AI Project Teams #3
Fall 2025

Fruit Detector AI Project Teams

Contents

Objectives

- Reproduce the core YOLO-style loss function from scratch
- Build a detection model capable of localizing and classifying multiple fruit categories

Project roadmap

Data Curation

Collected and preprocessed the fruit dataset

Model Architecture

Configured the CNN (YOLO) Architecture and divided input image into SxS grid

Training & Loss Opt.

Trained model using multipart loss function, optimizing for Localization and Objectiveness

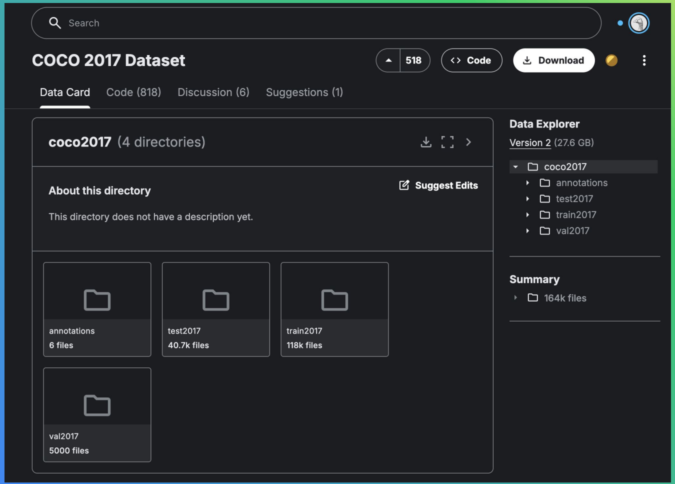
Validation Metrics

Evaluated Model Performance with IoU (Intersection over Union)

Real Time Inference

Fully trained model capable of accepting raw images and outputting accurate Bounding boxes with high confidence

Dataset & Filtering



- Sourced the COCO 2017 dataset from Kaggle
- Filtered over 200k images to extract specific fruit categories (Apples, Bananas, Oranges) to create a focused training set.

COCO 2017 Dataset

ultralytics/cfg/datasets/coco.yaml

```
41: cup
42: fork
43: knife
44: spoon
45: bowl
46: banana
47: apple
48: sandwich
49: orange
50: broccoli
51: carrot
52: hot dog
```

About COCO

- Common Object in Context
- A large-scale datasets for object detection using bounding boxes and segregation marks

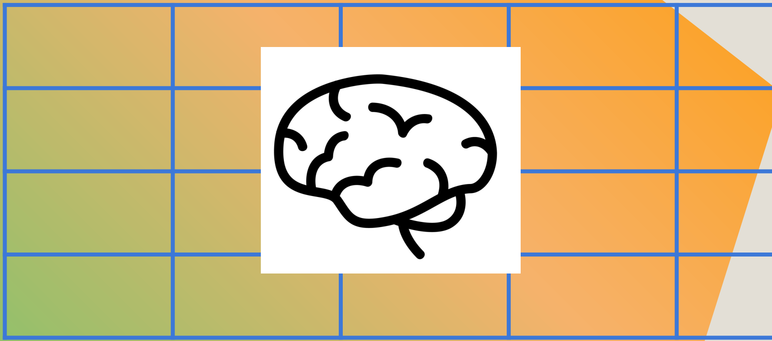
Data Statistics

- 200,000 images
- 80 object categories
- 500,000 annotated object instances

Limitation of COCO

- only a few fruit categories
  - Apple
  - Banana
  - Orange

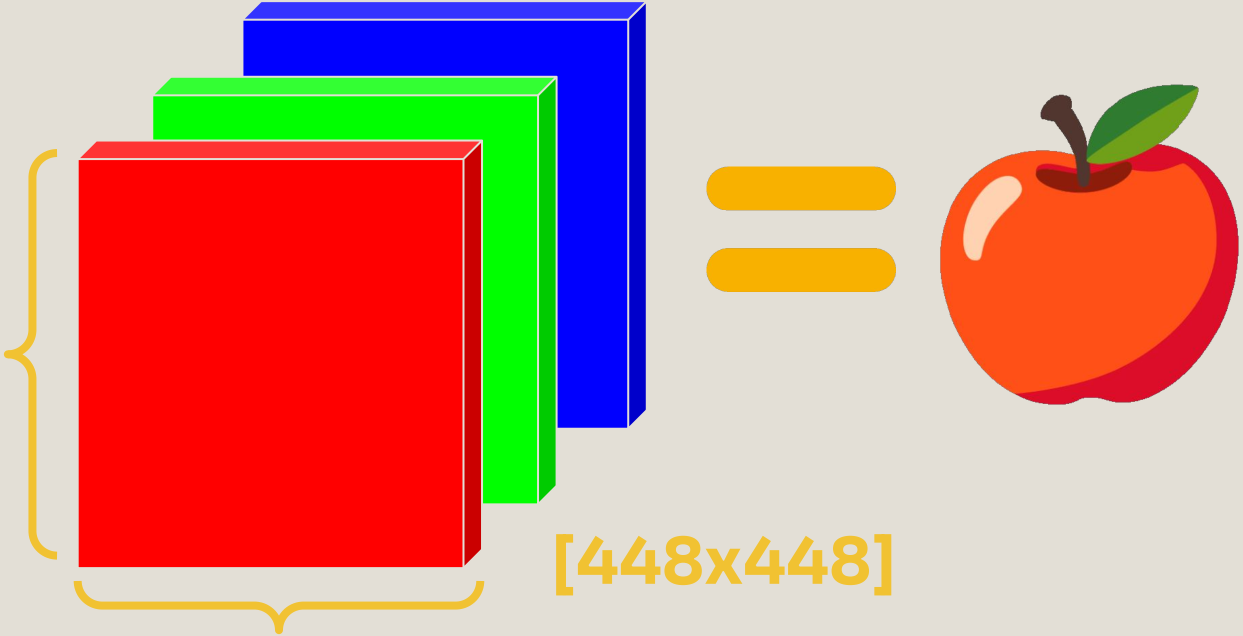
Theory & Architecture



- Selected YOLO for its single pass speed
- We applied linear algebra concepts to map input pixels into bounding box vectors on an SxS grid.

Linear Algebra: How Machines "See" Data

- The Matrix: A 2D grid of numbers (rows x columns)
- The Tensor: A multi dimensional array. (we use rank-3 tensors)
- Takeaway: The model doesn't see a banana, it sees a 448x448x3 block of values



You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

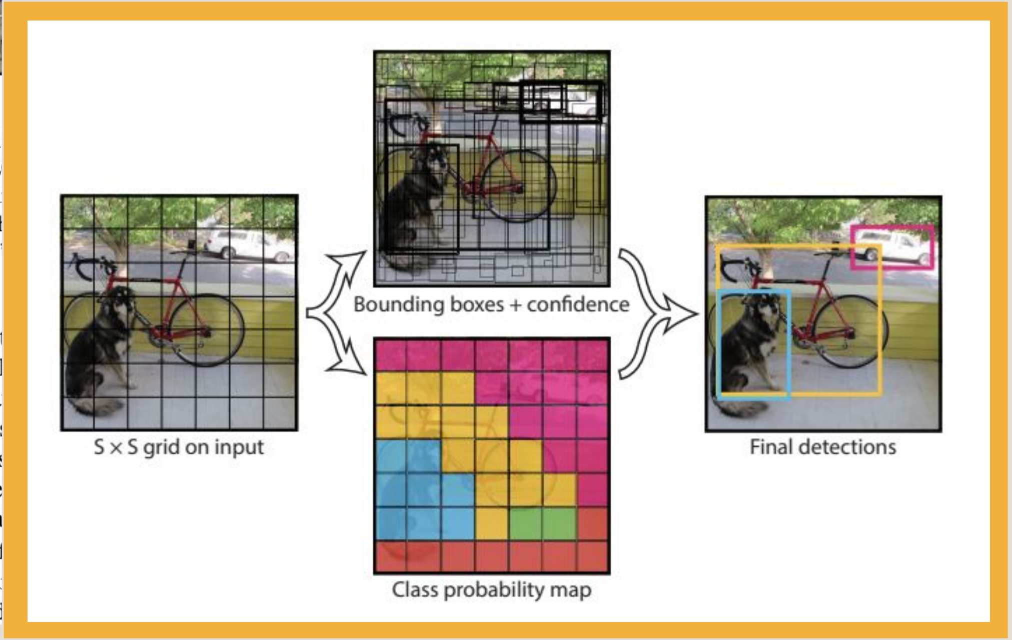
We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

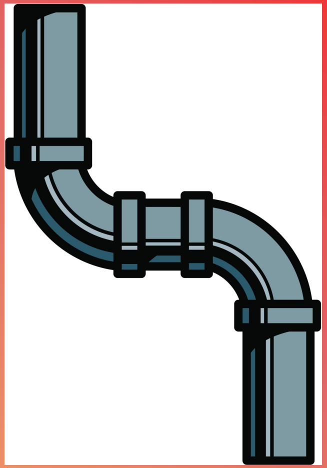
Figure 1 with YOL the input i work on th the model

methods age and th classifica ing boxe boxes bas plex pipe individua

We ref lem, stra nates and



Coding the Pipeline



- Built the model architecture from scratch using PyTorch. We utilized einx for efficient tensor manipulation and reshaping during the forward pass.

Building the Pipeline:

PyTorch + Einx

Flexibility: Allowed us to build the custom YOLO architecture from scratch rather than just using a pre-made "black box."

Speed: Handles the heavy lifting of matrix multiplication on the GPU.

Ecosystem: Huge community support for debugging and optimization.

The Problem: Deep learning involves 4D and 5D tensors. In standard code, you often lose track of which dimension is which. It's confusing and error-prone.

Einx lets us write code using named dimensions.

Key Benefit: It makes the math "readable" in the code itself.

Loss and Refinement

- Implemented the Custom multipart loss function
- Trained the model to balance Localization (box coordinates) loss against Confidence loss to reduce false positives

loss function:

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

Center coordinate loss (localization accuracy)

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

width–height loss

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

object confidence loss for grids containing objects

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

no-object confidence loss to suppress false positives

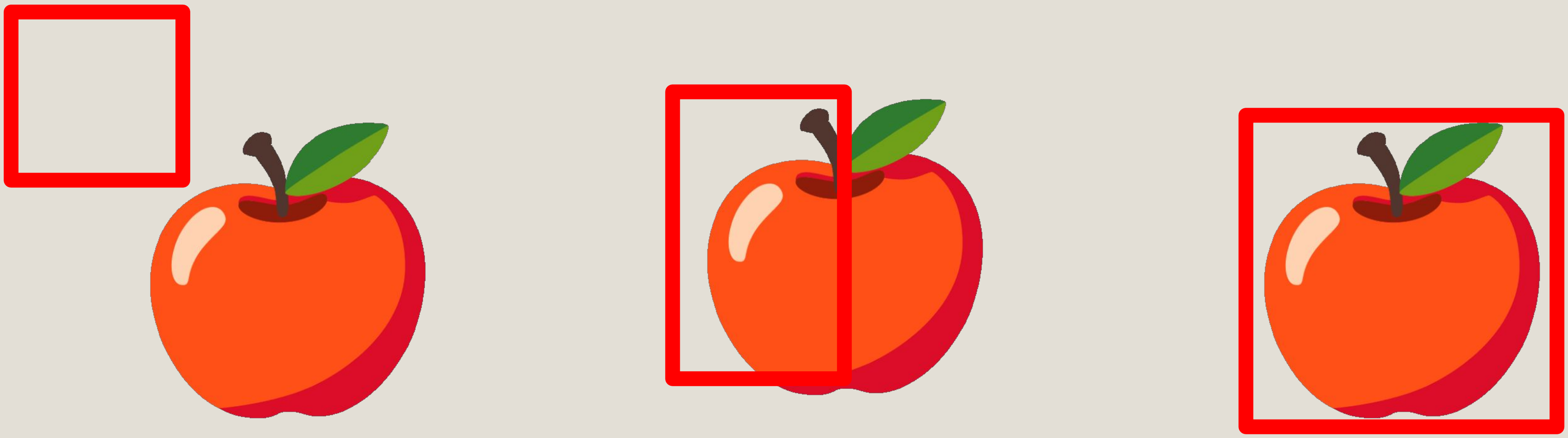
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

classification loss to ensure correct class prediction

Purpose of Loss Function:

- To quantify the error between a model's predictions and the true values

Rewarding Good Predictions

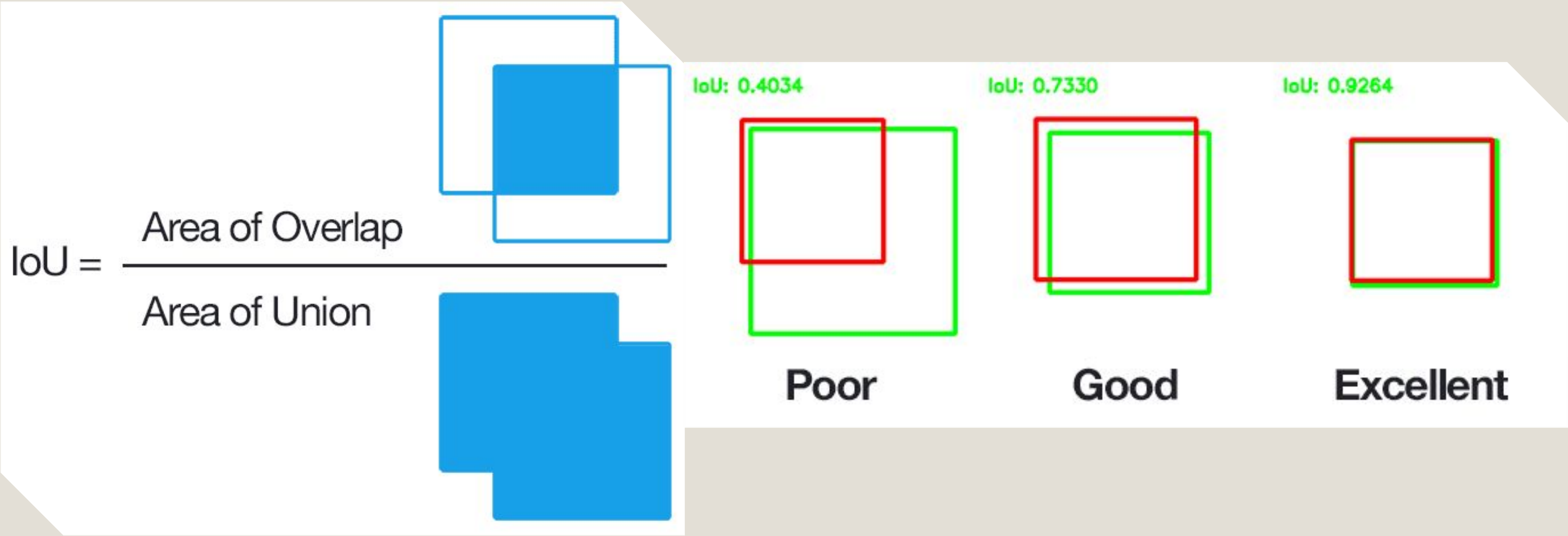


A rewarding system guides model learning by defining desirable outcomes, stabilizing optimization, and aligning training behaviors with task objectives.

Some examples:

- IoU based reward
- coordinated reward
- penalizing incorrect object confidence

Which Prediction Is The Best?



Intersection over Union (IoU)
Calculated as the area of intersection divided by the area of their union

Optimization 1: Localization Loss

Arbitrary Weighting Coefficient

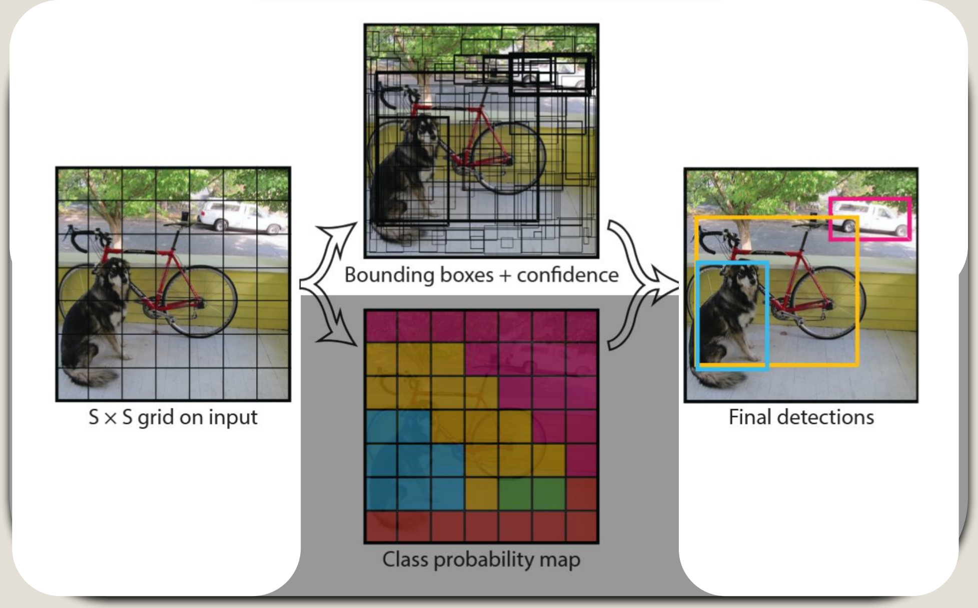
$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

Iterate over all predictions

Quantify how far away the center of the best predicted box is from the center of the actual object.

Ignore "bad" predictions by the model.

Only regress the predictions to make the **best** predictions better.



Optimization 2: Dimension Loss

Arbitrary Weighting Coefficient

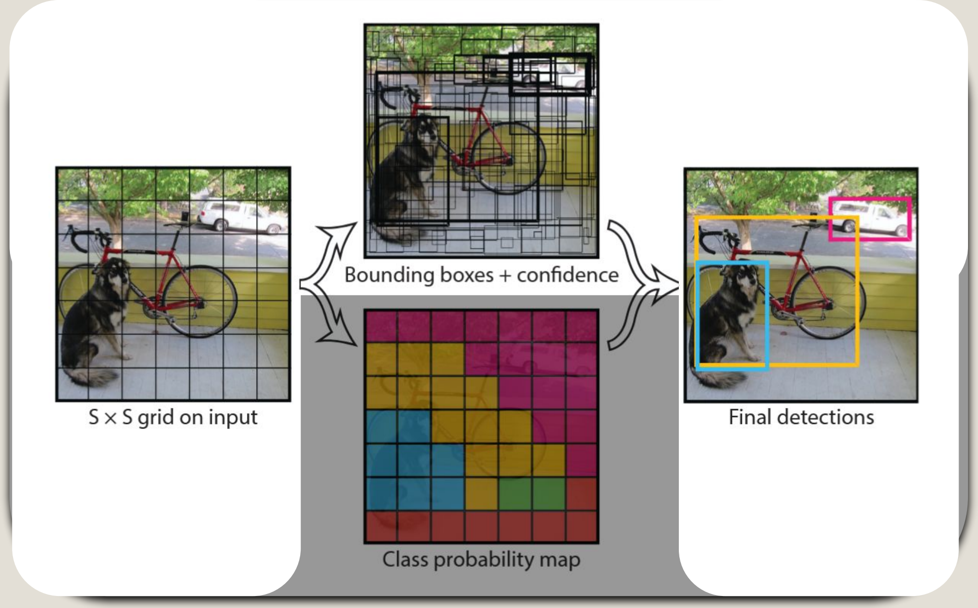
$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

Quantify difference between the size (width and height) of the best predicted box and the size of the actual object.

Iterate over all predictions

Ignore “bad” predictions by the model.

Only regress the predictions to make the **best** predictions better.



Intuition & Desirable Model Properties

- Model always predicts 7*7*2=98 bounding boxes, regardless of the image.
    - Most images do not have 98 objects!
    - Model needs to self-assess which predictions are relevant -> confidence score

Optimization 3: Confidence Loss

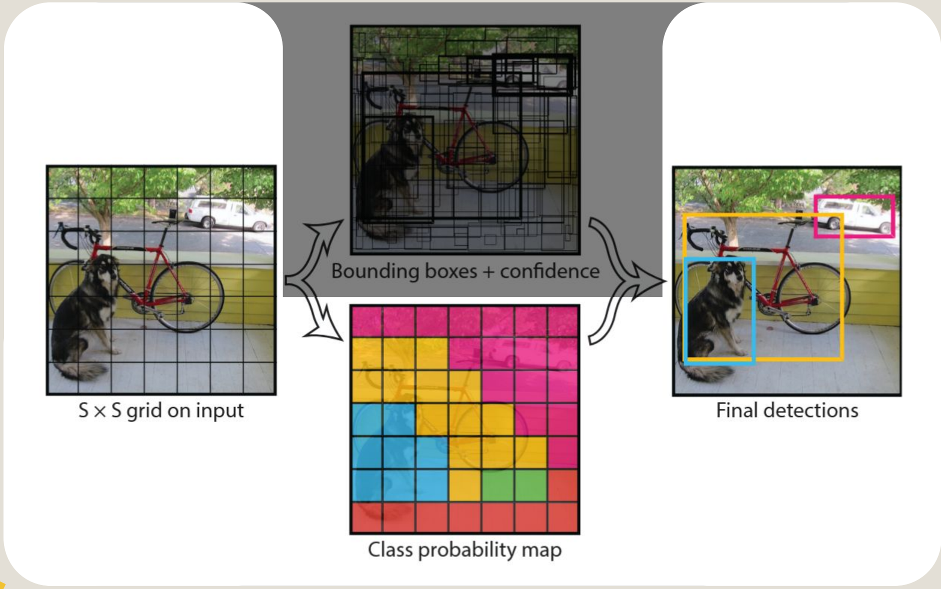
$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

Iterate over all predictions

Ignore “bad” predictions by the model.

Only regress the predictions to make the **best** predictions better.

Regress the model’s own confidence for the best predicted boxes to be close to 1.



Penalizing Bad Predictions

$$\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{noobj}}} \left(C_i - \hat{C}_i \right)^2$$

$\hat{C}_i$ is 0 for all predictors **except** the one with the highest IOU to the ground truth.
This regresses the predictor confidence to be 0.

YOLO Network Architecture Overview

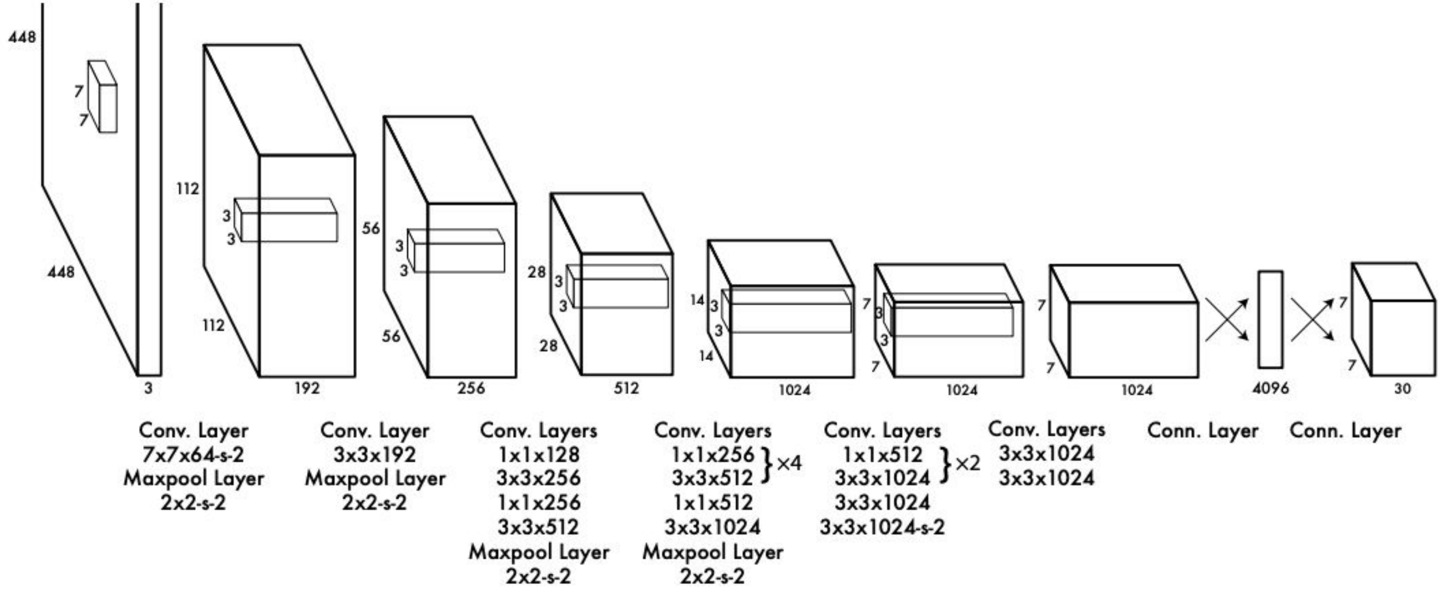


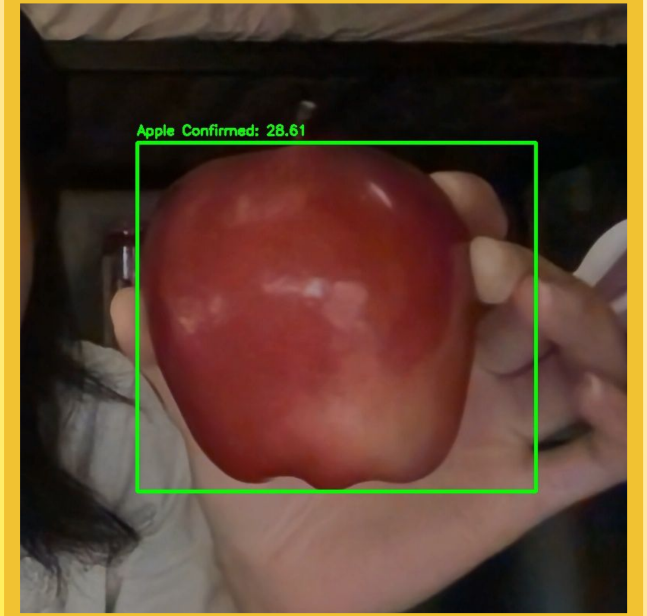
Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Input: 448 × 448 RGB Image

CNN

Output: 7 × 7 × 30 predictions
(bounding boxes + confidence + classes)

Real Time Inference (Final Product)



- A fully trained CNN capable of taking raw input images and outputting accurate bounding boxes with high confidence scores for multiple fruit types


THANK YOU