
TableSmith

A tool to bring out tabular data from everyday files

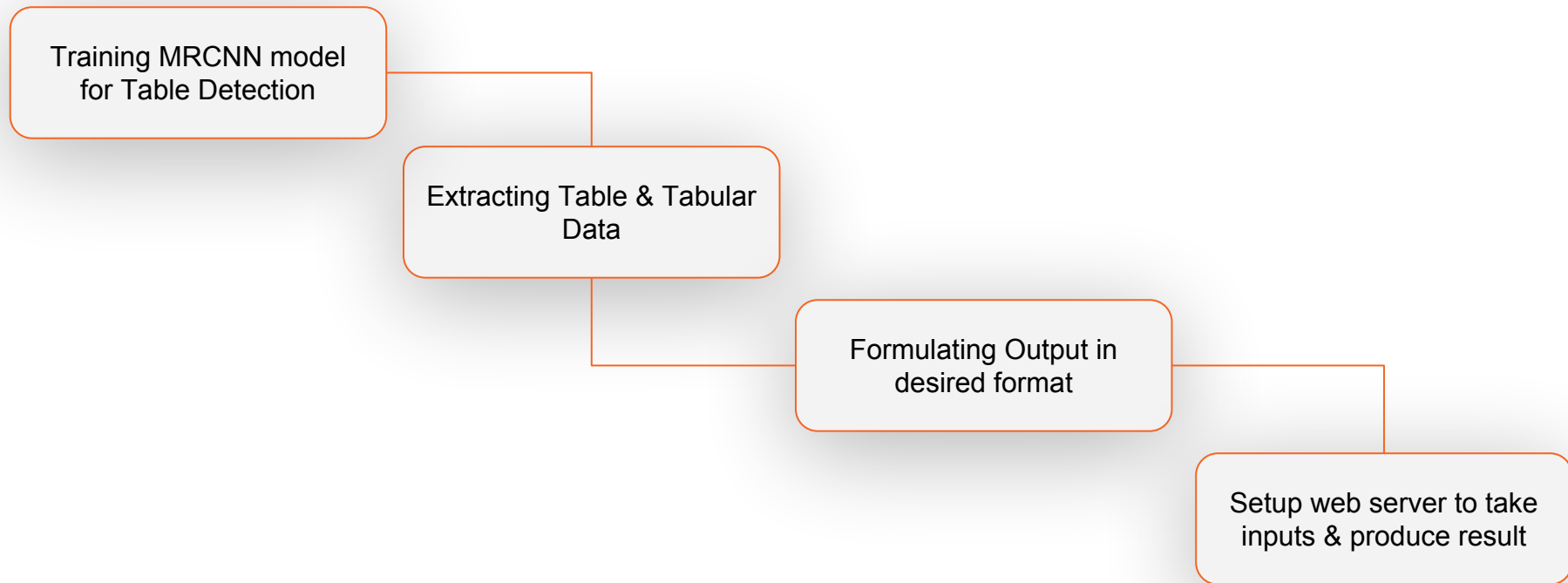
Team: Cyberpunk

Name: Rohit Sharma

Mobile: 9762450616

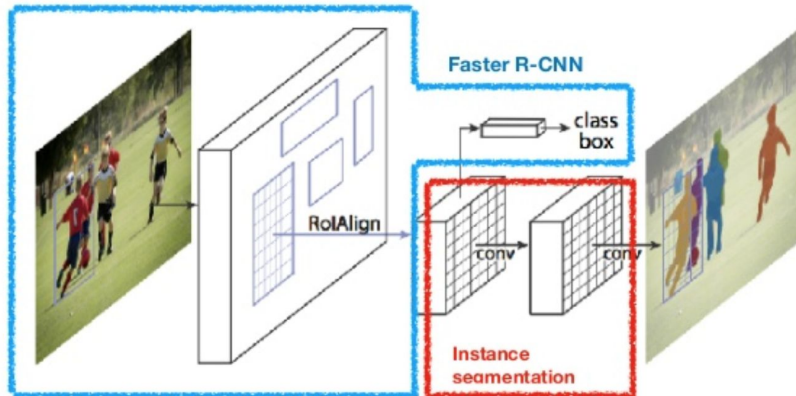
Theme: Table Reading & Understanding in Documents/Images

Final solution design



Algorithm & models applied

- **Mask RCNN** is an instance segmentation model that can identify pixel by pixel location of any object.
- It extends Faster R-CNN to pixel-level **image segmentation**. The key point is to decouple the classification and the pixel-level mask prediction tasks. Based on the framework of **Faster R-CNN**, it added a third branch for predicting an object **mask** in parallel with the existing branches for classification and localization. The mask branch is a small fully-connected network applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner.
- The multi-task loss function of Mask R-CNN combines the loss of classification, localization and segmentation mask: $\text{Loss} = L(\text{cls}) + L(\text{box}) + L(\text{mask})$, where $L(\text{cls})$ and $L(\text{box})$ are same as in Faster R-CNN.



Tech stack

MRCNN model
written in Keras
Trained on K80 GPU
in Google Colab



TensorFlow
inference
on CPU



Flask(Python)
web server



Deployment on
AWS/Heroku

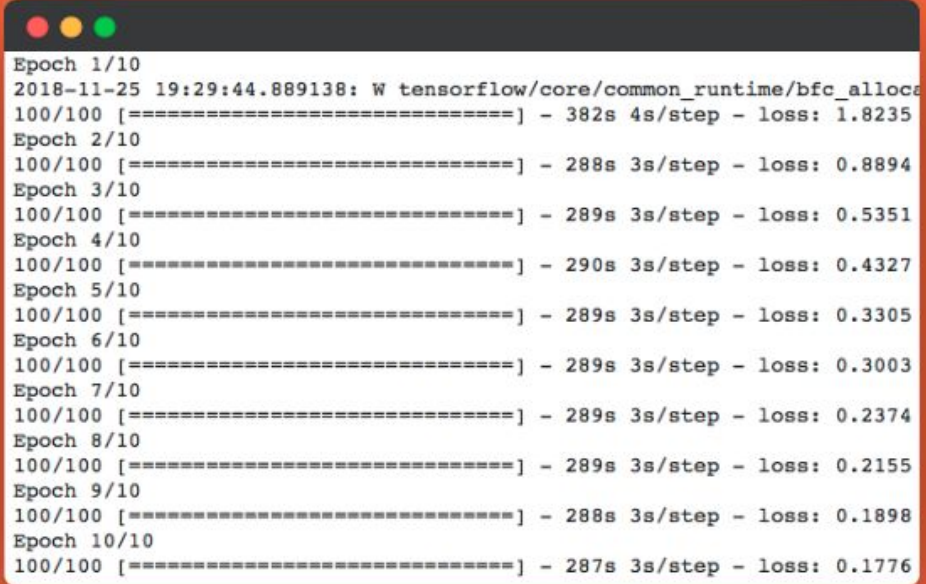


heroku

Metrics - Accuracy & loss attained

Dataset consisted of 40 training images & 10 validation images. They were full page scans, which were manually annotated.

Training was performed in 50 mins on Tesla K80 Nvidia GPUs on google colaboratory.



```
Epoch 1/10
2018-11-25 19:29:44.889138: W tensorflow/core/common_runtime/bfc_allocator.cc:1116: BFCAllocator: 100/100 [=====] - 382s 4s/step - loss: 1.8235
Epoch 2/10
100/100 [=====] - 288s 3s/step - loss: 0.8894
Epoch 3/10
100/100 [=====] - 289s 3s/step - loss: 0.5351
Epoch 4/10
100/100 [=====] - 290s 3s/step - loss: 0.4327
Epoch 5/10
100/100 [=====] - 289s 3s/step - loss: 0.3305
Epoch 6/10
100/100 [=====] - 289s 3s/step - loss: 0.3003
Epoch 7/10
100/100 [=====] - 289s 3s/step - loss: 0.2374
Epoch 8/10
100/100 [=====] - 289s 3s/step - loss: 0.2155
Epoch 9/10
100/100 [=====] - 288s 3s/step - loss: 0.1898
Epoch 10/10
100/100 [=====] - 287s 3s/step - loss: 0.1776
```

Input sample

Table 2: Accuracy results across all participants and puzzles.

Participant	A	B	C	D	E	F	G	Avg
Puzzle 1	0.78	0.92	0.93	0.78	0.81	0.73	0.90	0.84
Puzzle 2	0.69	0.60	0.89	0.91	0.72	0.88	0.90	0.80
Puzzle 3	0.92	0.76	0.88	0.91	0.92	0.91	0.82	0.87
Puzzle 4	0.81	0.69	0.84	0.99	0.85	0.96	0.73	0.84
Puzzle 5	0.61	0.83	0.89	0.73	0.55	0.68	0.70	0.71
Puzzle 6	0.78	n/a	0.77	0.72	0.74	0.90	0.86	0.80
Avg	0.77	0.76	0.87	0.84	0.77	0.84	0.82	0.81

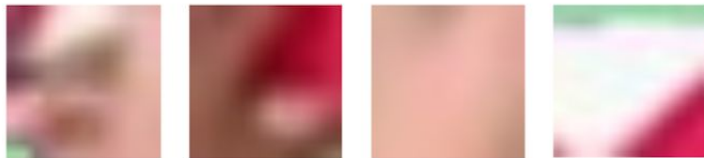


Figure 4: Examples of misclassification by our block identification system. From left to right, calculated labels were: *SW, Red, White, White.*

Output sample

Table 2: Accuracy results across all participants and puzzles.

Participant	A	B	C	D	E	F	G	Avg
Puzzle 1	0.78	0.92	0.93	0.78	0.81	0.73	0.90	0.84
Puzzle 2	0.69	0.60	0.89	0.91	0.72	0.88	0.90	0.80
Puzzle 3	0.92	0.76	0.88	0.91	0.92	0.91	0.82	0.87
Puzzle 4	0.81	0.69	0.84	0.99	0.85	0.96	0.73	0.84
Puzzle 5	0.61	0.83	0.89	0.73	0.55	0.68	0.70	0.71
Puzzle 6	0.78	n/a	0.77	0.72	0.74	0.90	0.86	0.80
Avg	0.77	0.76	0.87	0.84	0.77	0.84	0.82	0.81