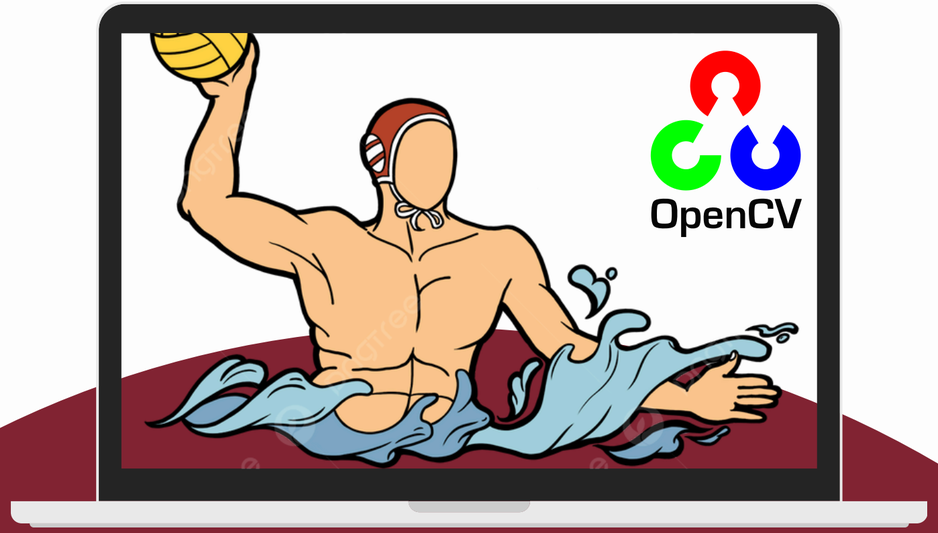
SAPIENZA
UNIVERSITÀ DI ROMA

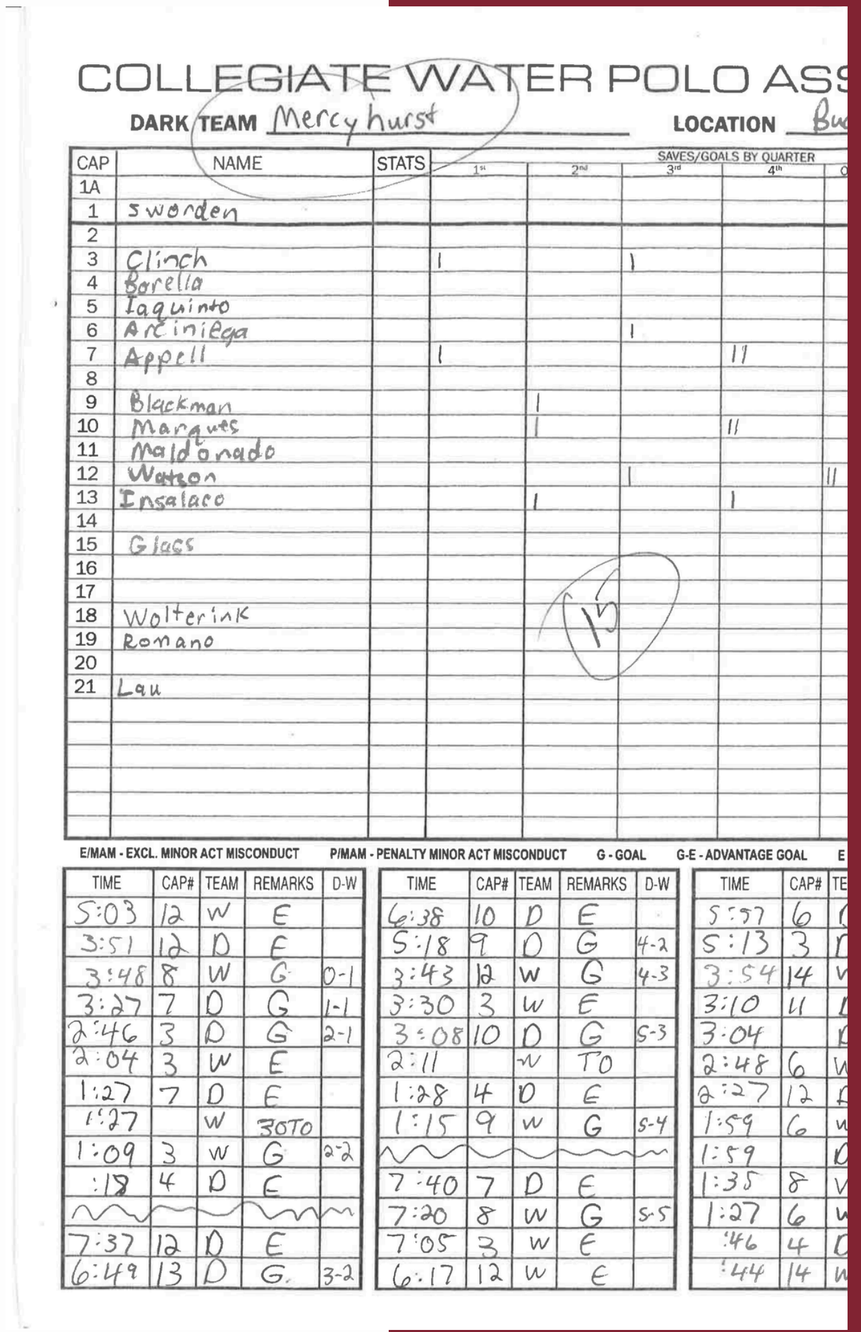
Water Polo Scoresheet Reader and Data Organizer Using Optical Character Recognition (OCR)



Computer Vision 2024

By: Chris Kreienkamp

Overview



COLLEGIATE WATER POLO ASS

DARK TEAM Mercyhurst LOCATION Bu

CAP	NAME	STATS	SAVES/GOALS BY QUARTER 1st	2nd	3rd	4th	
1A							
1	sworden						
2							
3	Clinch		1		1		
4	Borella						
5	Iaquinto						
6	Arciniega				1		
7	Appell		1			11	
8							
9	Blackman			1			
10	Marques			1		11	
11	Maldonado						
12	Watson				1		11
13	Insalaco			1		1	
14							
15	Glass						
16							
17							
18	Wolterink						
19	Romano						
20							
21	Lau						

E/MAM - EXCL. MINOR ACT MISCONDUCT P/MAM - PENALTY MINOR ACT MISCONDUCT G - GOAL G-E - ADVANTAGE GOAL E

TIME	CAP#	TEAM	REMARKS	D-W
5:03	12	W	E	
3:51	12	D	E	
3:48	8	W	G	0-1
3:27	7	D	G	1-1
2:46	3	D	G	2-1
2:04	3	W	E	
1:27	7	D	E	
1:27		W	30TO	
1:09	3	W	G	2-2
:18	4	D	E	
7:37	12	D	E	
6:49	13	D	G	3-2

TIME	CAP#	TEAM	REMARKS	D-W
6:38	10	D	E	
5:18	9	D	G	4-2
3:43	12	W	G	4-3
3:30	3	W	E	
3:08	10	D	G	5-3
2:11		W	TO	
1:28	4	D	E	
1:15	9	W	G	5-4
7:40	7	D	E	
7:20	8	W	G	5-5
7:05	3	W	E	
6:17	12	W	E	

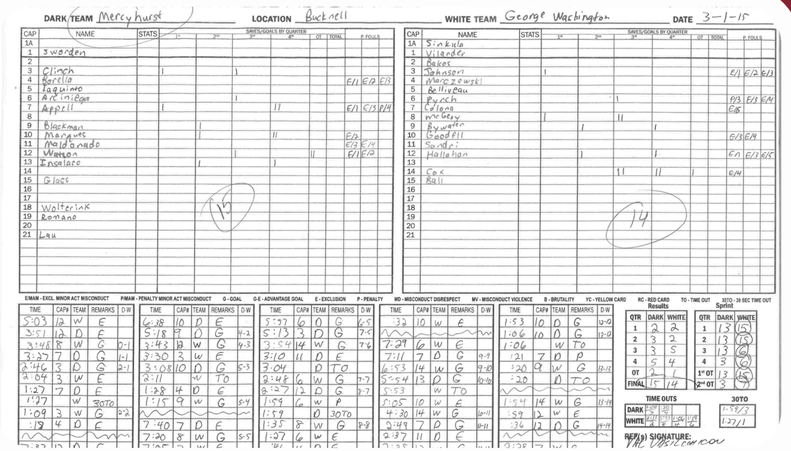
TIME	CAP#	TE
5:57	6	
5:13	3	
3:54	14	
3:10	11	
3:04		
2:48	6	
2:27	12	
1:59	6	
1:59		
1:35	8	
1:27	6	
:46	4	
:44	14	

Introduction

Tracking Statistics

During a water polo game, a group of people (table):

- start + stop the clock
- update the scoreboard with scores and penalties
- keep an official scoresheet



Scoresheet

- time and location of the game
- rosters for both teams
- game log -- running record of the time and details for every major event

Problem

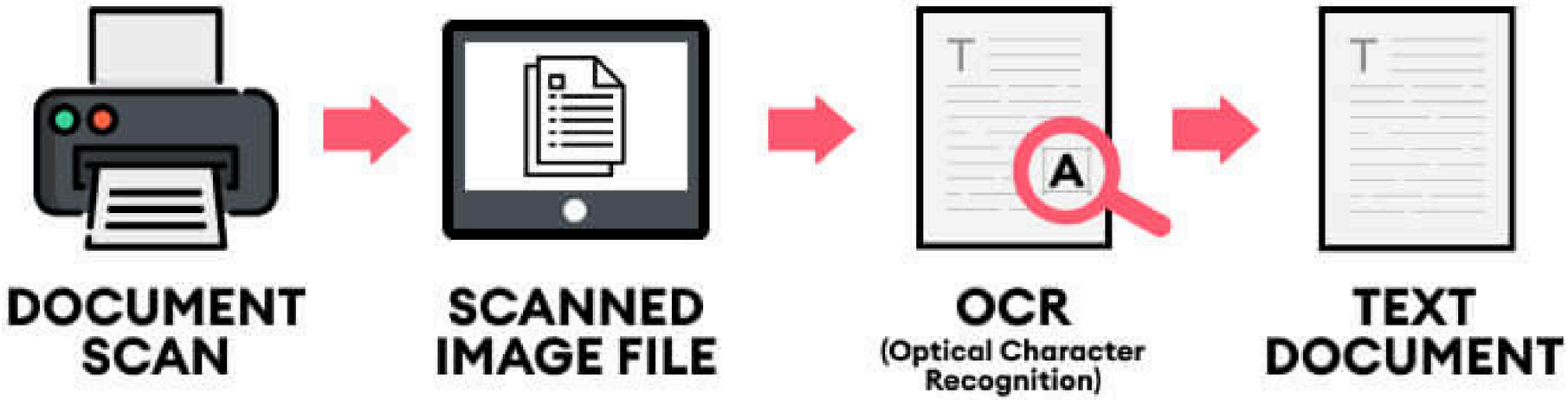
- often scoresheets / stats forgotten
- only way to track stats, especially in leagues without funding is by manually typing in data
- could a photo be automatically turned into text?

Related Works

Optical Character Recognition (OCR) is the process that converts an image of text into a machine-readable text format.

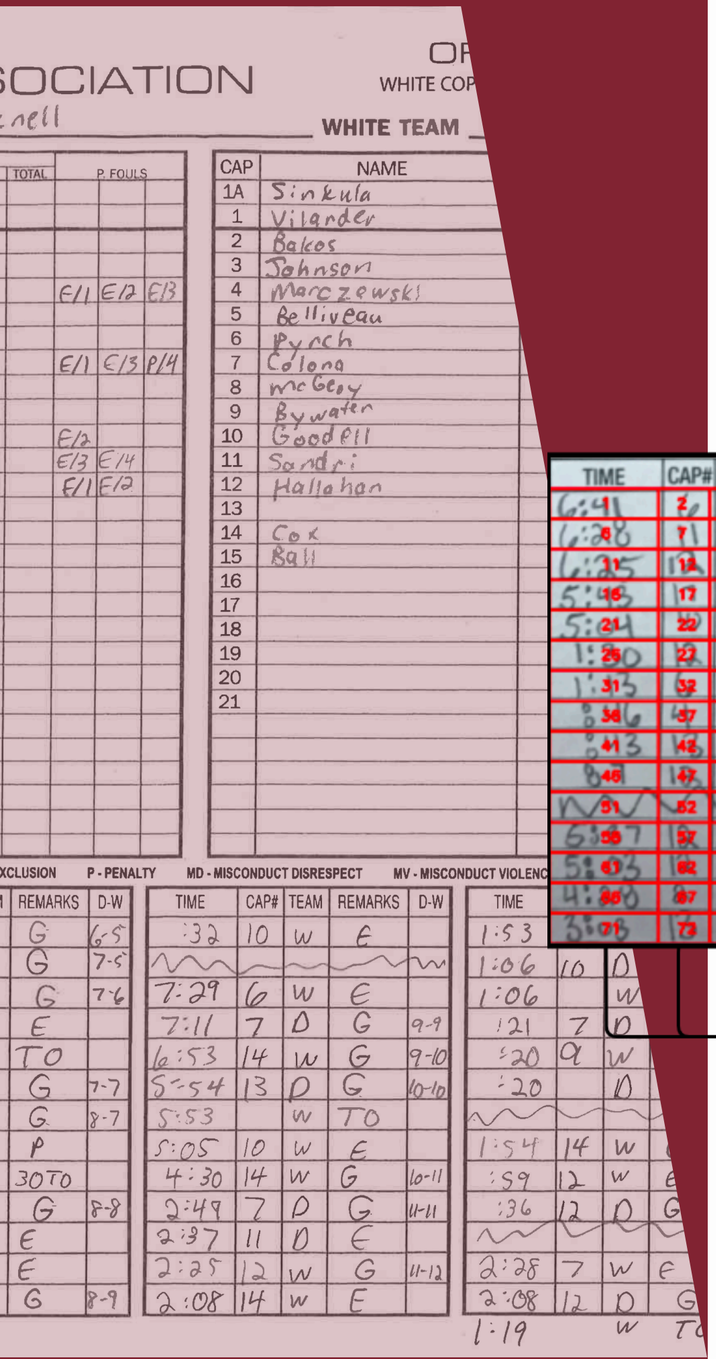
Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994. [1]

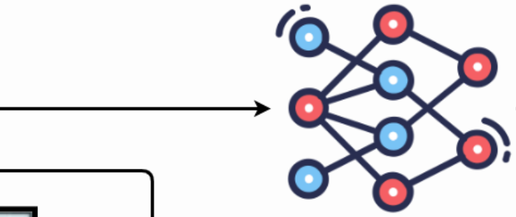
The Tesseract classifier was trained on 20 samples of 94 characters from 8 fonts with 4 attributes (normal, bold, italic, bold italic), making a total of 60160 training samples. --> no handwritten training data



[1] R. Smith, "An Overview of the Tesseract OCR Engine," Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, 2007, pp. 629-633, doi: 10.1109/ICDAR.2007.4376991.

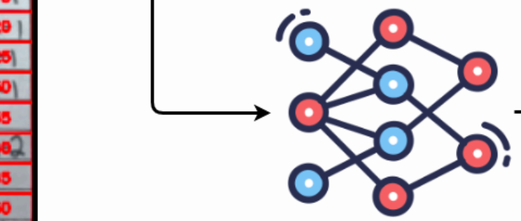
Proposed Method





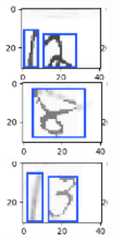
TEAM MODEL

D | W | -----



REMARKS MODEL

G-goal	TO-timeout
E-ejection	C-card
P-penalty	-----



CONTOUR DETECTION

MNIST MODEL

0	1	2	3	4
5	6	7	8	9

0	1	2	3	4
5	6	7	8	9

...

0	1	2	3	4
5	6	7	8	9

ROW	TIME	CAP	TEAM	REMARKS	SCORE
1	270	12	D	E-ejection	-----
2	2	11	W	E-ejection	-----
3	458	-----	W	TO-timeout	-----
4	9	2	W	G-goal	51
5	218	-----	D	TO-timeout	-----
6	21	9	W	G-goal	42
7	558	3	D	E-ejection	-----
8	-----	-----	-----	-----	-----
9	21	4	W	E-ejection	-----
10	85	9	W	G-goal	03
11	83	8	W	G-goal	04
12	43	-----	D	TO-timeout	-----
13	13521	6	W	E-ejection	-----
14	122	-----	W	30TO-30secondtimeout	-----
15	122	4	W	G-goal	-----
16	-----	-----	-----	-----	-----
17	244	4	W	G-goal	06
18	78	5	W	E-ejection	-----

...

Dataset and Metrics

Scoresheets

37 scoresheets
34 train/test/validation
3 not in dataset

metric: accuracy of correctly classified cells in 3 scoresheets not in the dataset

Team

Train Size: 1200
Test Size: 400
Validation Size: 400

Classes: [D , W , empty]
Shape: 28x28 grayscale

Remarks

Train Size: 1200
Test Size: 400
Validation Size: 400

Classes: [G , E , P , TO , C, empty]
Shape: 73x28 grayscale

MNIST

Train Size: 50000
Test Size: 10000
Validation Size: 10000

Classes: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Shape: 28x28 binary

Implementation Details

01

data_collection.ipynb
convert scoresheet from PDF/JPG --> PNG
select game log corner points with GUI and save to txt file
de-skew game log and save every TEAM + REMARKS cell
export TEAM + REMARKS cells to Roboflow for training

02

Roboflow
import TEAM + REMARKS images into 2 different projects
classify 2000+ images for each dataset, removing bad data
remove images in data-heavy classes (i.e. empty cells)
stretch to desired size, make grayscale, get download link

03

train_classifier.ipynb
load TEAM / REMARKS (Roboflow) or MNIST (PyTorch Datasets)
80% train / 10% test / 10% validation split
convert to dataloaders with batch size 32
train Simple CNN or TinyVGG then evaluate training results

04

wpolo_scoresheet_ocr.ipynb
choose a scoresheet, de-skew game log, separate every cell
cycle through rows and send cell through correct classifier
if MNIST cell, use contour detection to separate numbers
display the output predictions in a table

01

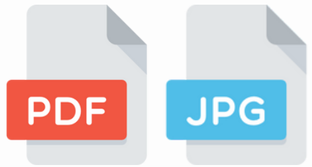
data_collection.ipynb

convert scoresheet from PDF/JPG --> PNG

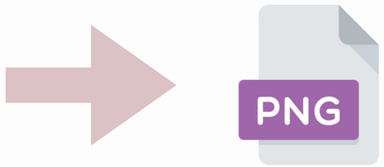
select game log corner points with GUI and save to txt file

de-skew game log and save every TEAM + REMARKS cell

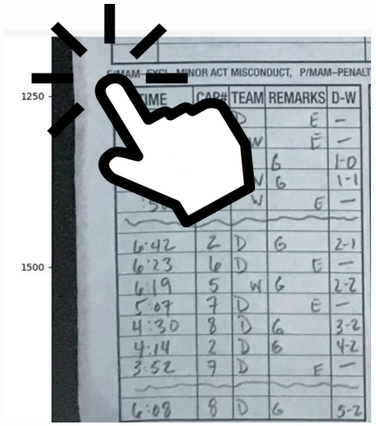
export TEAM + REMARKS cells to Roboflow for training



scoresheet



scoresheet



select corner points with GUI

txt file of game log corner points

straightened game log with every cell separated and numbered

02 **Roboflow**

03 **train_classifier.ipynb**

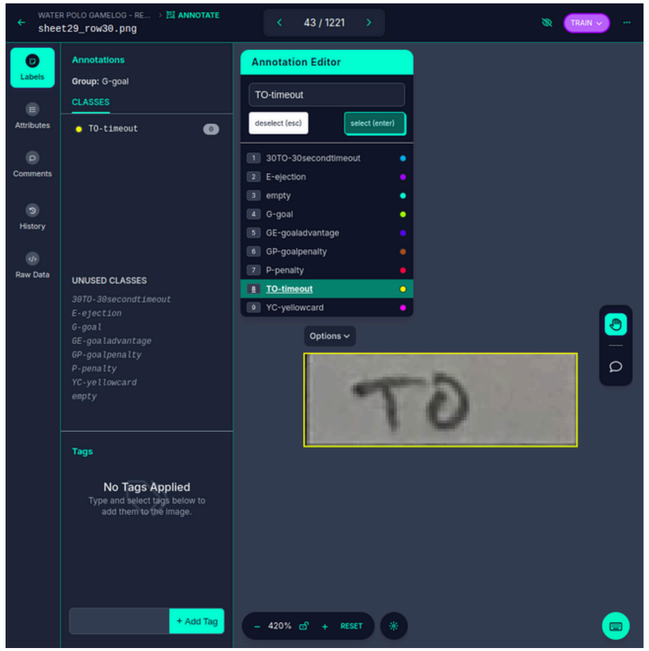
04 **wpolo_scoresheet_ocr.ipynb**

01 data_collection.ipynb

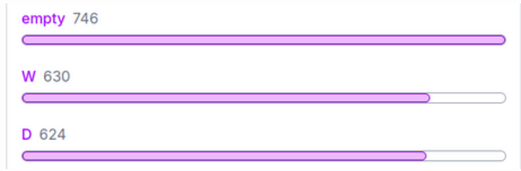
02

Roboflow

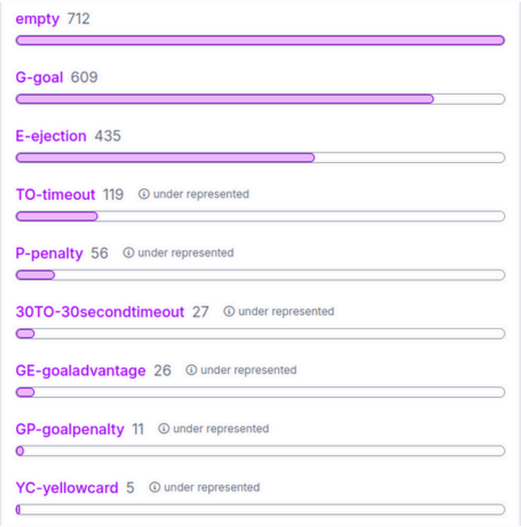
import TEAM + REMARKS images into 2 different projects
classify 2000+ images for each dataset, removing bad data
remove images in data-heavy classes (i.e. empty cells)
stretch to desired size, make grayscale, get download link



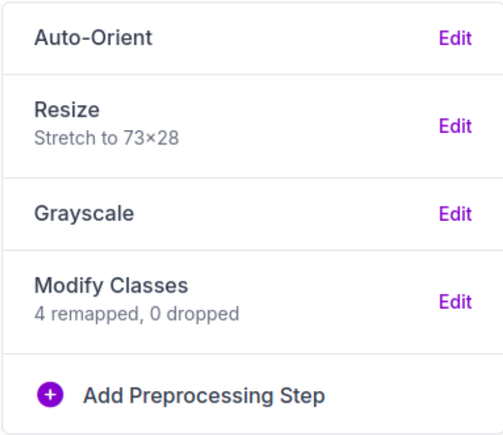
Roboflow labeling interface



TEAM class distribution



REMARKS class distribution



preprocessing choices for REMARKS dataset

03 train_classifier.ipynb

04 wpolo_scoresheet_ocr.ipynb

01 **data_collection.ipynb**

02 **Roboflow**

03

train_classifier.ipynb

load TEAM / REMARKS (Roboflow) or MNIST (PyTorch Datasets)

80% train / 10% test / 10% validation split

convert to dataloaders with batch size 32

train Simple CNN or TinyVGG then evaluate training results

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
TeamModel                                [1, 10]                   --
├─Conv2d: 1-1                            [1, 3, 28, 73]            30
├─MaxPool2d: 1-2                         [1, 3, 14, 36]            --
├─Conv2d: 1-3                            [1, 16, 10, 32]           1,216
├─MaxPool2d: 1-4                         [1, 16, 5, 16]            --
├─Linear: 1-5                            [1, 120]                  153,720
├─Linear: 1-6                            [1, 84]                   10,164
├─Linear: 1-7                            [1, 10]                   850
==========================================================================================
Total params: 165,980
Trainable params: 165,980
Non-trainable params: 0
Total mult-adds (M): 0.62
==========================================================================================
Input size (MB): 0.01
Forward/backward pass size (MB): 0.09
Params size (MB): 0.66
Estimated Total Size (MB): 0.76
==========================================================================================
```

Simple CNN Model

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
TinyVGG                                  [1, 3]                    --
├─Sequential: 1-1                        [1, 10, 14, 14]           --
│    └─Conv2d: 2-1                       [1, 10, 28, 28]           100
│    └─ReLU: 2-2                         [1, 10, 28, 28]           --
│    └─Conv2d: 2-3                       [1, 10, 28, 28]           910
│    └─ReLU: 2-4                         [1, 10, 28, 28]           --
│    └─MaxPool2d: 2-5                    [1, 10, 14, 14]           --
├─Sequential: 1-2                        [1, 10, 7, 7]             --
│    └─Conv2d: 2-6                       [1, 10, 14, 14]           910
│    └─ReLU: 2-7                         [1, 10, 14, 14]           --
│    └─Conv2d: 2-8                       [1, 10, 14, 14]           910
│    └─ReLU: 2-9                         [1, 10, 14, 14]           --
│    └─MaxPool2d: 2-10                   [1, 10, 7, 7]             --
├─Sequential: 1-3                        [1, 3]                    --
│    └─Flatten: 2-11                     [1, 490]                  --
│    └─Linear: 2-12                      [1, 3]                    1,473
==========================================================================================
Total params: 4,303
Trainable params: 4,303
Non-trainable params: 0
Total mult-adds (M): 1.15
==========================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.16
Params size (MB): 0.02
Estimated Total Size (MB): 0.18
==========================================================================================
```

TinyVGG Model

04 **wpolo_scoresheet_ocr.ipynb**

01 data_collection.ipynb

02 Roboflow

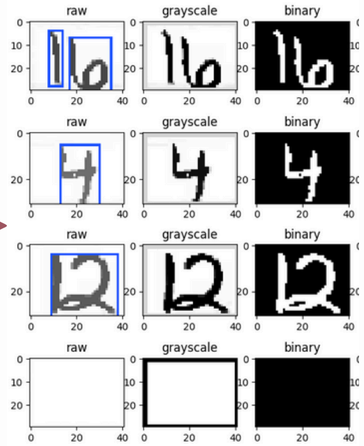
03 train_classifier.ipynb

04

wpolo_scoresheet_ocr.ipynb

choose a scoresheet, de-skew game log, separate every cell
cycle through rows and send cell through correct classifier
if MNIST cell, use contour detection to separate numbers
display the output predictions in a table

- grayscale
- binary (threshold=230)
- dilate (iterations=2)
- dilate (top+bottom only)
- erode
- find contours
- resize numbers
- pad numbers



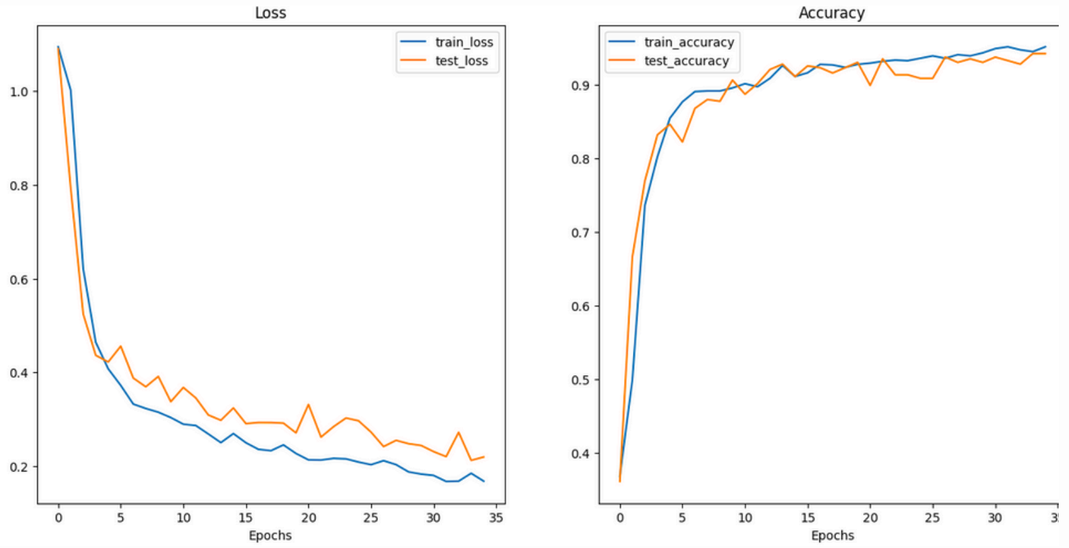
process to prepare the digits in a TIME, CAP, or SCORE column to be classified by MNIST

ROW	TIME	CAP	TEAM	REMARKS	SCORE
1	270	12	D	E-ejection	-----
2	2	11	W	E-ejection	-----
3	458	-----	W	TO-timeout	-----
4	9	2	W	G-goal	51
5	218	-----	D	TO-timeout	-----
6	21	9	W	G-goal	42
7	558	3	D	E-ejection	-----
8	-----	-----	-----	-----	-----
9	21	4	W	E-ejection	-----
10	85	9	W	G-goal	03
11	83	8	W	G-goal	04
12	43	-----	D	TO-timeout	-----
13	13521	6	W	E-ejection	-----
14	122	-----	W	30TO-30secondtimeout	-----
15	122	4	W	G-goal	-----
16	-----	-----	-----	-----	-----
17	244	4	W	G-goal	06
18	78	5	W	E-ejection	-----

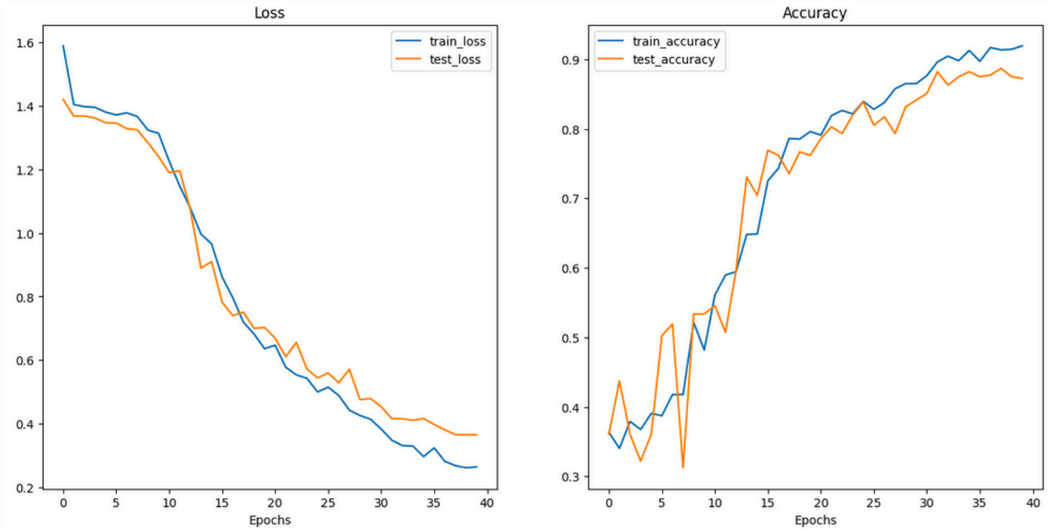
output from the wpolo_scorehsset_ocr Python notebook

Experimental Results

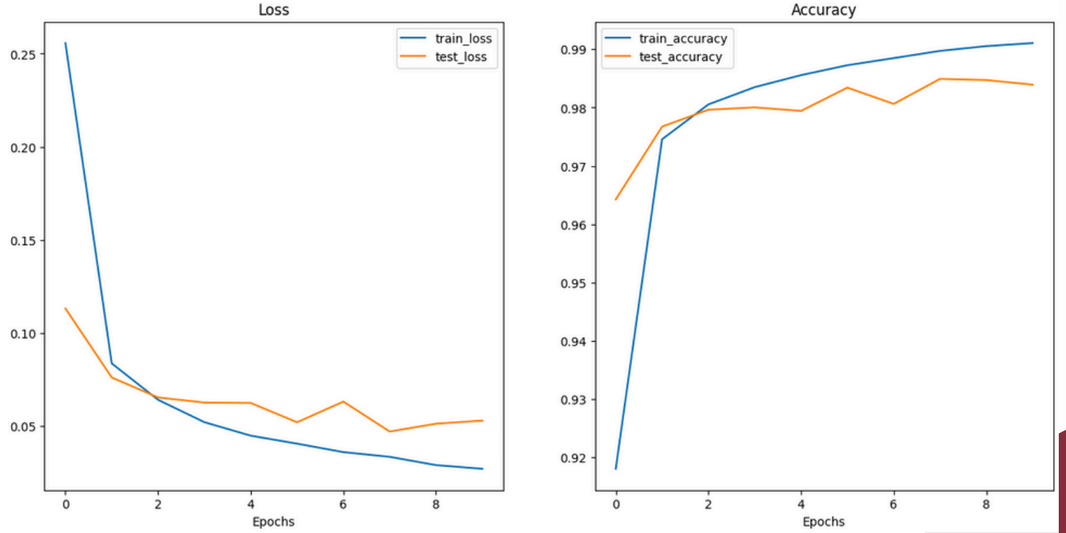
the model and the number of epochs were chosen for each dataset based on the validation accuracy and the loss/accuracy curves, paying attention to avoid overfitting



TEAM: TinyVGG with 35 epochs
validation accuracy = 93.75%



REMARKS: Simple CNN with 40 epochs
validation accuracy = 86.25%



MNIST: TinyVGG with 10 epochs
validation accuracy = 98.70%

Experimental Results

scoresheet rows = 40 + 45 + 35 = 120
total cells = 120 * 5 = 600

23%
TIME accuracy
correct: 11 + 11 + 6

49%
CAP accuracy
correct: 22 + 22 + 15

93%
TEAM accuracy
correct: 36 + 43 + 33

88%
REMARKS accuracy
correct: 30 + 44 + 31

58%
SCORE accuracy
correct: 21 + 27 + 22

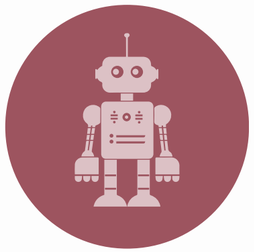
62%
TOTAL accuracy
correct: 120 + 147 + 107

ROW	TIME	CAP	TEAM	REMARKS	SCORE
1	108	1	W	E-ejection	1
2	8	11	W	E-ejection	-----
3	123	12	D	G-goal	11
4	54	17	D	G-goal	21
5	55	0	D	G-goal	31
6	132	12	D	G-goal	41
7	113	6	D	E-ejection	-----
8	44	4	W	G-goal	42
9	24	13	W	E-ejection	-----
10	27	12	D	G-goal	-----
11	0	2	-----	G-goal	0
12	5827	2	D	G-goal	62
13	20	12	D	G-goal	0
14	24	8	D	G-goal	07
15	836	11	-----	G-goal	82
16	0	-----	D	G-goal	52
17	120	11	W	E-ejection	-----
18	126	12	W	G-goal	02
19	2	9	W	E-ejection	23
20	1	0	D	-----	2
21	459	11	W	G-goal	24
22	544	12	D	G-goal	84
23	585	8	D	E-ejection	-----
24	2	11	W	G-goal	115
25	240	6	D	G-goal	05
26	24	10	W	P-penalty	-----
27	2	12	D	E-ejection	5
28	11	8	D	G-goal	45
29	08	2	D	G-goal	-----
30	800	-----	W	P-penalty	-----
31	02	2	W	E-ejection	-----
32	1	11	W	E-ejection	-----
33	19	3	D	G-goal	55
34	-----	-----	-----	-----	-----
35	609	1	W	E-ejection	-----
36	609	5	D	G-goal	65
37	543	12	W	TO-timeout	-----
38	405	21	D	G-goal	175
39	211	2	W	G-goal	76
40	2	6	W	-----	4

ROW	TIME	CAP	TEAM	REMARKS	SCORE
1	-----	-----	D	E-ejection	-----
2	7	0	W	G-goal	21
3	-----	-----	D	G-goal	-----
4	-----	-----	D	G-goal	1
5	-----	-----	W	G-goal	22
6	-----	-----	D	G-goal	2
7	-----	-----	-----	E-ejection	-----
8	-----	-----	D	G-goal	42
9	-----	-----	D	E-ejection	-----
10	-----	-----	D	G-goal	2
11	-----	2	W	E-ejection	-----
12	1	10	D	G-goal	62
13	-----	-----	D	E-ejection	-----
14	-----	0	W	G-goal	63
15	-----	-----	-----	-----	-----
16	0	10	D	G-goal	32
17	240	14	D	G-goal	83
18	-----	2	D	G-goal	93
19	5	8	D	G-goal	10
20	-----	2	D	E-ejection	-----
21	8	4	W	G-goal	184
22	12	10	D	G-goal	4
23	11	10	D	E-ejection	-----
24	-----	-----	-----	-----	-----
25	02	14	D	G-goal	524
26	940	10	W	E-ejection	-----
27	850	6	D	G-goal	134
28	400	5	D	G-goal	84
29	340	13	-----	E-ejection	-----
30	380	5	-----	G-goal	154
31	28	1	D	E-ejection	-----
32	226	3	D	G-goal	64
33	200	2	D	G-goal	14
34	110	2	D	E-ejection	-----
35	43	8	D	E-ejection	-----
36	20	15	D	G-goal	184
37	2	4	W	G-goal	-----
38	-----	-----	-----	-----	-----
39	450	13	W	E-ejection	-----
40	120	16	D	G-goal	114
41	81	11	D	E-ejection	-----
42	436	3	D	G-goal	24
43	900	4	D	G-goal	44
44	301	5	D	G-goal	28
45	40	0	D	E-ejection	4

ROW	TIME	CAP	TEAM	REMARKS	SCORE
1	-----	-----	W	G-goal	01
2	3	5	W	E-ejection	-----
3	2	3	W	G-goal	-----
4	0	-----	W	E-ejection	-----
5	05	2	D	G-goal	2
6	-----	-----	-----	-----	-----
7	2	-----	D	E-ejection	-----
8	105	5	W	G-goal	13
9	3	-----	W	E-ejection	-----
10	231	01	D	G-goal	23
11	235	6	D	E-ejection	-----
12	00	2	W	G-goal	24
13	045	10	W	E-ejection	-----
14	72	4	D	G-goal	34
15	57	4	D	G-goal	44
16	4	1	D	G-goal	4
17	-----	0	W	TO-timeout	-----
18	-----	-----	-----	G-goal	-----
19	129	10	D	E-ejection	-----
20	46	-----	W	E-ejection	-----
21	-----	4	D	G-goal	64
22	0	7	W	G-goal	60
23	74	2	D	G-goal	3
24	748	8	D	G-goal	05
25	20	8	W	G-goal	86
26	01	01	D	G-goal	86
27	-----	-----	-----	-----	-----
28	01	5	W	G-goal	13
29	22	2	D	G-goal	105
30	416	11	D	G-goal	153
31	-----	5	-----	E-ejection	18
32	39	13	-----	E-ejection	-----
33	22	-----	D	---	08
34	-----	10	W	G-goal	29
35	23	11	D	E-ejection	-----

Conclusion and Future Work



Game Log Corner Pts

- automatically detect corner pts
- ability to work on more scoresheet formats

Increase Accuracy

- test more deep learning models
- use logic in the model (match SCORE w/ goal)
- research better methods for contour detection

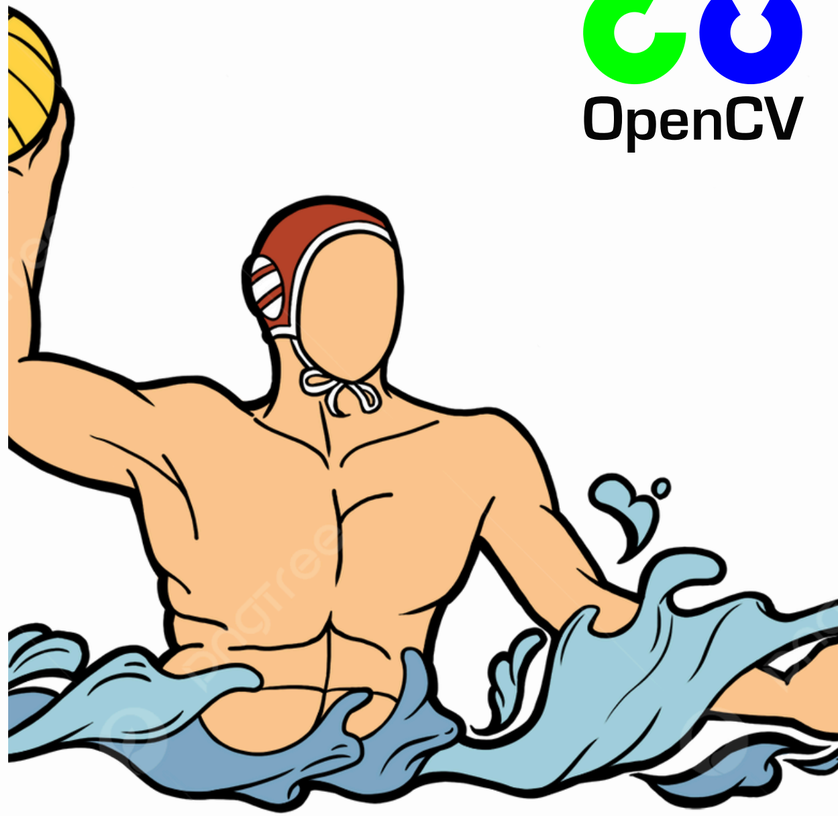
More Balanced Data

- collect more samples (MNIST was clearly the most accurate and it had the most samples)
- balance training classes (less empty cells, more penalties)

SAPIENZA
UNIVERSITÀ DI ROMA

OpenCV

THANK YOU!



Github and Dataset Links

https://github.com/cjkreienkamp/wpolo-scoresheet-ocr

https://app.roboflow.com/chris-kreienkamp/water-polo-gamelog-team/3

https://app.roboflow.com/chris-kreienkamp/water-polo-gamelog-remarks/2

https://pytorch.org/vision/main/generated/torchvision.datasets.MNIST.html