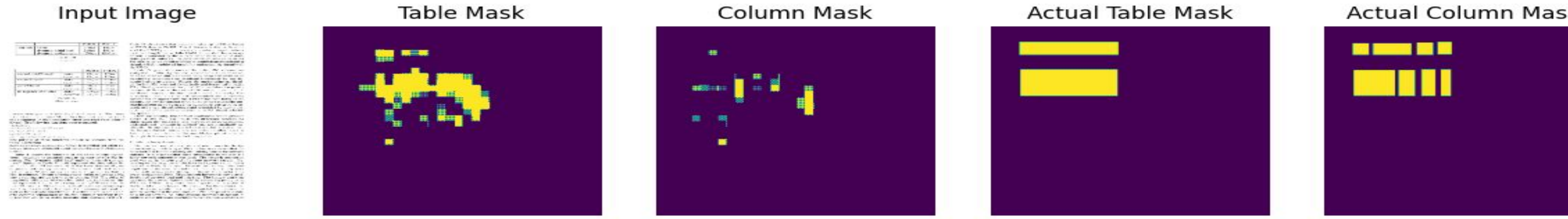
Results for ResNet50 - IoU score for table mask = 0.51, column mask = 0.25



Results for DenseNet121 - IoU score for table mask = 0.82, column mask = 0.72



Results for VGG19 - IoU score for table mask = 0.67, column mask = 0.45



Results for ResNet50 - IoU score for table mask = 0.30, column mask = 0.09

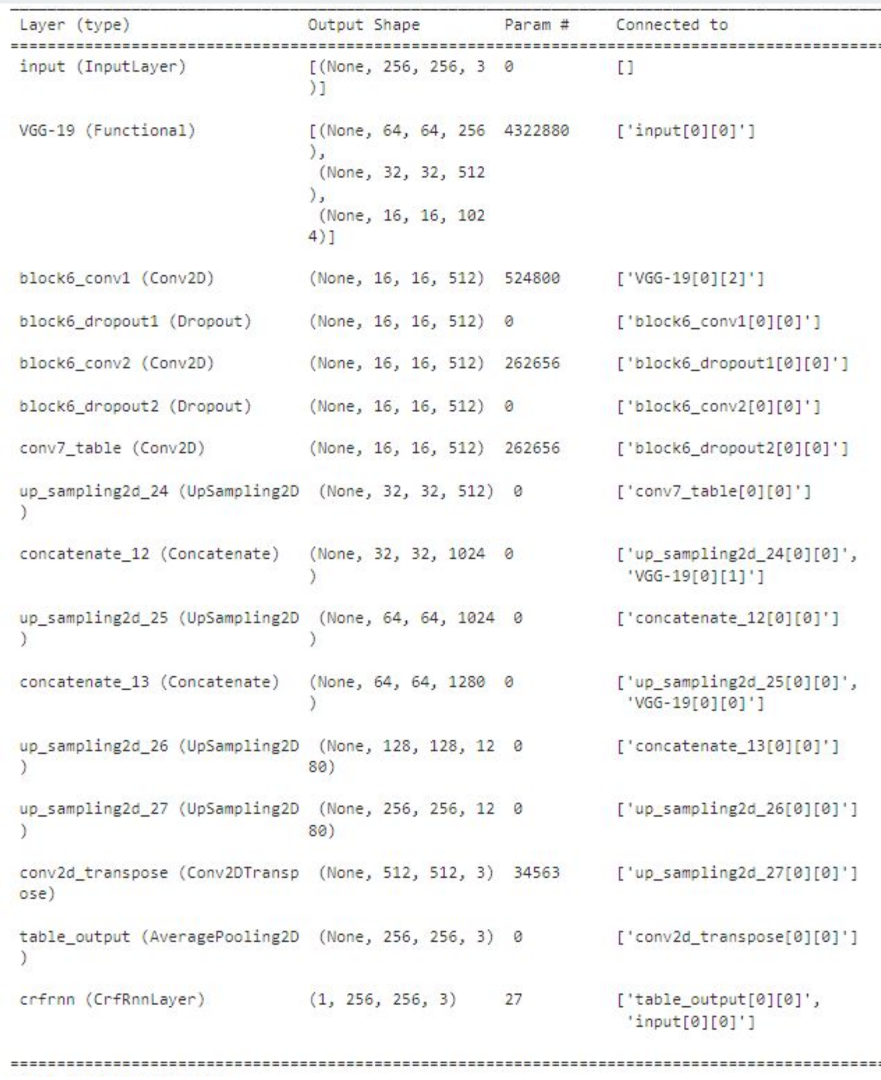
Post Processing of Image Segmentation using Conditional Random Fields

- CRF model exploits the dependencies within input domains
- Different types of CRFs depending on how many neighbours to consider:
  - Linear
  - Grid
  - Dense
- CRFs often used for sequence pattern recognition tasks
- It incorporates subjective and non-independent features of information

CRF

- No direct implementation of CRF available for image segmentation task
- CRF-RNN repository used which provides an implementation of CRF layer for image segmentation tasks
- Easy to use implementation
- Problem with some versions of tensorflow
- Batch size can only be set to 1

```
crf_layer = CrfRnnLayer(image_dims=(256, 256),
      num_classes=3, theta_alpha=3.,
      theta_beta=160., theta_gamma=3.,
      num_iterations=100,
      name='crfrnn')
([original_model.outputs[0], original_model.inputs[0]]
```



```
 ______________________________________________________________________________________________
 Layer (type)                   Output Shape          Param #      Connected to
 ==============================================================================================
 input (InputLayer)             [(None, 256, 256, 3  0            []
                                )]

 VGG-19 (Functional)            [(None, 64, 64, 256  4322880      ['input[0][0]']
                                ),
                                 (None, 32, 32, 512
                                ),
                                 (None, 16, 16, 102
                                4)]

 block6_conv1 (Conv2D)          (None, 16, 16, 512)  524800       ['VGG-19[0][2]']

 block6_dropout1 (Dropout)      (None, 16, 16, 512)  0            ['block6_conv1[0][0]']

 block6_conv2 (Conv2D)          (None, 16, 16, 512)  262656       ['block6_dropout1[0][0]']

 block6_dropout2 (Dropout)      (None, 16, 16, 512)  0            ['block6_conv2[0][0]']

 conv7_table (Conv2D)           (None, 16, 16, 512)  262656       ['block6_dropout2[0][0]']

 up_sampling2d_24 (UpSampling2D  (None, 32, 32, 512)  0           ['conv7_table[0][0]']
 )

 concatenate_12 (Concatenate)   (None, 32, 32, 1024  0            ['up_sampling2d_24[0][0]',
                                )                                  'VGG-19[0][1]']

 up_sampling2d_25 (UpSampling2D  (None, 64, 64, 1024  0           ['concatenate_12[0][0]']
 )                              )

 concatenate_13 (Concatenate)   (None, 64, 64, 1280  0            ['up_sampling2d_25[0][0]',
                                )                                  'VGG-19[0][0]']

 up_sampling2d_26 (UpSampling2D  (None, 128, 128, 12  0           ['concatenate_13[0][0]']
 )                              80)

 up_sampling2d_27 (UpSampling2D  (None, 256, 256, 12  0           ['up_sampling2d_26[0][0]']
 )                              80)

 conv2d_transpose (Conv2DTransp  (None, 512, 512, 3)  34563       ['up_sampling2d_27[0][0]']
 ose)

 table_output (AveragePooling2D  (None, 256, 256, 3)  0           ['conv2d_transpose[0][0]']
 )

 crfrnn (CrfRnnLayer)           (1, 256, 256, 3)     27           ['table_output[0][0]',
                                                                   'input[0][0]']

 ==============================================================================================
```

Implementation

- Added CRF layer after the table output and freezed the weights of original model
- Sparse Categorical Cross entropy loss used
- Adam optimizer
- Number of iterations = 100
- Epochs = 1

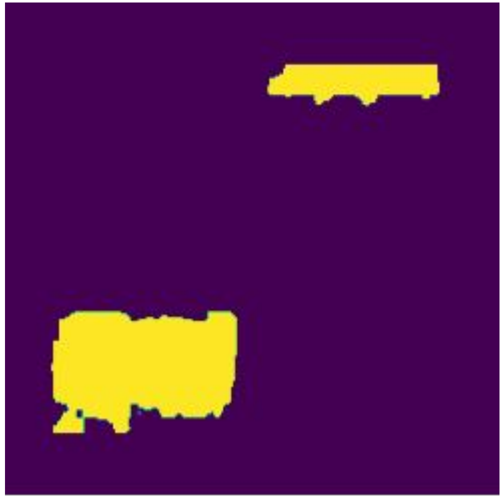
Results

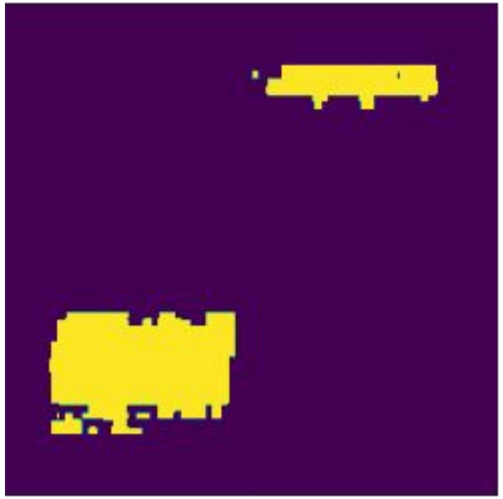
Table mask accuracy values with CRF layers

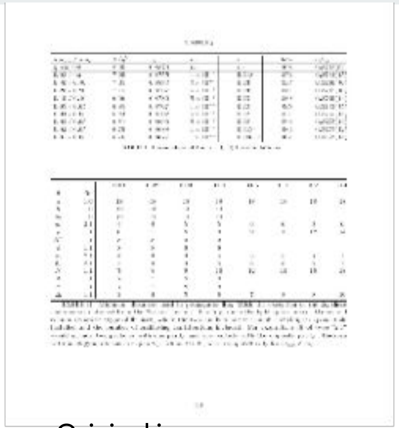
Learning rate	Table mask accuracy
0.01	0.9663
0.001	0.9673
0.0001	0.9658

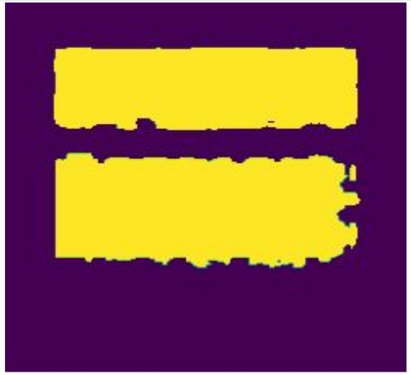
Comparison of CRF with DenseNet121

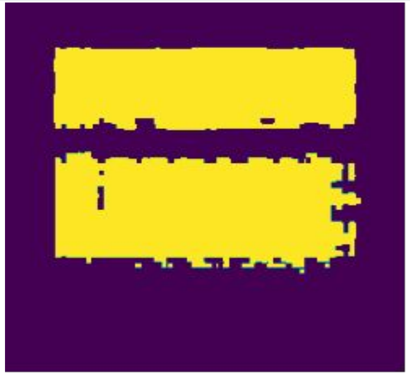
Original image


CRF Table mask - IoU score = 0.72

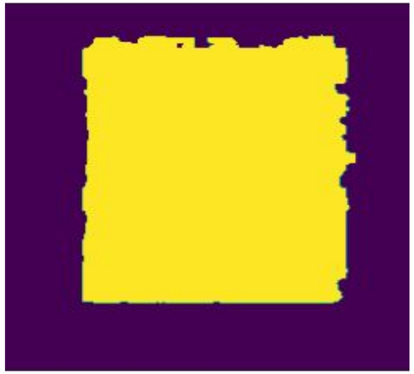

DenseNet Table mask - IoU score = 0.7

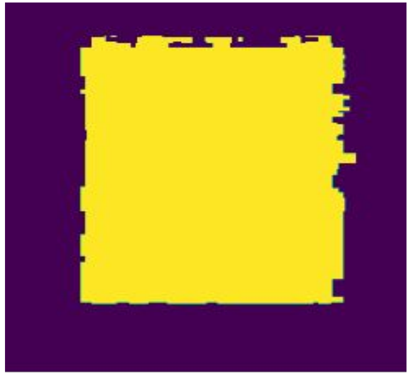

Original image


CRF Table mask - IoU score = 0.822


DenseNet Table mask - IoU score = 0.815

Original image


CRF Table mask - IoU score = 0.947


DenseNet Table mask - IoU score = 0.938

Results on provided images

Original image

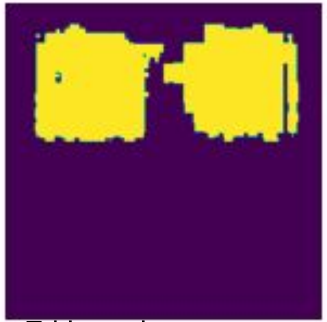
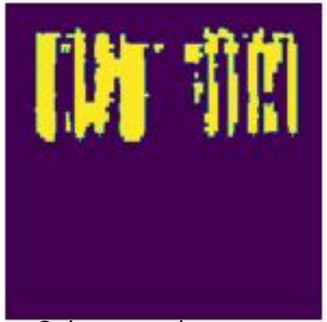


Table mask


Column mask

Original image

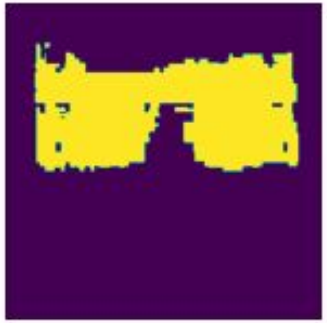
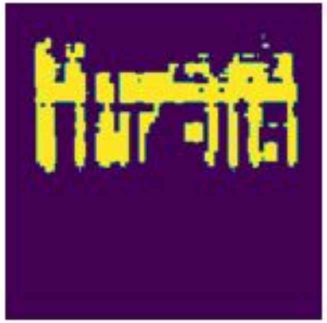
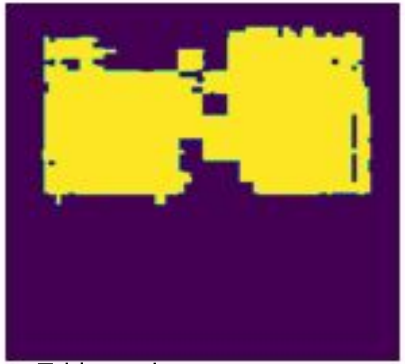
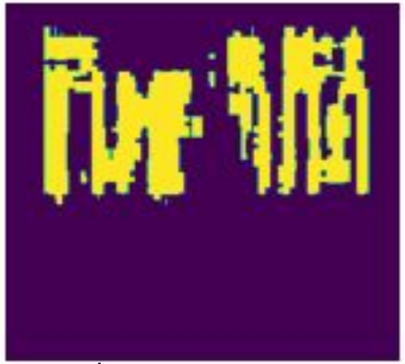


Table mask


Column mask

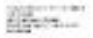


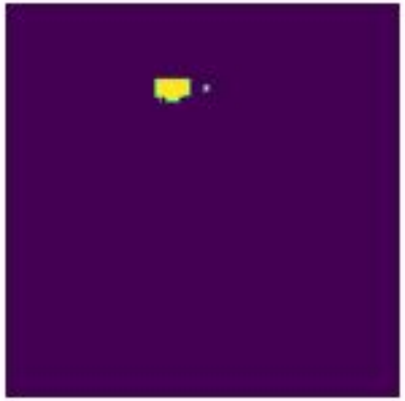


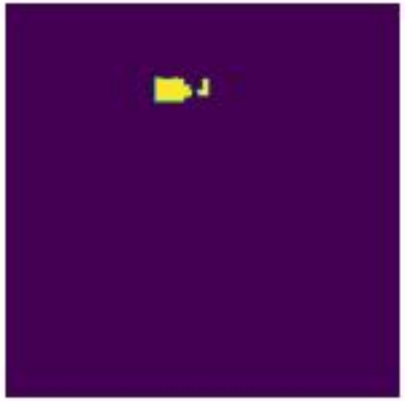
Original image

Table mask

Column mask







Original image

Table mask

Column mask

Future work

- Using IoU loss instead of Cross Entropy loss or mixture of both
- Varying values of column mask weight and table mask weight
- More improvement required in column mask
- Extraction of data from the table

References

- Shubham Paliwal, Vishwanath D, Rohit Rahul, Monika Sharma, & Lovekesh Vig. (2020). TableNet: Deep Learning model for end-to-end Table detection and Tabular data extraction from Scanned Document Images
- Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, & Philip H. S. Torr (2015). Conditional Random Fields as Recurrent Neural Networks
- crfasrnn_keras: CRF-RNN Keras/Tensorflow version
- Dhawan, A., Bodani, P., & Garg, V. (2019). Post Processing of Image Segmentation using Conditional Random Fields

Thank You!