COMPUTER VISION 23/24

Gerardo Paladino 1918178

SEMI-AUTOMATIC OFFSIDE

SAPIENZA
UNIVERSITÀ DI ROMA

Offside

- In the game of football, offside is a position where a player **is beyond the last opposing defender** at the moment the ball is passed, unless they are in their own half.
- This rule aims to prevent players from staying close to the opponent's goal, waiting for easy scoring opportunities.

Semi-Automatic Offside

- Semi-automatic offside in soccer uses advanced technology to assist referees in accurately and quickly detecting offside situations by tracking player and ball positions with sensors and cameras.
- This technology reduces errors and speeds up decisions, introduced in the Italian football championship from the second half of the 2022/2023 season.

Solutions

- **COMPUTER VISION ORIENTED SOLUTIONS**

Players detection

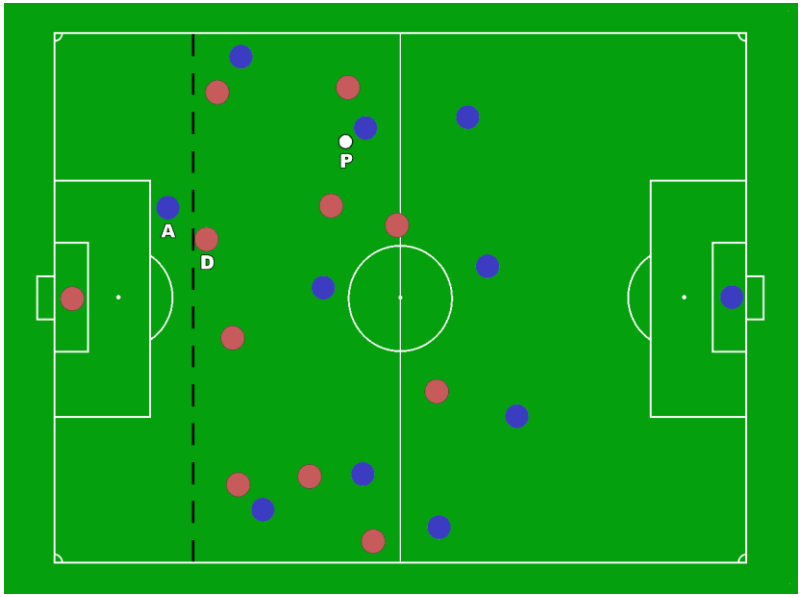
- **DEEP LEARNING ORIENTED SOLUTIONS**

Convolutional Neural Networks

Dataset Idea

The idea comes from news that most stadiums have added the dynamic camera,
a remote-controlled camera attached by wires to the roof of the stadium that collects an overhead view of the entire match and the entire field.

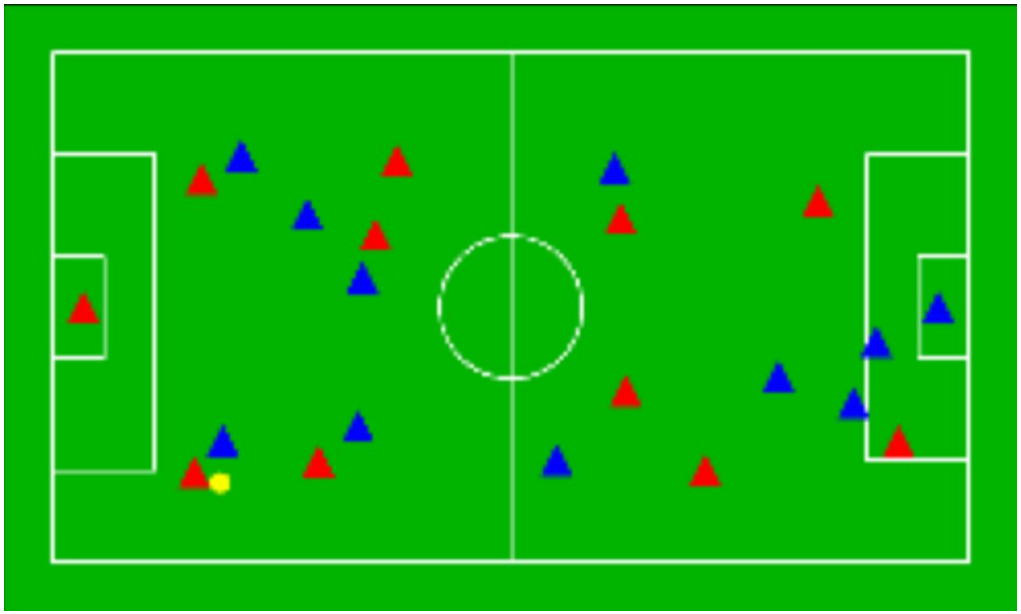
This camera, among other things, provides an instantaneous representation
of all players on the pitch at any moment
of the match.



Dataset

CV Solutions

- **Green background**: The pitch
- **White lines**: Field’s lines
- **Red triangles**: Attacking team
- **Blue triangles**: Defending team
- **Yellow circle**: The ball

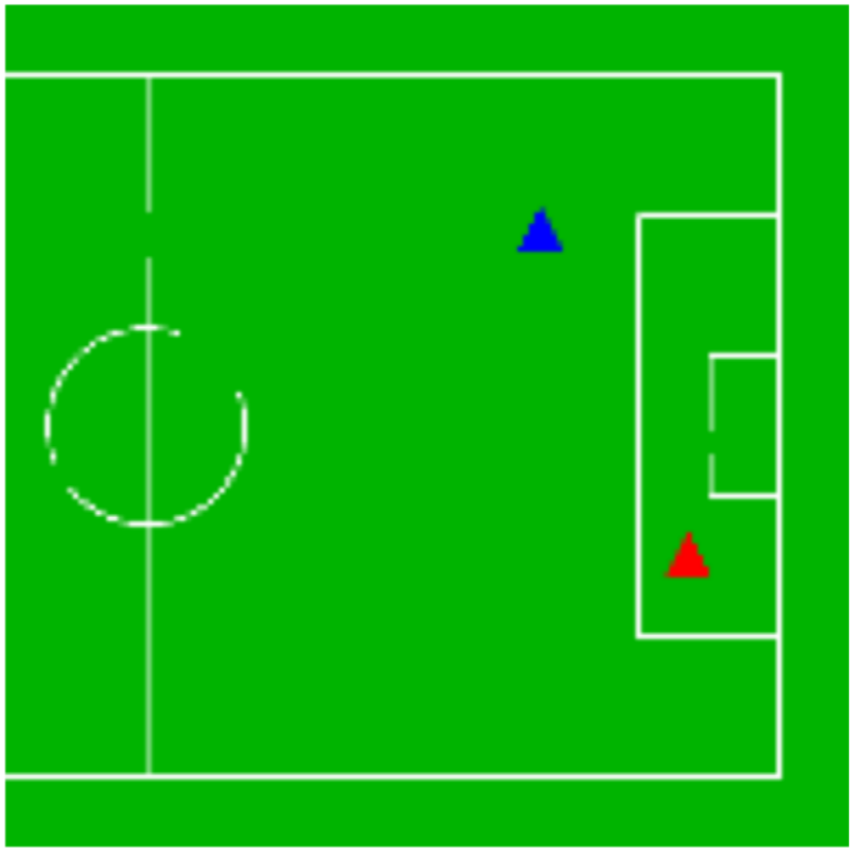


Dataset

DL Solutions (output of CV-based is input of DL-based)

The CV solutions correctly detect the players, the teams, and the offside/inside. It is used to modify the images for DL solutions, so:

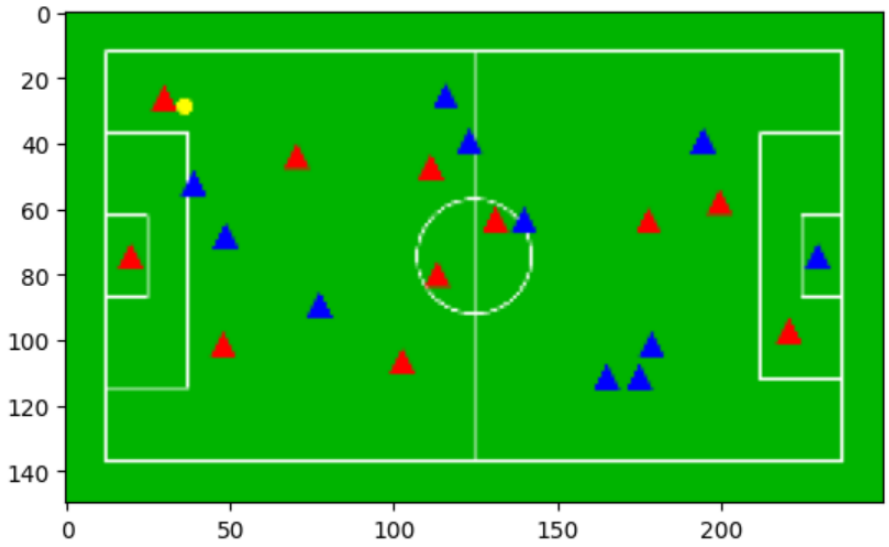
- All the players not important are removed.
- Only the last striker and the last defender remaining in the pitch.
- The image is cropped to make it square.



Computer Vision oriented solution

5 steps

1. Input image

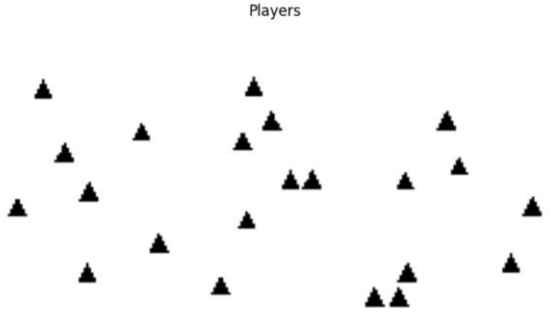


- Resize image: from 1000x600 to 250x150

Computer Vision oriented solution

1. Input image
2. **Players position detected**

Result of threshold function:



- cv2.threshold() to detect the triangles
- Verify if the contours found are triangles
- Save triangles coordinates

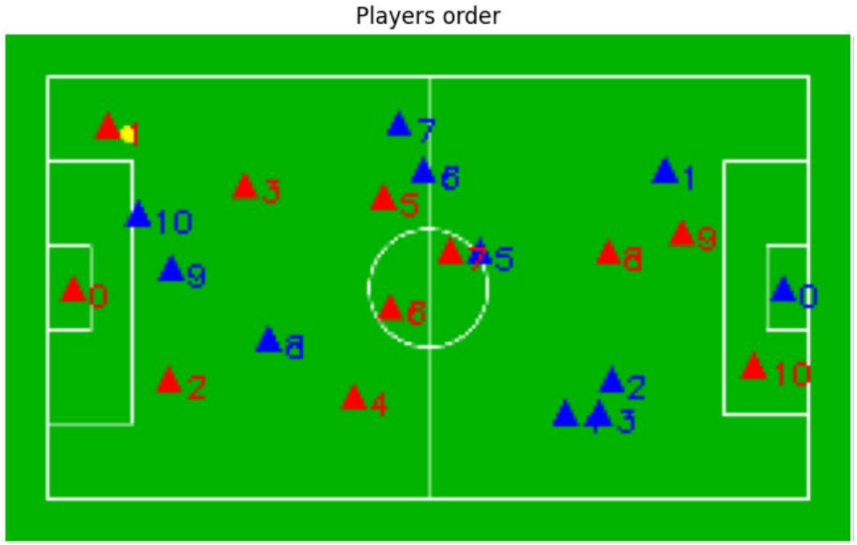
Computer Vision oriented solution

1. Input image
2. Players position detected
3. **Players teams detected**

- Iterate on the players coordinates
- Detect color area
- Split the players positions in the two teams

Computer Vision oriented solution

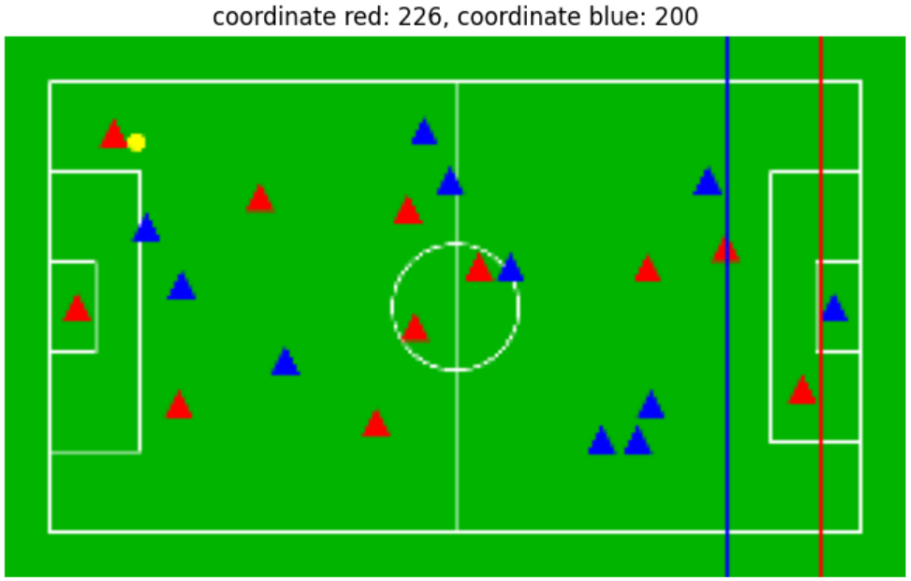
1. Input image
2. Players position detected
3. Players teams detected
4. **Players order detected**



- Sort the two players team lists in their attacking directions

Computer Vision oriented solution

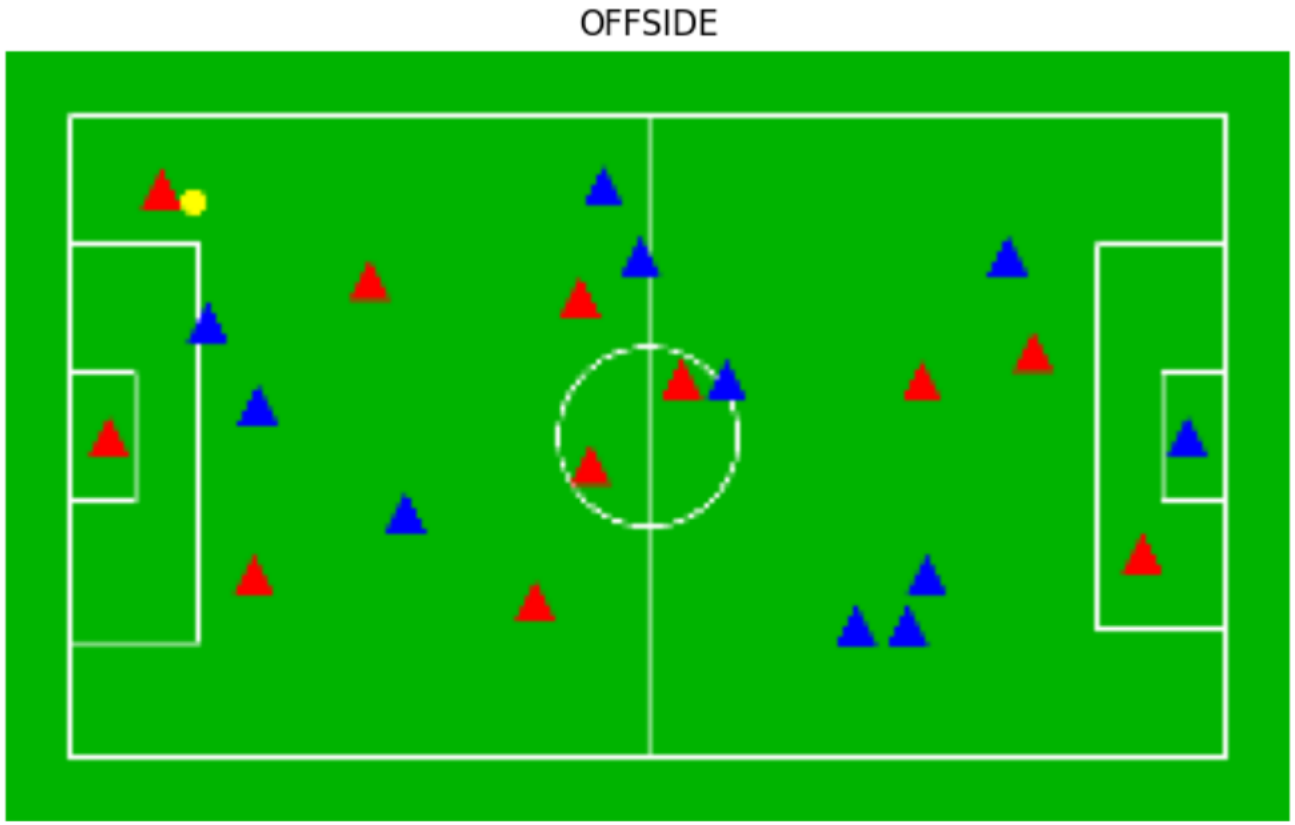
1. Input image
2. Players position detected
3. Players teams detected
4. Players order detected
5. **Offside lines detected**



- Consider the penultimate defender (index 1)
- Consider the last striker (index 10)
- Draw the offside line
- The striker is inside if its x-coordinate is less or equal then the x-coordinate of defender
- Otherwise the striker is in offside

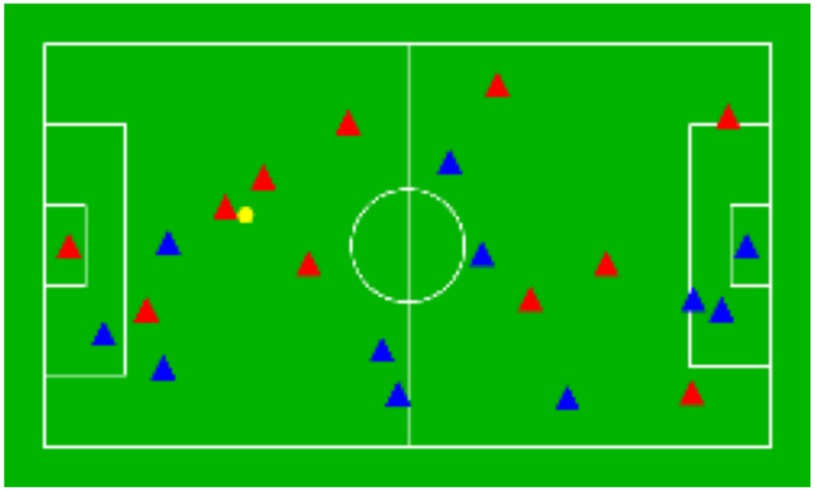
Computer Vision oriented solution

Result

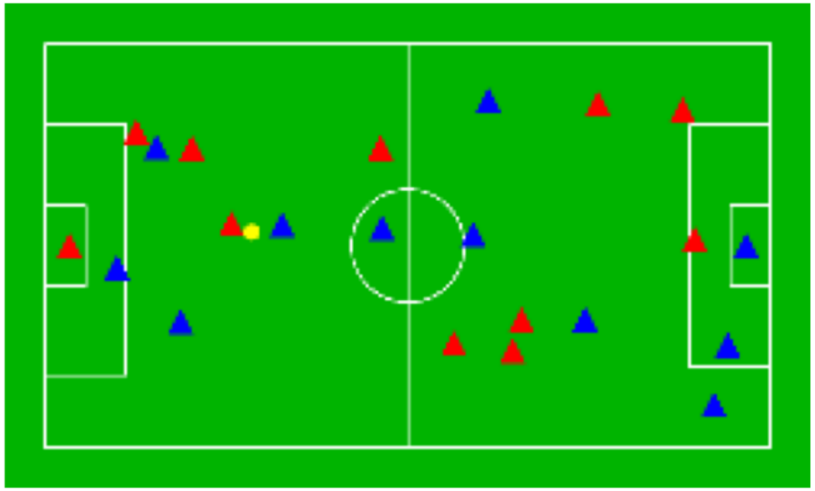


Computer Vision oriented solution

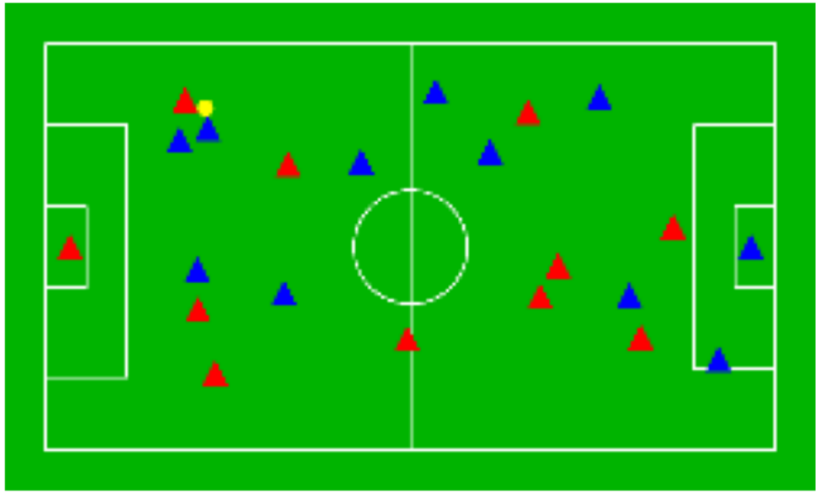
OFFSIDE



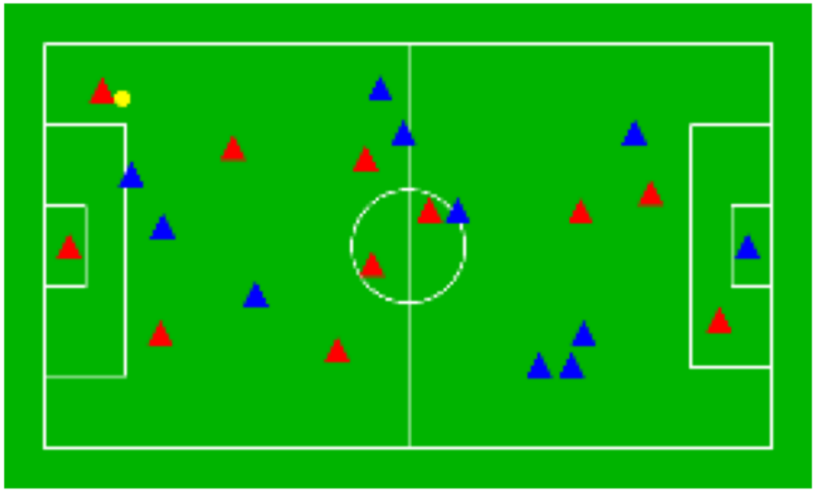
INSIDE



INSIDE



OFFSIDE

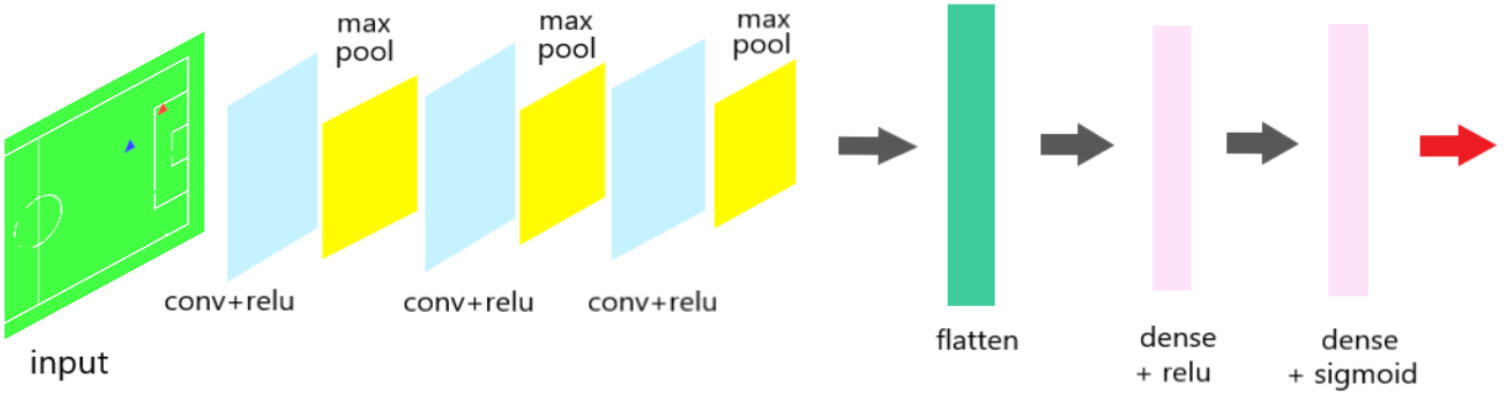


Dataset Distribution

Set	Size	Offside	Inside	Batch Size
Training	8000	3826	4174	256
Validation	1000	487	513	64
Test	1000	481	519	1

Deep Learning Oriented Solution

Extracting features with a CNN to **classify** the image as Offside or Inside

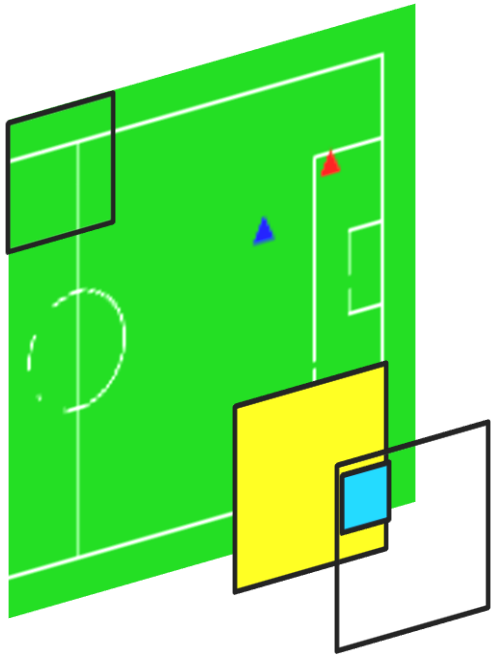


Deep Learning Oriented Solution

Convolutional Layer

- Extracting features
- 3 Conv Layers
- 5x5 filter (F)
- Stride = 1 (S)
- No pad (P)
- Relu

Dimension of the output = $\frac{W-F+2P}{S} + 1$

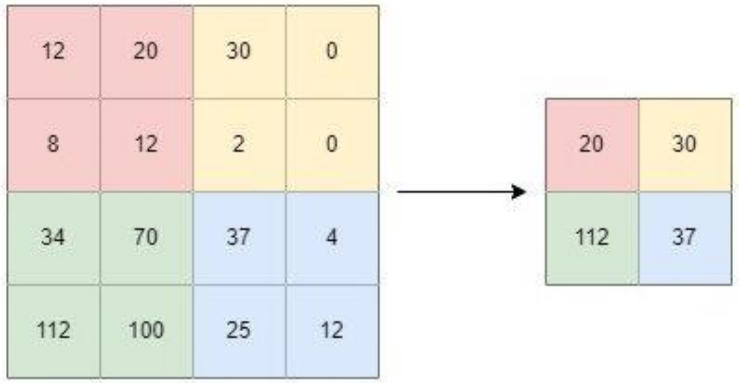


Deep Learning Oriented Solution

(Max) Pooling Layer

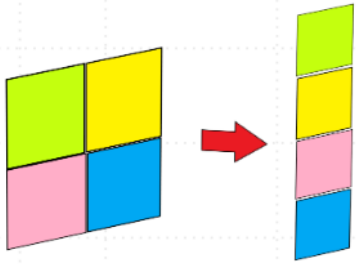
- Reduce spatial dimension
- Preserve important features
- Max Pooling
- 2X2 ($p_f = 2$)

Dimension of the output = w/p_f

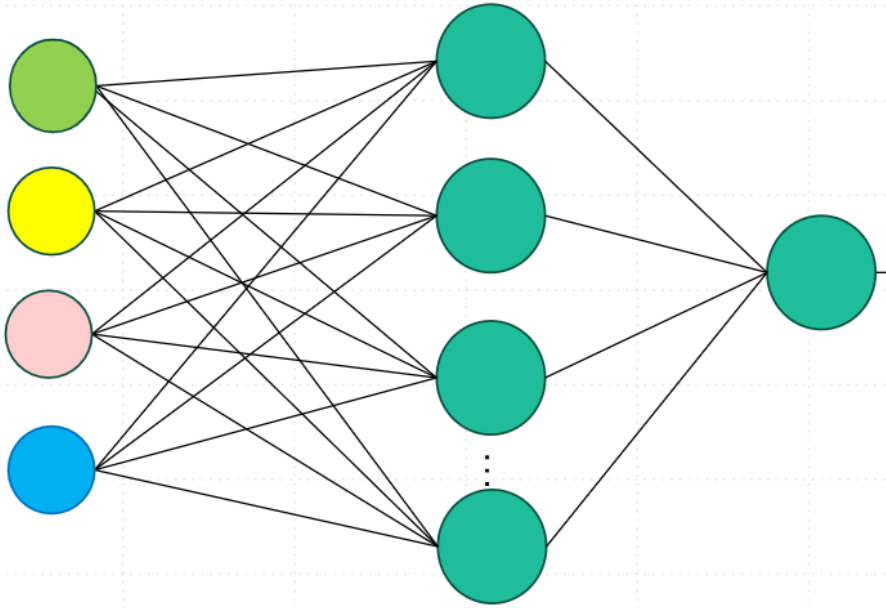


Deep Learning Oriented Solution

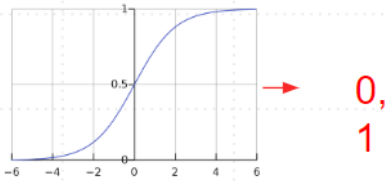
Flatten and Dense Layer



Convert a multidimensional array to a vector



Pass each cell to a NN



Predict the result

Deep Learning Oriented Solution

Parameters settings

Parameters	Value
Epochs	30 -> 50*
Steps per epoch	3
Learning rate	0.001
Optimazer	Adam
Loss	Binary Cross Entropy

*Before and after regularization techniques

Deep Learning Oriented Solution

Regularization: Early Stopping

Early stopping rules provide guidance as to how many iterations can be run before the learner begins to over-fit.

Monitor : val_accuracy
Patience : 5
Restore best weights : True

```
3/3 [==============================] - 22s 10s/step - loss: 0.1666 - accuracy: 0.9297 - val_loss: 0.1596 - val_accuracy: 0.9520
Epoch 31/50
3/3 [==============================] - 21s 9s/step - loss: 0.1633 - accuracy: 0.9466 - val_loss: 0.1608 - val_accuracy: 0.9470
Epoch 32/50
3/3 [==============================] - 28s 13s/step - loss: 0.1348 - accuracy: 0.9609 - val_loss: 0.1567 - val_accuracy: 0.9440
Epoch 33/50
3/3 [==============================] - 20s 9s/step - loss: 0.1329 - accuracy: 0.9518 - val_loss: 0.1592 - val_accuracy: 0.9450
Epoch 34/50
3/3 [==============================] - 21s 9s/step - loss: 0.1144 - accuracy: 0.9518 - val_loss: 0.1690 - val_accuracy: 0.9370
Epoch 35/50
3/3 [==============================] - 21s 10s/step - loss: 0.1356 - accuracy: 0.9453 - val_loss: 0.1462 - val_accuracy: 0.9520
```

Regularization: Data Augmentation

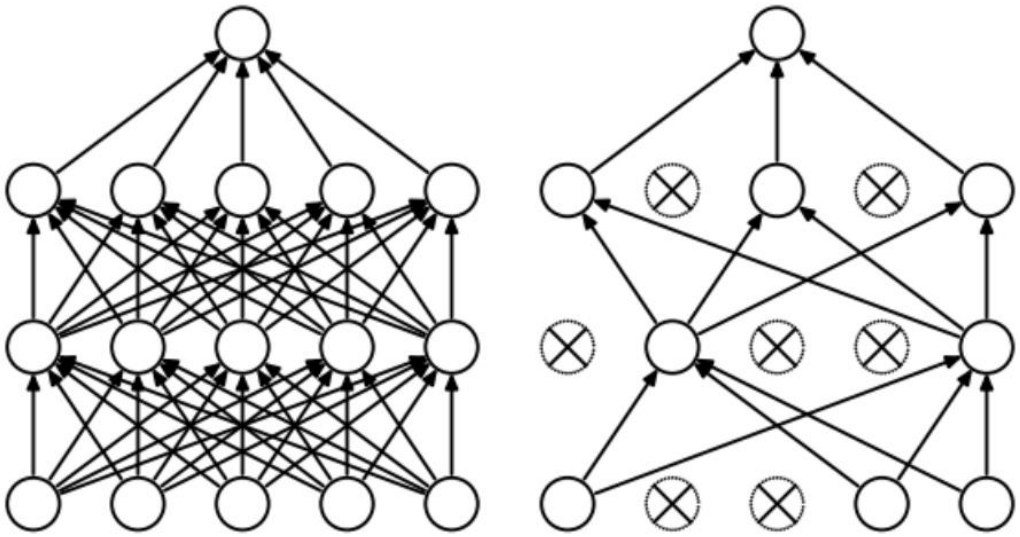
Data augmentation is a statistical technique which allows maximum likelihood estimation from incomplete data.

In this case:

- Rescale = 1/255
- Zoom_range = 0.1
- Brightness_range = [0.9,1.1]
- Fill_mode = ‘nearest’

Regularization: DropOut

"dropout" refers to the practice of disregarding certain nodes in a layer at random during training. It prevents the overfitting. In this case the probability of "turn off" is 50%.



(a) Standard Neural Net (b) After applying dropout.

Deep Learning Oriented Solution

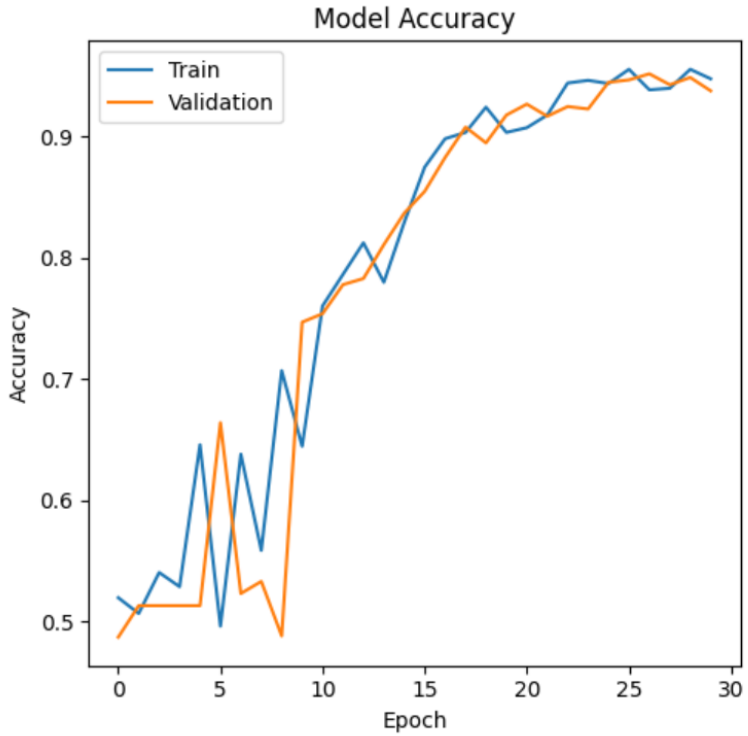
Structure in details

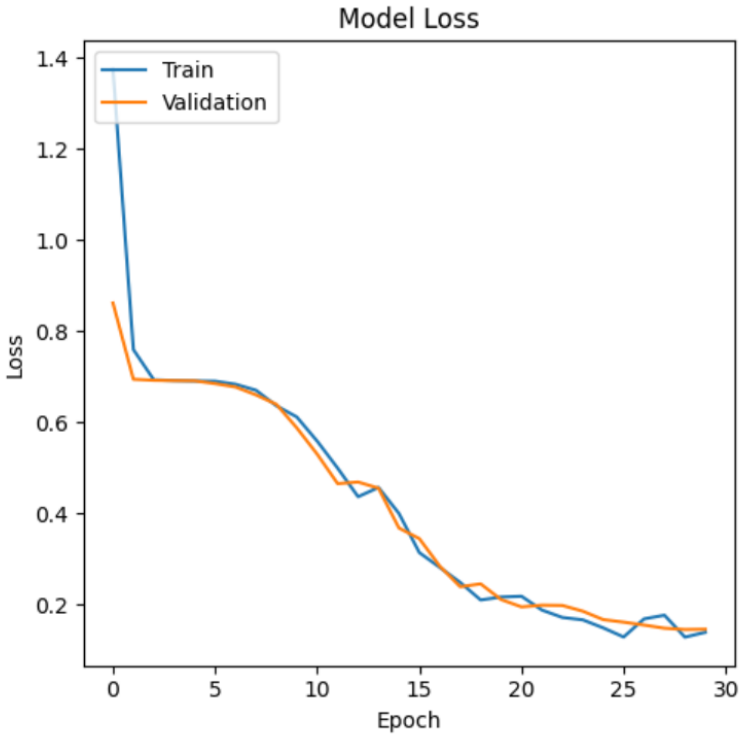
Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_7 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_7 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_8 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_8 (MaxPooling2D)	(None, 17, 17, 128)	0
flatten_2 (Flatten)	(None, 36992)	0
dense_4 (Dense)	(None, 512)	18940416
dropout_2 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 1)	513

Total params: 19034177 (72.61 MB)
Trainable params: 19034177 (72.61 MB)
Non-trainable params: 0 (0.00 Byte)

Deep Learning Oriented Solution

Results before DA and DO (with ES)



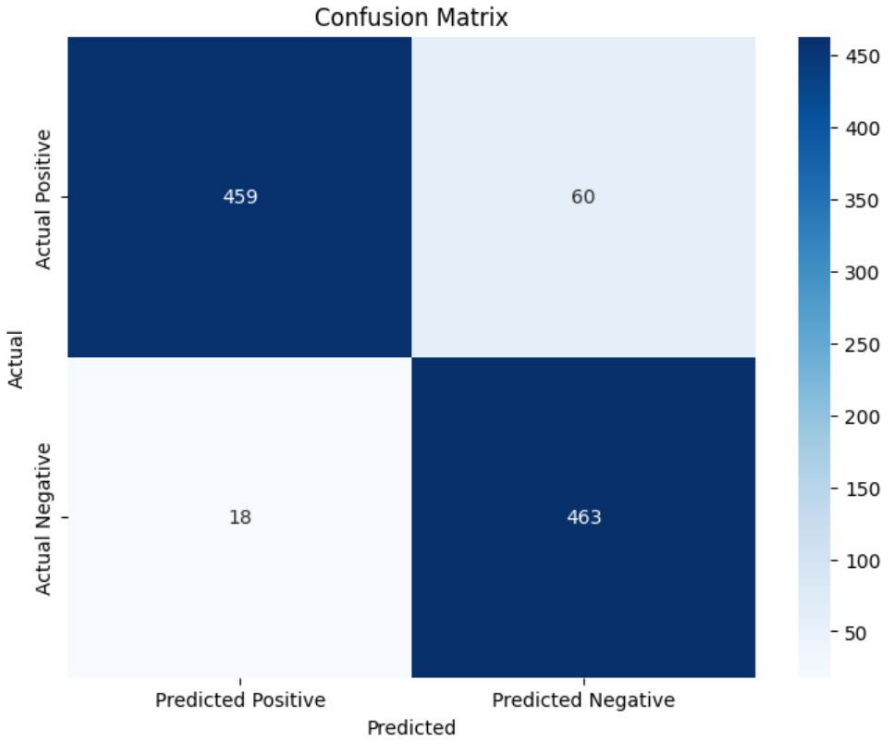


Training Accuracy: 0.9479
Validation Accuracy: 0.9380

Training Loss: 0.1397
Validation Loss: 0.1464

Deep Learning Oriented Solution

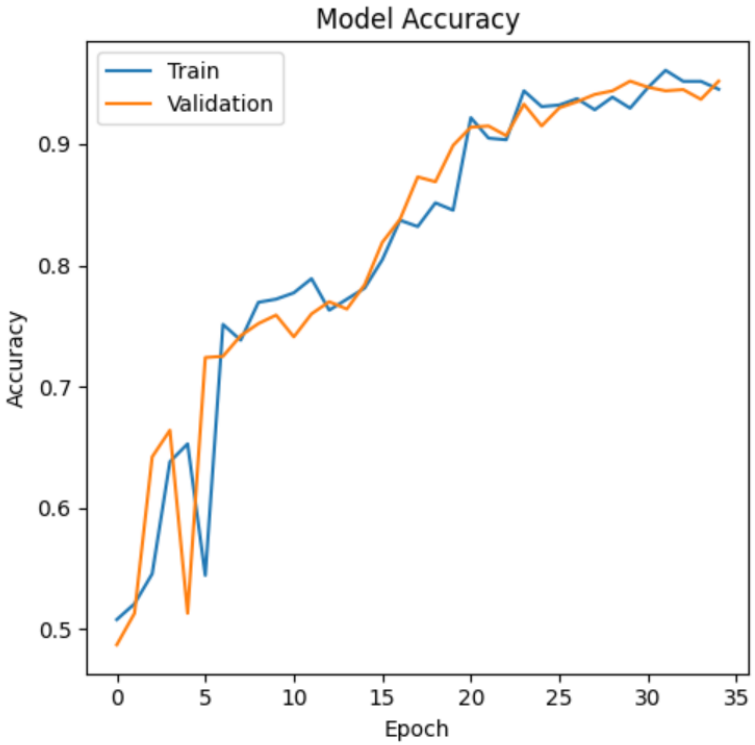
Metrics before DA and DO (with ES)

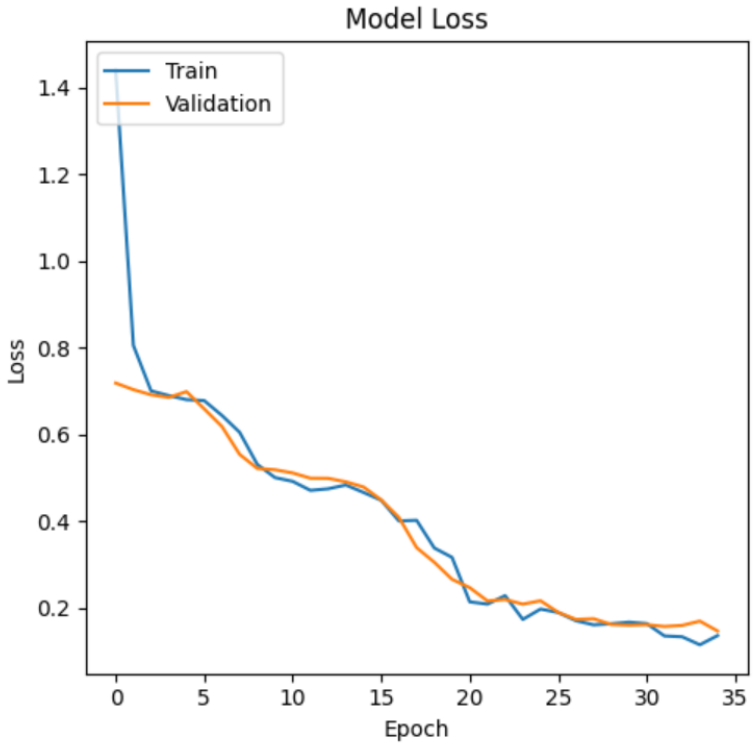


Metric	Value
Test Accuracy	0.922
Precision	0.884
Recall	0.962
F1-score	0.921

Deep Learning Oriented Solution

Results after DA and DO (with ES)



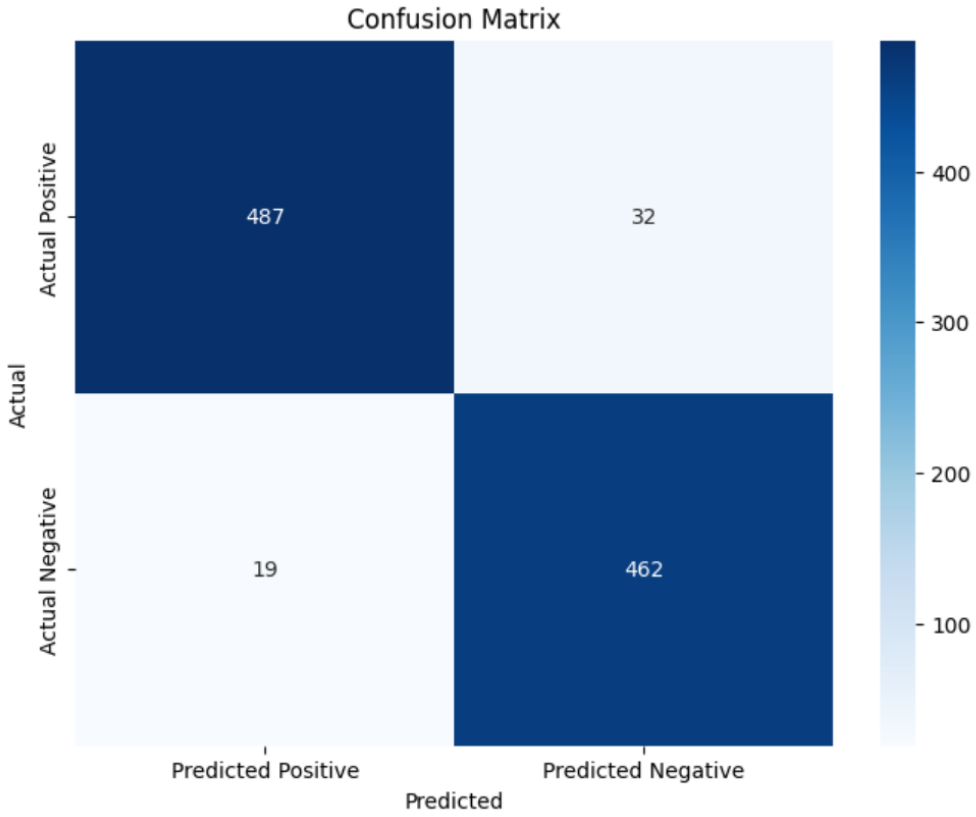


Training Accuracy: 0.9609
Validation Accuracy: 0.9440

Training Loss: 0.1347
Validation Loss: 0.1567

Deep Learning Oriented Solution

Metrics after DA and DO (with ES)



Metric	Value
Test Accuracy	0.949
Precision	0.938
Recall	0.962
F1-score	0.950

Links

[Dataset]

[CV Solution]

[DL Solution]

[GitHub repo]

Thanks for your attention!