DIRECTV - Enhanced Sports Replays

AI Studio Final Presentation

Break Through Tech Los Angeles @UCLA
December 4, 2023

Introductions

Meet Our Team!

Rachel Cheung
UC Berkeley

Joana Fang
UC Los Angeles

Nhu Nguyen
CSU Long Beach

Emi Viernes
CSU Northridge

Our AI Studio TA and Challenge Advisors

Kaifeng Pang
AI Studio TA

Taylor Hosbach
Challenge Advisor

Mia Calhoun Mummert
Challenge Advisor

Presentation Agenda

1. AI Studio Project Overview
2. Data Understanding & Data Preparation
3. Modeling & Evaluation
4. Final Thoughts

AI Studio Project Overview

Identifying Football Players to Enhance Sports Replays

Business Impact

- **70%** of DIRECTV customers watch at least **3 hours of sports each week**
- DIRECTV currently has Sports Mode
    - Enhanced Sports Replays would bring a more **personalized experience**
        - More engagement

Sports Mode

Timeline
Team Comparison
Game Leaders
Enhanced Replays

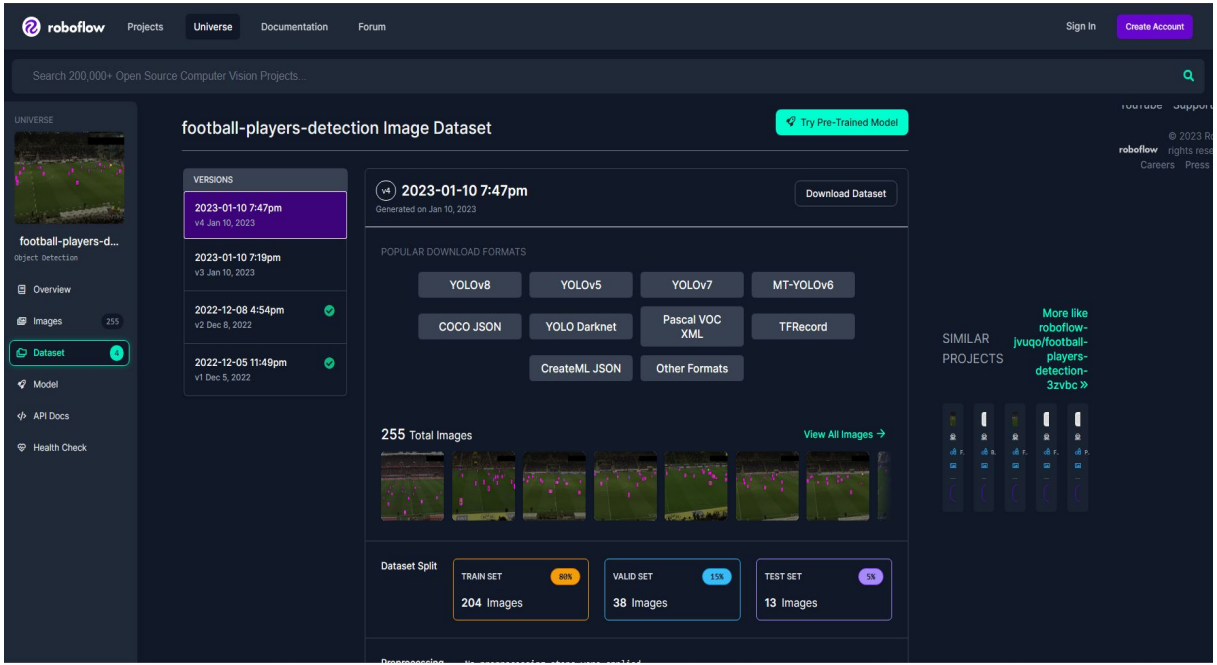
Our Goal

1. Extract information from image data.
2. Finetune pre-trained models.
3. Evaluate model accuracy and select the best model.

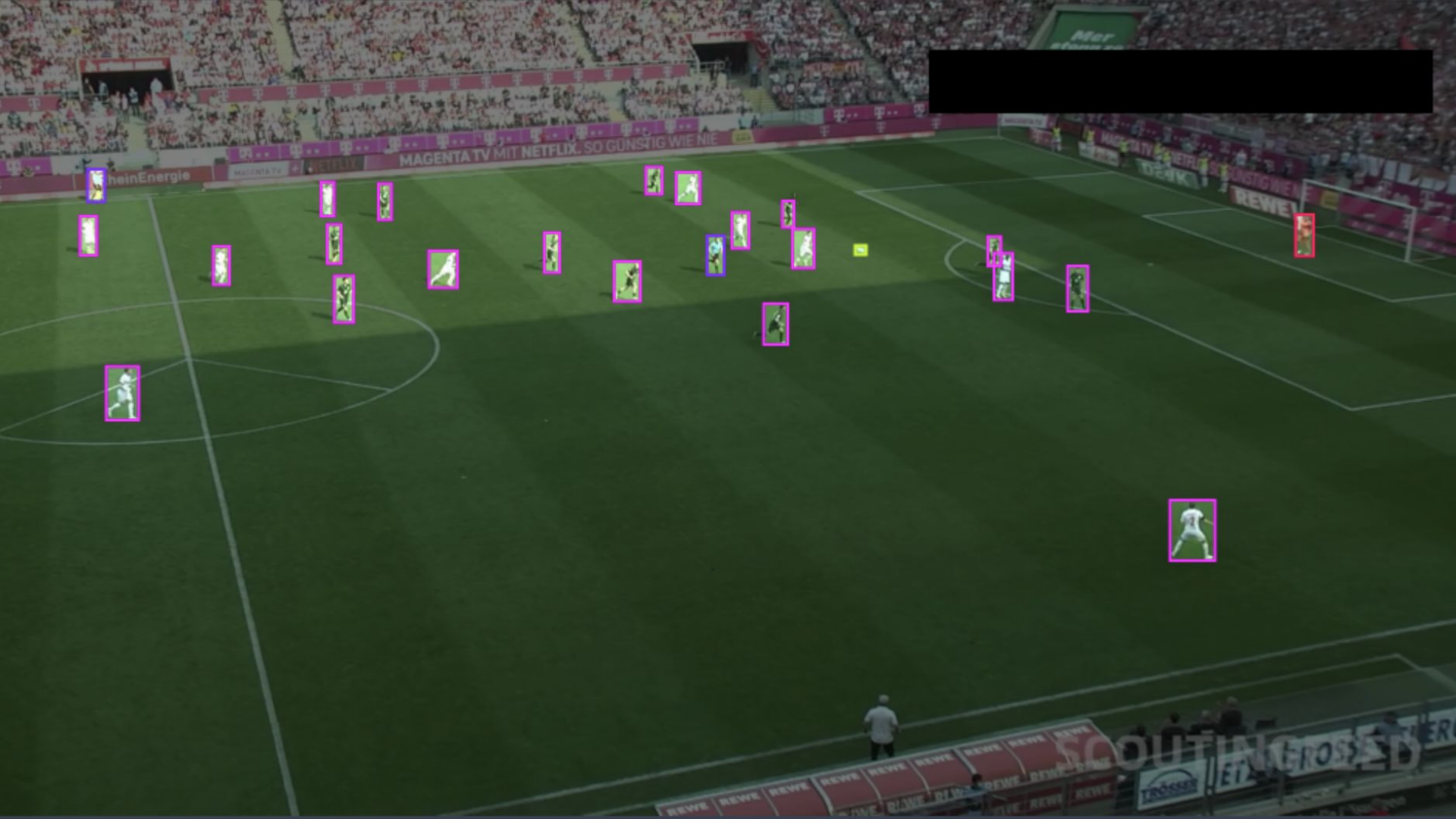
Resources We Leveraged

Data Understanding & Data Preparation

Our Dataset from RoboFlow



- Given by DIRECTV
- Download data format: COCO JSON


MAGENTA TV MIT NETFLIX. SO GÜNSTIG WIE NIE
REWE

JSON Dataset

First half:


```
"categories":[{"id":1,"name":"ball"},{"id":2,"name":"goalkeeper"},
{"id":3,"name":"player"},{"id":4,"name":"referee"}]
,"images":[{"id":0,"license":1,"file_name":"121364_7_8_png.rf.d492
bc91a93a35073d085c5110bc7c10.jpg","height":1280,"width":1280,"date
_captured":"2023-08-07T22:52:26+00:00"},...
```


Second half:


```
"annotations":[{"id":0,"image_id":0,"category_id":2,"bbox":[0,465,
10,49],"area":490,"segmentation":[],"iscrowd":0},{"id":1,"image_id
":0,"category_id":3,"bbox":[520,417,16.5,38],"area":627,"segmentat
ion":[],"iscrowd":0},{"id":2,"image_id":0,"category_id":3,"bbox":[
623,412,13.5,47.5],"area":641.25,"segmentation":[],"iscrowd":0},..
..
```

JSON Dataset

First half:


```
"categories":[{"id":1,"name":"ball"},{"id":2,"name":"goalkeeper"},
{"id":3,"name":"player"},{"id":4,"name":"referee"}]
,"images":[{"id":0,"license":1,"file_name":"121364_7_8_png.rf.d492
bc91a93a35073d085c5110bc7c10.jpg","height":1280,"width":1280,"date
_captured":"2023-08-07T22:52:26+00:00"},...
```


Second half:


```
"annotations":[{"id":0,"image_id":0,"category_id":2,"bbox":[0,465,
10,49],"area":490,"segmentation":[],"iscrowd":0},{"id":1,"image_id
":0,"category_id":3,"bbox":[520,417,16.5,38],"area":627,"segmentat
ion":[],"iscrowd":0},{"id":2,"image_id":0,"category_id":3,"bbox":[
623,412,13.5,47.5],"area":641.25,"segmentation":[],"iscrowd":0},..
..
```

JSON Dataset

First half:


```
"categories":[{"id":1,"name":"ball"},{"id":2,"name":"goalkeeper"},
{"id":3,"name":"player"},{"id":4,"name":"referee"}]
,"images":[{"id":0,"license":1,"file_name":"121364_7_8_png.rf.d492
bc91a93a35073d085c5110bc7c10.jpg","height":1280,"width":1280,"date
_captured":"2023-08-07T22:52:26+00:00"},...
```


Second half:


```
"annotations":[{"id":0,"image_id":0,"category_id":2,"bbox":[0,465,
10,49],"area":490,"segmentation":[],"iscrowd":0},{"id":1,"image_id
":0,"category_id":3,"bbox":[520,417,16.5,38],"area":627,"segmentat
ion":[],"iscrowd":0},{"id":2,"image_id":0,"category_id":3,"bbox":[
623,412,13.5,47.5],"area":641.25,"segmentation":[],"iscrowd":0},..
..
```

JSON Dataset

First half:

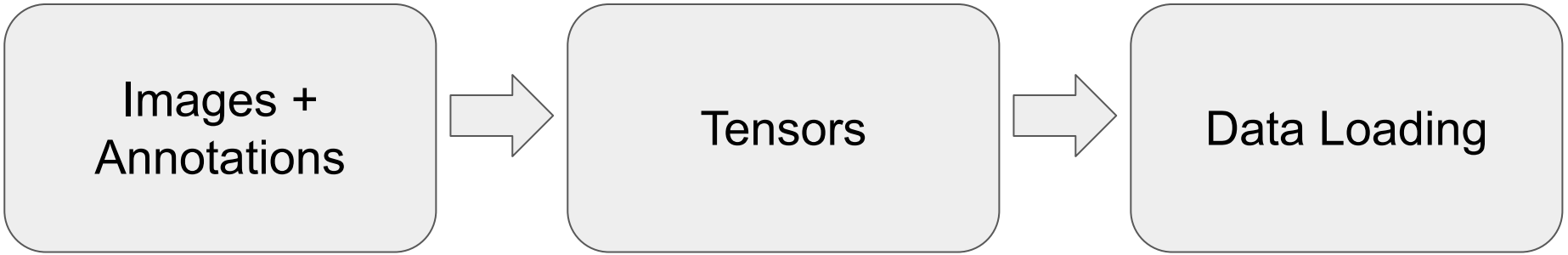

```
"categories":[{"id":1,"name":"ball"},{"id":2,"name":"goalkeeper"},
{"id":3,"name":"player"},{"id":4,"name":"referee"}]
,"images":[{"id":0,"license":1,"file_name":"121364_7_8_png.rf.d492
bc91a93a35073d085c5110bc7c10.jpg","height":1280,"width":1280,"date
_captured":"2023-08-07T22:52:26+00:00"},...
```


Second half:


```
"annotations":[{"id":0,"image_id":0,"category_id":2,"bbox":[0,465,
10,49],"area":490,"segmentation":[],"iscrowd":0},{"id":1,"image_id
":0,"category_id":3,"bbox":[520,417,16.5,38],"area":627,"segmentat
ion":[],"iscrowd":0},{"id":2,"image_id":0,"category_id":3,"bbox":[
623,412,13.5,47.5],"area":641.25,"segmentation":[],"iscrowd":0},..
..
```

Football Dataset Class and Data Loading

- Implemented functions to convert given images and their annotations to tensors
  - Tensors: multidimensional numerical arrays
- Loaded the data



Modeling & Evaluation

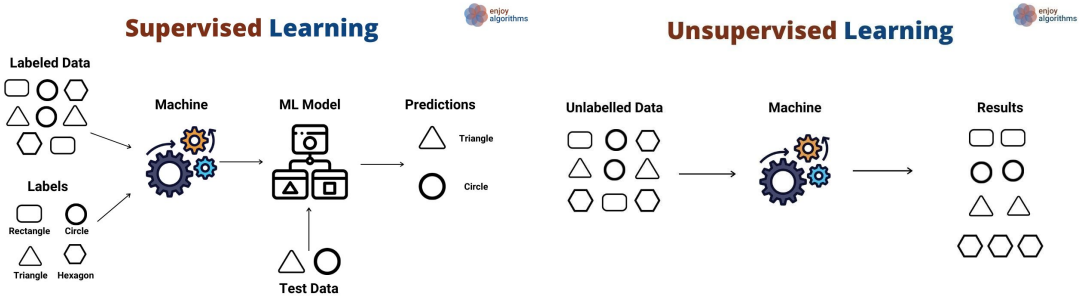
Model Approach

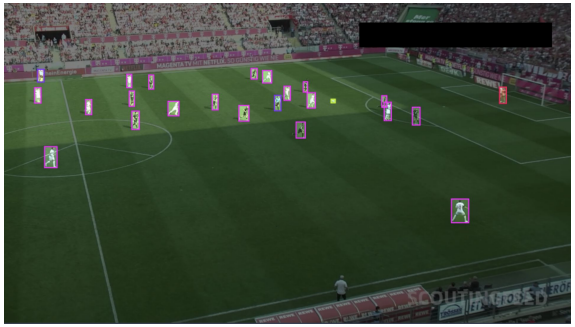
Two types of Machine Learning processes: Supervised Learning vs Unsupervised

- Ours is supervised learning
  - Utilize the labeled data (JSON annotations) to train model and have the model output labeled data
  - Classification problem: our model should classify each object in the image to its respective category/class (player, goalkeeper, ball, and referee).
  - Utilize classification algorithms/methods

Classification vs Object Detection Algorithms (classification isn't enough)

- Classification algorithms: Take the whole image to classify object
- Object detection algorithms: Utilize bounding boxes (boxes around objects) within the image, more specific





Models: R-CNN vs YOLO

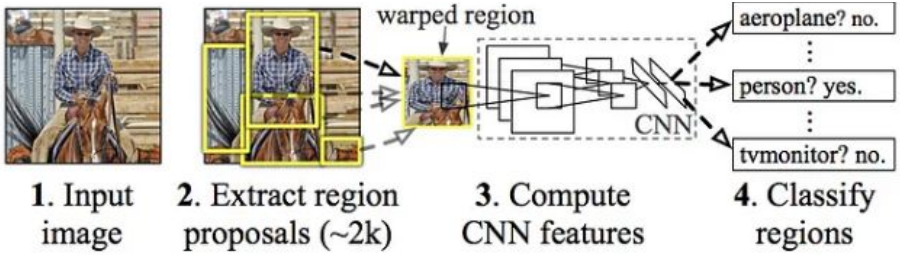
Existing popular object detection models we are using:

- **R-CNN** (Region-Based Convolutional Neural Network)
- **YOLO** (You Only Look Once)
  - Both utilize **CNN** (Convolutional Neural Network), more about this in the next slide

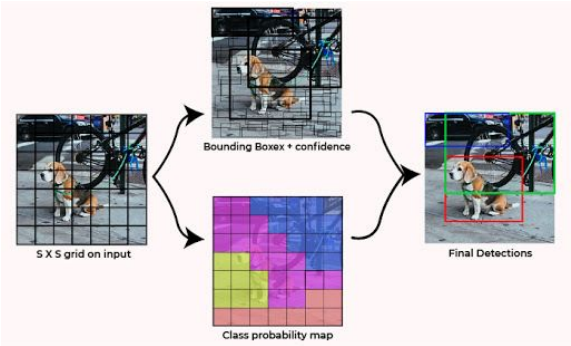
R-CNN vs YOLO

- R-CNN is a two-stage model: Propose regions with objects and then classify the objects in those regions
  - More steps = slower
- YOLO is a single-stage model: Predict objects and their bounding boxes at one look (You Only Look Once)
  - Less steps = faster

R-CNN

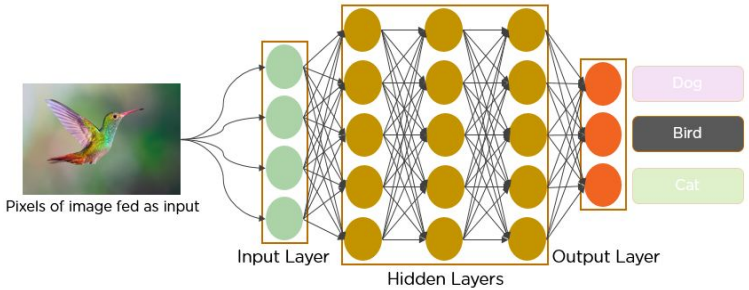


YOLO



CNN (Convolutional Neural Network)

Neural Network

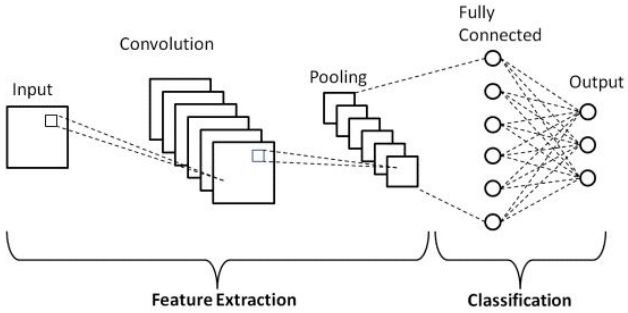


Input layer: Capture pixel data/features from the image like color of pixel

Hidden Layers: Recognize patterns in the pictures (like edges, patterns, complex shapes, etc, with each subsequent layer getting more complex) and deciding which features of the image are more important (weights)

Output Layer: The final outputs (the final predicted class and the score of how accurate it is) like 80% sure it is bird and 20% cat.

CNN



Conv layers/hidden: reduce images into a matrix of pixels that is easier to process (conv feature map) and define the weights

Pooling layers/hidden: simplifying data by keeping only the most important features

FC (fully connected)/hidden: connect all the features in the pooled feature maps

Softmax/Output Layer: convert raw scores to probability that we can use (in the format of 0.6, 0.7, etc.)

(conv feature map)

4	3	4
2	4	3
2	3	4

Intro to R-CNN Family Models

Three generations:

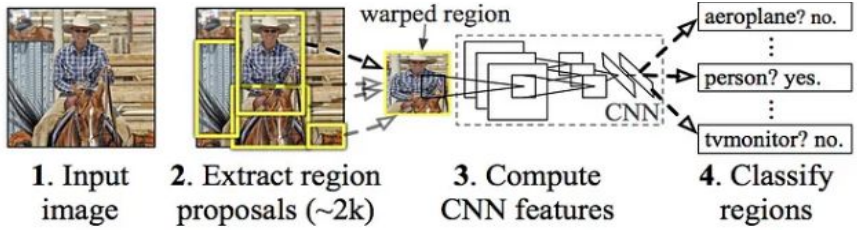
- R-CNN
- Fast R-CNN
- **Faster R-CNN (the model we chose, more on the next slide)**

Basic three steps for all R-CNN models:

1. Create **region proposals** (2000) using a selective search algorithm: the regions in image that might contain object (bounding boxes)
2. Extract **CNN features** (patterns in the image like colors, lighting, etc.) from region proposals
3. Classify objects using features

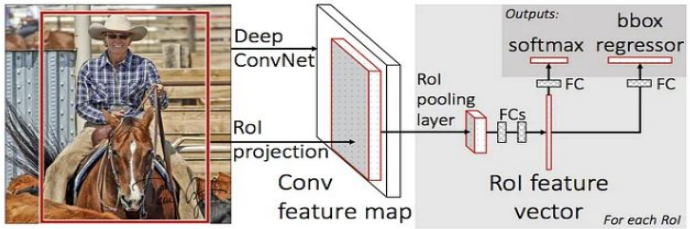
The problem with R-CNN and Fast R-CNN: **they are slow** because they utilize **selective search algorithm** (an external algorithm) to generate region proposals

RCNN



Region proposals, then CNN and then classification

Fast RCNN

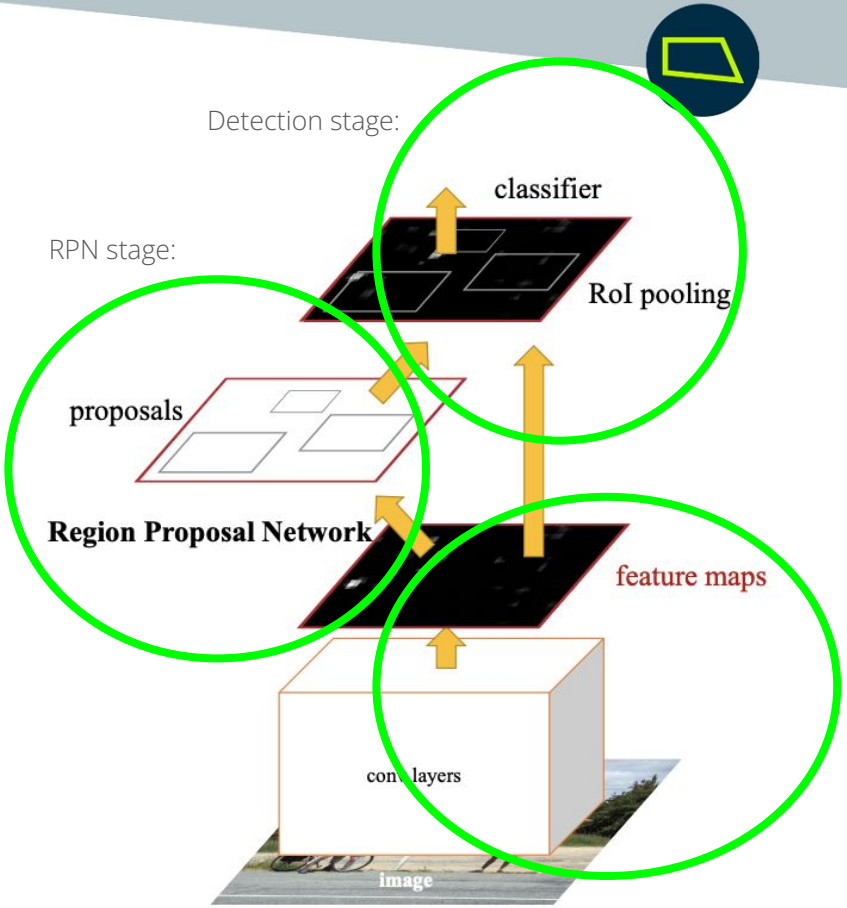


CNN (ConvNet + ROI/pool) with region proposals (step 2 and 3 combined) and then classification

Faster R-CNN Model

Why Faster R-CNN? The fastest out of all RCNN models, allowing for real-time object detection.

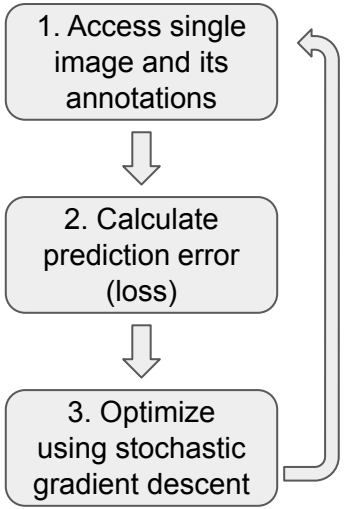
- Region proposal network (RPN): Propose regions directly from the feature maps (building on the integration of CNN and region proposals in Fast R-CNN) and replacing selective search algorithm
- Detects objects with the proposed regions through ROI pooling and classifier
    - ROI pooling: makes sure all regions are reduced into fixed-size and only keep important features, reducing computational costs (better than Fast R-CNN by doing ROI pooling once)
    - Classifier: identify object label/class (player, referee, etc.) and refine coordinates of bounding box
- Shared CNN Backbone: Allow for model to utilize the same feature extractions in both the RPN stage and the detection stage (we can see the feature maps being used for both stages).



Training Faster-RCNN Model

Steps:

1. Train all 600+ images in our training dataset with model by loading images and annotations
2. Have the model predict where the bounding boxes are and calculate the error between the prediction output and the provided target value (our annotations) to get loss
3. With our loss, optimize the model using stochastic gradient descent for each image to adjust the parameters of the model (the weights of each feature) and minimize loss (the learning step!)
4. And loop! In our case, we trained 600+ images 10 times (or epochs = passings of the entire dataset) for the best results.



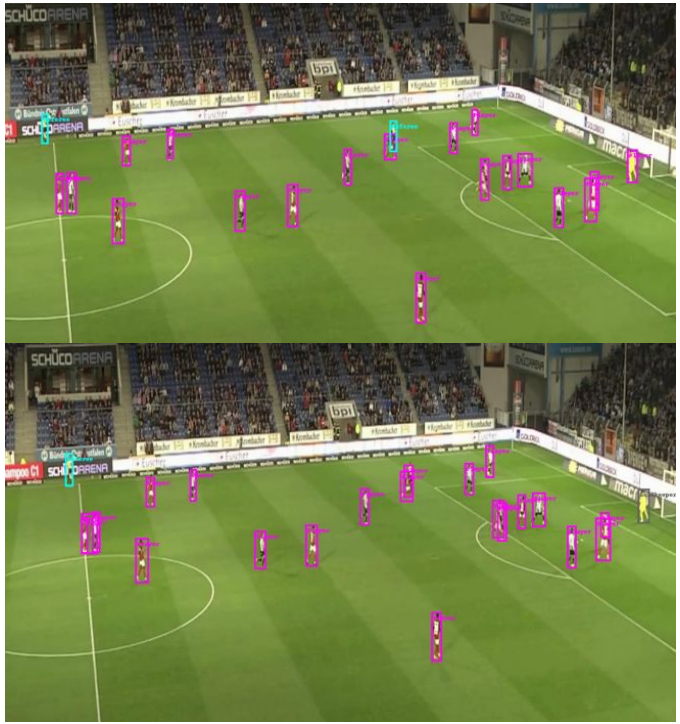

```
Iteration: 1/10, Average iteration loss: 1.1697361469268799
Iteration: 2/10, Average iteration loss: 0.8446316123008728
Iteration: 3/10, Average iteration loss: 0.6970969438552856
Iteration: 4/10, Average iteration loss: 0.5963253378868103
Iteration: 5/10, Average iteration loss: 0.5403541326522827
Iteration: 6/10, Average iteration loss: 0.4978560507297516
Iteration: 7/10, Average iteration loss: 0.4836512506008148
Iteration: 8/10, Average iteration loss: 0.42549413442611694
Iteration: 9/10, Average iteration loss: 0.46847689151763916
Iteration: 10/10, Average iteration loss: 0.4042800962924957
```

Output of Faster RCNN Model

Snippet of output predictions for 13 test images

```
[{'boxes': tensor([[ 325.3285,
470.2320,  340.2840,  519.5388]],
device='cuda:0'), 'labels':
tensor([2, 2, 2, 2],
device='cuda:0'), 'scores':
tensor([0.9995, 0.9992, 0.9991,
0.9991], device='cuda:0')}]
```

Draw function

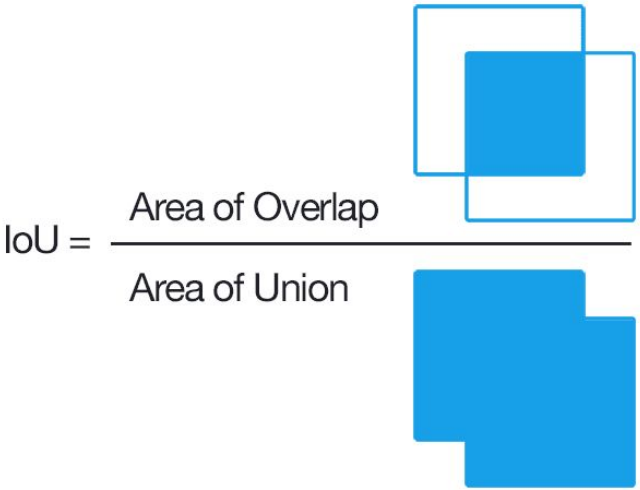


Actual labels

Predictions

Evaluate Faster R-CNN Model: Intersection over Union (IoU metrics)

- Calculated accuracy scores with intersection over union
  - Compared predicted boxes with actual boxes
  - Higher IoU score = better accuracy



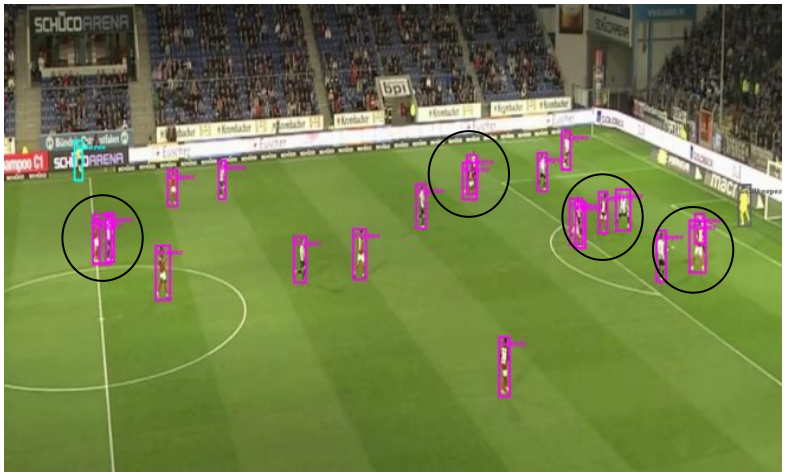

```
{'iou': tensor(0.0471, device='cuda:0')}
{'iou': tensor(0.0447, device='cuda:0')}
{'iou': tensor(0.0331, device='cuda:0')}
```

NMS

- Problem: Faster R-CNN can produce multiple bounding boxes for one object
- Solution: **NMS** (Non-maximum Suppression)
    - Reduces the number of output bounding boxes using IOU by iteratively removing boxes with low scores and IOUs that are bigger than the IOU threshold we want (small IOU means keeping only the most precise boxes)

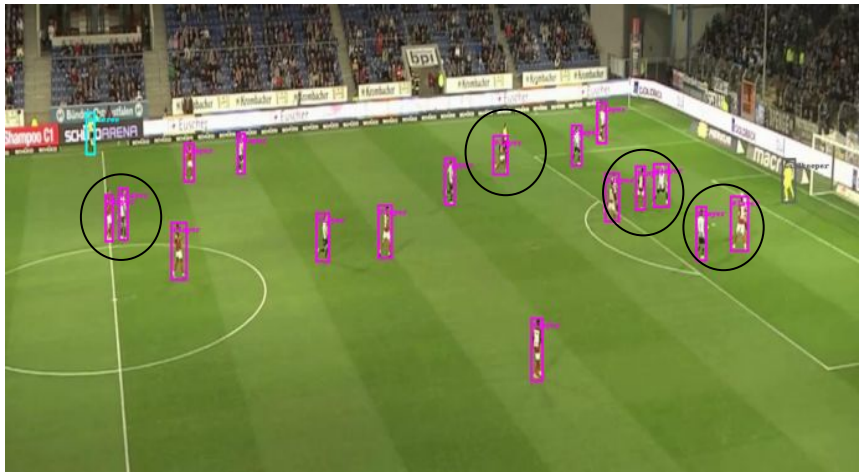
```
indices = torchvision.ops.batched_nms(boxes=pred[0]['boxes'], scores=pred[0]['scores'], idxs = pred[0]['labels'], iou_threshold = 0.2) *WE WANT LOW IOU THRESHOLD (but at some point, lowering the iou threshold will not increase score because there are no overlapping boxes left)*
```

Before



{'iou': tensor(0.0436, device='cuda:0')}

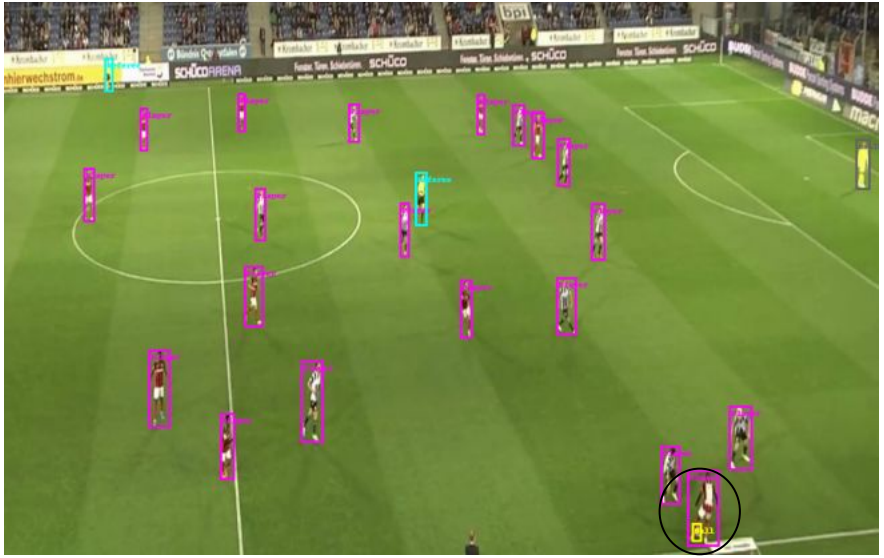
After



{'iou': tensor(0.0461, device='cuda:0')}

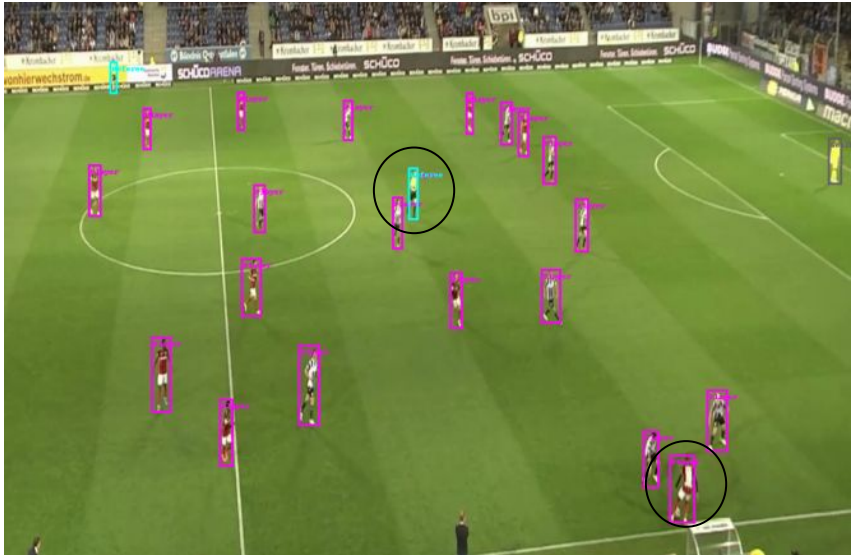
More NMS

Actual Labels



```
{'iou': tensor(0.0591, device='cuda:0')}
```

NMS Predictions



```
{'iou': tensor(0.0461, device='cuda:0')}
```

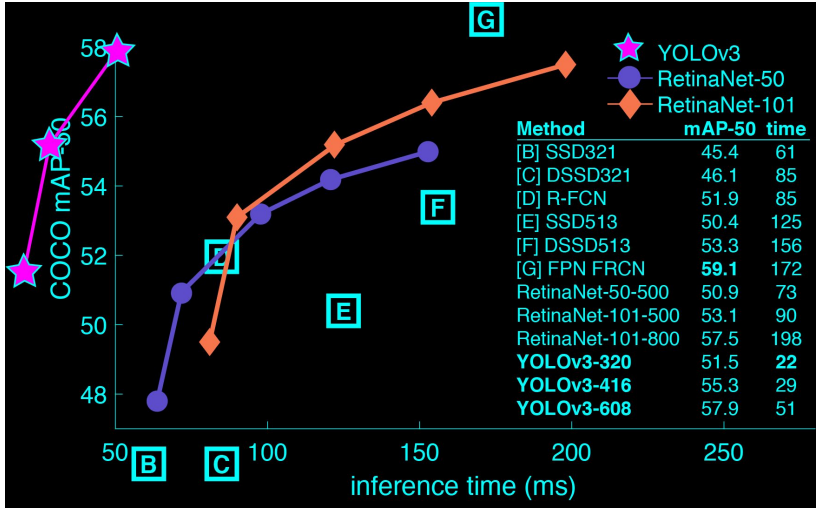
- Problem with our implementation of NMS: potential overlapping between two classes (minimize class bounding boxes), strange IOU scores for actual labels
- Problem with this model training: can't detect ball at all

What About YOLO?

YOLO: Real-Time Object Detection

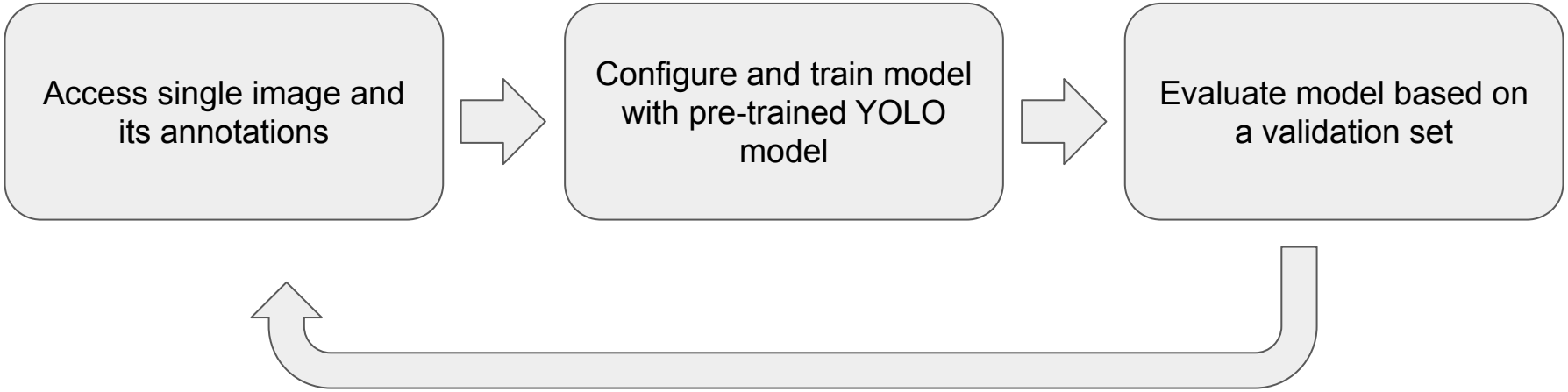
Goal: Create a **YOLOv8** model with **Darknet** framework and finetune it to our specific 4 classes

- Why YOLO? YOLO is extremely **fast** and **accurate**
  - Speed vs accuracy adjustments are made via **changing** model **size**
  - **NO RETRAINING!!!!**



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

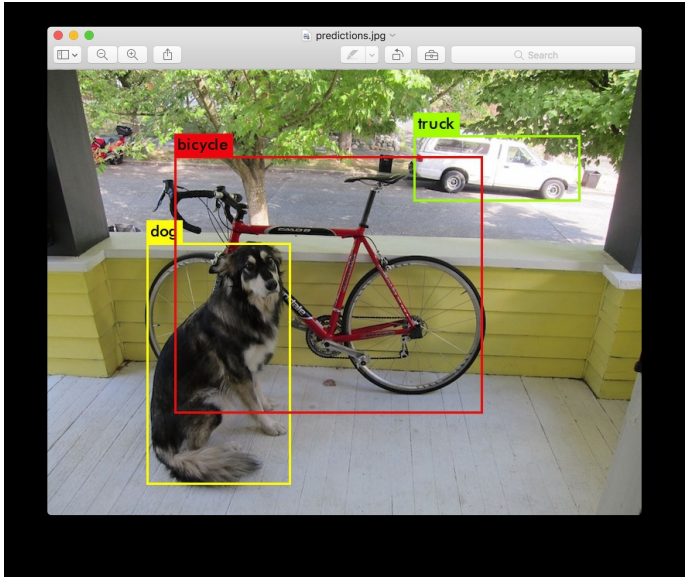
Training YOLO Model



Darknet framework used alongside YOLO

Utilizing **darknet** over a **traditional** approach

- Why darknet? Utilizes **different approach**
  - **Standard**: Repurpose **classifiers** or **localizers** by applying the model at multiple **locations** and **scales** of images, **region scoring** for detection
  - **Standard** (R-CNN)**:** Makes predictions that require thousands of images of neural networks for 1 image R
  - **Darknet**: makes it **extremely fast,** more than 1000x faster than R-CNN and 100x faster than Fast CNN



Evaluation of YOLO Model PT.1-Understanding Format

```
# Classes (80 COCO classes)
names:
  0: person
  1: bicycle
  2: car
  ...
  77: teddy bear
  78: hair drier
  79: toothbrush
```

- YOLO Formatting- **NO JSON allowed**
  - Converted into one row per object in class **x_center y_center width height format**
- Very specific pathing needed when importing our data set

```
1 0.384375 0.277734375 0.00625 0.01015625
3 0.44453125 0.2294921875 0.0109375 0.043359375
4 0.2396484375 0.2224609375 0.012109375 0.043359375
```

SAMPLE SET

zidane.txt

```
0  0.481719 0.634028 0.690625 0.713278
0  0.741094 0.524306 0.314750 0.933389
27 0.364844 0.795833 0.078125 0.400000
```

Evaluation of YOLO Model PT.2- YOLO on a single Image

- Exploratory Test
  - Checked how information would be outputted
  - First look at the accuracies

```
Loading weights from yolov3.weights...Done!
/content/data/test/40cd38_7_6_png.rf.dc7957fcce915b
sports ball: 54%
person: 100%
person: 99%
person: 98%
person: 97%
person: 97%
person: 97%
person: 95%
person: 95%
person: 94%
person: 87%
person: 86%
person: 79%
person: 77%
person: 76%
person: 75%
person: 69%
person: 69%
person: 67%
```

Evaluation of YOLO Model PT.3- Yolo Darknet Player Detection

- Model detection
  - Before anything the cfg coco.data must be specified to fit your pathing given names and number of classes.
  - The yolo.cfg model is modified for training
  - Then you are ready to train
    - `./darknet detector train cfg/coco.data cfg/yolov3.cfg darknet53.conv.74`

```
[net]
# Testing
# batch=1
# subdivisions=1
# Training
batch=64
subdivisions=8
....
```

```
classes= 80
train  = <path-to-coco>/
valid  = <path-to-coco>/
names = data/coco.names
backup = backup
```

```
Region 94 Avg IOU: 0.244487, Class: 0.641122, Obj: 0.508777, No Obj: 0.518971, .5R:
Region 106 Avg IOU: 0.206882, Class: 0.393139, Obj: 0.472929, No Obj: 0.416488, .5R: 0.063
```

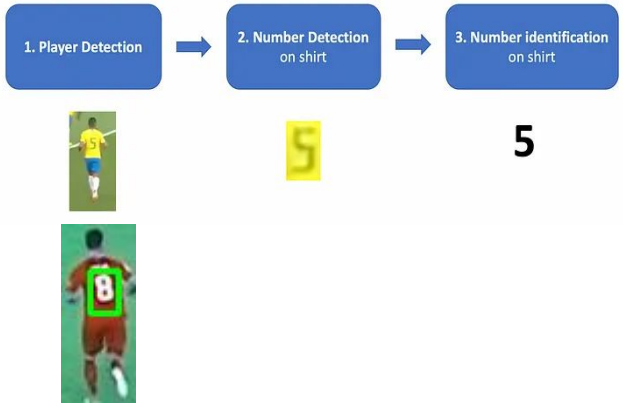
Evaluation of YOLO Model PT.4- Yolo Darknet Detection Evaluation

- 3 main components to be created for complete player tracking in something like a video or frame
  1. Player detection
  2. Number detection
  3. Number identification

- BBox regression loss on training = large component of prediction like prev models
  - Comparing **Epoch** count to **Loss**

- Confusion Matrix Heat Maps
  - Later Allowed for easy prediction visualization of numbers to results

Fig. 3 Main Components of the YOLO notebook



Model Comparison

Model Name	Description	Results	Pros	Cons
Faster-RCNN	Pre-trained deep network model that can predict object boxes and scores	Slow training but accurate predictions (with NMS) in relative to actual IOU	Identified the main objects correctly, fastest out of all R-CNN models	Multiple bounding boxes for one object, can't identify ball, slow when doing multiple epoch/iterations for large datasets
YOLOv8	Computer vision model for object detection and classification	Higher initial loss with IOU sadly below threshold	Worked fast, weights are able to help identify all the pretrained class names in images	Constant pathing and formatting issues, accuracy drops with object size

Final Thoughts

What We Learned

- New tools, concepts, technologies, and frameworks
  - Google Colab, COCO JSON file + API, tensors, PyTorch, Darknet, IoU, NMS
- Object detection using pre-trained models (Faster-RCNN, YOLO)
  - First experience with **computer vision**!
- How to approach projects with fewer specifications

Takeaways:

- Importance of organization and coordination
- Understand the tools and their limitations
- Utilize online resources
- Don’t be afraid to ask for help

Potential Next Steps

Models:

- Finetune R-CNN NMS more (there are still some overlaps, particularly for different classes)
- Implement R-CNN IOU better: strange IOU scores for actual labels

More complicated usage:

- Track players based on jersey number
- Detect objects on videos
- Specific replay angles based on player


Questions?