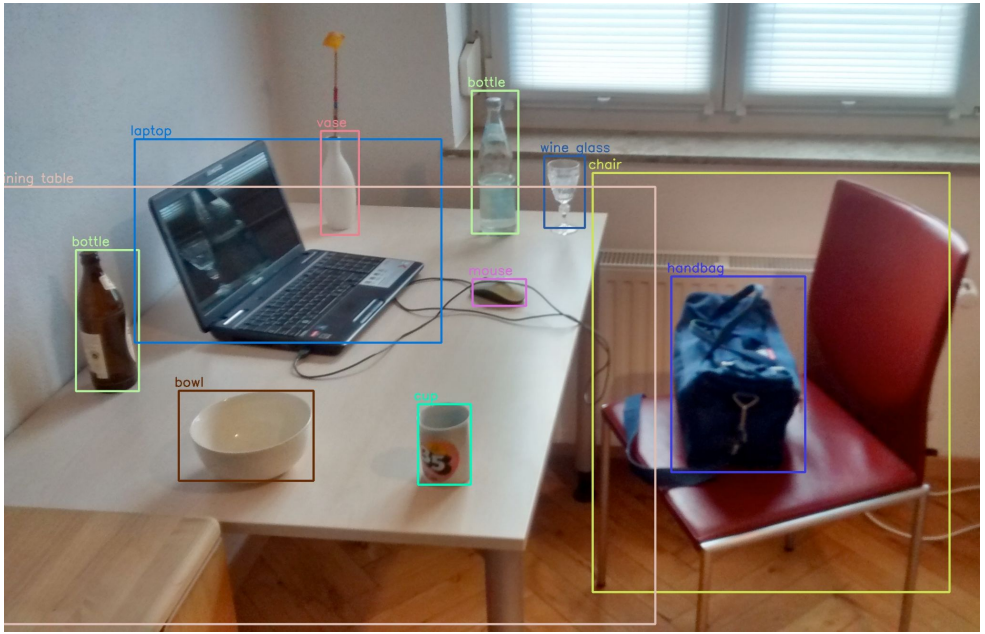
Object Detection

John Mikos + Kenneth Hernandez

What is Object Detection

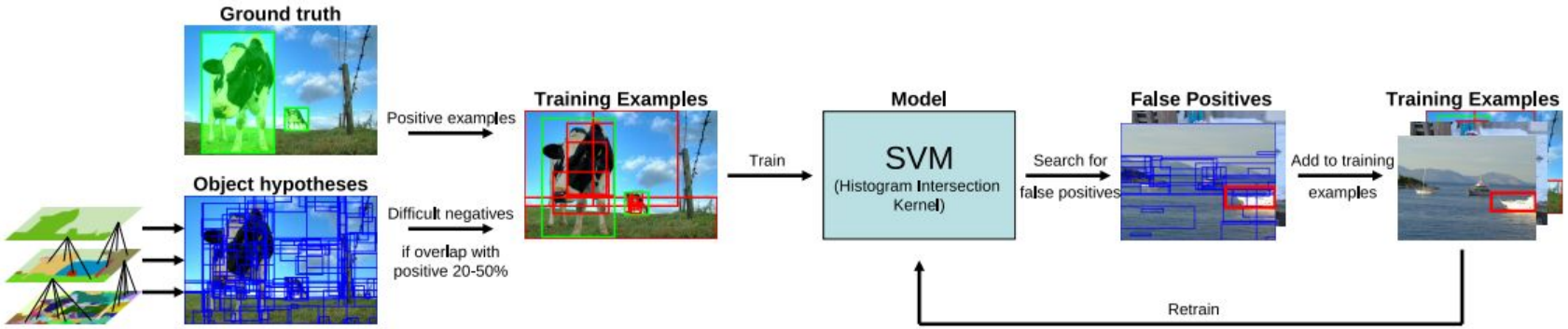
- Process of locating objects in images or videos
- Basis for different CV tasks such as instance segmentation, image captioning, object tracking and more.
- Some uses consist of Surveillance systems, diagnosing diseases from MRI/CT scans, and Autonomous Vehicles.



Selective Searching

- Uses a diverse set of complementary and hierarchical grouping strategies
- Uses image structure to guide sampling process
- Aims to capture all possible object location
- Fast to compute
- Able to work at all scales

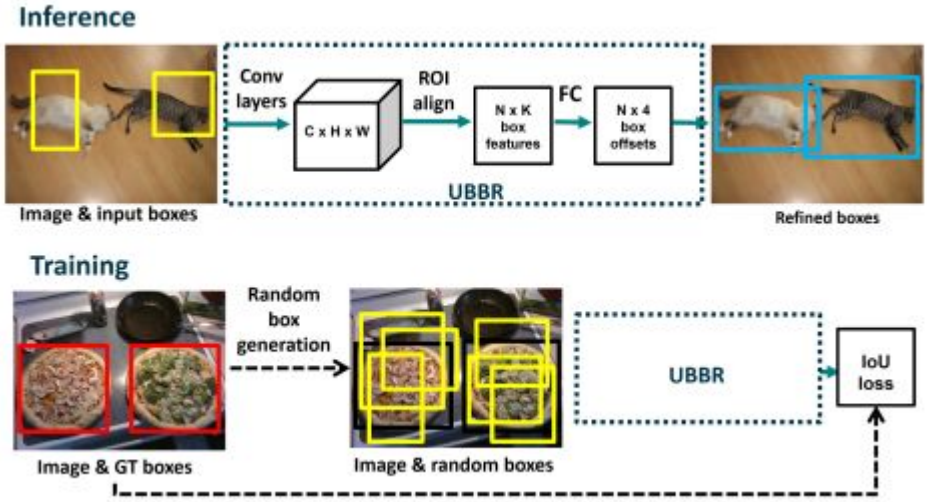
Basis behind Histogram of Oriented Gradients (HOG) and Bag-of-words



Universal Bounding Box Regression (UBBR)

Takes an image with roughly localized bounding boxes and refine them so that they tightly enclose nearby objects.

The network takes bounding boxes randomly generated around ground-truth boxes, and is learned to transform each input box so that Intersection-over-Union between the box and its nearest ground-truth is maximized



PASCAL VOC 2012 Challenge

- PASCAL Visual Object Classes
  - Train/validation has 11,530 images containing 27,450 Regions of Interest (ROI) annotated objects and 6,929 segmentations
  - Private testing set
- 50% training/validation and 50% for testing
- Dataset contains 20 object categories/classes
  - Person
  - Animal: bird, cat, cow, dog, horse, sheep
  - Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
  - Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor
- Each image has pixel-level segmentation annotations, boundboxes annotations, and object class annotations

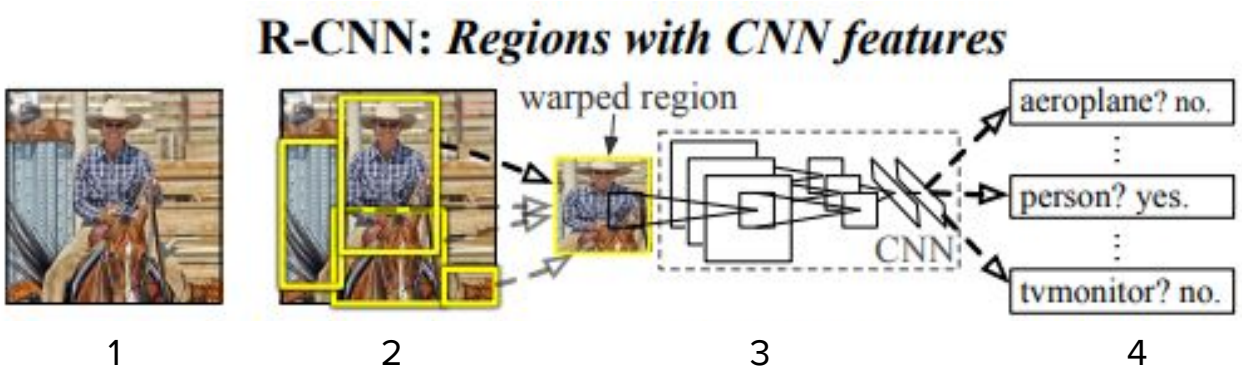
Rich feature hierarchies for accurate object detection and semantic segmentation

Overview

Object Detection models were complex ensemble systems that typically combine multiple low-level image features with high-level context. So how to improve?

Combine two key insights:

1. Apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects
2. Supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost when labeled training data is scarce



1. Takes an input image
2. Extracts around 2000 bottom-up region proposals
3. Computes features for each proposal using a large CNN
4. Classifies each region using class-specific linear SVMs

Called R-CNN due to this mixture of regions with CNN features

Out performed by over 15% on PASCAL VOC 2010 dataset

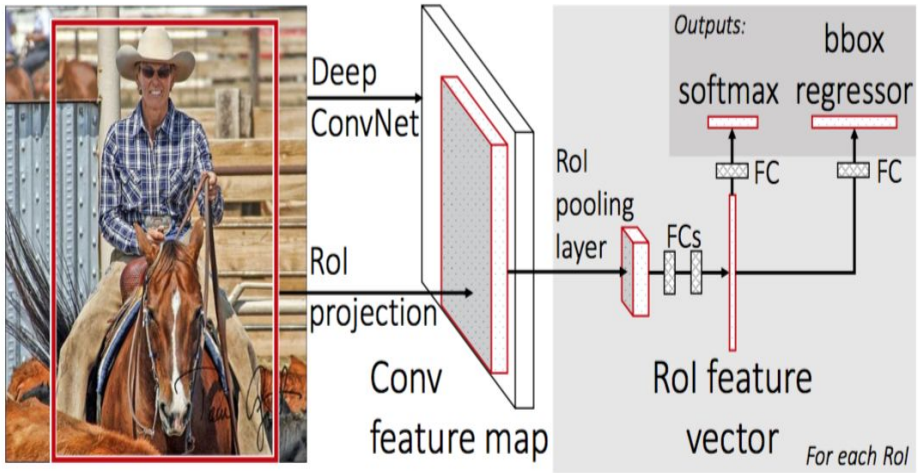
Key Takeaways

1. Simplified object detection methods
2. Gave a 30% relative improvement over the best previous results on PASCAL VOC 2012
3. Apply high-capacity CNNs to bottom-up region proposals in order to localize and segment objects
4. Supervised pre training on a large auxiliary dataset, followed by domain-specific fine-tuning on a small database, is an effective paradigm for learning high-capacity CNNs when data is scarce

Fast R-CNN

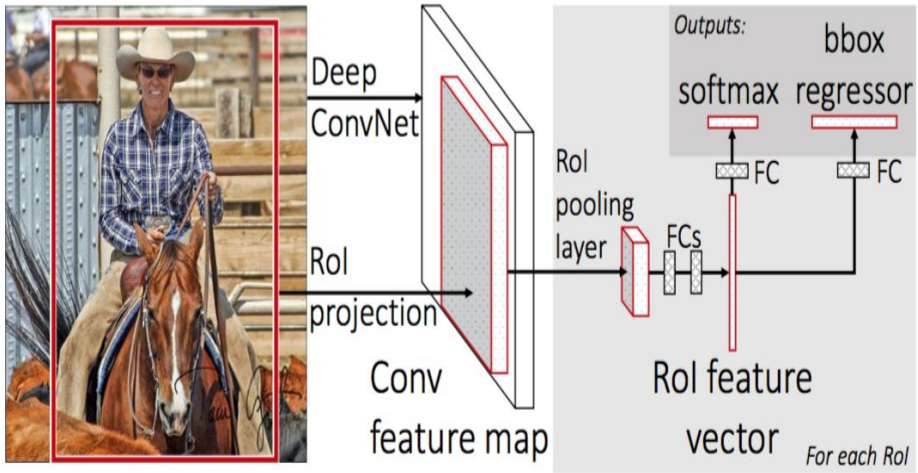
Overview

- Builds off R-CNN to classify objects using DNNs
  - 9x Faster than R-CNN
    - 243x faster during inference(Test-time)
- End-to-End trainable
- Resolves some issues with R-CNN
  - Training is a multi-stage pipeline.
  - Training is expensive in space and time.
  - Object detection is slow



Training

- Uses RoI pooling, pretrained networks, Fine-tuning, Multi-task loss, mini-batch sampling, back-propagation through RoI Pooling layers
- Takes an entire image and set of object proposals as an input
- Processes the whole image with multiple convolutional and max pooling layers to create a conv feature map.
- A region of interest pooling layer extracts a fixed-length feature vector from the feature map for each object proposal
- Each feature vector is fed into fully connected layers that branch into two output layers, softmax and bounding box regressor.



Results

- Main results:
  - SOTA mAP on VOC07, 2010, and 2012
  - Fast training and testing compared to R-CNN, SPPnet
  - Fine-Tuning conv layers in VGG16 improves mAP

	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

Table 4. Runtime comparison between the same models in Fast R-CNN, R-CNN, and SPPnet. Fast R-CNN uses single-scale mode. SPPnet uses the five scales specified in [11]. †Timing provided by the authors of [11]. Times were measured on an Nvidia K40 GPU.

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11]†	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	07	73.4	77.0	63.4	45.4	**44.6**	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	**35.6**	66.8	67.2	70.4	**71.1**	66.0
FRCN [ours]	07	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
FRCN [ours]	07 \ diff	74.6	**79.0**	68.6	57.0	39.3	79.5	**78.6**	81.9	**48.0**	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	07+12	**77.0**	78.1	**69.3**	**59.4**	38.3	**81.6**	**78.6**	**86.7**	42.8	**78.8**	**68.9**	**84.7**	**82.0**	**76.6**	**69.9**	31.8	**70.1**	**74.8**	**80.4**	70.4	**70.0**

Table 1. **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07 \ diff**: **07** without "difficult" examples, **07+12**: union of **07** and VOC12 trainval. †SPPnet results were prepared by the authors of [11].

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

Table 2. **VOC 2010 test** detection average precision (%). BabyLearning uses a network based on [17]. All other methods use VGG16. Training set key: **12**: VOC12 trainval, **Prop.**: proprietary dataset, **12+seg**: **12** with segmentation annotations, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval.

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	**43.0**	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	**38.6**	**68.3**	58.0	68.7	63.3	63.8
R-CNN BB [10]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours]	07++12	**82.3**	**78.4**	**70.8**	**52.3**	38.7	**77.8**	**71.6**	**89.3**	**44.2**	**73.0**	**55.0**	**87.5**	**80.5**	**80.8**	**72.0**	35.1	**68.3**	**65.7**	**80.4**	**64.2**	**68.4**

Table 3. **VOC 2012 test** detection average precision (%). BabyLearning and NUS_NIN_c2000 use networks based on [17]. All other methods use VGG16. Training set key: see Table 2, **Unk.**: unknown.

Key Takeaways

1. Higher detection quality than R-CNN
2. Training is single-stage, using a multi-task loss
3. Training can update all network layers
4. No disk storage is required for feature caching
5. Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Overview

Fast R-CNN achieves near real-time rates using very deep networks, when ignoring the time spent on region proposals.

- But above still relies on Selective Search which is much slower, 2s per image in CPU Implementation. Even with EdgeBoxes it is still 0.2s per image
- Now, proposals are the computational bottleneck in state-of-the-art detection systems

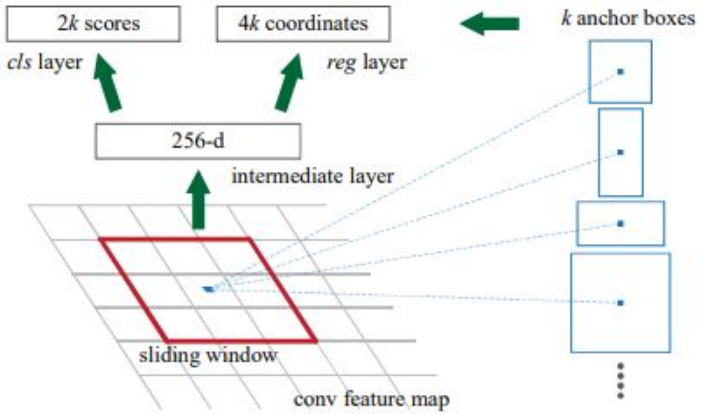
Region finding is still done with CPU, why not add onto GPU for faster processing of input image?

Region Proposal Networks (RPNs)

Takes image of any size as input and outputs a set of rectangular object proposals, each is provided an objectiveness score

At each sliding-window location predict k region proposals where:

- the reg layer has 4k outputs encoding the coordinates of k boxes
- the cls layer outputs 2k scores that estimate probability of object / not-object



Key Takeaways

1. Speeds up region proposals time by processing on GPU rather than a CPU
    a. Computing proposals goes from 0.2s (EdgeBoxes) to ~0.01s (20x improvement)
    b. Nearly cost free!
2. The learned RPN also improves region proposal quality and thus the overall object detection accuracy

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

Overview

- Introduced a new approach to objection detection
  - Single regression problem
    - CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes.
    - You only look at the image once.
- Real-Time Object Detector
  - Tested on Titan X GPU
  - Processes images at 45 fps
    - Fast Yolo at 155 fps
      - Allows for streaming with 25 milliseconds of latency
- Achieves more than 2x mean average precision (mAP) of other real-time systems.

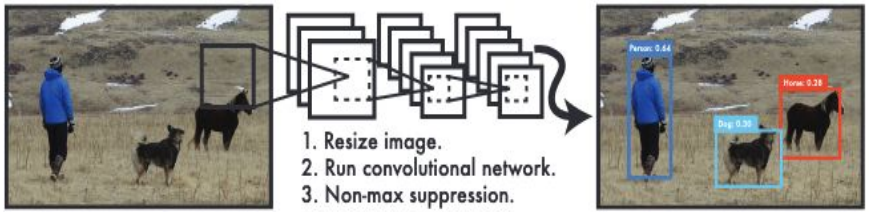


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Training

- Pretrain convolutional layers on ImageNet 1000-class comp dataset.
  - Pretrain first 20 conv layers which are followed by an average-pooling layer and a fully connected layer.
  - Final layer predicts class probabilities and bounding box coordinates.

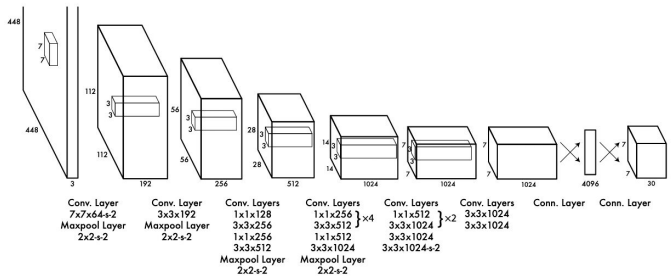


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

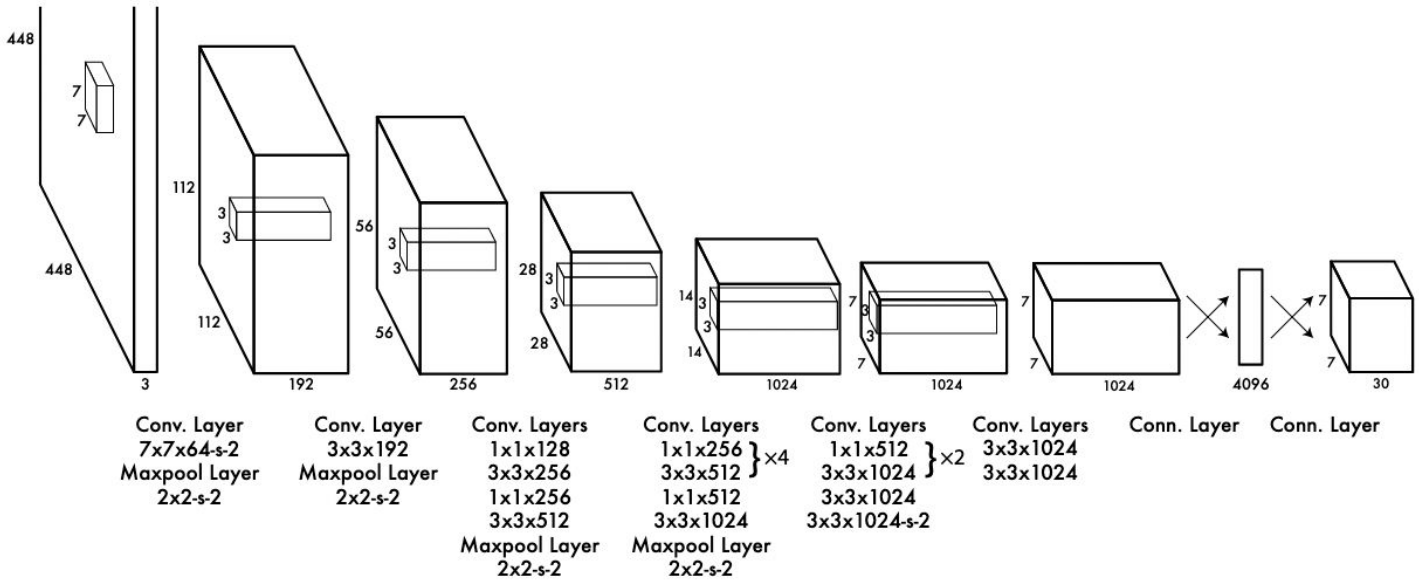


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

object.

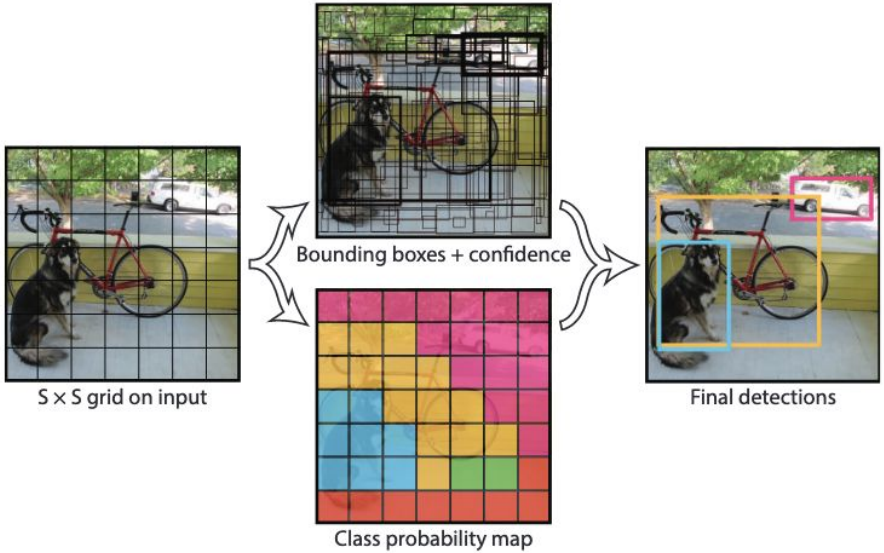
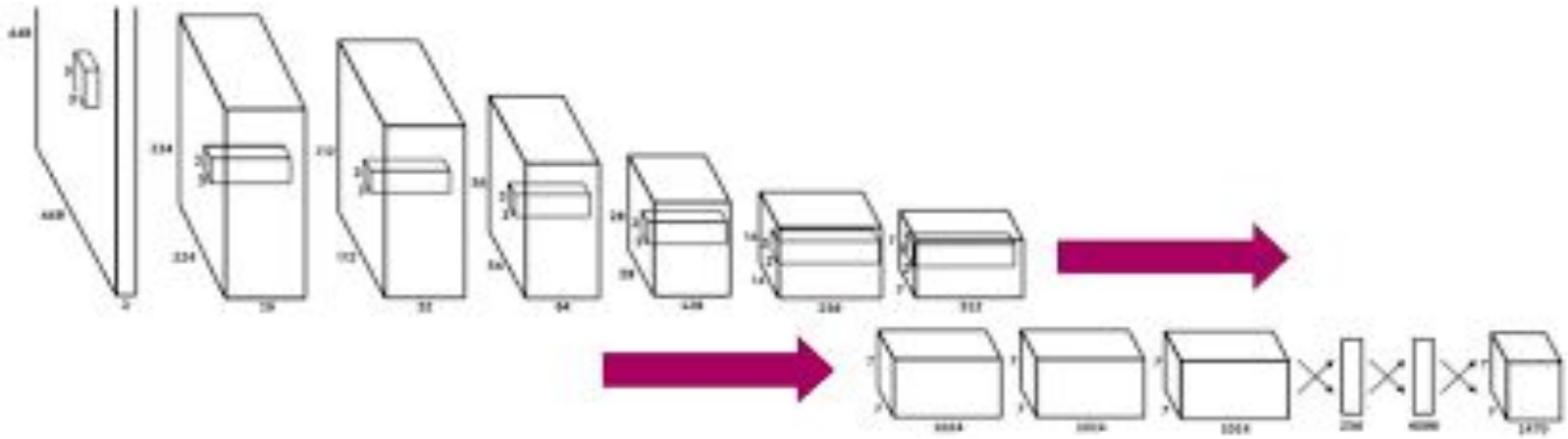


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Fast YOLO

- Fastest object detection method on PASCAL.
- Uses a CNN with 9 convolutional layer, instead of 24.
- Training at testing are the same.



Person Detection in Artwork

- Used pretrained network on Picasso art and People-Art to test generalizability/robustness.
- R-CNN mAP falls off, YOLO maintains accuracy.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

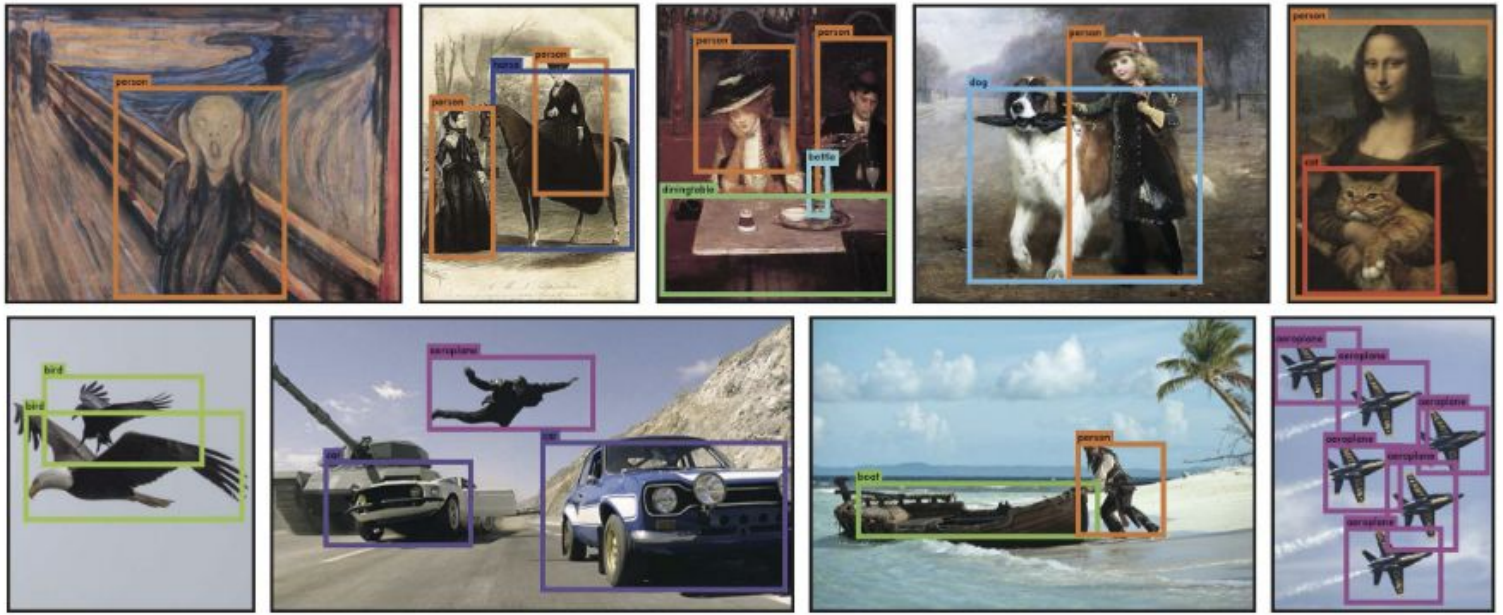


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

Results

- Makes less than half the number of background errors compared to Fast R-CNN
- Best mAP for Real-Time Detectors
- Fastest image detection at the time.

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

Results Cont.

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [27]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [28]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [32]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

Table 3: PASCAL VOC 2012 Leaderboard. YOLO compared with the full `comp4` (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

YOLO VISUAL

https://www.youtube.com/watch?v=ruDXYYIdV1E

https://www.youtube.com/watch?v=MPU2HistivI

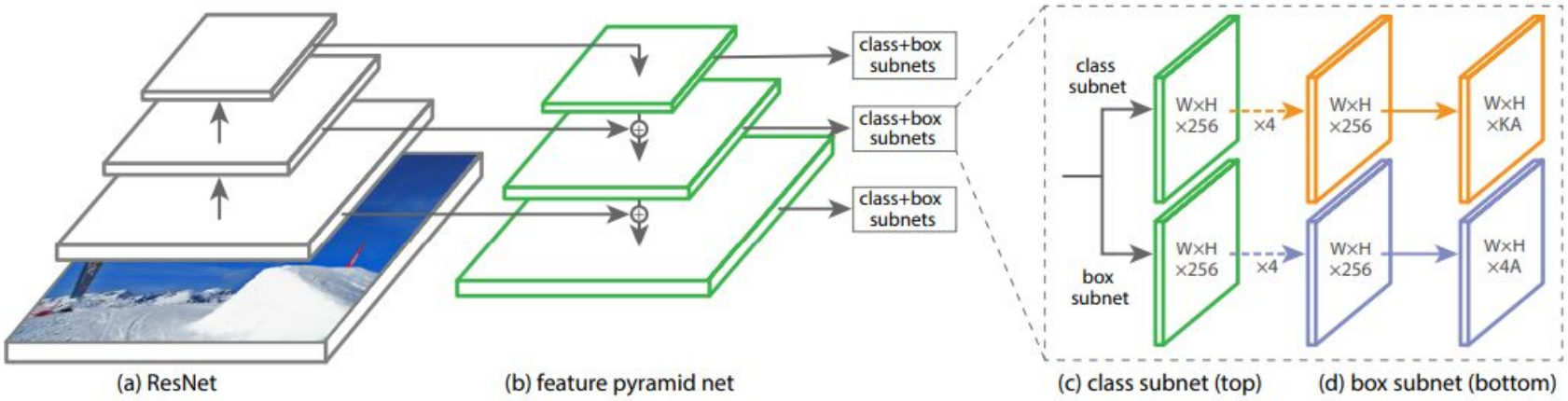
Key Takeaways

1. Trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly
2. Generalizes well to new domains making it ideal for applications that rely on fast, robust object detection
3. Makes more localization errors but is less likely to predict false positives on background

Focal Loss for Dense Object Detection

Overview

- Previously all models are two-stage
  - region proposals and later classifying either background or foreground classes
- It is not simplified to one-stage is due to the extreme foreground-background class imbalance encountered during training of dense detectors
- Propose new method of focal loss which will down-weigh the loss assigned to well-classified examples
- So, create a simple one-stage object detector, RetinaNet, because it does dense sampling of object locations in any input image
  - Only concern is that one-stage detectors must process a much larger set of candidate object locations regularly sampled across an image



a. Generate a rich, multi-scale convolutional feature pyramid
b. Attach 2 subnetworks, one for classifying anchor boxes
c. The other for regressing from anchor boxes to ground-truth object boxes
d. Focuses on novel focal loss function which eliminates the accuracy gap

Key Takeaways

1. Class imbalance as the primary obstacle preventing one-stage object detectors from surpassing top-performing, two-stage methods
2. Propose focal loss which applies a modulating term to the cross entropy loss in order to focus learning on hard negative examples

Discussion Questions

- What other methods from other sections do you think might be helpful to improve performance?

References

- https://www.mathworks.com/discovery/object-detection.html#:~:text=Object%20detection%20is%20a%20computer,learning%20to%20produce%20meaningful%20results
- https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f
- http://host.robots.ox.ac.uk/pascal/VOC/voc2010/index.html
- http://www.huppelen.nl/publications/selectiveSearchDraft.pdf
- https://az659834.vo.msecnd.net/eventsairseasiaprod/production-eecw-public/a67993fffadc49948edb6824ce168fc9#:~:text=Bounding%2Dbox%20regression%20is%20a,pre%2Ddefined%20target%20object%20classes