CY 7790

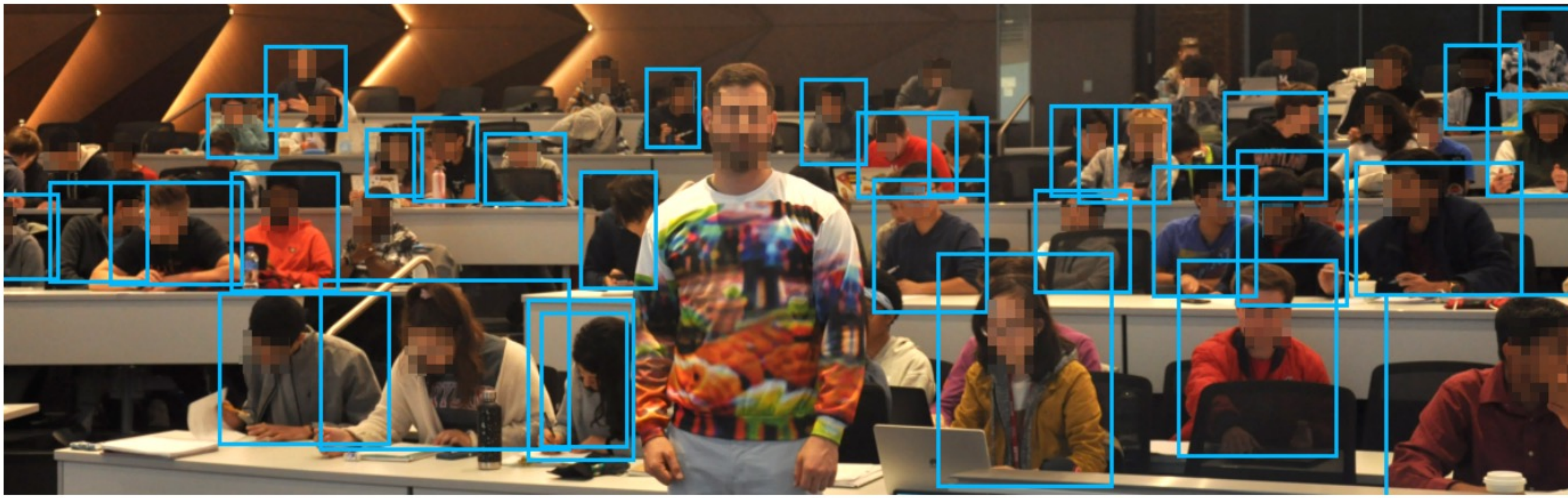
Special Topics in Security and Privacy: Machine Learning Security and Privacy
Fall 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science

October 28 2021

Wu et al. Making an Invisibility Cloak: Real World Adversarial Attacks on Object Detectors. ECCV 2020

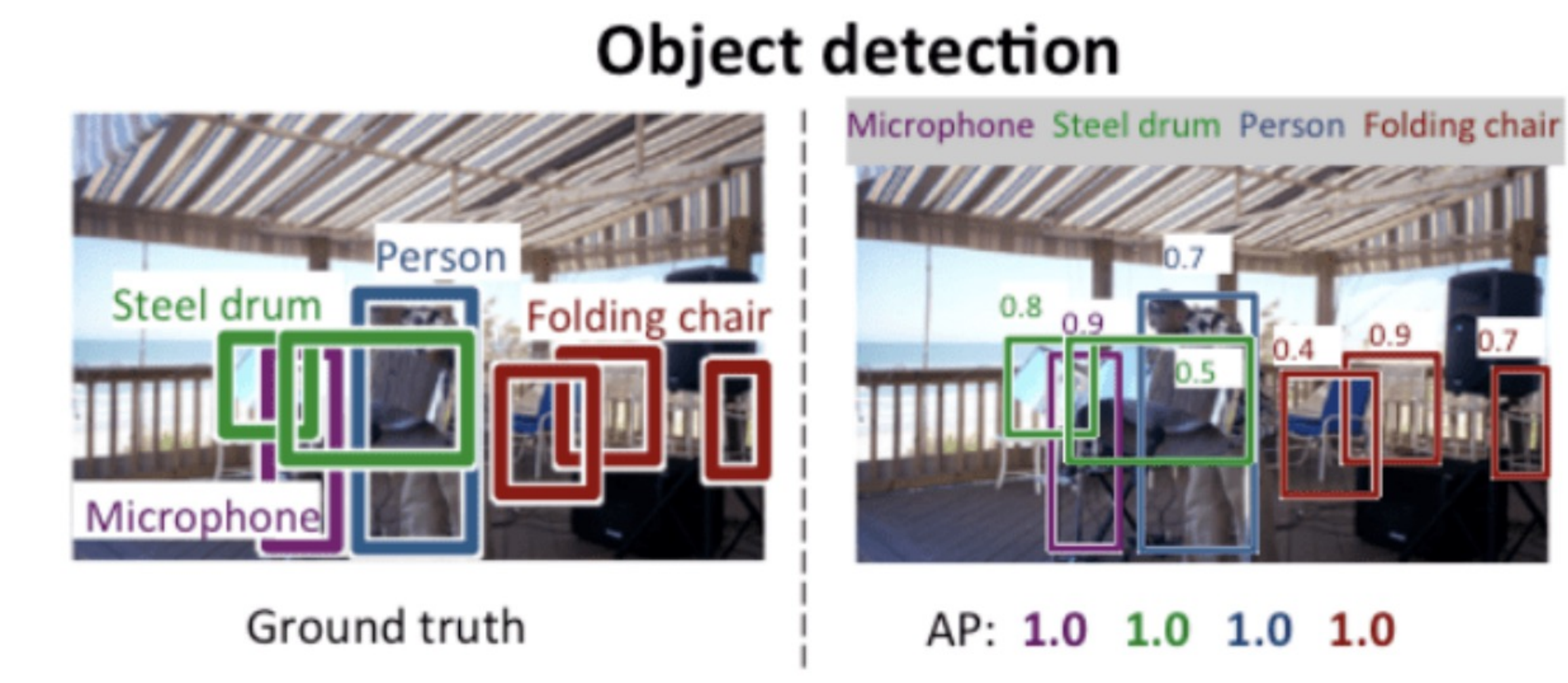
End Goal



Problem Statement

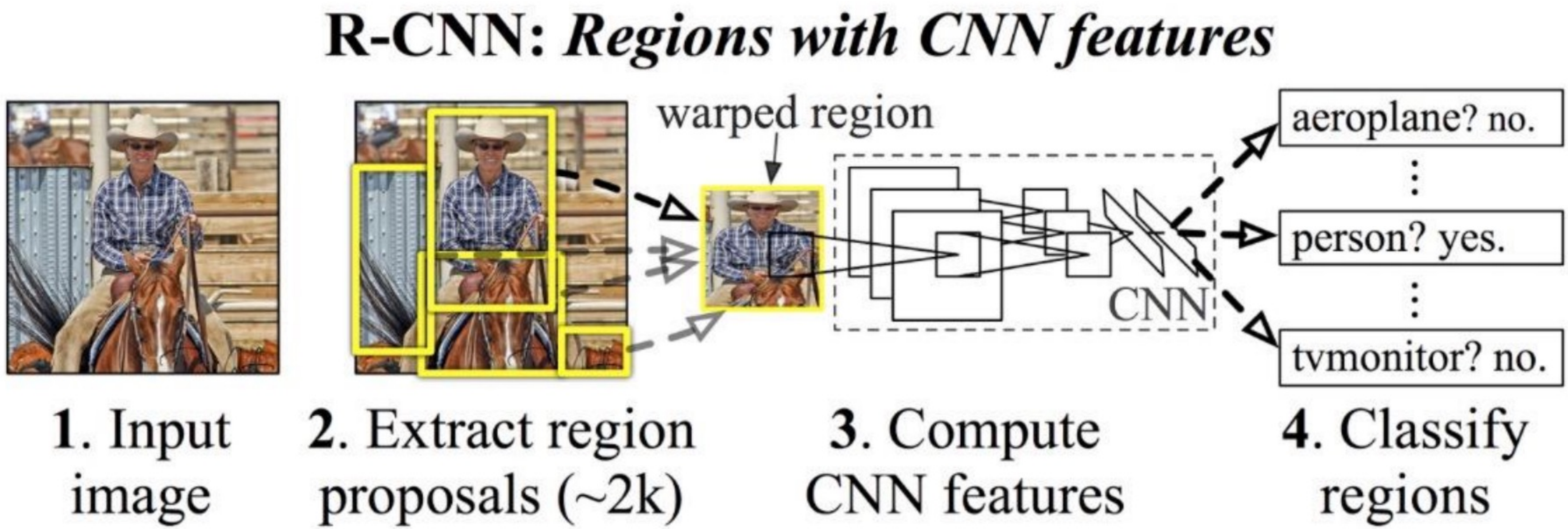
- How to evade object detectors to not identify a particular class (e.g., “person”)
- Goals
  - Universal patch (applicable to all images)
  - Transferable (applies to many types of detectors)
  - Dataset agnostic
  - Robust to viewing conditions
  - Physically realizable (patterns remain adversarial when printed over 3D objects)

Detour on Object Detectors



- Detection is considered accurate if the bounding boxes overlap by a threshold (e.g., 50%)

Existing Object Detectors

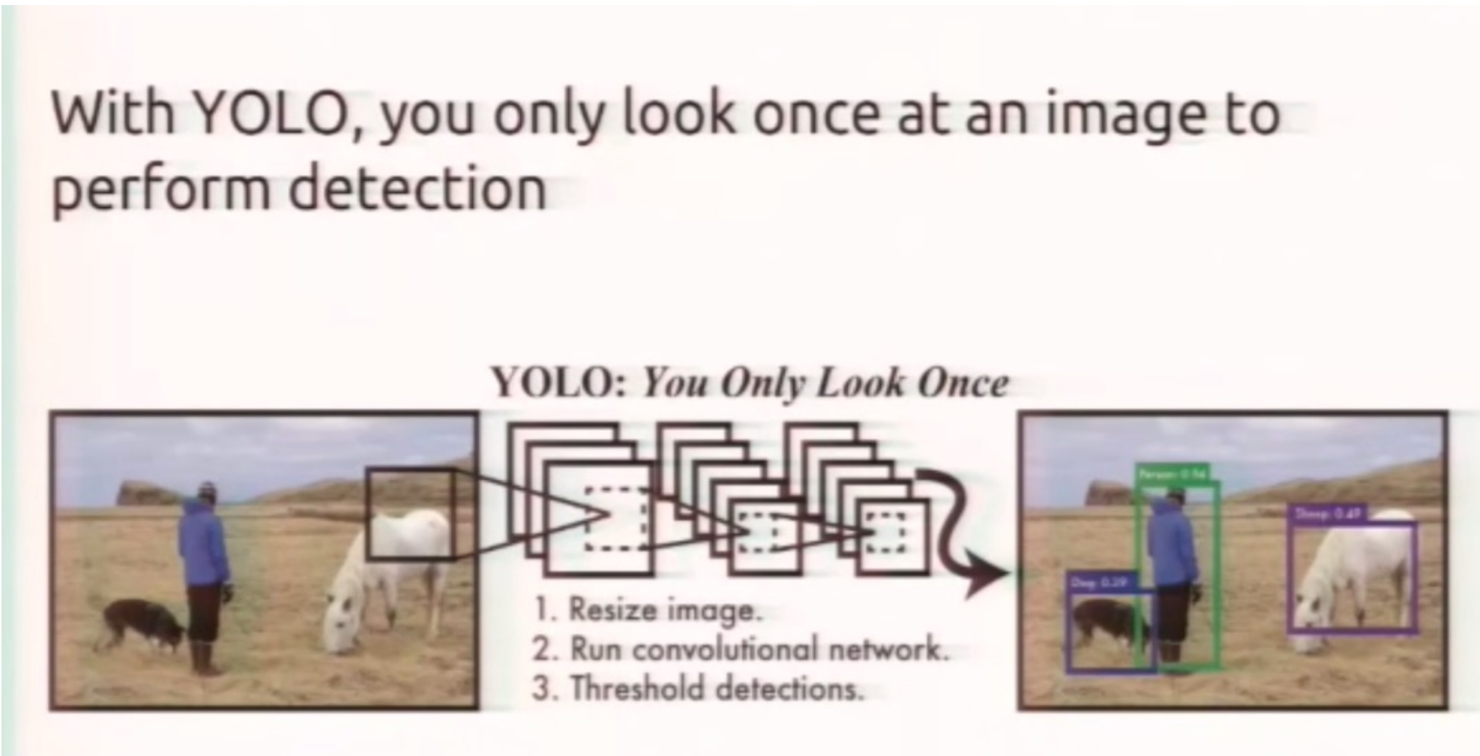


Backbone: Feature extractor network
(e.g., model pre-trained on ImageNet)

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/



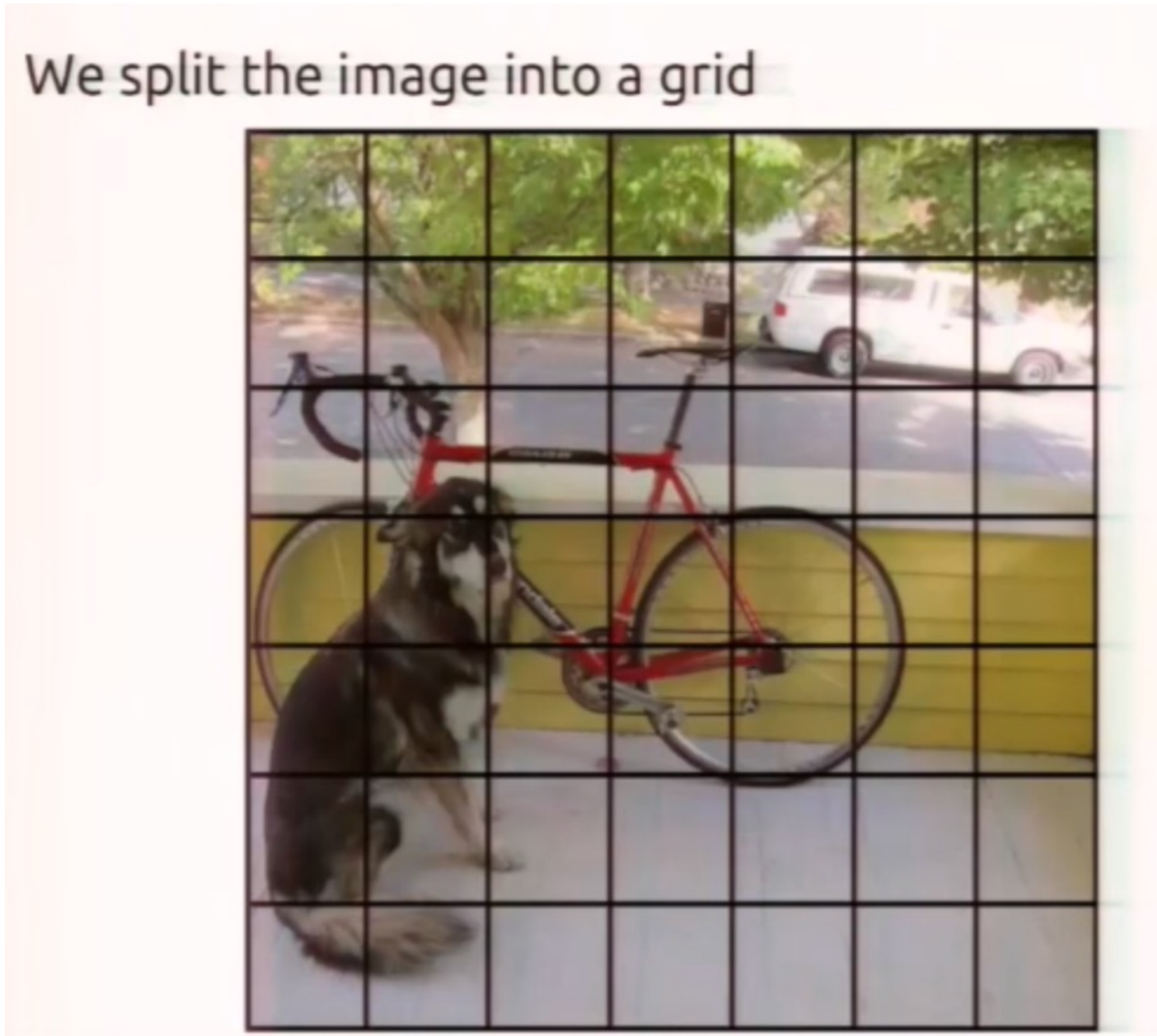
Slides from Joe Redmon, presentation at CVPR 2016

Limitations

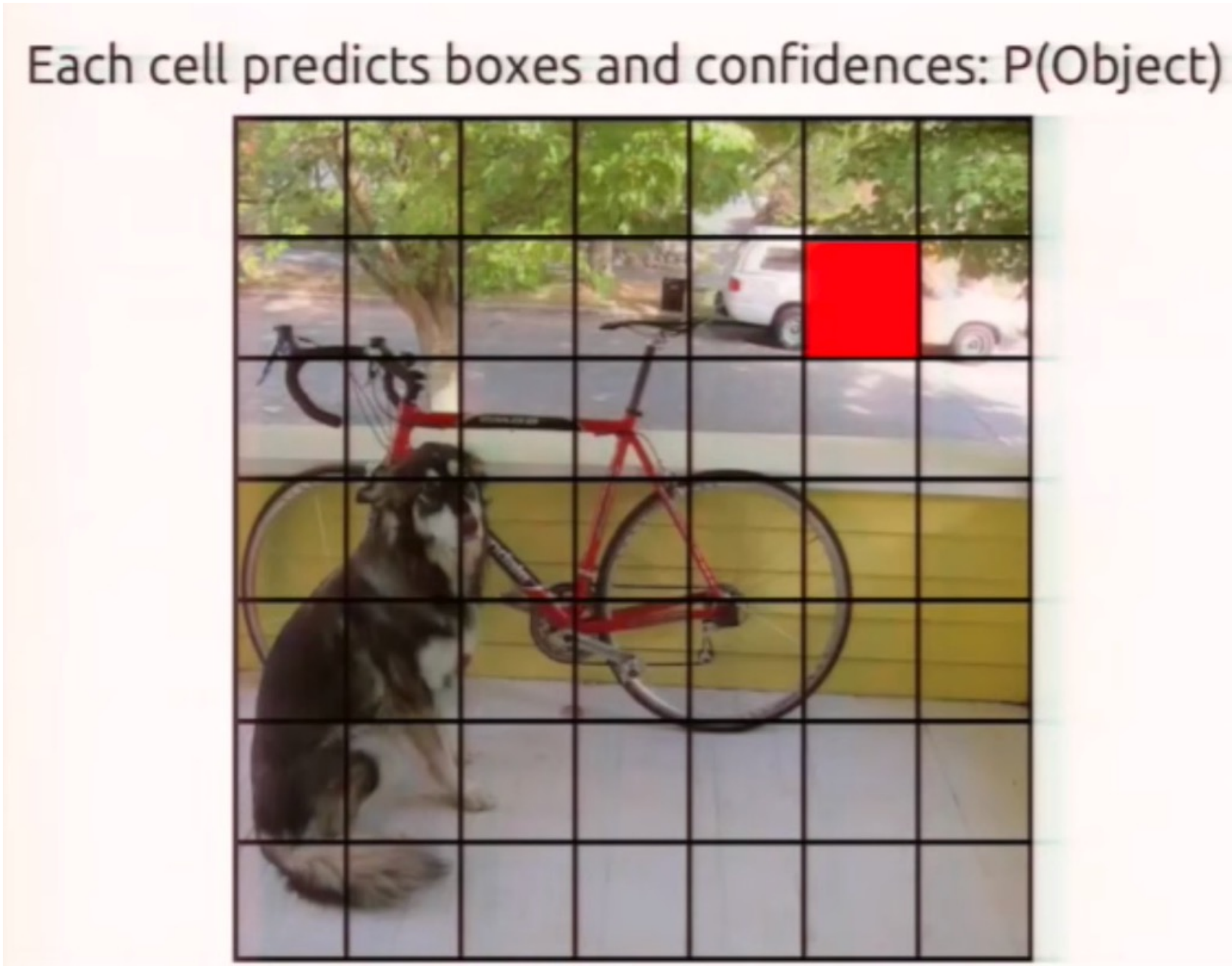
Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img

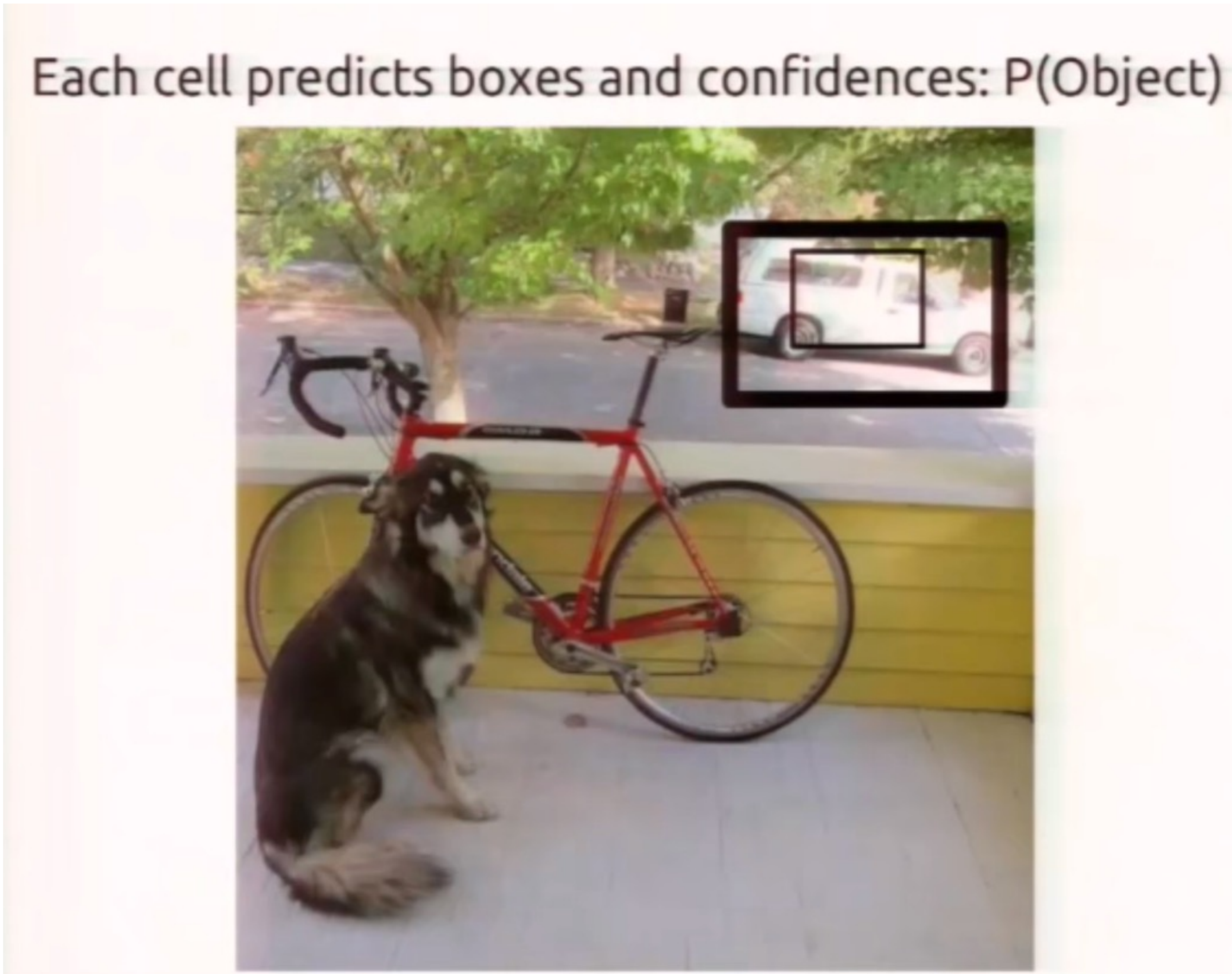
YOLO methodology



YOLO methodology

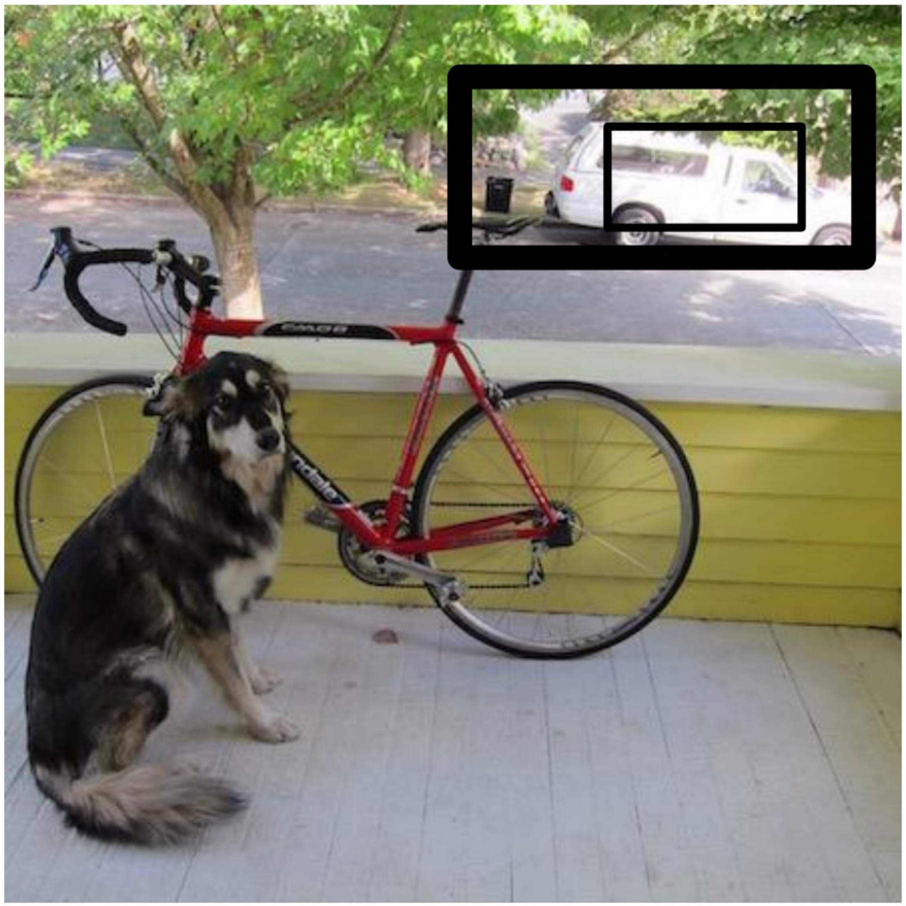


YOLO methodology

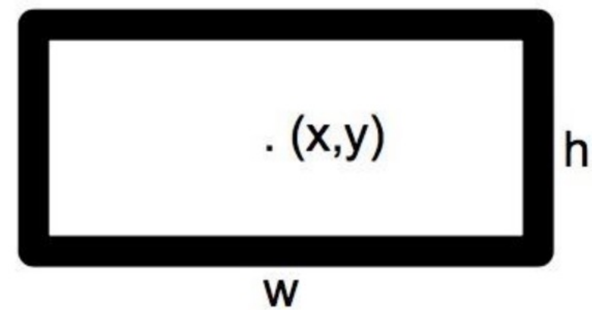


Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)

B = 2

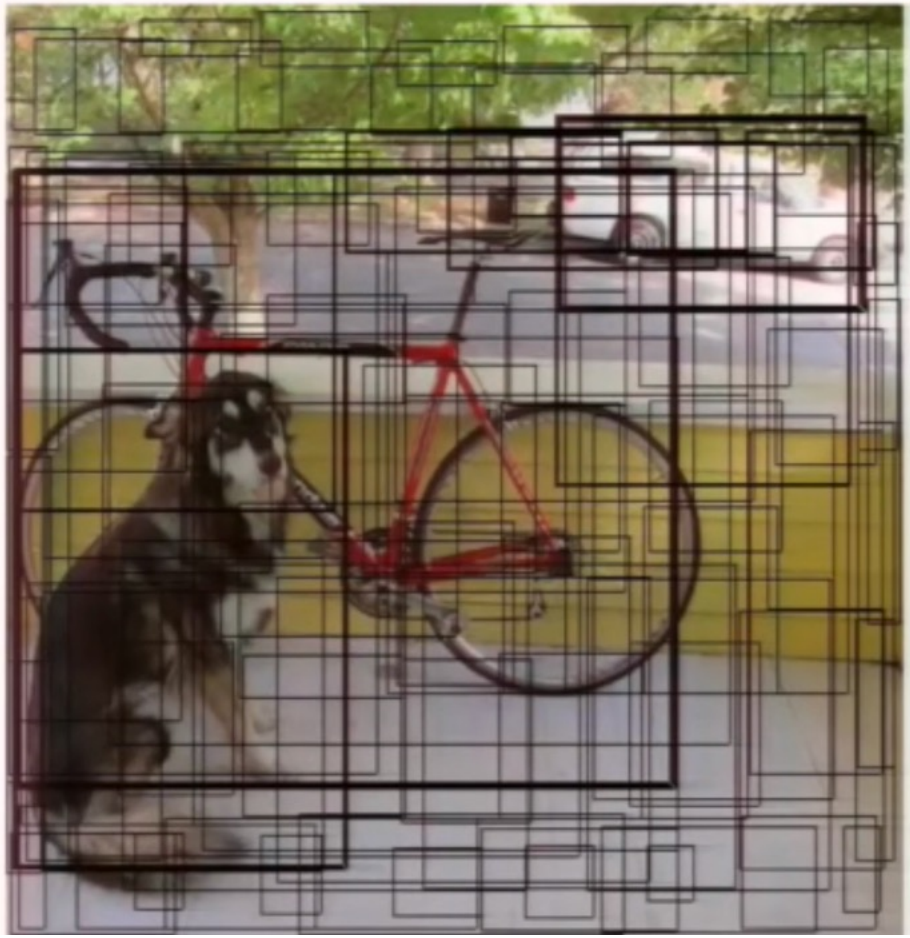


each box predict:

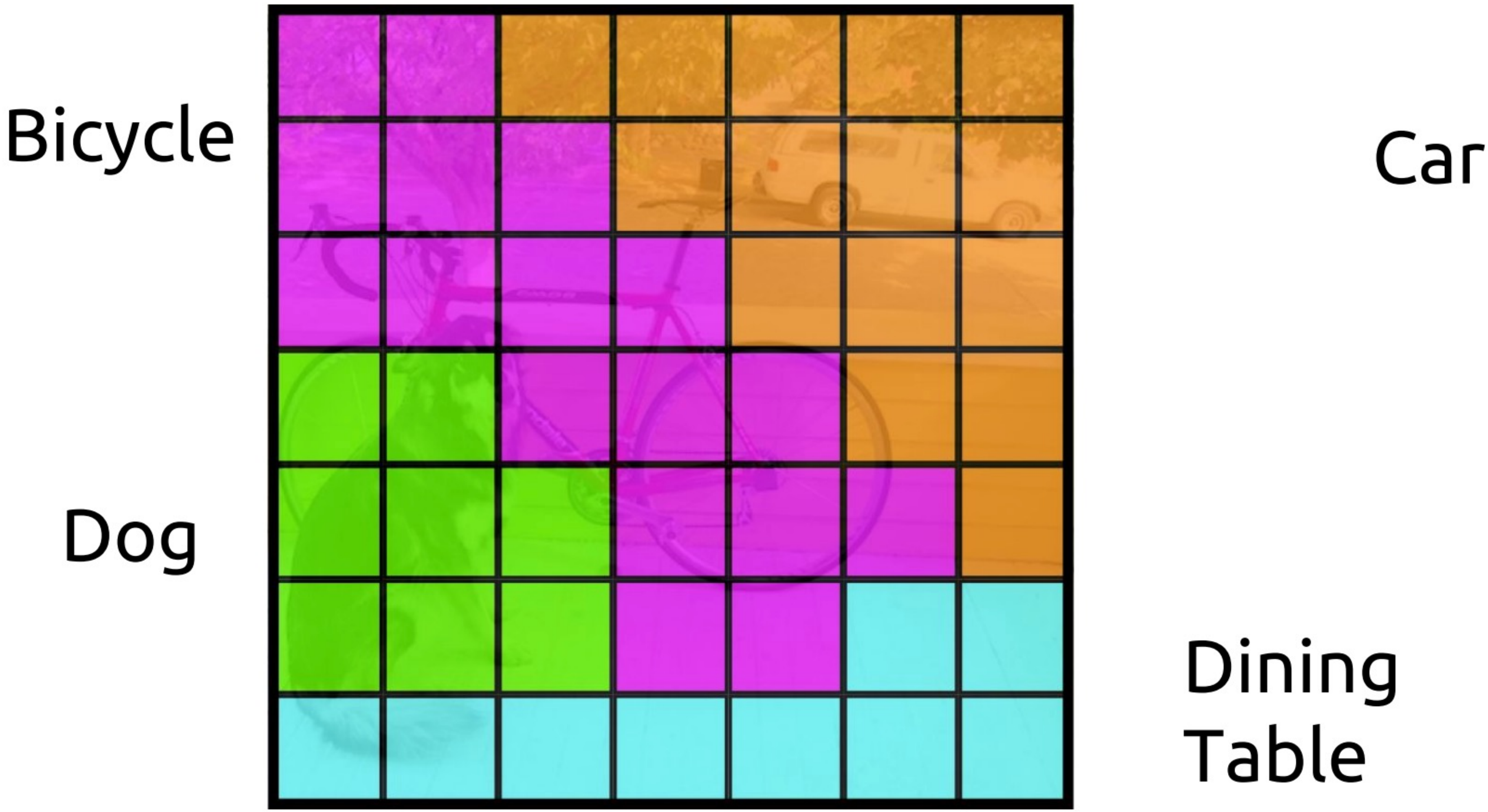


P(Object): probability that the box contains an object

Each cell predicts boxes and confidences: P(Object)



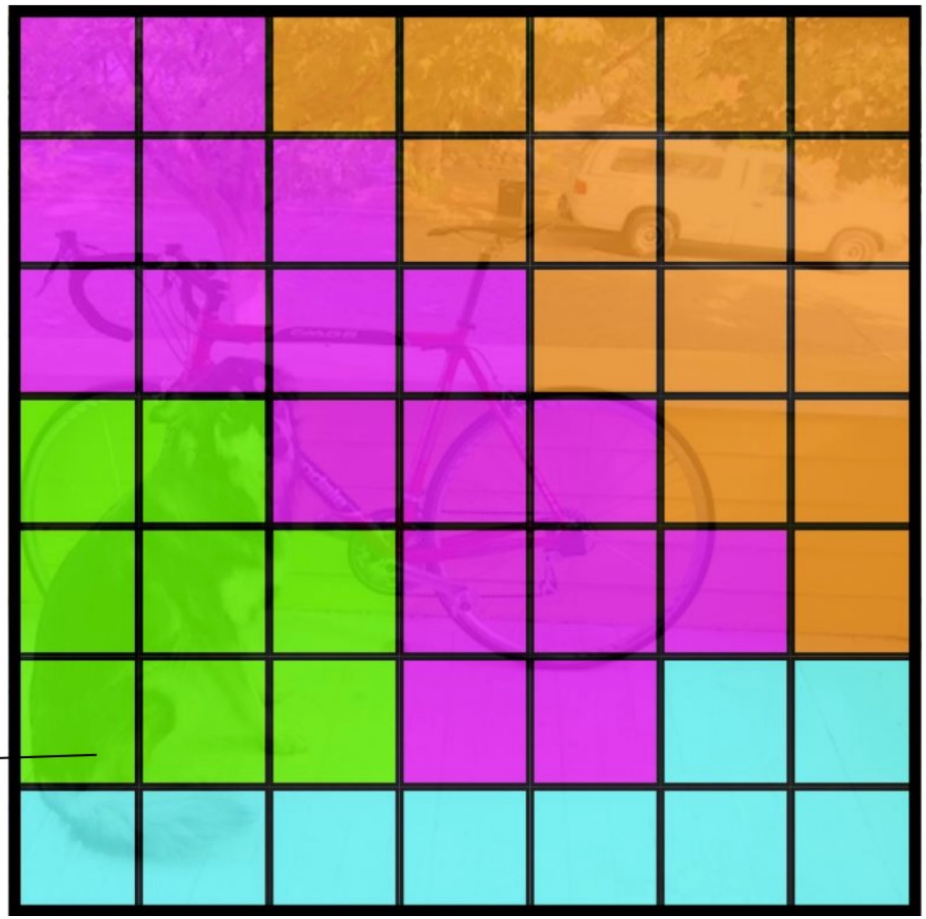
Each cell also predicts a class probability.



Conditioned on object: P(Car | Object)

Bicycle

Car

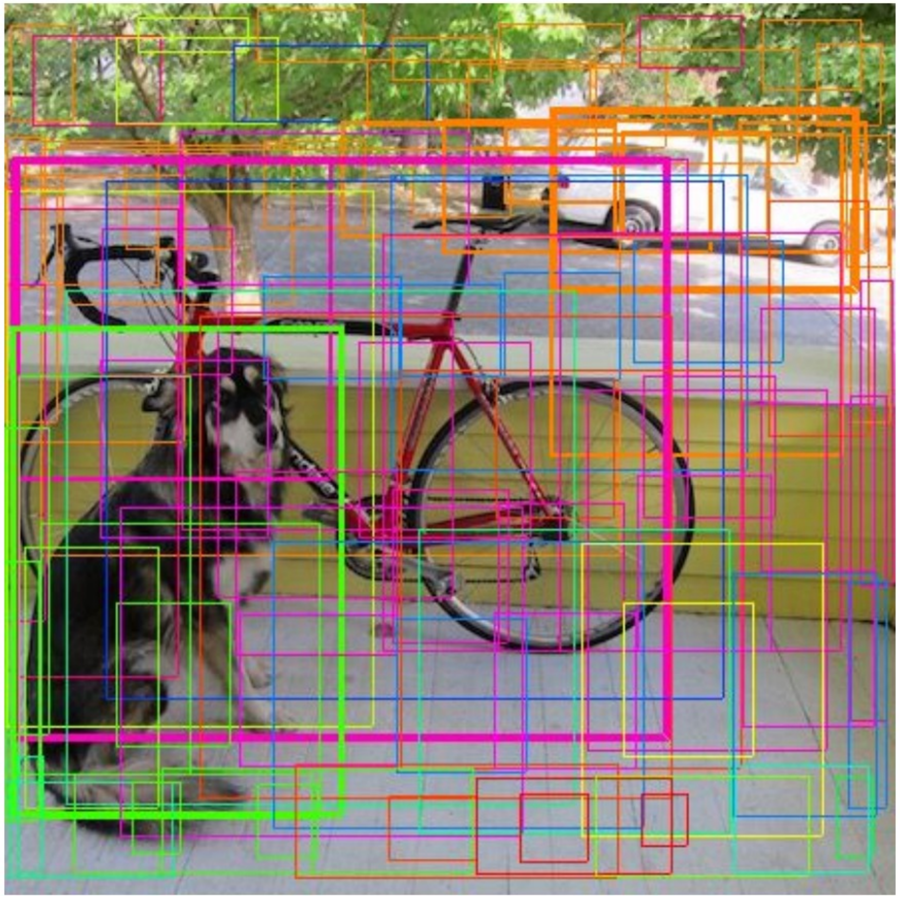


Dog

Eg.
Dog = 0.8
Cat = 0
Bike = 0

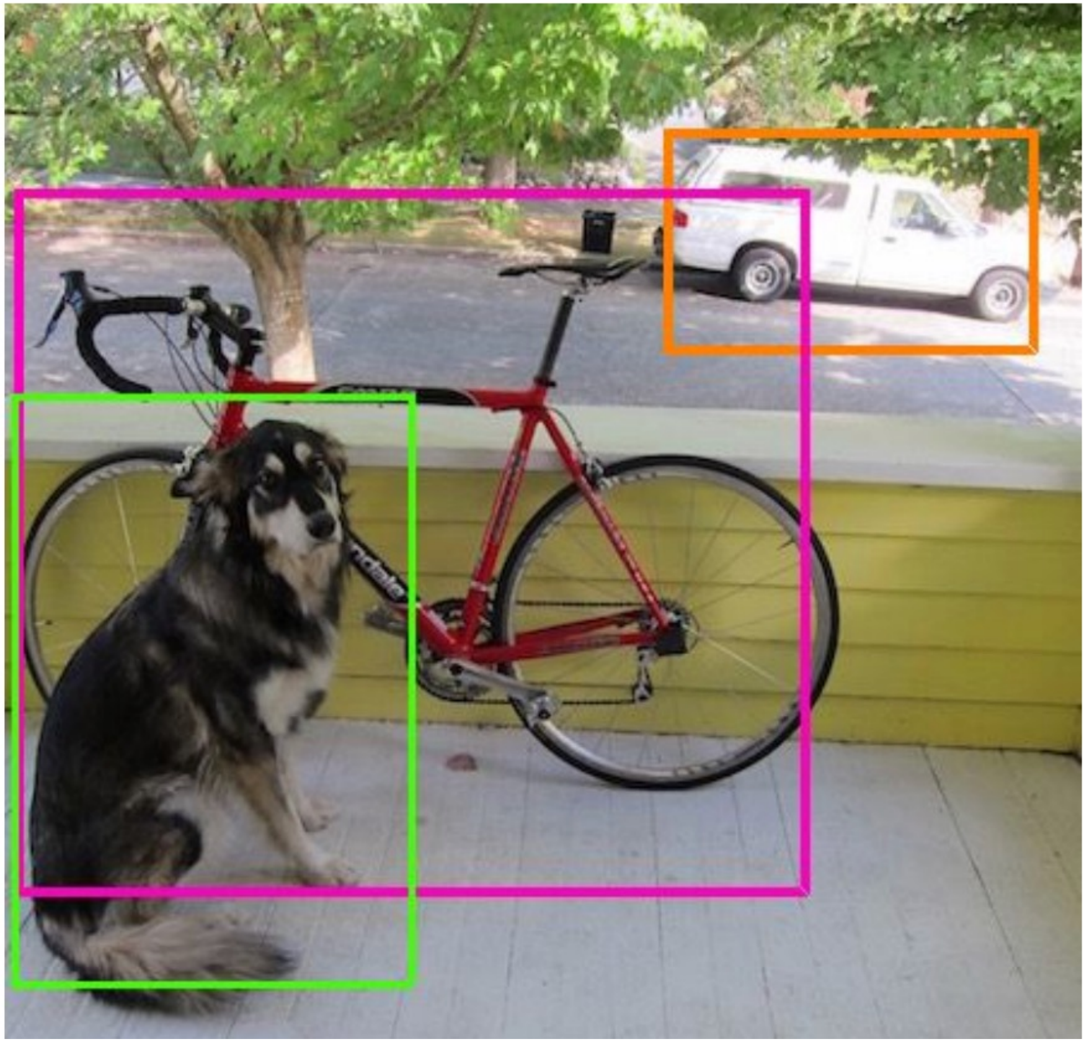
Dining Table

Then we combine the box and class predictions.



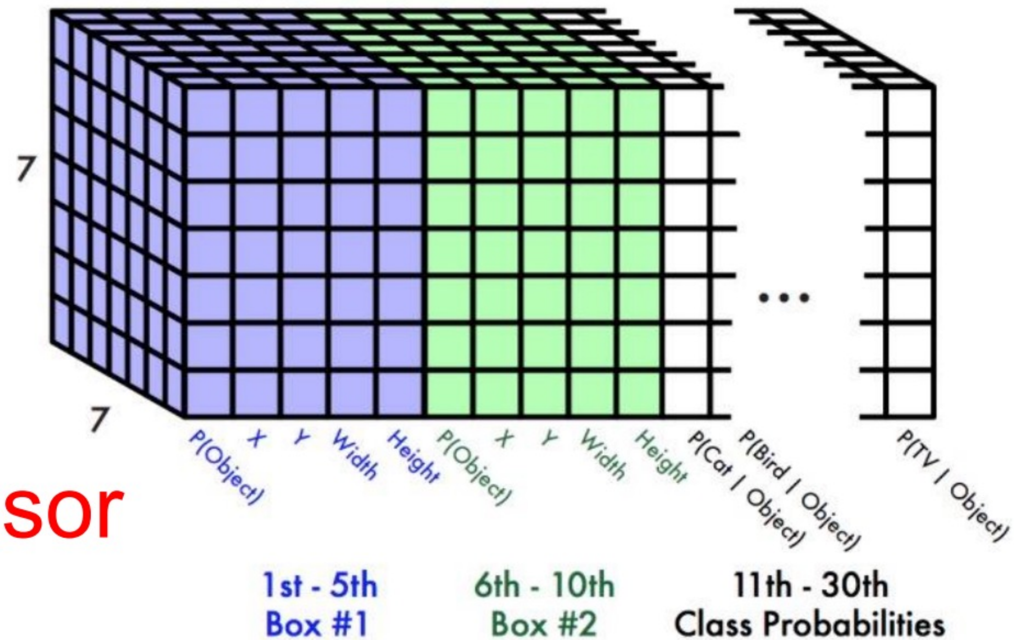
$P(class|Object) * P(Object) = P(class)$

Finally we do threshold detections and NMS



Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities



S * S * (B * 5 + C) tensor

Given ground truth, it maximize P[Object] for the bounding box, and P[Correct Class|Object], while it minimizes P[Object] if no object is present and the conditional probabilities of other classes.

Concrete Parameterization

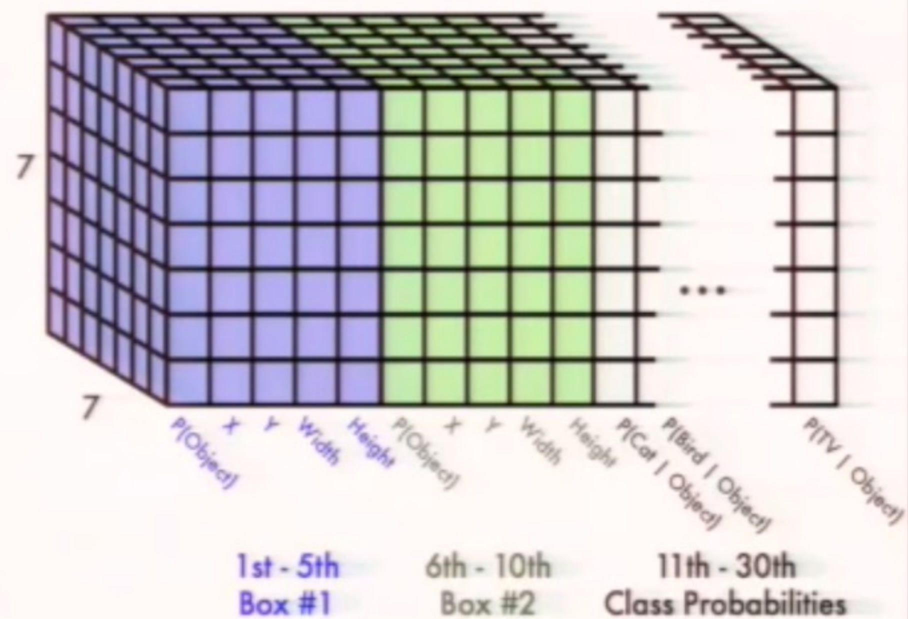
This parameterization fixes the output size

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

For Pascal VOC:

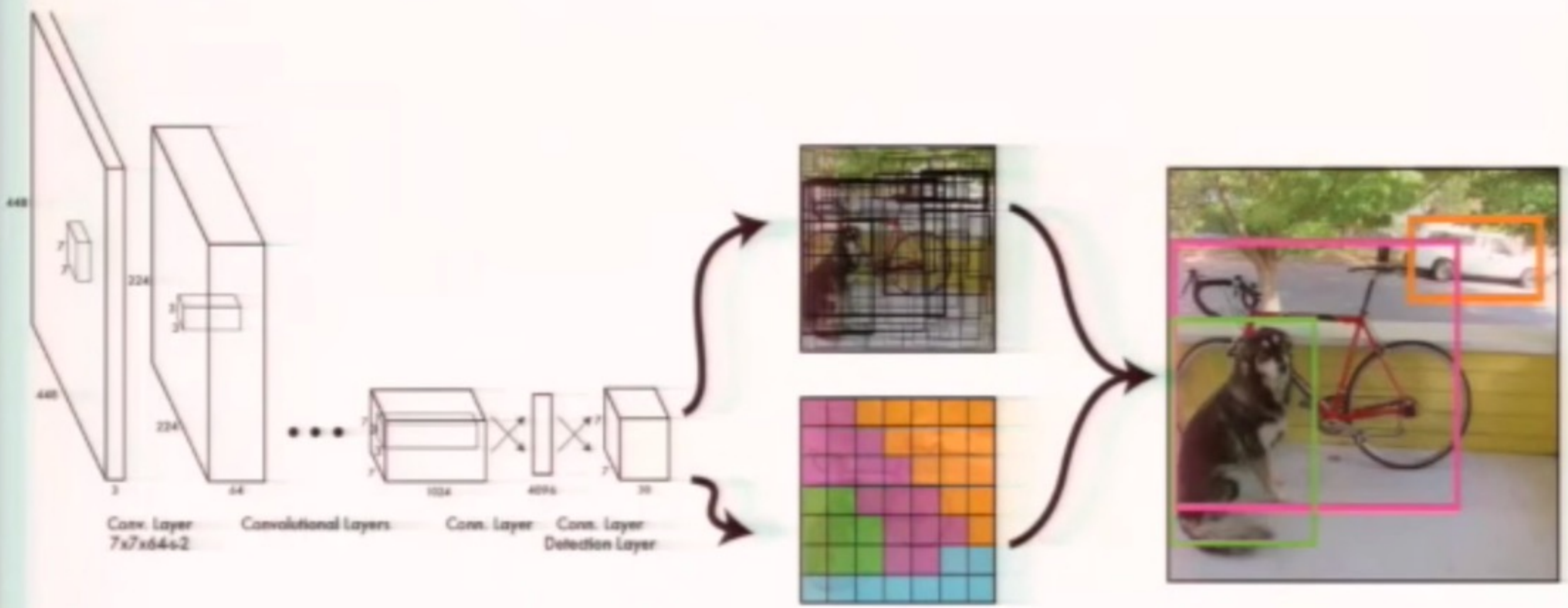
- 7x7 grid
- 2 bounding boxes / cell
- 20 classes



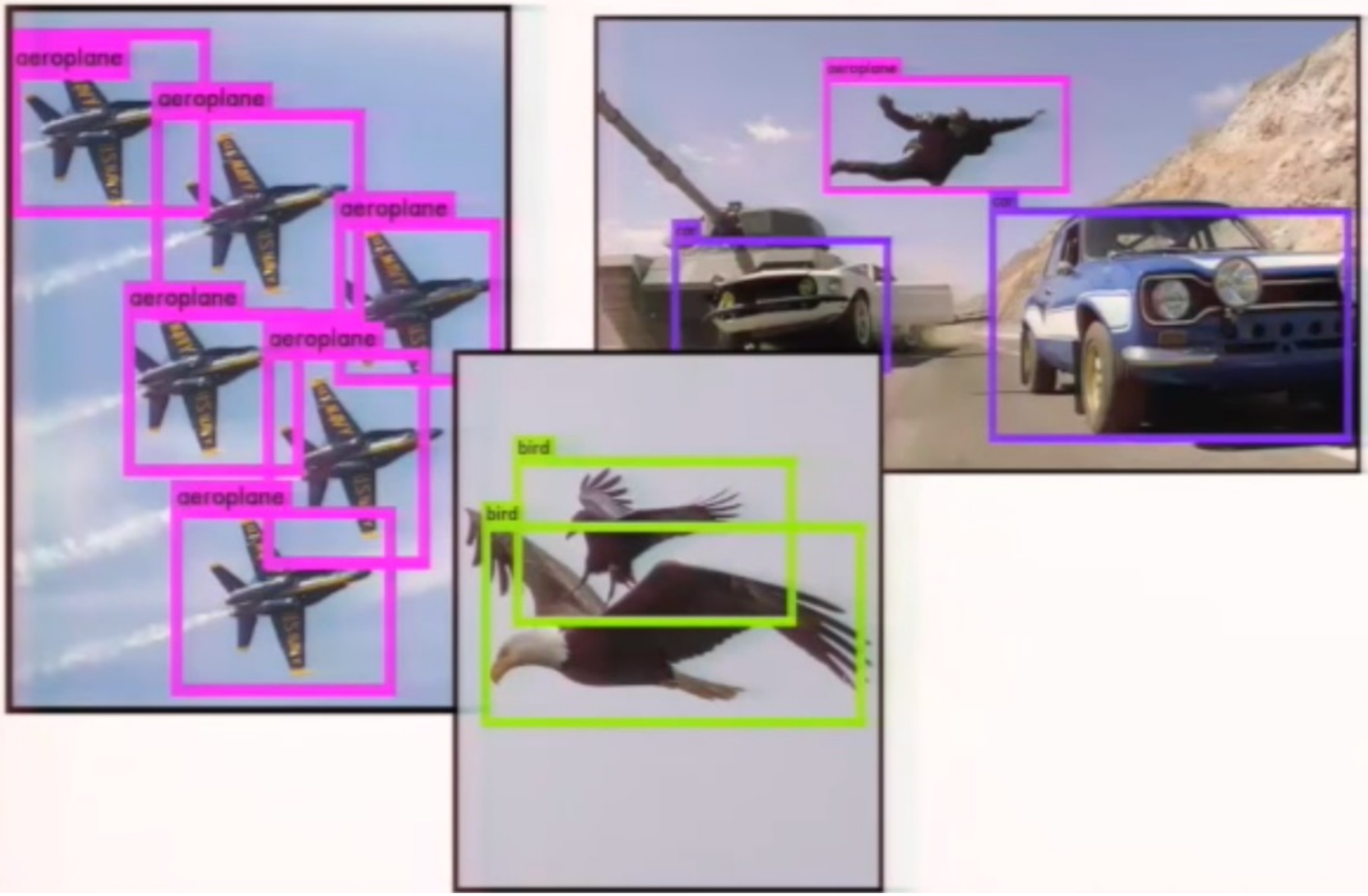
$7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30$ tensor = **1470 outputs**

Training YOLO

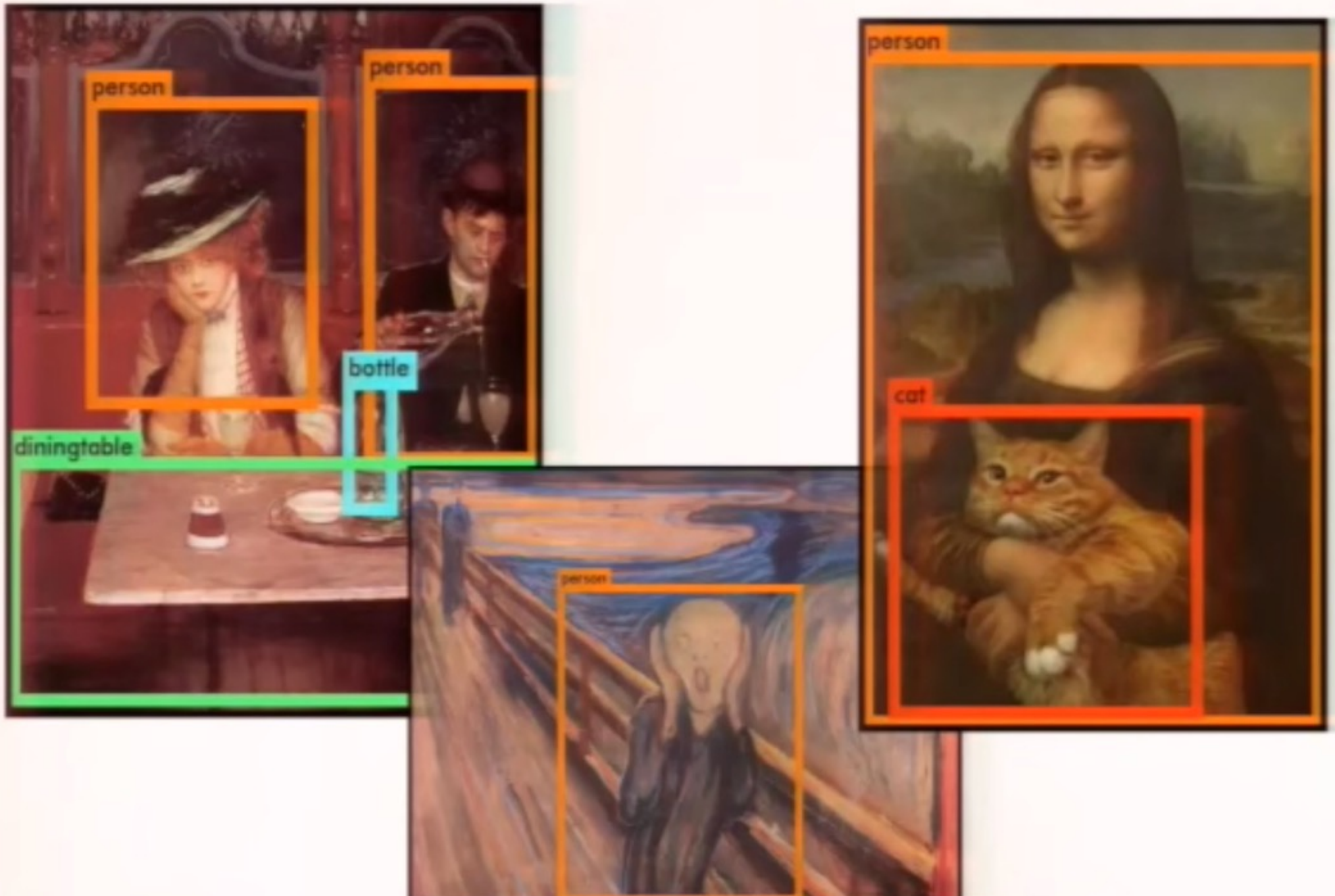
Thus we can train one neural network to be a whole detection pipeline



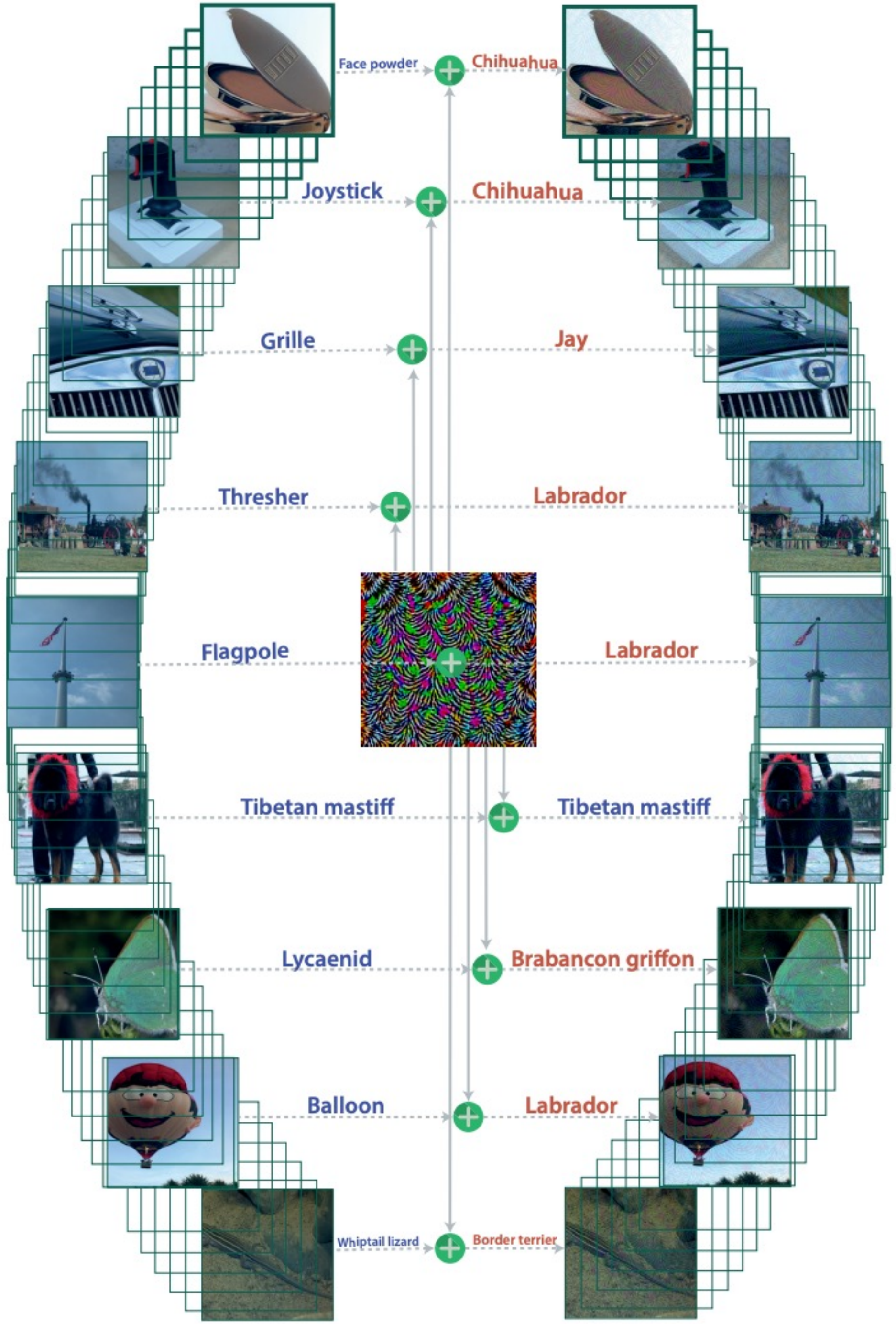
YOLO works across a variety of natural images



It also generalizes well to new domains (like art)



Universal Adversarial Perturbations Moosavi-Dezfooli et al. 2017



Robust Physical-World Attacks on Deep Learning Visual Classification

Kevin Eykholt*[1], Ivan Evtimov*[2], Earlence Fernandes[2], Bo Li[3], Amir Rahmati[4], Chaowei Xiao[1], Atul Prakash[1], Tadayoshi Kohno[2], and Dawn Song[3]

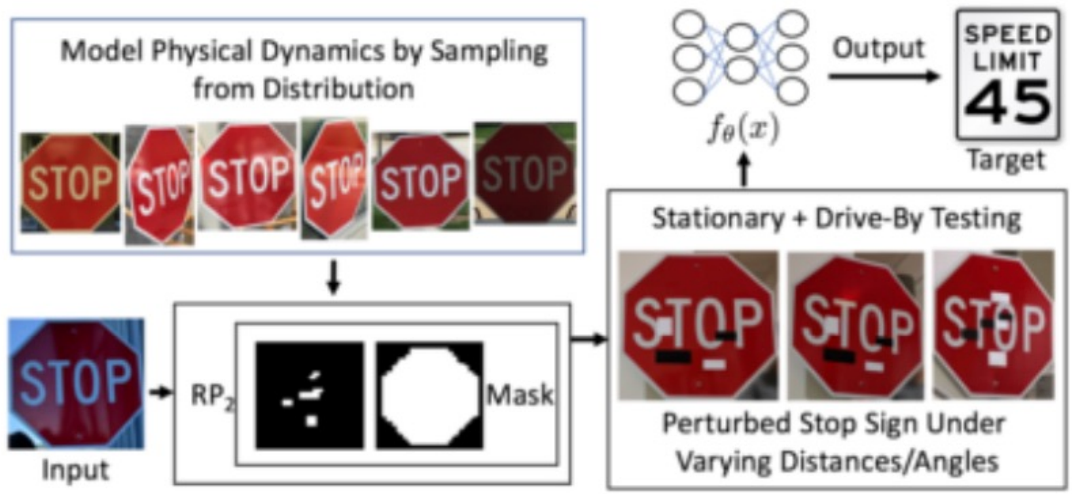


Figure 2: RP_2 pipeline overview. The input is the target Stop sign. RP_2 samples from a distribution that models physical dynamics (in this case, varying distances and angles), and uses a mask to project computed perturbations to a shape that resembles graffiti. The adversary prints out the resulting perturbations and sticks them to the target Stop sign.

Back to Problem Statement

- How to evade object detectors to not identify a particular class (e.g., “person”)
- Goals
  - Universal patch (applicable to all images)
  - Transferable (applies to many types of detectors)
  - Dataset agnostic
  - Robust to viewing conditions
  - Physically realizable (patterns remain adversarial when printed over 3D objects)
- Attack model: white-box and black-box

Why it's a hard problem

Why are detectors hard to fool? A detector usually produces hundreds or thousands of priors that overlap with an object. Usually, non-maximum supression (NMS) is used to select the bounding box with highest confidence, and reject overlapping boxes of lower confidence so that an object is only detected once. Suppose an adversarial attack evades detection by one prior. In this case, the NMS will simply select a different prior to represent the object. For an object to be completely erased from an image, the attack must simultaneously fool the ensemble of all priors that overlap with the object—a much harder task than fooling the output of a single classifier.

Overview

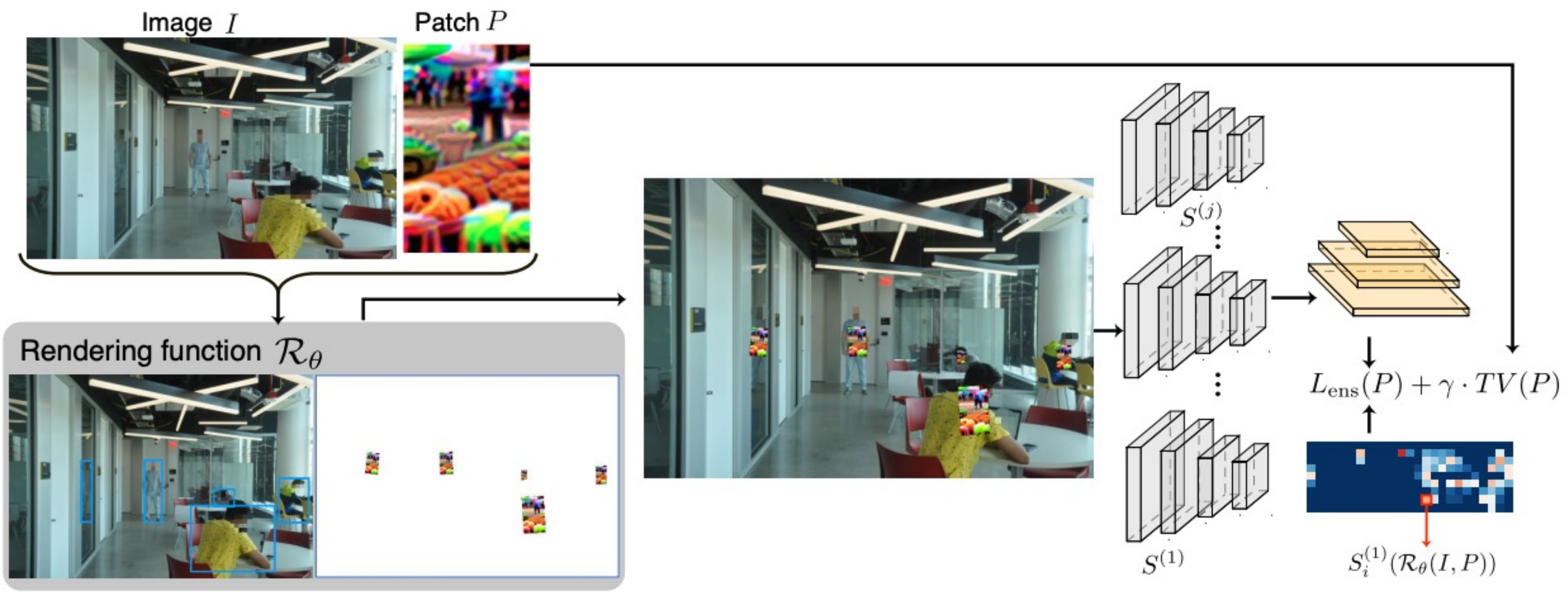


Fig. 2: **An overview of the framework**. Given a patch and an image, the rendering function uses translations and scalings, plus random augmentations, to overlay the patch onto detected persons. The patch is then updated to minimize the objectness scores produced by a detector while maintaining its smoothness.

Optimization

A detector network takes a patched image $\mathcal{R}_\theta(I, P)$ as its input, and outputs a vector of objectness scores, $\mathcal{S}(\mathcal{R}_\theta(I, P))$ one for each prior. These scores rank general objectness for a two-stage detector, and the strength of the "person" class for a one-stage detectors. A positive score is taken to mean that an object/person overlaps with the corresponding prior, while a negative score denotes the absence of a person. To minimize objectness scores, we use the objectness loss function

$$L_{\text{obj}}(P) = \mathbb{E}_{\theta, I} \sum_i \max\{\mathcal{S}_i(\mathcal{R}_\theta(I, P)) + 1,\, 0\}^2. \tag{1}$$

Expectation over:

- Set of images I (universal)
- Set of transforms θ: brightness, contrast, rotation, translation (robustness to distortions)

- Optimum of 0 is achieved for all scores lower than -1 (equivalent to "person" not detected)
- The more positive the score is, the larger the loss

Transferability

Ensemble training. To help adversarial patterns generalize to detectors that were not used for training (*i.e.*, to create a black-box attack), we also consider training patches that fool an ensemble of detectors. In this case we replace the objectness loss (1) with the ensemble loss

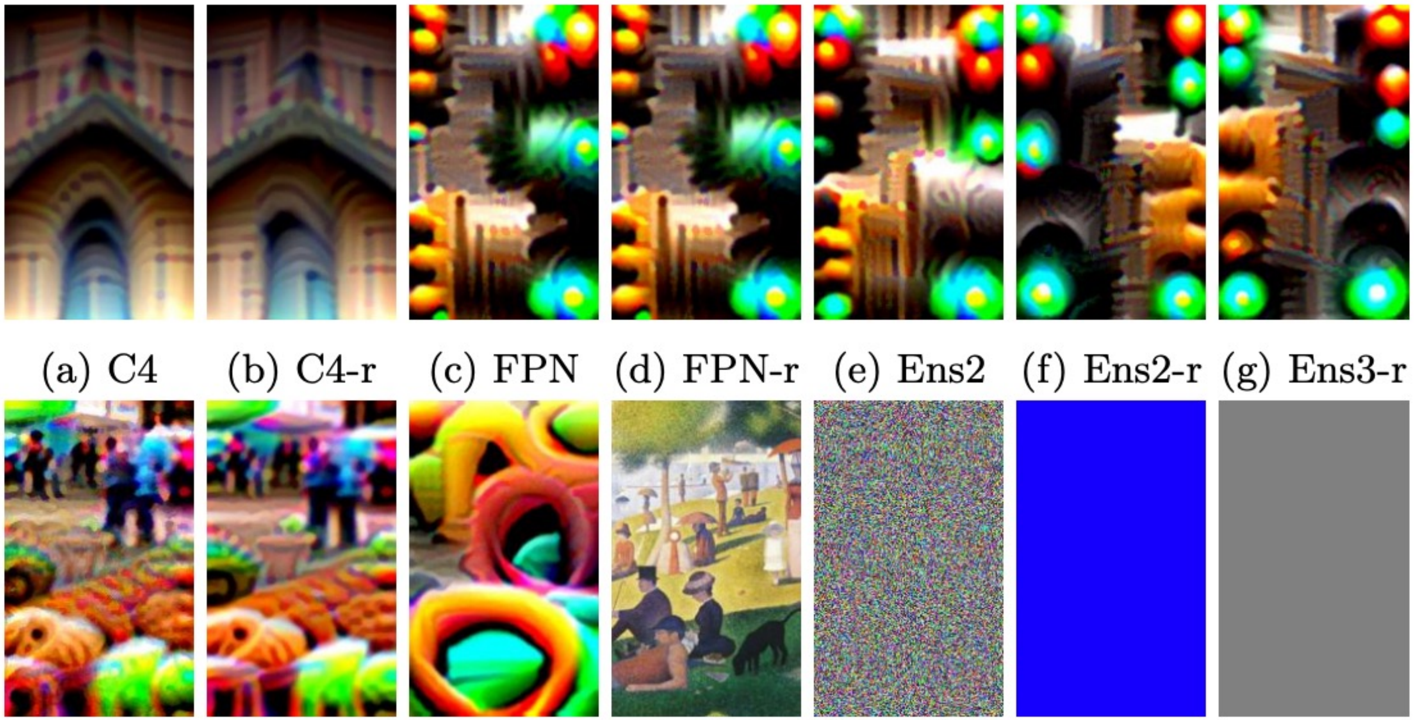
$$L_{\text{ens}}(P) = \mathbb{E}_{\theta,I} \sum_{i,j} \max\{\mathcal{S}_i^{(j)}(\mathcal{R}_\theta(I,P)) + 1,\, 0\}^2, \tag{3}$$

where $\mathcal{S}^{(j)}$ denotes the jth detector in an ensemble.

Evaluation

- COCO dataset
  - 123,000 samples
  - Select 10,000 images with people for training
- Object detectors
  - YOLOv1, YOLOv3
  - Two-stage: R50-C4, R50- FPN
- Metrics
  - Average Precision (AP): Area under the Precision-Recall curve
- Digital and Physical World experiments

Digital Results: Patches



(a) C4 (b) C4-r (c) FPN (d) FPN-r (e) Ens2 (f) Ens2-r (g) Ens3-r

(h) Fted (i) Yolov2 (j) Yolov3 (k) Seurat (l) Rand (m) G++ (n) Grey

Fig. 3: **Adversarial patches**, and comparisons with control patches. Here, (a)-(d) are based on R50, and G++ denotes Grey++.

Digital Results

Patch \ Victim	R50-C4	R50-C4-r	R50-FPN	R50-FPN-r	YOLOv2	YOLOv2-r	YOLOv3	YOLOv3-r
R50-C4	24.5	24.5	31.4	31.4	37.9	42.6	57.6	48.3
R50-C4-r	25.4	23.9	30.6	30.2	37.7	42.1	57.5	47.4
R50-FPN	20.9	21.1	23.5	19.6	22.6	12.9	40.2	40.3
R50-FPN-r	21.5	21.7	25.4	18.8	17.6	11.2	37.5	36.9
YOLOV2	21.1	19	21.5	21.4	10.7	7.5	18.1	25.7
YOLOV3	28.3	28.9	31.5	27.2	20	15.9	17.8	36.1
FTED	25.6	23.9	24.2	24.4	18.9	16.4	31.6	28.2
ENS2	20	20.3	23.2	19.3	17.5	11.3	39	38.8
ENS2-r	19.7	20.2	23.3	16.8	14.9	9.7	36.3	34.1
ENS3-r	21.1	21.4	24.2	17.4	13.4	9.0	29.8	33.6
SEURAT	47.9	52	51.6	52.5	43.4	39.5	62.6	57.1
RANDOM	53	58.2	59.8	59.7	52	52.5	70	63.5
GREY	45.9	49.6	50	50.8	48	47.1	65.6	57.5
GREY++	46.5	49.8	51.4	52.7	48.5	49.4	64.8	58.6
CLEAN	78.7	78.7	82.2	82.1	63.6	62.7	81.6	74.5

Table 1: **Impact of different patches on various detectors, measured using average precision (AP).** The left axis lists patches created by different methods, and the top axis lists different victim detectors. Here, "r" denotes retrained weights instead of pretrained weights downloaded from model zoos.

Transferability Across Datasets

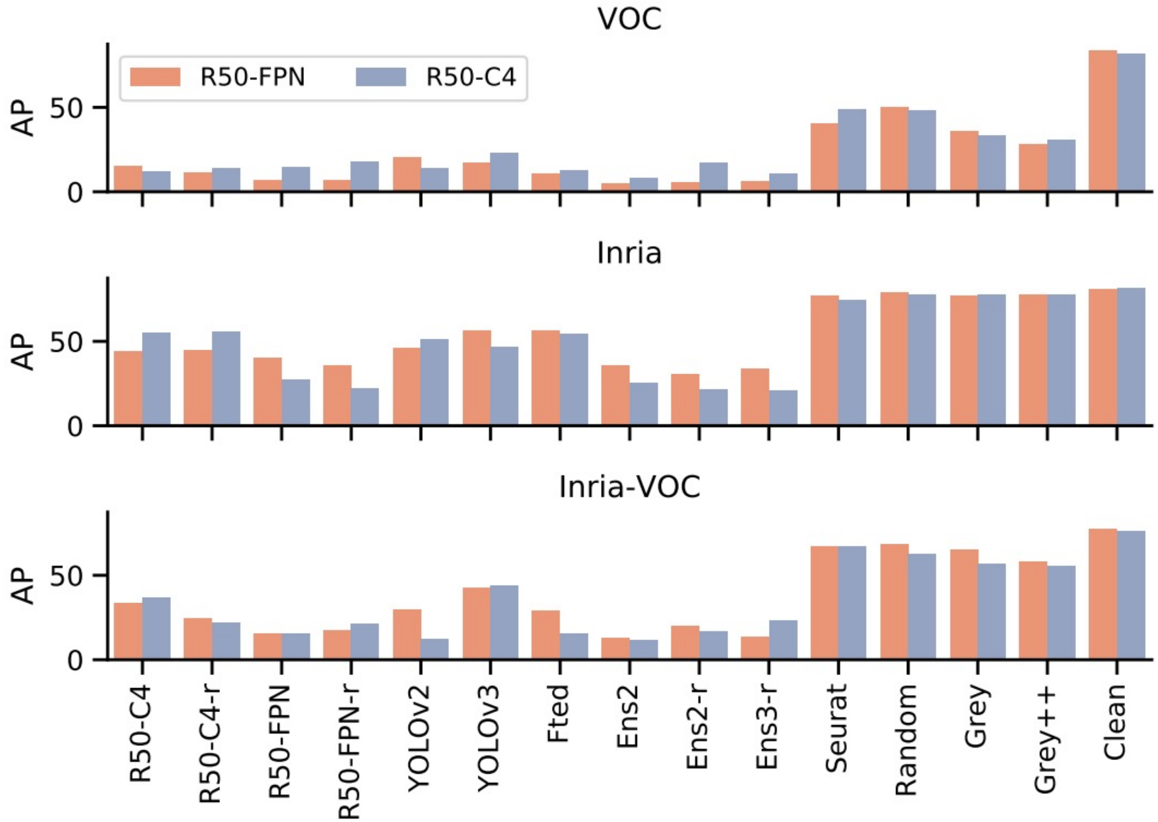


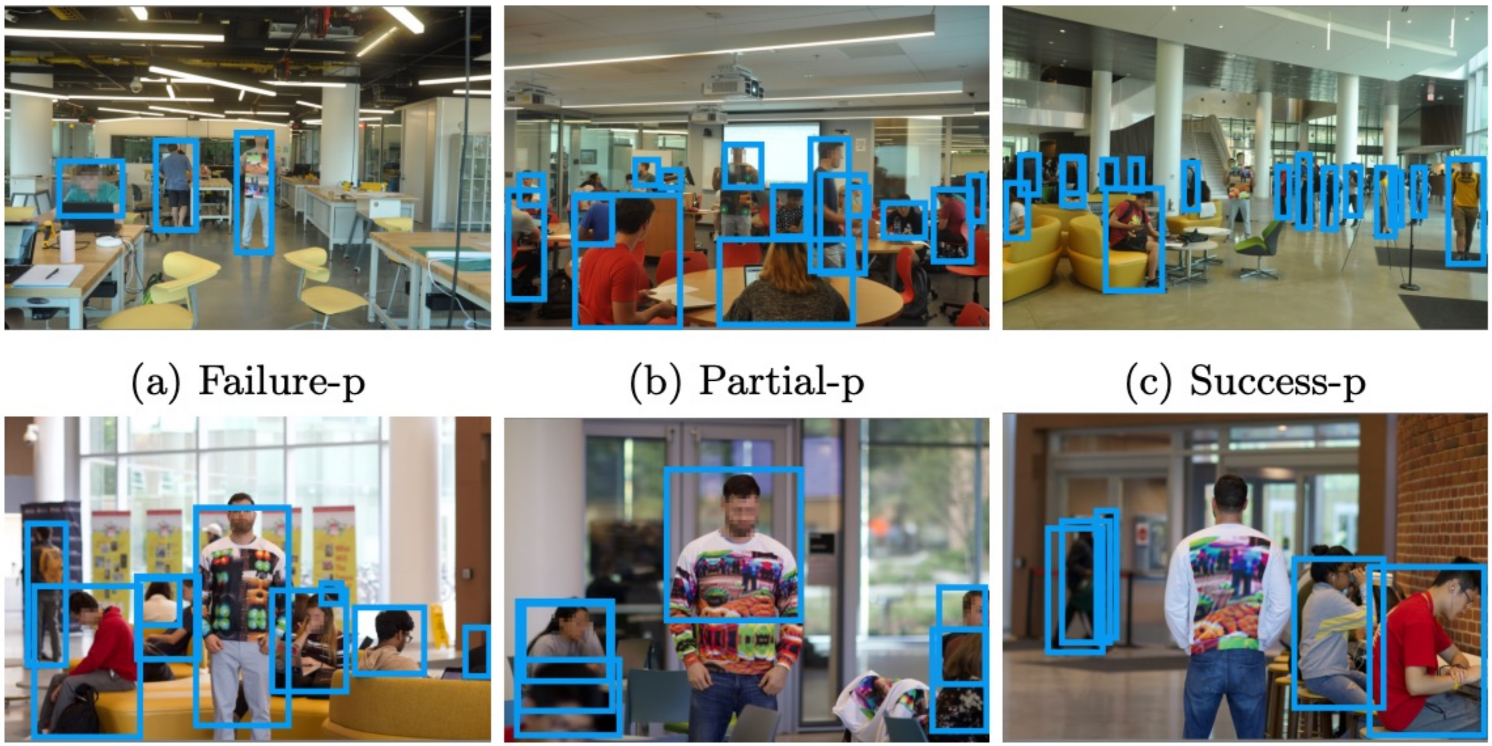
Fig. 6: **Results of different patches, trained on COCO, tested on the person category of different datasets**. Top two panels: COCO patches tested on VOC and Inria, respectively, using backbones learned on COCO; The bottom panel: COCO patches tested on Inria with backbones trained on VOC.

Transferability Across Classes

Patch \ Class	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
PERSON	2.0	14.6	1.0	1.8	2.7	13.5	10.7	2.3	0.1	2.4	6.4	2.3	8.3	12.3	5.5	0.3	2.2	1.3	3.8	12.4
HORSE	5.0	31.9	4.7	4.1	2.5	26.4	17.6	10.6	2.3	26.0	24.7	9.5	27.9	26.6	16.0	7.6	12.4	13.4	13.2	35.3
BUS	3.1	30.6	8.5	4.4	1.9	18.4	15.6	7.8	2.7	25.7	39.8	5.3	20.8	20.7	16.0	8.9	12.3	9.5	9.3	29.5
GREY	3.0	19.0	6.4	14.6	8.5	26.9	19.6	9.9	9.8	28.6	24.4	7.4	22.7	15.9	35.8	6.1	18.7	8.7	11.4	61.8
CLEAN	77.5	82.2	76.3	63.6	64.5	82.9	86.5	83.0	57.2	83.3	66.2	84.9	84.5	81.4	83.3	48.0	76.7	70.1	80.1	75.4

Table 2: **Transferability of patches across classes from VOC**, measured with average precision (AP).

Physical-World Attacks



(a) Failure-p (b) Partial-p (c) Success-p

(d) Failure-c (e) Partial-c (f) Success-c

Fig. 7: **Examples of attack failure, partial success, and full success**, using posters (top) and shirts (bottom).

Physical-World Attacks

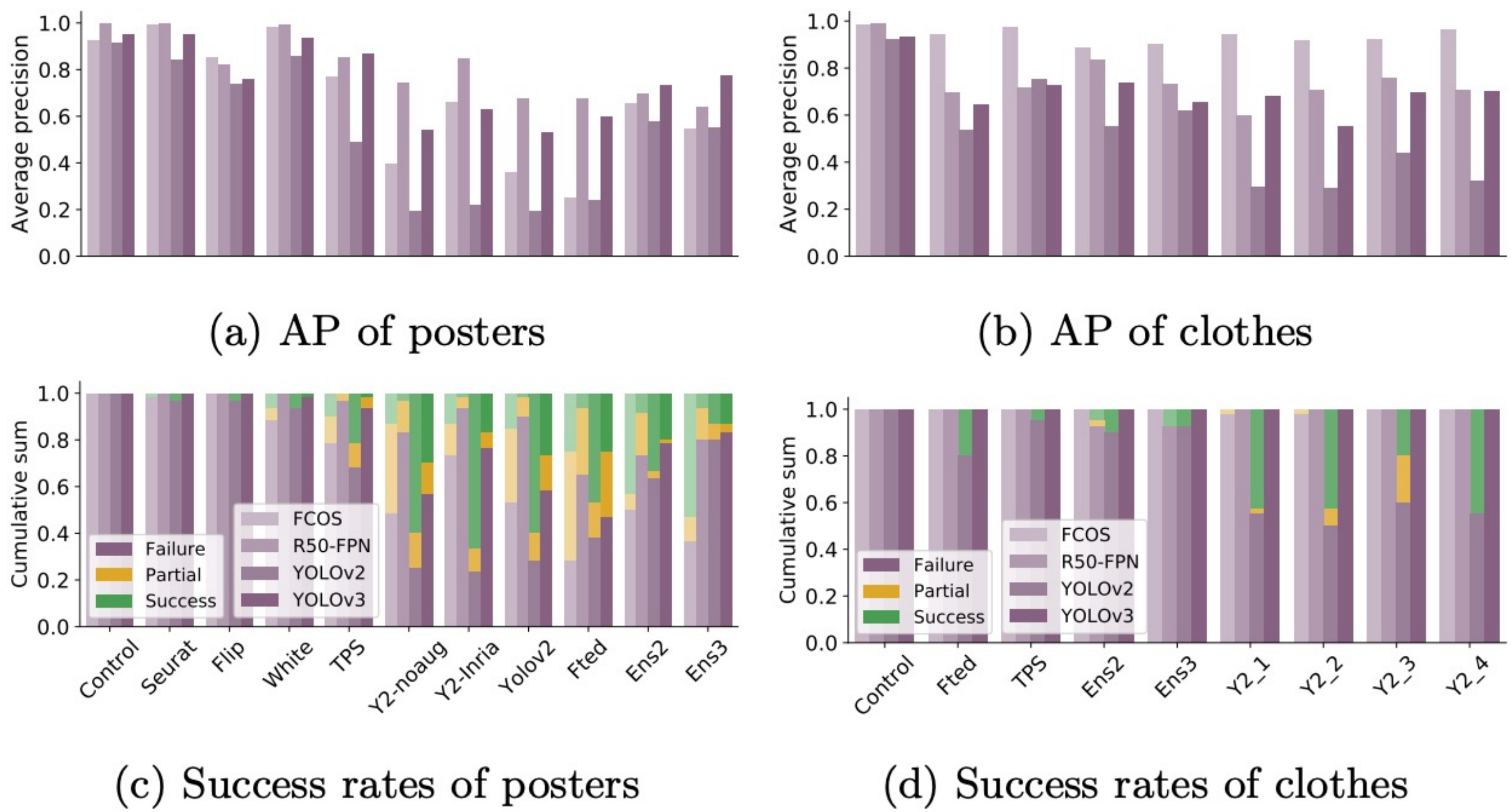


Fig. 8: **AP and success rates for physical attacks**. Top: AP of different printed posters (left) and clothes (right). Lower is better. Bottom: success rates of different printed posters (left) and clothes (right). Y2 denotes YOLOV2.

Discussion

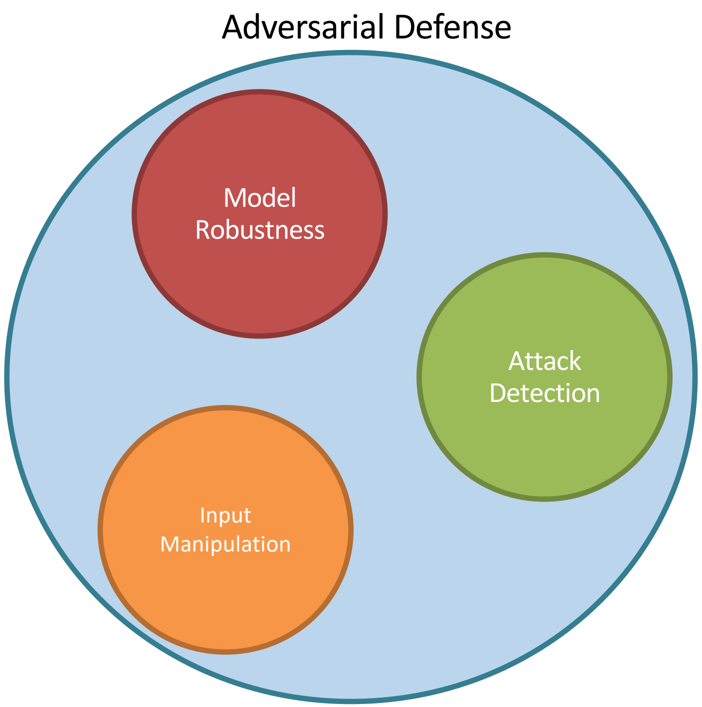
- Universal attacks
  - Need to optimize for a range of test samples
- Transferability
  - Attack an ensemble of diverse models
- Physical-world attacks
  - How to make the attack realizable? For object detection: poster, wearable (domain specific)
  - Success of attacks is lower than in the digital world

Cost Aware Tree Ensembles for Security Applications

Yizheng Chen, Shiqi Wang, Weifan Jiang, Asaf Cidon, and Suman Jana

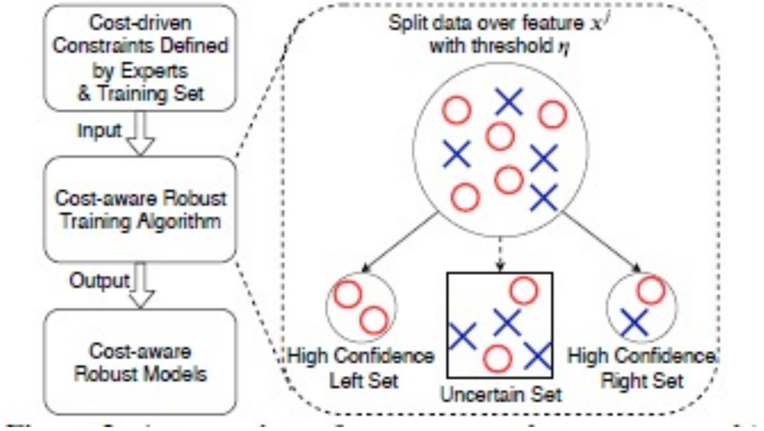
Problem Statement

- Unlike perturbing features in an image, attacks have a cost to manipulate different security features
- Most robustness research focuses on the neural network
  - Security applications predominantly use trees, such as Random Forest or Gradient Boosted Decision Trees
- L_p*-norm* bounding not suitable for security applications (hence, cost)



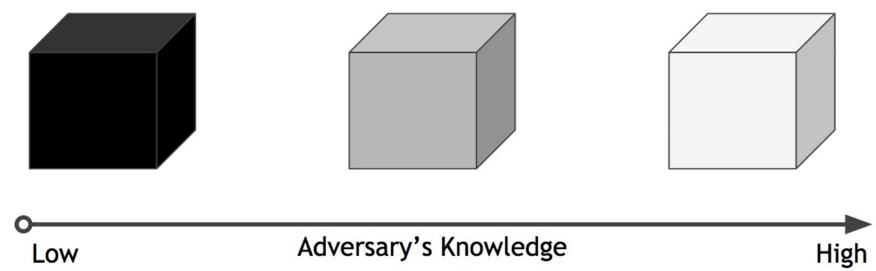
Problem Statement

- Develop systematic method to train cost-aware tree ensembles for security
  - Domain knowledge
  - Feature manipulation cost for the attacker
- First, model the cost
- Second, integrate the modeled cost into the training process



Threat Model

- White box attack
  - Attacker has knowledge of model
- Adaptive attack:
  - Attack has full knowledge of both the security application *and* defense
  - Attack objective specifically targets the cost-driven constraint



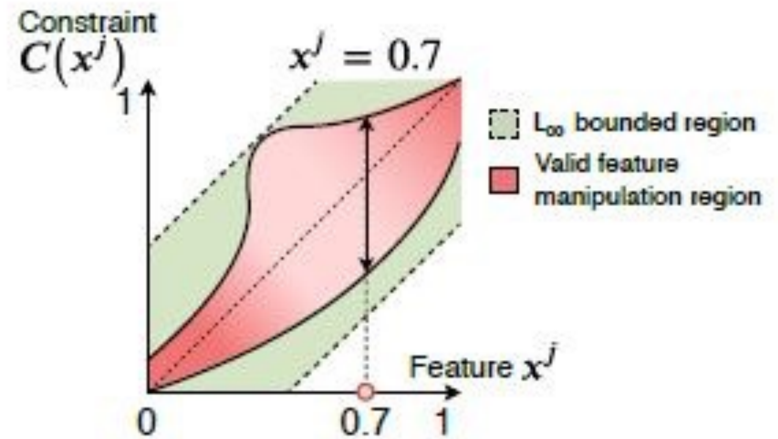
Attacks on Tree Ensembles

- MILP Attack (Kantchelian et al.)
  - Minimize a distance between the evasive example and the attacked data point
  - Finds adversarial example with *minimal evasion distance*
  - Linear solving program, which can be used to minimize any objective in the linear form

Methodology

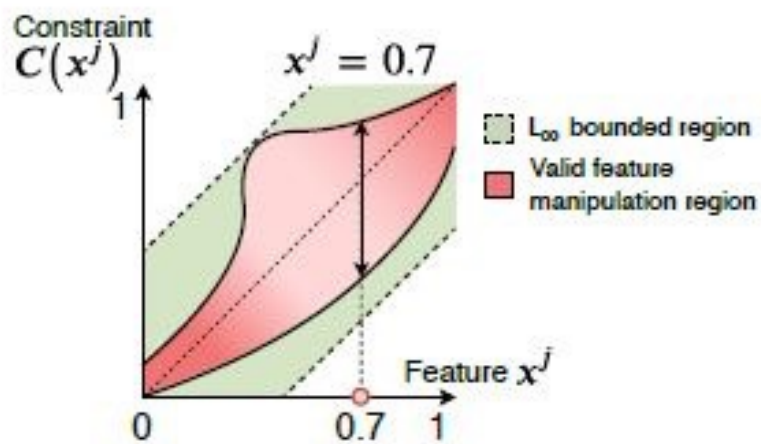
Attack Cost-Driven Constraint

- For each feature x^j, $C(x^j)$ is the cost constraint
  - Mapping from $[0, 1]$ to a set in $[0, 1] \times [0, 1]$
  - Gives the valid feature manipulation interval for any bounded attacker according to the cost of changing the feature, for all training data points



Cost Factors

- How do we decide the cost of a particular feature manipulation?
  - Human domain expert
- Economic factors
- Functionality
- Suspiciousness
- Monotonicity
- Attack Seed
- **All cost factors can be translated to some ROI for attackers.**



Box Cost Constraint

- After ranking feature manipulation cost, categorize cost
  - Negligible
  - Low
  - Medium
  - High
- Map the categories into a high dimensional box

$$C(x^j) = [x^j - l_j, x^j + h_j], j = 1,2,3,\dots,d$$

- For the j-th feature, maps to the interval of allowable changes

Cost	Value for l_j, h_j
Negligible	α
Low	β
Medium	γ
High	μ
Relationship	$\mu < \gamma < \beta < \alpha$

Table 1: Feature manipulation cost categories based on domain knowledge. For each feature j, we categorize the cost of increasing and decreasing its values and assign the bound for the box constraint using variables l_j and h_j.

Conditioned Cost Constraint

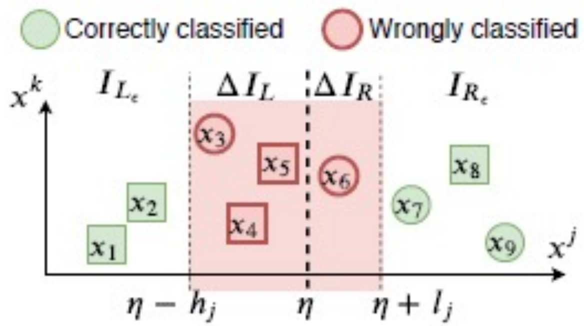
- If benign, it is extremely difficult for the attacker to change the j feature
- Can use this constraint to derive:
  - Set of training data points under attack for every feature dimension j
  - Every split threshold

$$C(x_i^j) = \begin{cases} 0 & x_i \text{ is benign} \\ [x_i^j, 1] & x_i \text{ is mal, pred score} > 0.9 \\ [-0.1, 0.1] * x_i^j & x_i \text{ is mal, pred score} <= 0.9 \end{cases} \quad (6)$$

Optimization Problem

- Goal: Maximize the gain computed from potential splits
  - Uses the domain knowledge
- Optimize for the maximal value of the score after the split, given the different children sets under the constraint

$$
\begin{aligned}
s(I_L, I_R, C) &= \max_{I'_L, I'_R, C} s(I'_L, I'_R) \\
&= \max_{\Delta I_L, \Delta I_R} s(I_{L_c} \cup \Delta I_L, I_{R_c} \cup \Delta I_R)
\end{aligned} \tag{8}
$$



Robust Training Algorithm

- Efficiently solves the optimization problem from previous slide
- Works for different types of trees, different ensembles, and different splitting metrics

Algorithm 1 Robust Training Algorithm

Input: training set $D = \{(x_i, y_i)\}, |D| = N$ $(x_i \in \mathbb{R}^d, y \in \mathbb{R})$.
Input: data points of the current node $I = \{(x_i, y_i)\}$, $|I| = m$.
Input: attack cost-driven constraint C.
Input: the score function s.
Output: the best split at the current node j^*, η^*.

Initialize $Gain^* = 0; j^* = 0; \eta^* = 0$
for $j = 1$ **to** d **do**
  Sort $I = \{(x_i, y_i)\}$ along the j-th feature as $\{(x_{t_i}, y_{t_i})\}$
  for $t_i = t_1$ **to** t_m **do**
    if $t_i = t_1$ **then**
      $\eta \leftarrow x^j_{t_1}$
    else
      $\eta \leftarrow \frac{1}{2}(x_{t_i} + x_{t_{i-1}})$
    end if
    Project C to the uncertain set ΔI.
    $I_L = \{(x_i, y_i) | x^j_i < \eta, x \notin \Delta I\}$
    $I_R = \{(x_i, y_i) | x^j_i > \eta, x \notin \Delta I\}$
    /* Greedily put (x_k, y_k) to whichever side that has a larger score to solve Equation (8). */
    for every (x_k, y_k) in ΔI **do**
      $ls = s(I_L \cup \{(x_k, y_k)\}, I_R)$
      $rs = s(I_L, I_R \cup \{(x_k, y_k)\})$
      if $ls > rs$ **then**
        $I_L = I_L \cup \{(x_k, y_k)\}$
      else
        $I_R = I_R \cup \{(x_k, y_k)\}$
      end if
    end for
    /* Find the maximal gain. */
    $Gain(j, \eta, I) = s(I) - s(I_L, I_R)$
    if $Gain(j, \eta, I) > Gain^*$ **then**
      $j^* = j; \eta^* = \eta$
      $Gain^* = Gain(j, \eta, I)$
    end if
  end for
end for
return j^*, η^*

Adaptive Attacker

- Has knowledge of the proposed robustness measure
- Wants to minimize the total feature manipulation cost to generate an adversarial example

$$\text{minimize} \sum_j a_j w_{xj} |\hat{x}^j - x^j| + \sum_j (1 - a_j) w'_{xj} |\hat{x}^j - x^j|$$

$$a_j = \begin{cases} 0 & \hat{x}^j \leq x^j \\ 1 & \hat{x}^j > x^j \end{cases}$$

Evaluation

GBDT Results

Dataset	# of trees	Trained ε		Tree Depth			Test ACC (%)			Test FPR (%)			Avg. l_∞			Improv.	
		Chen's	ours	natural	Chen's	ours	natural	Chen's	ours	natural	Chen's	ours	natural	Chen's	ours	natural	Chen's
breast-cancer	4	0.30	0.30	6	8	8	97.81	96.35	99.27	0.98	0.98	0.98	.2194	.3287	**.4405**	**2.01x**	**1.34x**
cod-rna	80	0.20	0.035	4	5	5	96.48	88.08	89.64	2.57	4.44	7.38	.0343	.0560	**.0664**	**1.94x**	**1.19x**
ijcnn1	60	0.20	0.02	8	8	8	97.91	96.03	93.65	1.64	2.15	1.62	.0269	.0327	**.0463**	**1.72x**	**1.42x**
MNIST 2 vs. 6	1,000	0.30	0.30	4	6	6	99.30	99.30	98.59	0.58	0.68	1.65	.0609	.3132	**.3317**	**5.45x**	**1.06x**

Table 3: Test accuracy and robustness of GBDT models trained by our algorithm (ours), compared to regularly trained models (natural) and the models trained by Chen and Zhang et al.'s method [11] (Chen's), in XGBoost. The improvement (Improv.) here denotes the average l_∞ robustness distance on our models over regularly trained ones and Chen and Zhang's, by measuring adversarial examples found by Kantchelian's MILP attack [29], the strongest whitebox attack.

Random Forest Results

Dataset	Trained ε		Tree Num / Depth			Test ACC (%)			Test FPR (%)			Avg. l_∞			Improv.	
	Chen's	ours	natural	Chen's	ours	natural	Chen's	ours	natural	Chen's	ours	natural	Chen's	ours	natural	Chen's
breast-cancer	0.30	0.30	20 / 4	20 / 4	80 / 8	99.27	99.27	98.54	0.98	0.98	1.96	.2379	.3490	**.3872**	**1.63x**	**1.11x**
cod-rna	0.03	0.03	40 / 14	20 / 14	40 / 14	96.54	92.63	89.44	2.97	3.65	5.69	.0325	.0512	**.0675**	**2.08x**	**1.32x**
ijcnn1	0.03	0.03	100 / 14	100 / 12	60 / 8	97.92	93.86	92.26	1.50	0.78	0.08	.0282	.0536	**.1110**	**3.94x**	**2.07x**
MNIST 2 vs. 6	0.30	0.30	20 / 14	100 / 12	100 / 14	99.35	99.25	99.35	0.68	0.68	0.48	.0413	.1897	**.2661**	**6.44x**	**1.40x**

Table 4: Test accuracy and robustness of random forest models trained by our algorithm (ours) compared to regularly trained models (natural), in scikit-learn. The improvement (Improv.) here denotes the average l_∞ robustness distance increase.

Twitter Case Study

- Twitter spam detection application
- Section 4.3
  - Walks through how each component of the proposed mechanism was implemented
- Results:
  - Improves cost-aware robustness by 10.6x

Classifier Model	Constraint Variables				Adaptive Objective	Model Quality			Robustness against MILP Average					
	N	L	M	H										
	α	β	γ	μ		Acc	FPR	AUC	L_1	L_2	$Cost_1$	$Cost_2$	$Cost_3$	$Cost_4$
Natural	-	-	-	-	-	99.38	0.89	.9994	.007	.006	.010	.009	.009	.008
C1 $\varepsilon = 0.03$	-	-	-	-	-	96.59	5.49	.9943	.046	.036	.080	.070	.062	.054
C2 $\varepsilon = 0.05$	-	-	-	-	-	94.51	7.27	.9910	.062	.053	.133	.109	.089	.085
C3 $\varepsilon = 0.1$	-	-	-	-	-	91.89	11.96	.9810	**.079**	**.062**	**.156**	**.133**	**.111**	**.099**
M1	0.08	0.04	0.02	0	$Cost_1$	98.24	2.05	.9984	.032	.027	.099	.051	.058	.056
M2	0.12	0.06	0.03	0		96.54	4.09	.9941	**.043**	**.036**	**.106**	**.078**	**.064**	**.062**
M3	0.20	0.10	0.05	0		96.96	4.10	.9949	.033	.027	.064	.025	.040	.040
M4	0.28	0.14	0.07	0		94.38	4.25	.9884	.024	.012	.043	.026	.039	.023
M5	0.32	0.16	0.08	0		93.85	9.62	.9877	.024	.015	.034	.025	.030	.025
M6	0.09	0.06	0.03	0.03	$Cost_2$	97.82	2.65	.9968	**.049**	.038	**.104**	**.090**	**.070**	**.068**
M7	0.15	0.10	0.05	0.05		96.60	4.91	.9929	.045	**.039**	.080	.072	.061	.060
M8	0.24	0.16	0.08	0.08		93.10	9.16	.9848	.041	.030	.082	.057	.050	.049
M9	0.30	0.20	0.10	0.10		92.28	12.16	.9836	.042	.028	.050	.044	.041	.038
M10	0.04	0.04	0.02	0	$Cost_3$	98.51	1.84	.9988	.025	.022	.087	.041	.052	.049
M11	0.06	0.06	0.03	0		97.31	3.65	.9953	.029	.017	.032	.027	.026	.025
M12	0.10	0.10	0.05	0		96.86	4.07	.9919	.044	.035	.062	.059	.049	.048
M13	0.16	0.16	0.08	0		94.54	5.91	.9900	**.051**	**.041**	**.109**	**.090**	**.075**	**.074**
M14	0.20	0.20	0.10	0		96.36	4.95	.9910	.033	.024	.054	.042	.043	.043
M15	0.28	0.28	0.14	0		93.81	6.57	.9851	.039	.039	.093	.070	.048	.048
M16	0.06	0.03	0.03	0	$Cost_4$	97.31	3.48	.9953	.036	.018	.038	.035	.034	.028
M17	0.10	0.05	0.05	0		97.41	2.70	.9964	.023	.020	**.084**	.034	.051	.049
M18	0.16	0.08	0.08	0		93.41	9.08	.9872	.035	.024	.074	**.044**	**.062**	**.051**
M19	0.20	0.10	0.10	0		96.48	4.75	.9918	**.047**	**.038**	.054	.041	.051	**.051**

Table 8: We trained classifiers with 19 different cost models under the box constraint, and we compare them against regular training (Natural) and three models from Chen's method [11] with different ε. We separate our models by four different cost families. Each cost family keeps the same proportion between the constraint variables and has the same adaptive attack objective. The best numbers within each cost family are highlighted in bold. We have also evaluated the recall of the models in Appendix A.2.

Conclusion

Discussion

Strengths

- Improves on state of the art
- Significant improvement in solving the optimization problem
- Generalizes to multiple types of trees and ensembles
- Realistic threat model

Weaknesses

- Constraint hyperparameter tuning can be difficult to perform
- Human input = potential for error
- Tradeoff with accuracy and robustness