

ROB 498/599: Deep Learning for Robot Perception (DeepRob)

Lecture 11: Object Detection - part 1

02/16/2026



<https://deeprob.org/w26/>

Announcements

- New course website: <https://deeprob.org/w26/>
- **P3** released, due **March 8, 2026**
(recommend finish before spring vacation)
- **Midterm**: March 11, 2026, 12pm-1:30pm EST
(Wednesday in class after spring vacation)
 - Pen/Pencil and paper exam
 - 1 A4/Letter-size note, front and back
 - No GenAI/phone/computer/internet
- **Final Project info**:
<https://deeprob.org/w26/projects/finalproject/>

Today

- Training NN (Cont'd)
 - After Training: Model Ensembles, Transfer Learning
- Object Detection
 - Object Detection Overview
 - Region Proposals
 - R-CNN
 - NMS and mAPs
 - Fast R-CNN
- Summary and Takeaways

Training NN (Cont'd)

1. One time setup:

- Activation functions, data preprocessing, weight initialization, regularization

2. Training dynamics:

- Learning rate schedules; hyperparameter optimization

3. After training:

- Model ensembles, transfer learning, large-batch training

Next Time

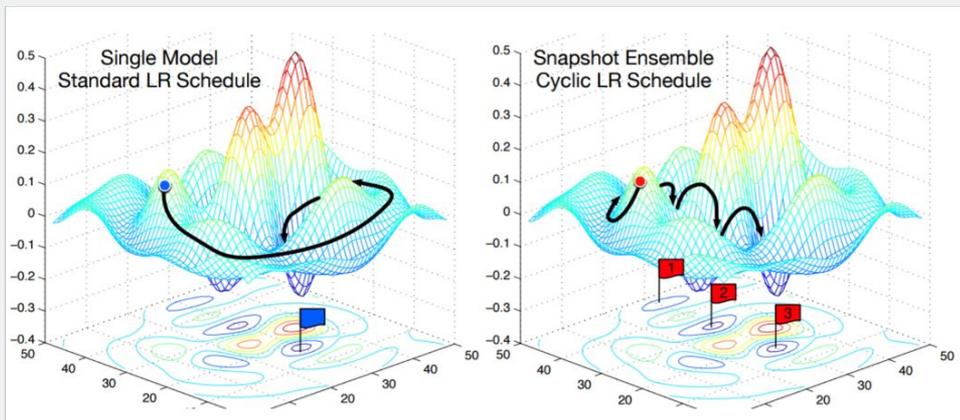
Model Ensembles

- 1. Train multiple independent models**
- 2. At test time average their results:**
(Take average of predicted probability distributions, then choose argmax)

Enjoy 2% extra performance

Model Ensembles: Tips and Tricks

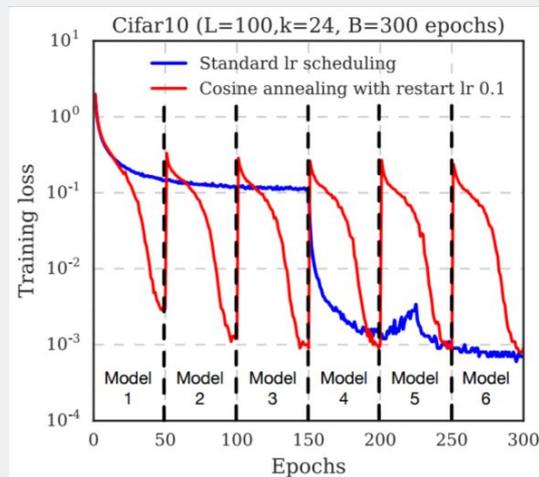
Instead of training independent models, use multiple snapshots of a single model during training!



Loshchilov and Hutter, "SGDR: Stochastic gradient descent with restarts", arXiv 2016

Huang et al, "Snapshot ensembles: train 1, get M for free", ICLR 2017

Figures copyright Yixuan Li and Geoff Pleiss, 2017. Reproduced with permission.



Cyclic learning rate schedules can make this work even better!

Model Ensembles: Tips and Tricks

Instead of using actual parameter vector, keep a moving average of the parameter vector and use that at test time (Polyak averaging)

```
while True:
    data_batch = dataset.sample_data_batch()
    loss = network.forward(data_batch)
    dx = network.backward()
    x += - learning_rate * dx
    x_test = 0.995*x_test + 0.005*x # use for test set
```

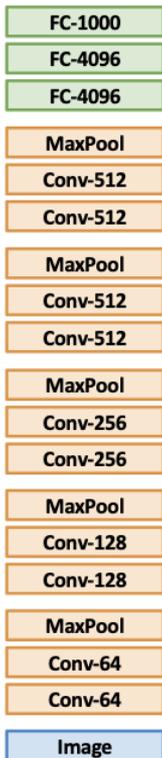
Polyak and Juditsky, "Acceleration of stochastic approximation by averaging", SIAM Journal on Control and Optimization, 1992.

Karras et al, "Progressive Growing of GANs for Improved Quality, Stability, and Variation", ICLR 2018
Brock et al, "Large Scale GAN Training for High Fidelity Natural Image Synthesis", ICLR 201

Transfer Learning: Generalizing to New Tasks

Transfer Learning with CNNs

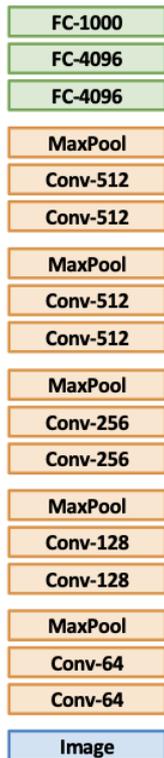
1. Train on ImageNet



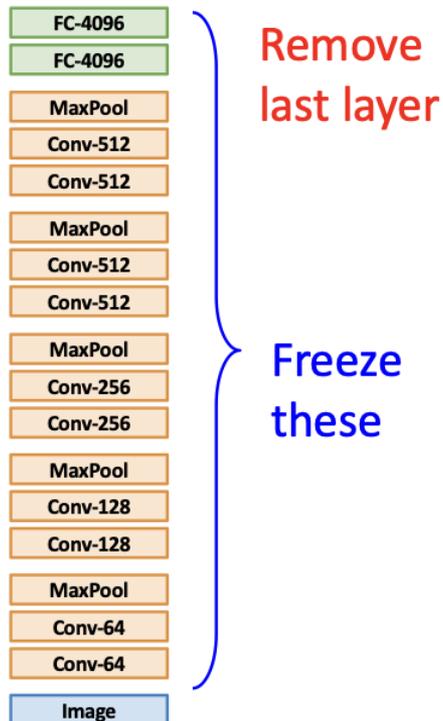
Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
<https://arxiv.org/abs/1310.1531>

Transfer Learning with CNNs

1. Train on ImageNet

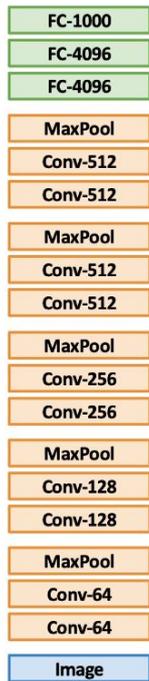


2. Use CNN as a feature extractor

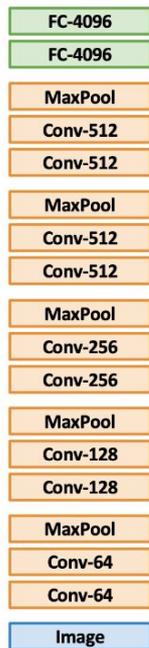


Transfer Learning: Feature Extraction

1. Train on ImageNet

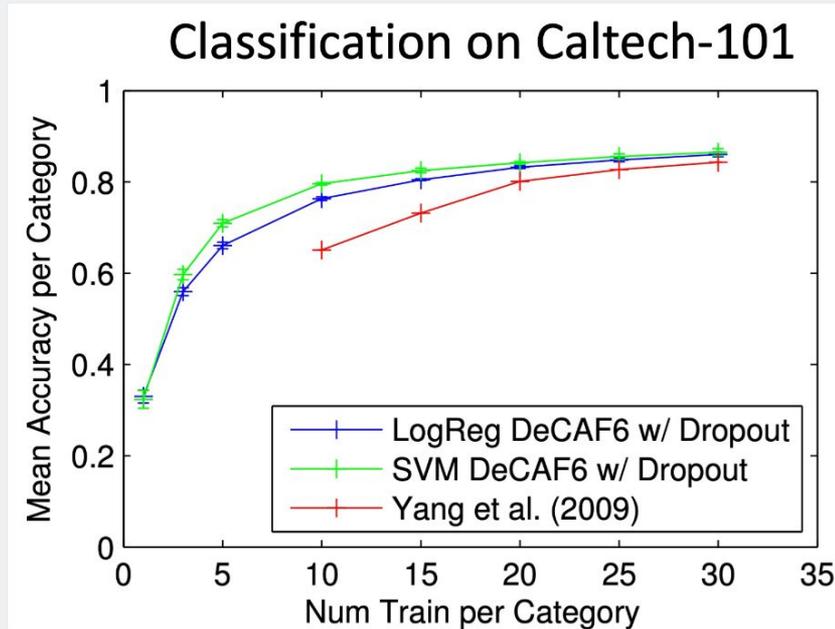


2. Use CNN as a feature extractor



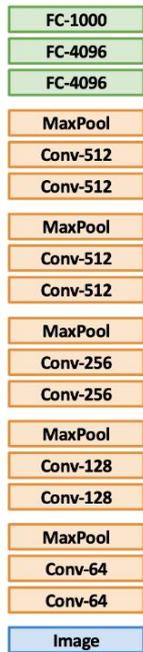
Remove last layer

Freeze these

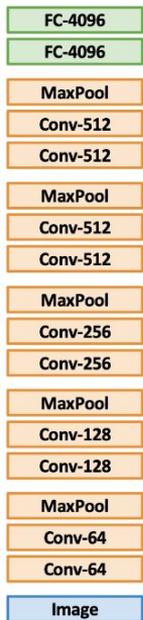


Transfer Learning: Feature Extraction

1. Train on ImageNet



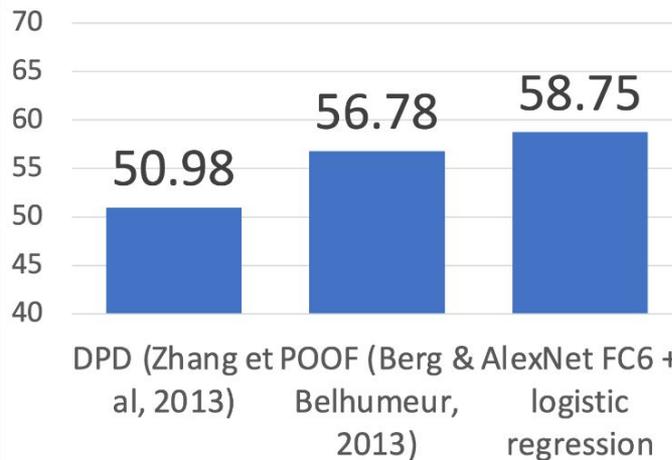
2. Use CNN as a feature extractor



Remove
last layer

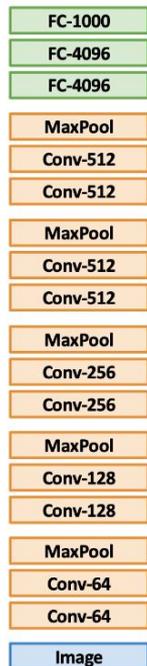
Freeze
these

Bird Classification on Caltech-UCSD

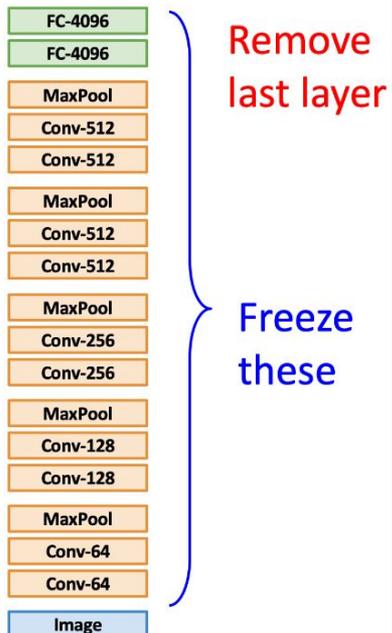


Transfer Learning: Feature Extraction

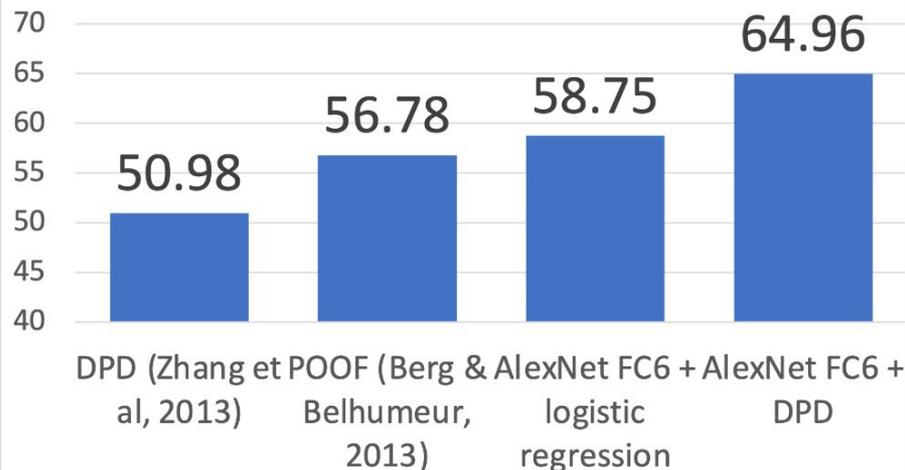
1. Train on ImageNet



2. Use CNN as a feature extractor

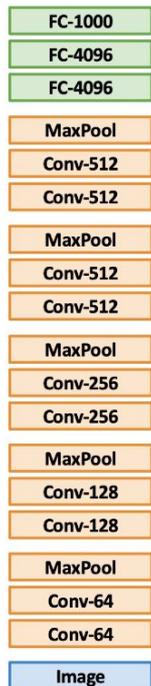


Bird Classification on Caltech-UCSD



Transfer Learning: Feature Extraction

1. Train on ImageNet



2. Use CNN as a feature extractor

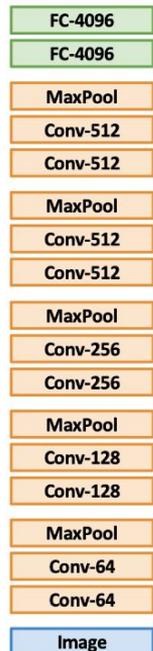
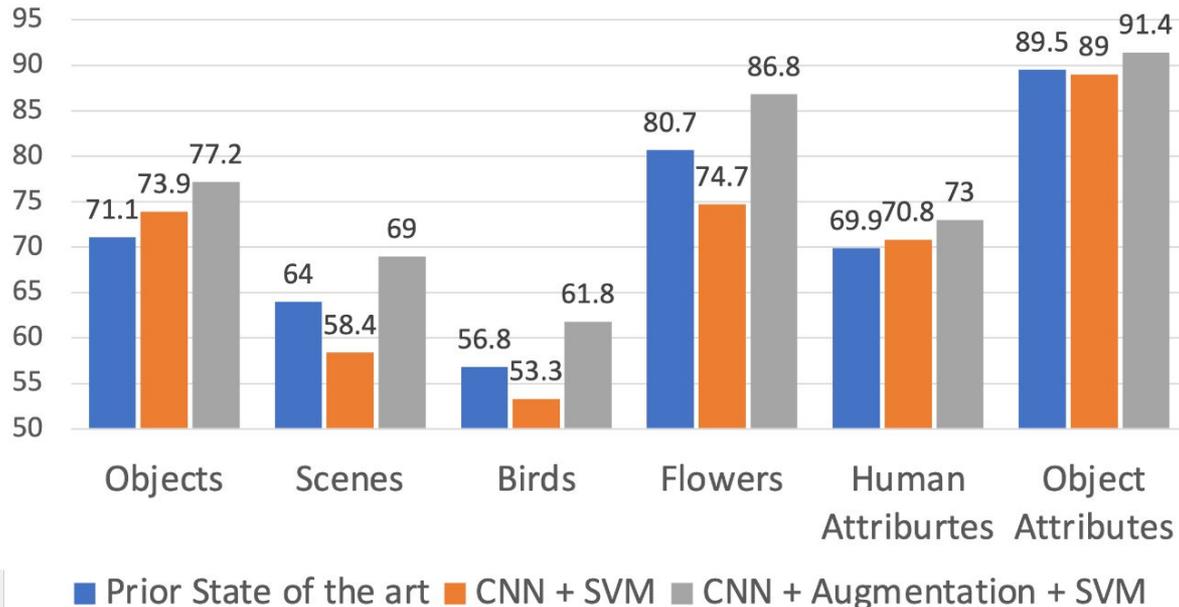
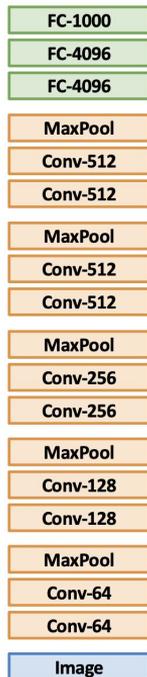


Image Classification



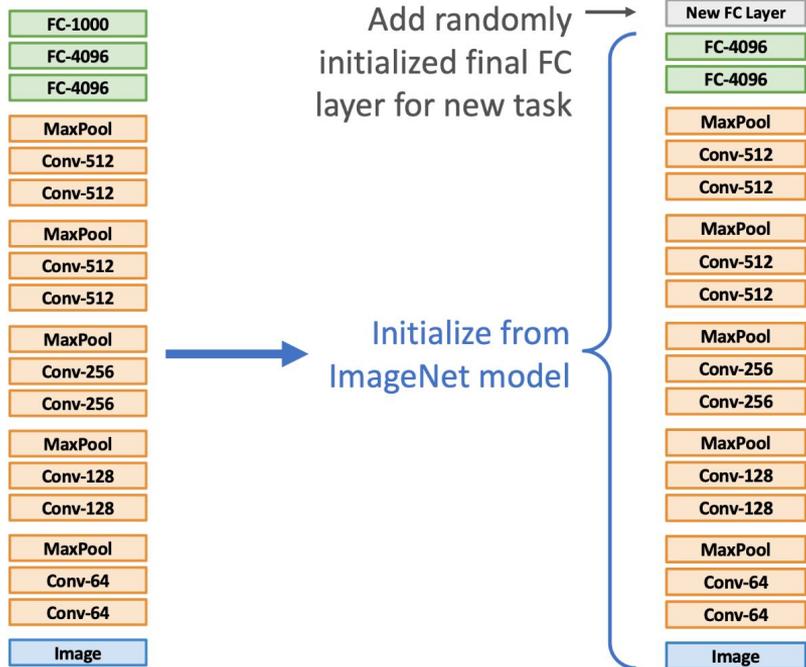
Transfer Learning: Fine Tuning

1. Train on ImageNet



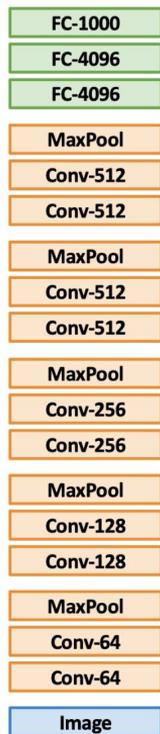
Transfer Learning: Fine Tuning

1. Train on ImageNet



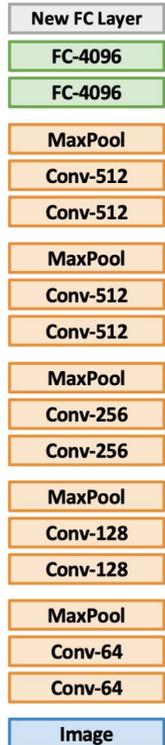
Transfer Learning: Fine Tuning

1. Train on ImageNet



Add randomly initialized final FC layer for new task

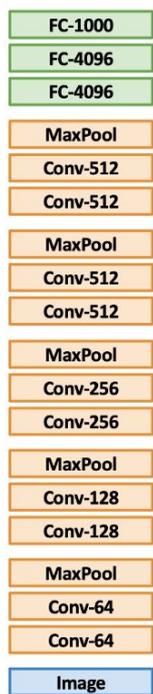
Initialize from ImageNet model



Continue training entire model for new task

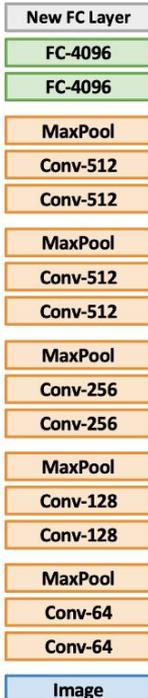
Transfer Learning: Fine Tuning

1. Train on ImageNet



Add randomly initialized final FC layer for new task

Initialize from ImageNet model



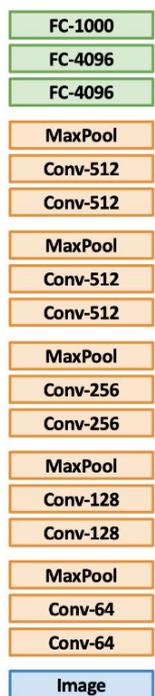
Continue training entire model for new task

Some tricks:

- Train with feature extraction first before finetuning
- Lower the learning rate: use $\sim 1/10$ of LR used in original training
- Sometimes freeze lower layers to save computation

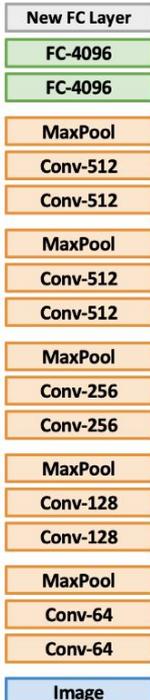
Transfer Learning: Fine Tuning

1. Train on ImageNet



Add randomly initialized final FC layer for new task

Initialize from ImageNet model

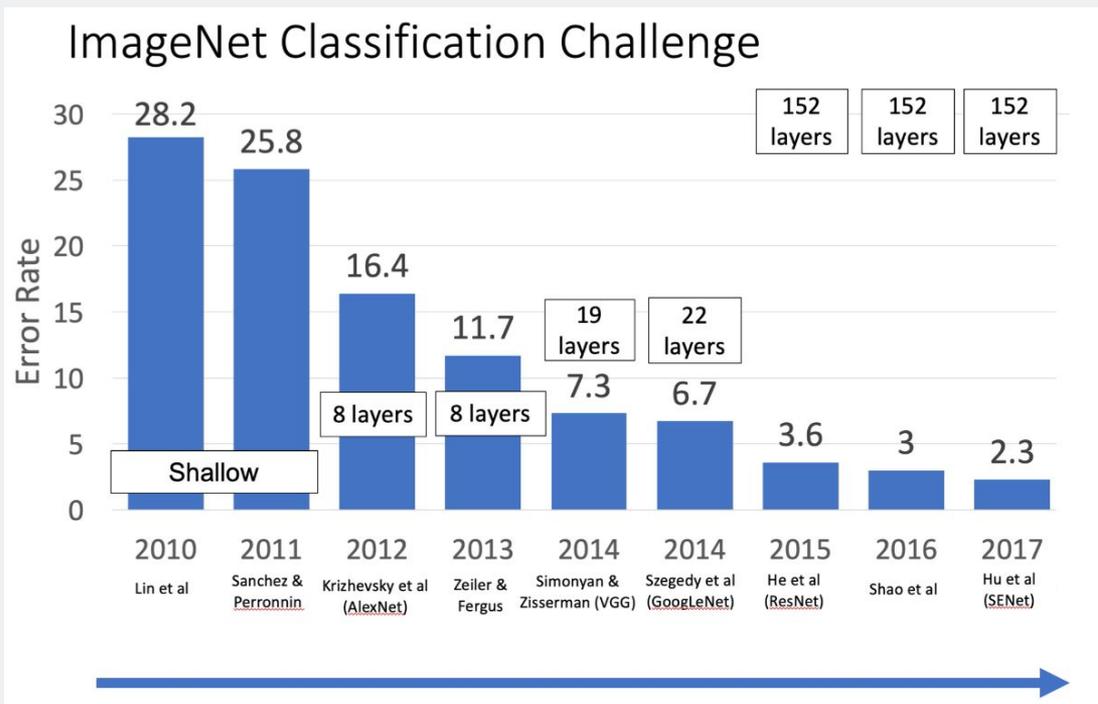


Continue training entire model for new task

Compared with feature extraction, fine-tuning:

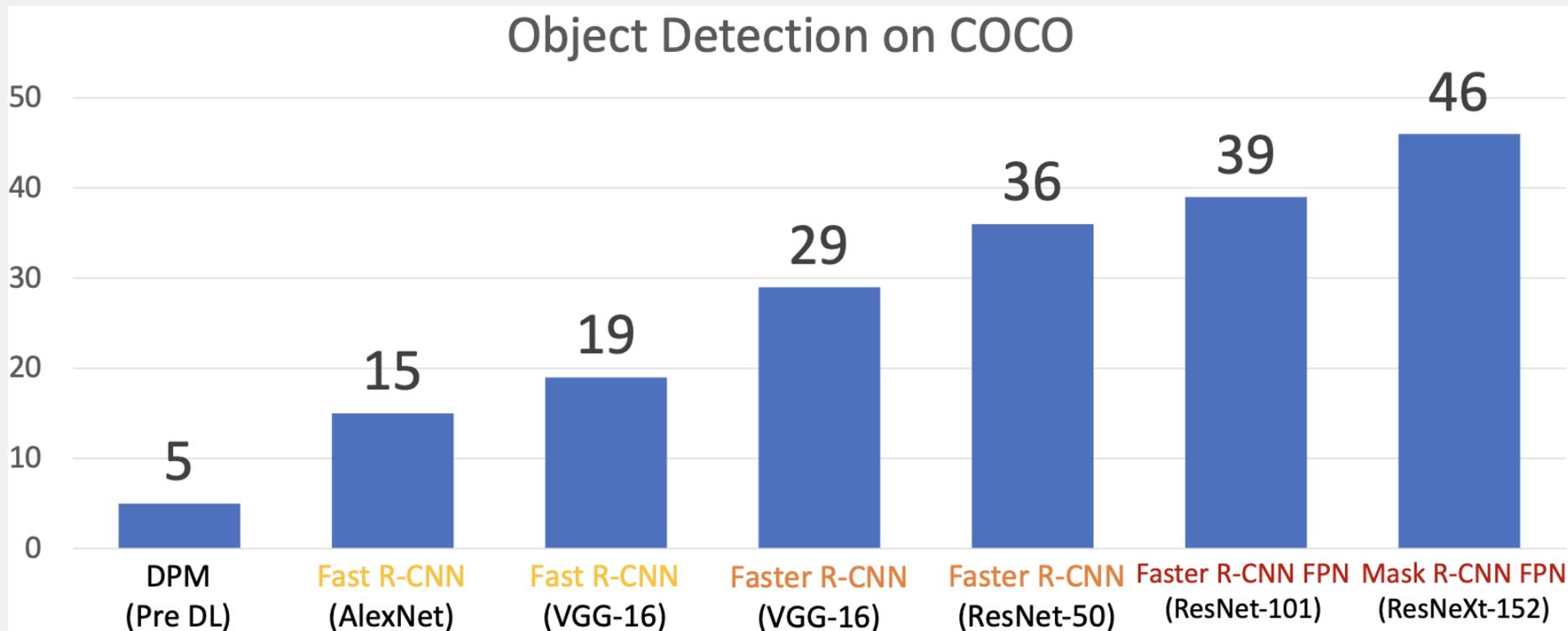
- Requires more data
- Is computationally expensive
- Can give higher accuracies

Transfer Learning: Architecture Matters!



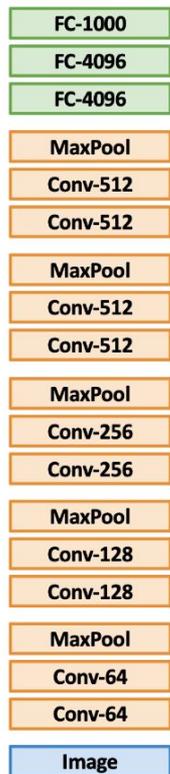
Improvements in CNN architecture leads to improvements in many down stream tasks thanks to transfer learning!

Transfer Learning: Architecture Matters!



Ross Girshick, "The Generalized R-CNN Framework for Object Detection", ICCV 2017 Tutorial on Instance-Level Visual Recognition
<https://instancetutorial.github.io/>

Transfer Learning with CNNs

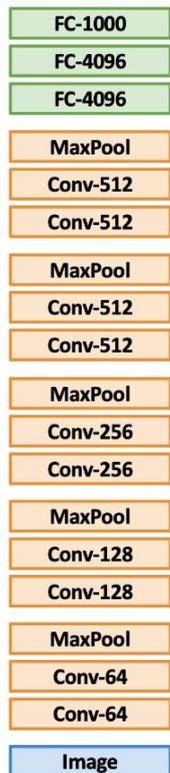


More specific

More generic

	Dataset similar to ImageNet	Dataset very different from ImageNet
Very little data (10s to 100s)	?	?
Quite a lot of data (100s to 1000s)	?	?

Transfer Learning with CNNs

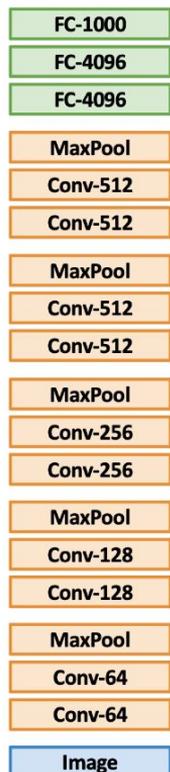


More specific

More generic

	Dataset similar to ImageNet	Dataset very different from ImageNet
Very little data (10s to 100s)	Use Linear Classifier on top layer	?
Quite a lot of data (100s to 1000s)	?	?

Transfer Learning with CNNs

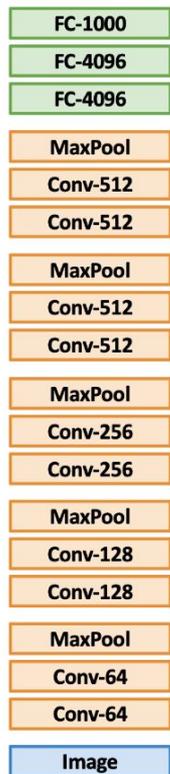


More specific

More generic

	Dataset similar to ImageNet	Dataset very different from ImageNet
Very little data (10s to 100s)	Use Linear Classifier on top layer	?
Quite a lot of data (100s to 1000s)	Finetune a few layers	?

Transfer Learning with CNNs

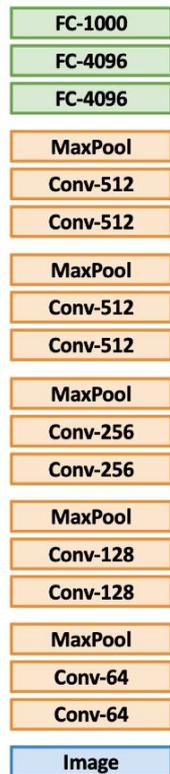


More specific

More generic

	Dataset similar to ImageNet	Dataset very different from ImageNet
Very little data (10s to 100s)	Use Linear Classifier on top layer	?
Quite a lot of data (100s to 1000s)	Finetune a few layers	Finetune a larger number of layers

Transfer Learning with CNNs

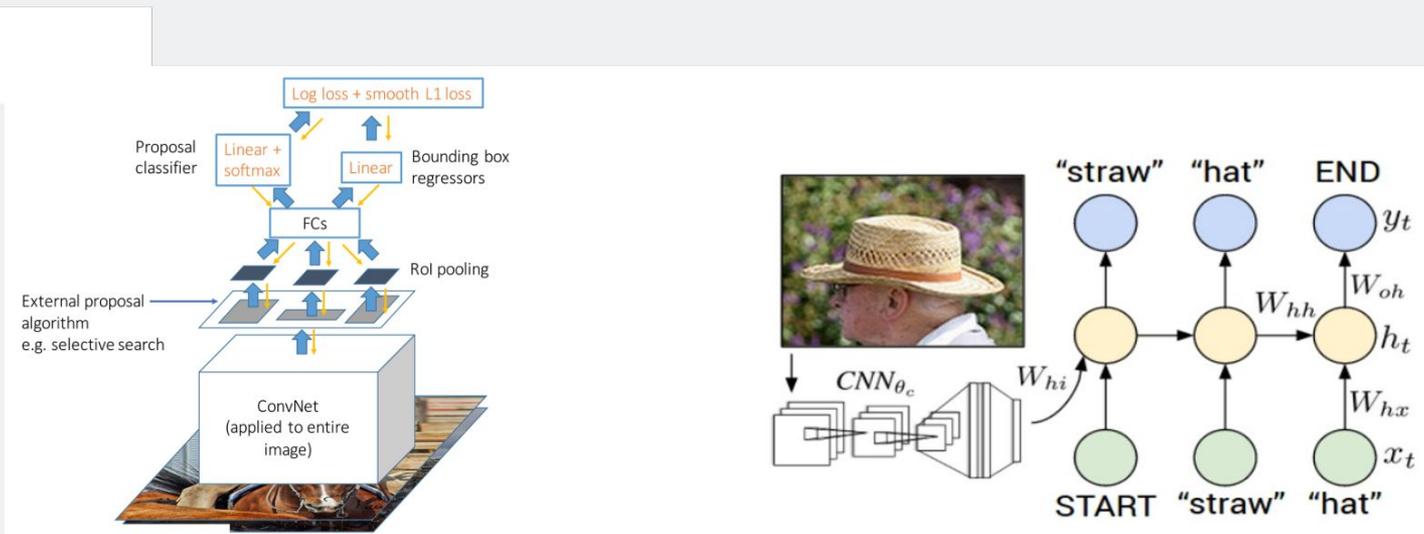


More specific

More generic

	Dataset similar to ImageNet	Dataset very different from ImageNet
Very little data (10s to 100s)	Use Linear Classifier on top layer	You're in trouble... Try linear classifier from different stages
Quite a lot of data (100s to 1000s)	Finetune a few layers	Finetune a larger number of layers

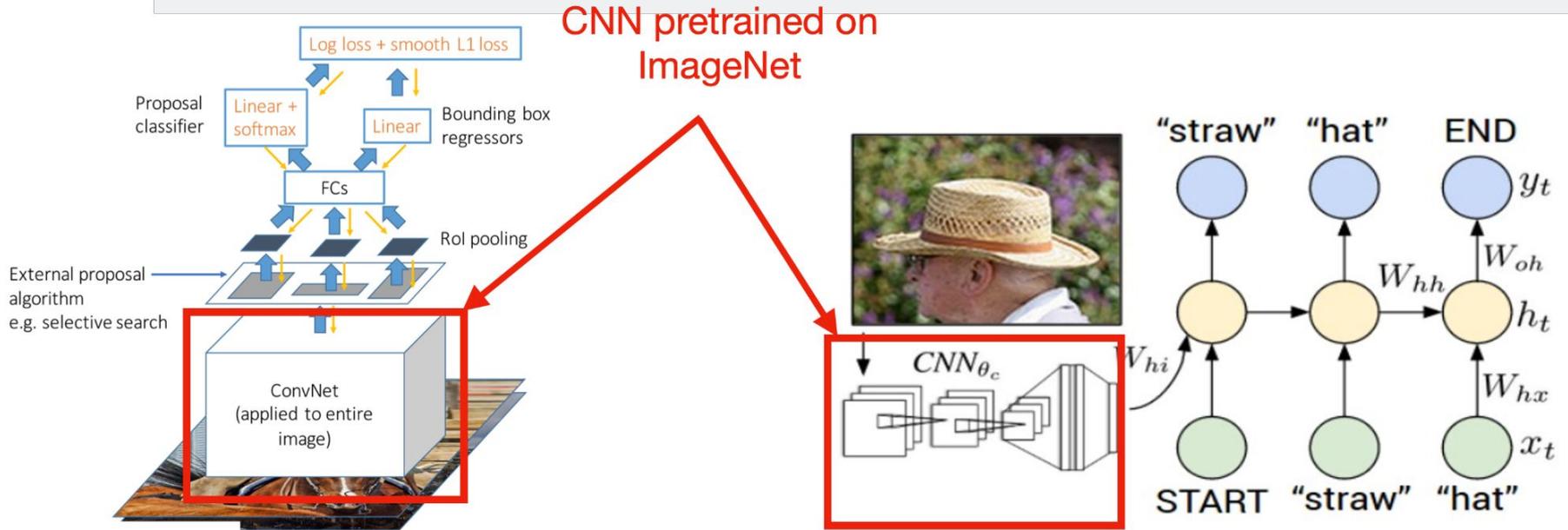
Transfer Learning - “the norm”, not the exception



Girshick, “Fast R-CNN”, ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission

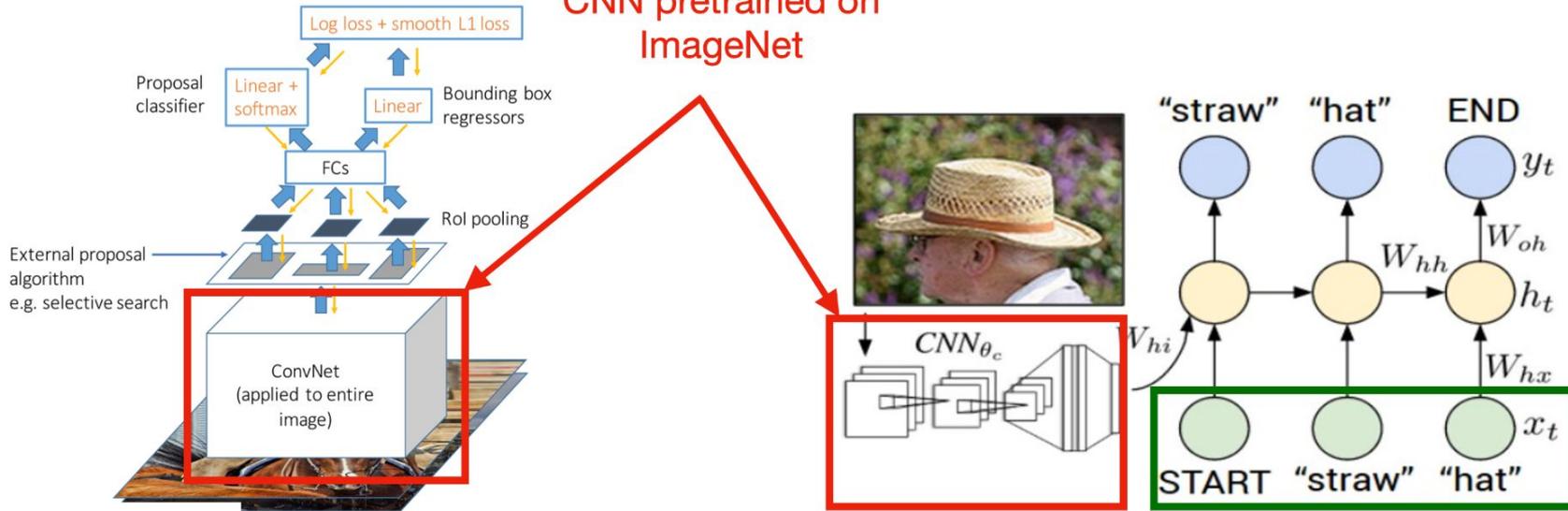
Karpathy and Fei-Fei, “Deep Visual-Semantic Alignments for Generating Image Descriptions”, CVPR 2015

Transfer Learning - “the norm”, not the exception



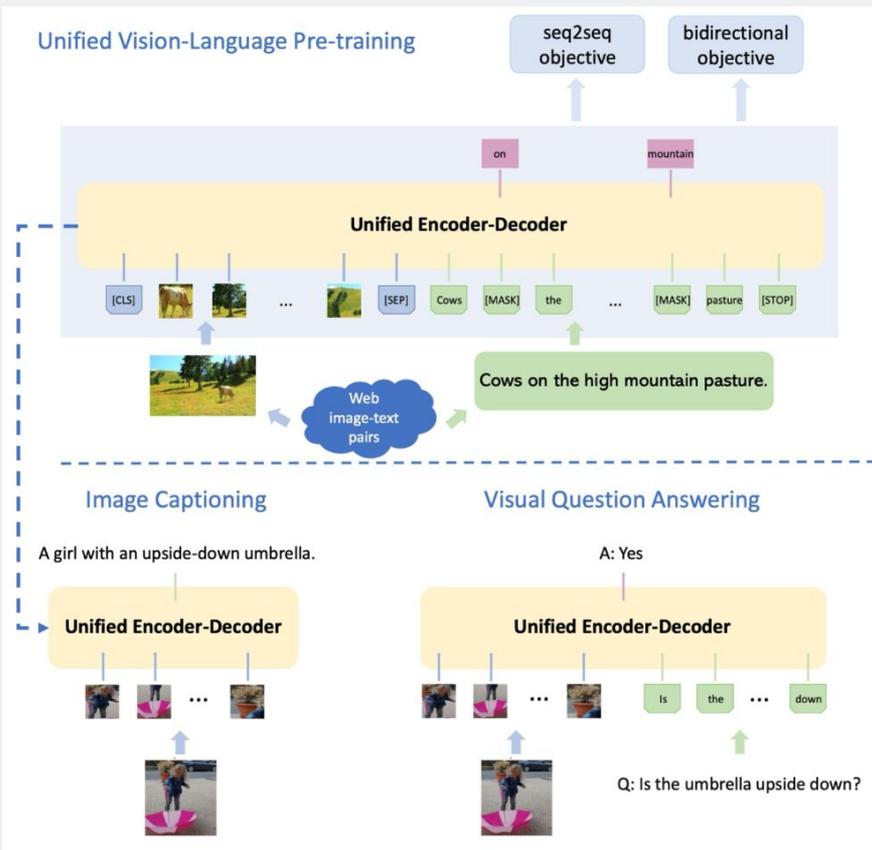
Transfer Learning - “the norm”, not the exception

CNN pretrained on ImageNet



Word vectors pretrained with word2vec

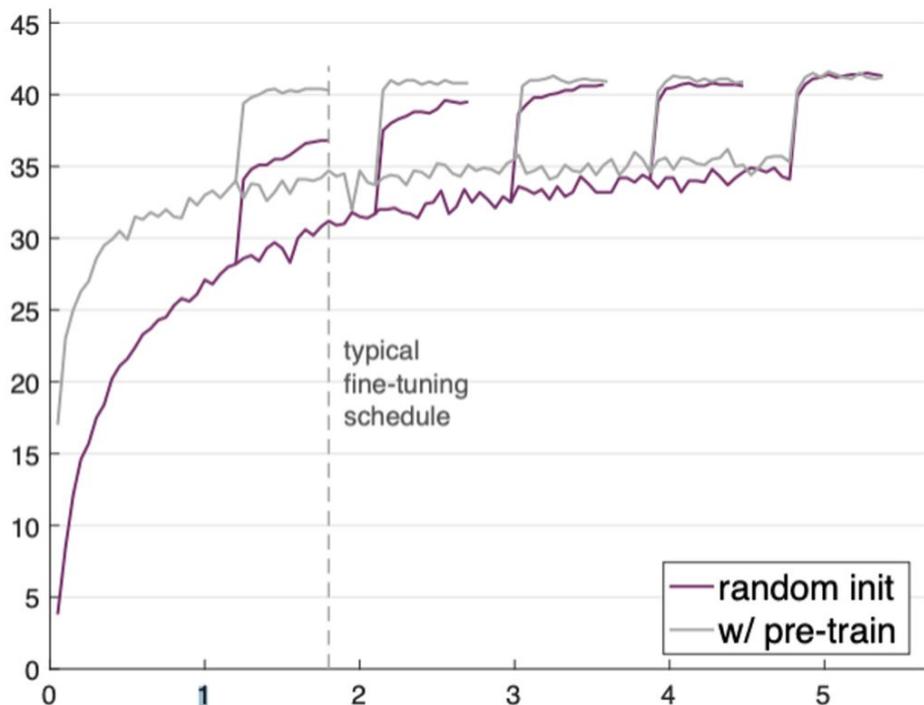
Transfer Learning - “the norm”, not the exception



1. Train CNN on ImageNet
2. Fine-Tune (1) for object detection on Visual Genome
3. Train BERT language model on lots of text
4. Combine (2) and (3), train for joint image / language modeling
5. Fine-tune (5) for image captioning, visual question answering, etc.

Transfer Learning: Helps you converge faster!

COCO object detection

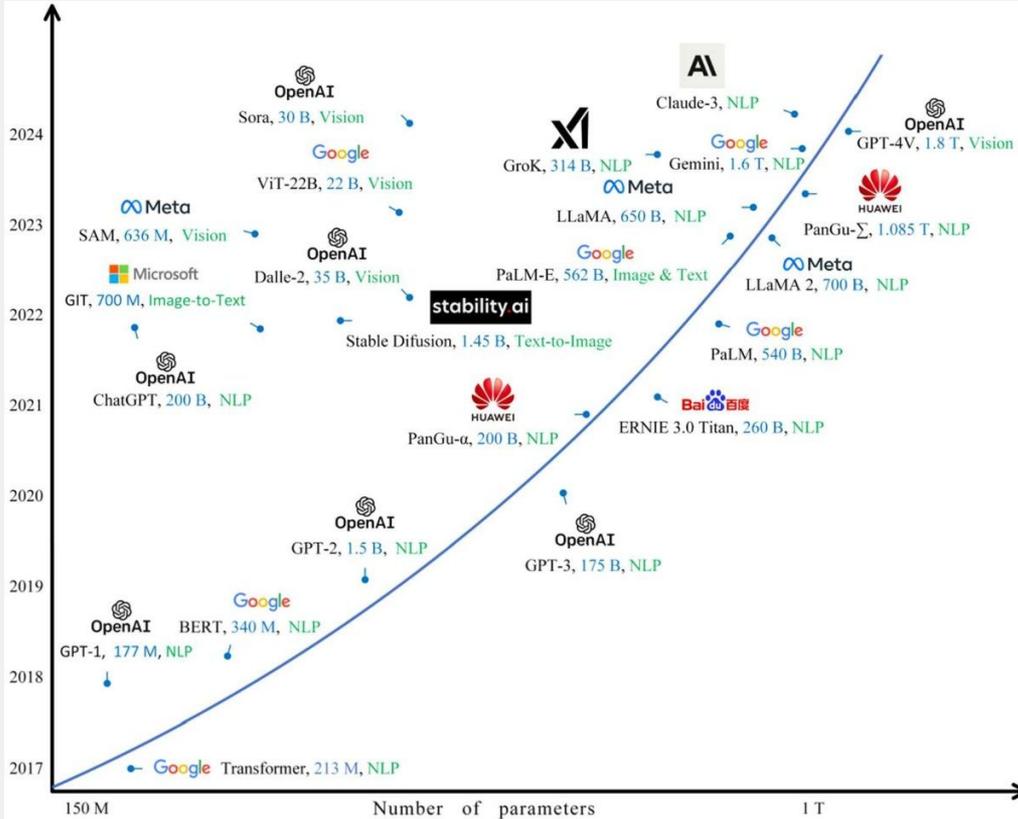


If you have enough data and train for much longer, random initialization can sometimes do as well as transfer learning

He et al, "Rethinking ImageNet Pre-Training", ICCV 2019

https://openaccess.thecvf.com/content_ICCV_2019/papers/He_Rethinking_ImageNet_Pre-Training_ICCV_2019_paper.pdf

The Scaling Problem



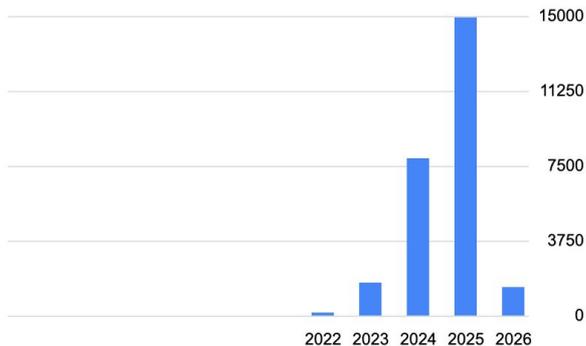
- Modern models have **billions of parameters**: standard finetuning is computationally challenging
- Require **Parameter-Efficient Fine-Tuning (PEFT)** paradigm

PEFT: Low-Rank Adaptation (LoRA)

Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." *ICLR 2022*.

LoRA: Low-Rank Adaptation of Large Language Models

Edward Hu* Yelong Shen* Phillip Wallis
Zeyuan Allen-Zhu Yuanzhi Li Shean Wang Weizhu Chen
Microsoft Corporation
{edwardhu, yeshe, phwallis, zeyuana, yuanzhil
swang, wzchen}@microsoft.com



Key idea: learning low-rank corrections to pre-trained weights!

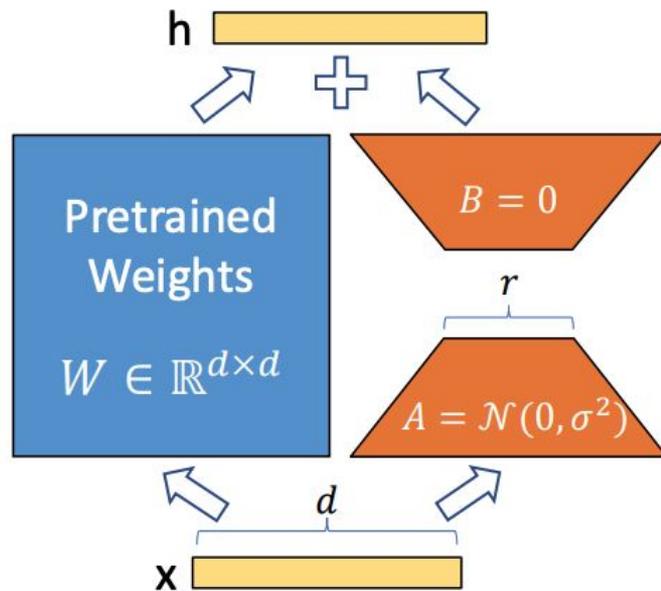


Figure 1: Our reparametrization. We only train A and B .

PEFT: Low-Rank Adaptation (LoRA)

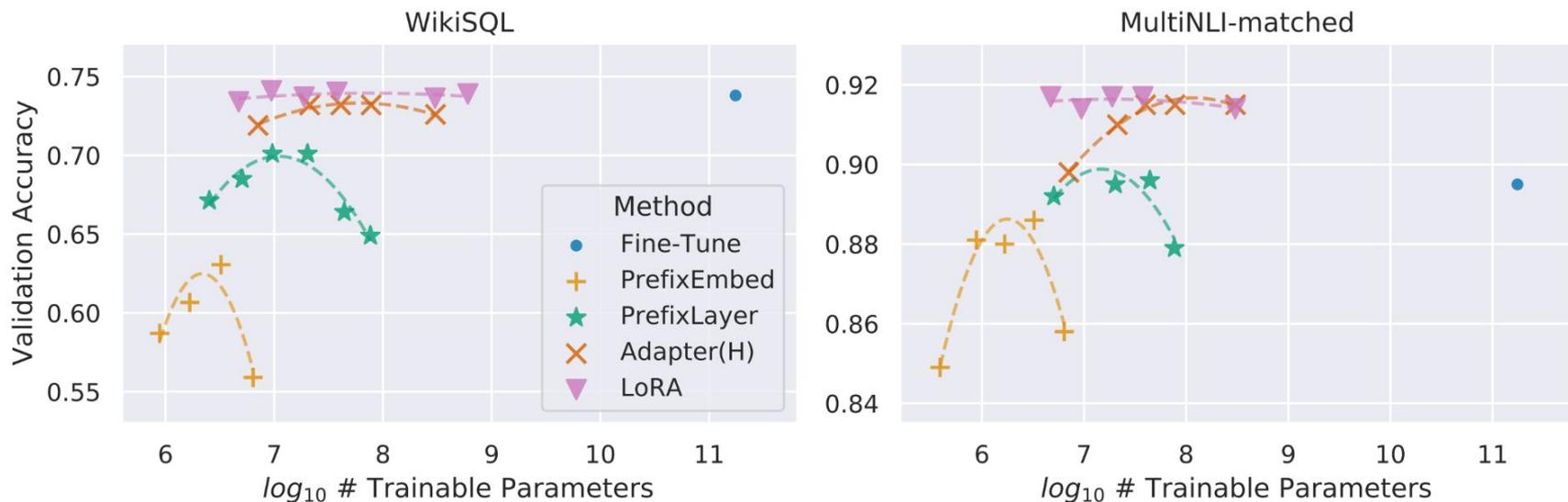


Figure 2: GPT-3 175B validation accuracy vs. number of trainable parameters of several adaptation methods on WikiSQL and MNLI-matched. LoRA exhibits better scalability and task performance. See Section F.2 for more details on the plotted data points.

PEFT: Low-Rank Adaptation (LoRA)

LoRA-FAIR: Federated LoRA fine-tuning with aggregation and initialization refinement

[J Bian](#), [L Wang](#), [L Zhang](#), [J Xu](#)

Proceedings of the IEEE/CVF International Conference on ..., 2025 · [openaccess.thecvf.com](#)

Lora3d: Low-rank self-calibration of 3d geometric foundation models

Authors [Ziqi Lu](#), [Heng Yang](#), [Danfei Xu](#), [Boyi Li](#), [Boris Ivanovic](#), [Marco Pavone](#), [Yue Wang](#)

Low-rank few-shot adaptation of vision-language models

[M Zanella](#), [I Ben Ayed](#)

Proceedings of the IEEE/CVF Conference on Computer Vision and ..., 2024 · [openaccess.thecvf.com](#)

Mtlora: Low-rank **adaptation** approach for efficient multi-task learning

[A Agiza](#), [M Neseem](#), [S Reda](#) - Proceedings of the IEEE/CVF ..., 2024 - [openaccess.thecvf.com](#)

Rethinking few-shot **adaptation** of vision-language models in two stages

[M Farina](#), [M Mancini](#), [G Iacca](#)... - Proceedings of the IEEE ..., 2025 - [openaccess.thecvf.com](#)

Transfer Learning: Helps you converge faster!

6th Robot Learning Workshop: Pretraining, Fine-Tuning, and Generalization with Large Scale Models

@NeurIPS 2023
16 Dec
Hall B2

Workshop on

Generalizable Priors for Robot Manipulation

CoRL 2025 | Seoul, Korea | Saturday, September 27th, 2025

Pretraining for Robotics (PT4R)

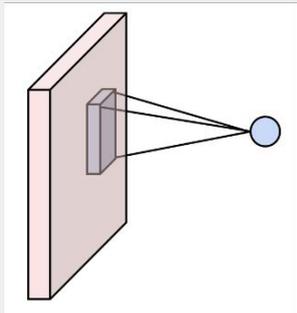
Workshop at the 2023 International Conference on Robotics and Automation - ICRA
London, May 29 2023, full-day workshop

Summary: Training NN

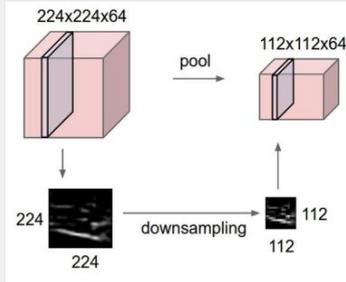
- **One-Time Setup**
 - Activation Functions
 - Data Preprocessing
 - Weight Initialization
 - Dropout
- **Training Dynamics**
 - Learning Rate Schedule
 - Hyperparameter Tuning
- **After Training**
 - Model Ensembles
 - Transfer Learning

Recap: Components of Convolutional Networks

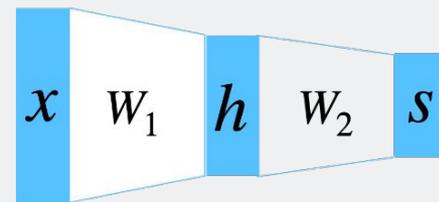
Convolution Layers



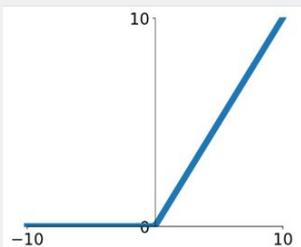
Pooling Layers



Fully-Connected Layers



Activation Function



Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

Next Up

Question: How should we put them together?

So far... Image Classification

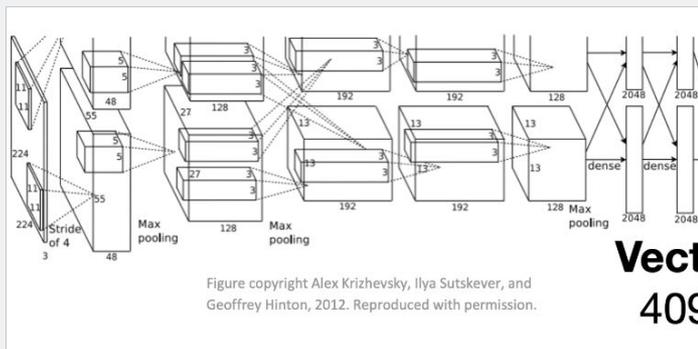


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Fully connected:

4096 to 10



Vector:
4096

Chocolate Pretzels

Granola Bar

Potato Chips

Water Bottle

Popcorn

Computer Vision Tasks

Classification



"Chocolate Pretzels"

No spatial extent

Semantic Segmentation



Chocolate Pretzels,
Shelf

No objects, just pixels

Object Detection



Flipz, Hershey's, Keese's

Multiple objects

Instance Segmentation



Computer Vision Tasks

Classification



“Chocolate Pretzels”

No spatial extent

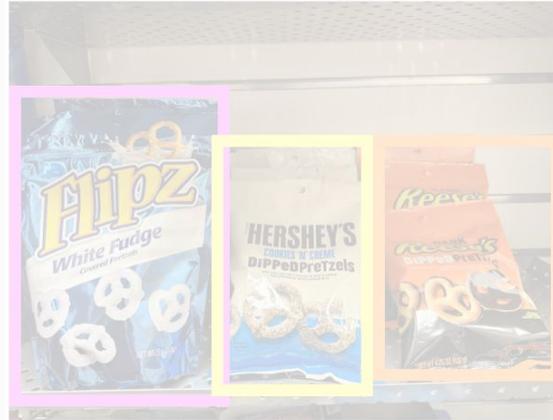
Semantic Segmentation



Chocolate Pretzels,
Shelf

No objects, just pixels

Object Detection



Flipz, Hershey's, Keese's

Multiple objects

Instance Segmentation



Classification: Transferring to New Tasks

Classification



“Chocolate Pretzels”

No spatial extent

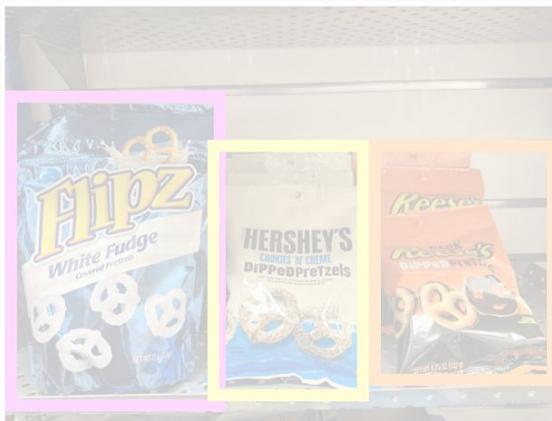
Semantic Segmentation



Chocolate Pretzels,
Shelf

No objects, just pixels

Object Detection



Flipz, Hershey's, Keese's

Multiple objects

Instance Segmentation



Object Detection

Classification



"Chocolate Pretzels"

No spatial extent

Semantic Segmentation



Chocolate Pretzels,
Shelf

No objects, just pixels

Object Detection



Flipz, Hershey's, Keese's

Multiple objects

Instance Segmentation



Object Detection: Task Definition

Input: Single RGB image

Output: A set of detected objects;
For each object predict:

1. Category label (from a fixed set of labels)
2. Bounding box (four numbers: x, y, width, height)



Object Detection: Challenges

Multiple outputs: Need to output variable numbers of objects per image

Multiple types of output: Need to predict "what" (category label) as well as "where" (bounding box)

Large images: Classification works at 224x224; need higher resolution for detection, often ~800x600



Object Detection: Bounding Boxes

Bounding boxes are typically *axis-aligned*



Object Detection

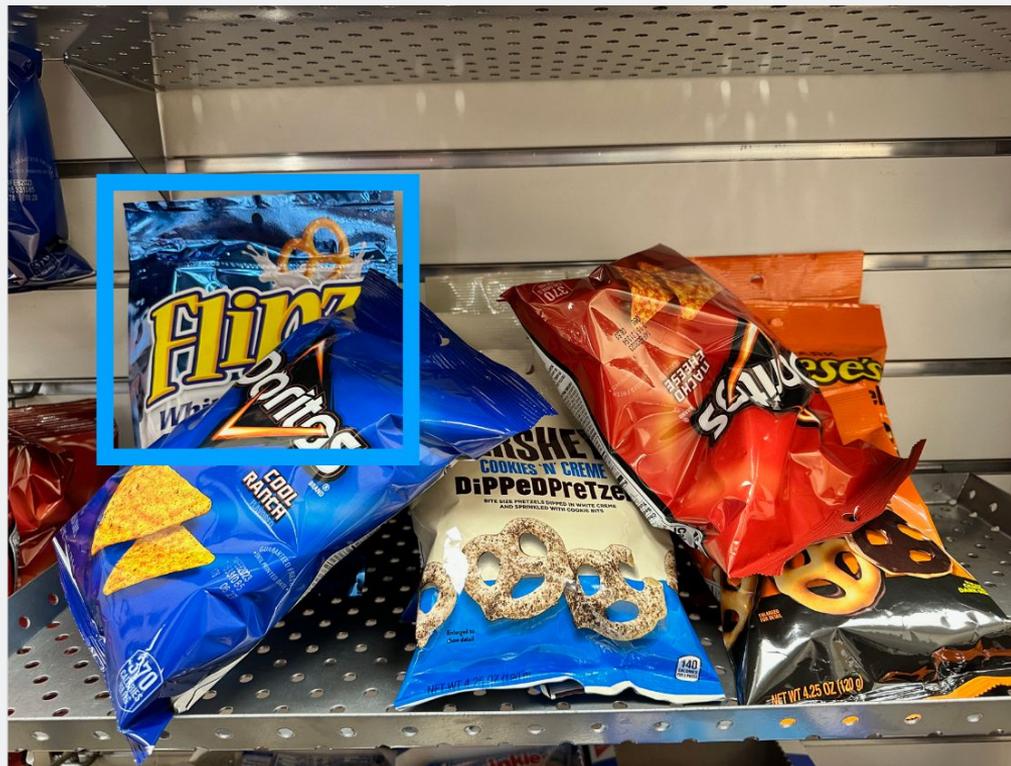
Bounding boxes are typically *axis-aligned*

Oriented boxes are much less common



Object Detection: Modal vs. Amodal Boxes

Bounding boxes cover only the visible portion of the object



Zhu et al, "Semantic Amodal Segmentation", CVPR 2017
https://openaccess.thecvf.com/content_cvpr_2017/papers/Zhu_Semantic_Amodal_Segmentation_CVPR_2017_paper.pdf

Object Detection: Modal vs. Amodal Boxes

Bounding boxes cover only the visible portion of the object

Amodal detection: box covers the entire extent of the object, even occluded parts



Object Detection: Modal vs. Amodal Boxes

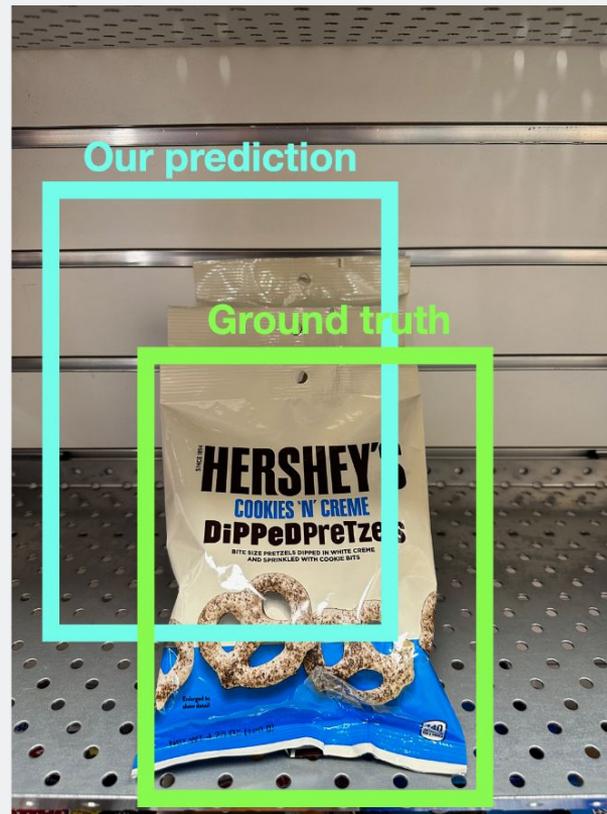
“Modal” detection: Bounding boxes (usually) cover only the visible portion of the object

Amodal detection: box covers the entire extent of the object, even occluded parts



Comparing Boxes: IoU (Intersection over Union)

How can we compare our prediction to the ground-truth box?



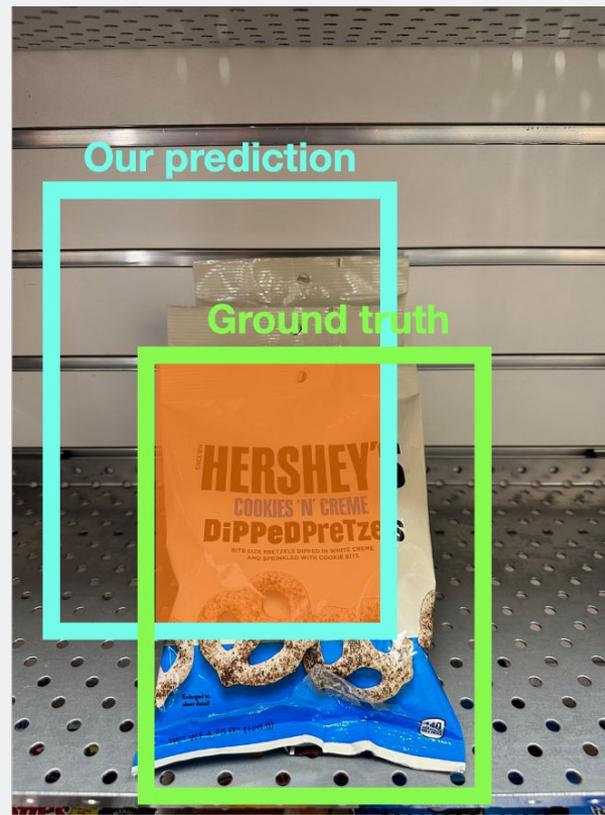
Comparing Boxes: IoU (Intersection over Union)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

Area of Intersection

Area of Union



Comparing Boxes: IoU (Intersection over Union)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

Area of Intersection

Area of Union



Comparing Boxes: IoU (Intersection over Union)

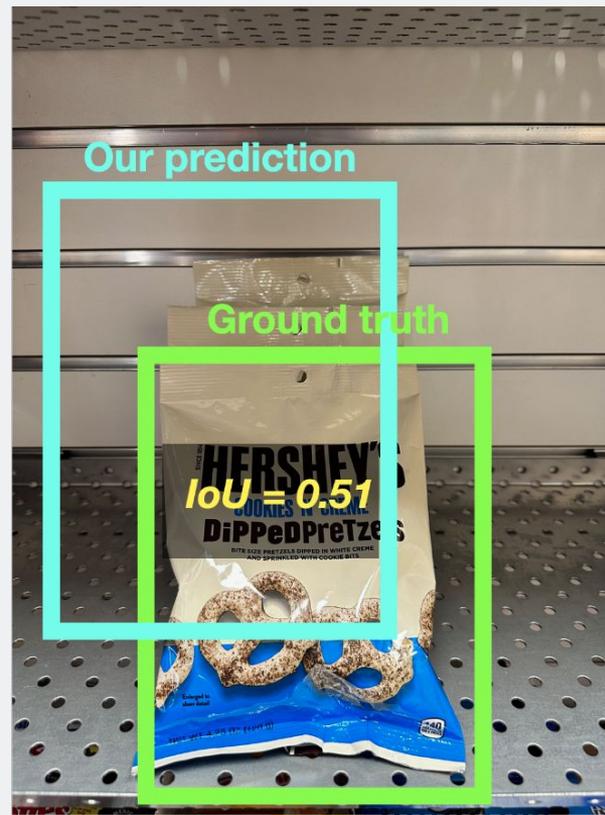
How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

Area of Intersection

Area of Union

IoU > 0.5 is “decent”,



Comparing Boxes: IoU (Intersection over Union)

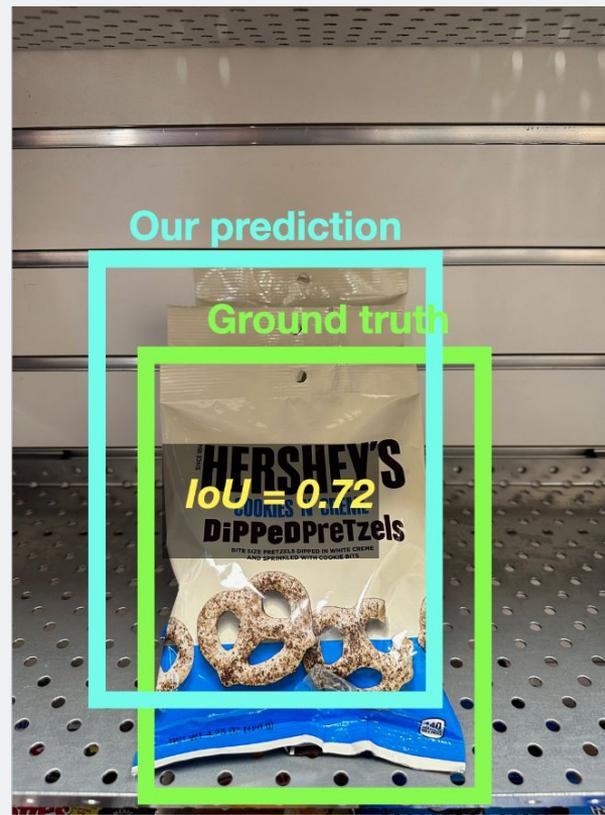
How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

Area of Intersection

Area of Union

IoU > 0.5 is “decent”,
IoU > 0.7 is “pretty good”,



Comparing Boxes: IoU (Intersection over Union)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

Area of Intersection

Area of Union

IoU > 0.5 is “decent”,
IoU > 0.7 is “pretty good”,
IoU > 0.9 is “almost perfect”



Detecting a single object

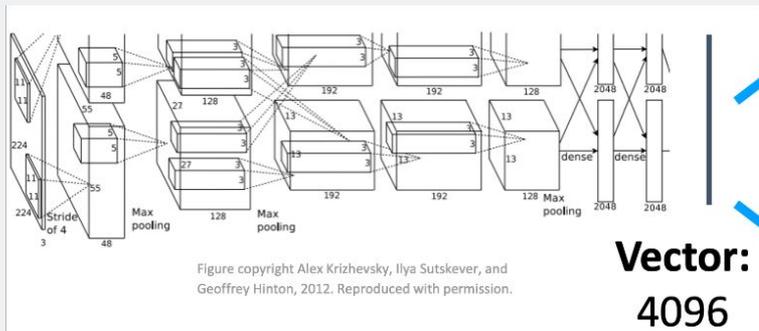


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Treat localization as a regression problem!

Fully connected:
4096 to 10

Vector:
4096

Fully connected:
4096 to 4

What??

Class scores:

Chocolate Pretzels: 0.9
Granola Bar: 0.02
Potato Chips: 0.02
Water Bottle: 0.02
Popcorn: 0.01
....

Box coordinates:
(x, y, w, h)

Where??

Correct Label:

Chocolate Pretzels

Softmax Loss

Weighted Sum

L2 Loss

Multitask Loss

Loss

$$L = L_{cls} + \lambda L_{reg}$$

Correct coordinates:
(x', y', w', h')

Detecting a single object

Often pretrained on ImageNet: Transfer learning

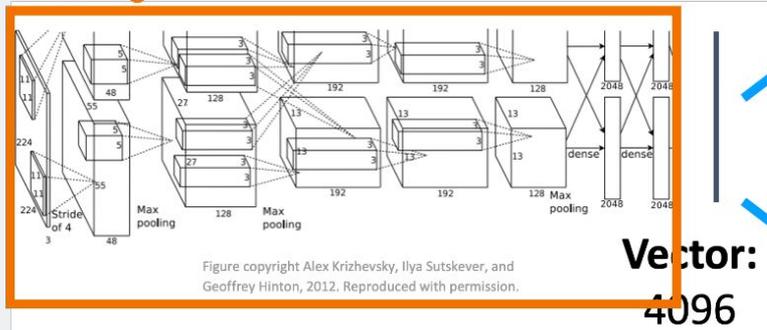


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

Fully connected:
4096 to 10

Fully connected:
4096 to 4

What??

Class scores:

- Chocolate Pretzels: 0.9
- Granola Bar: 0.02
- Potato Chips: 0.02
- Water Bottle: 0.02
- Popcorn: 0.01
-

Correct Label:

Chocolate Pretzels

Softmax Loss

Multitask Loss

Weighted Sum

Loss

$$L = L_{cls} + \lambda L_{reg}$$

Box coordinates:

(x, y, w, h)

L2 Loss

Where??

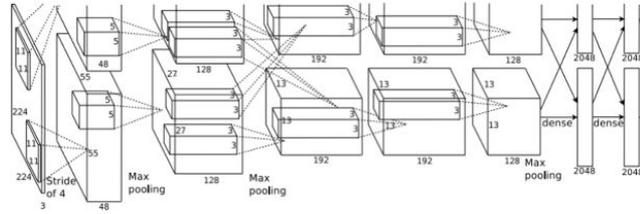
Correct coordinates:

(x', y', w', h')

Treat localization as a regression problem!

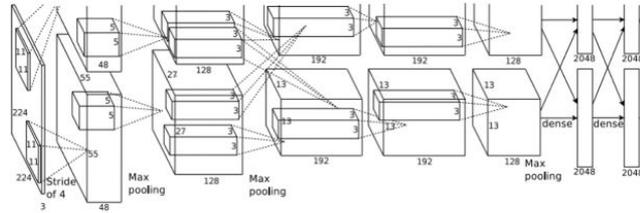
Problem: Images can have more than one object!

Detecting Multiple Objects



Hershey's: (x, y, w, h)

4 numbers

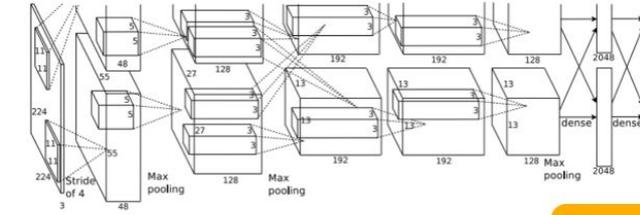


Hershey's: (x, y, w, h)

Flipz: (x, y, w, h)

Reese's: (x, y, w, h)

12 numbers



Chips: (x, y, w, h)

Chips: (x, y, w, h)

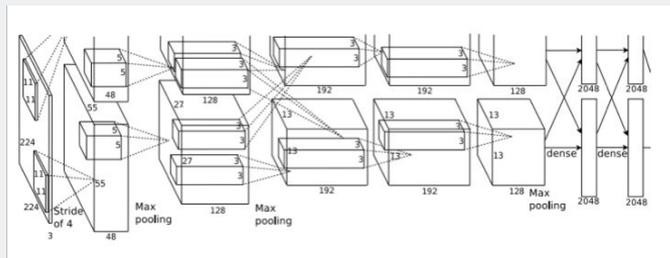
.....

Many numbers!

**Need different numbers of
output per image**

Detecting Multiple Objects: Sliding Window

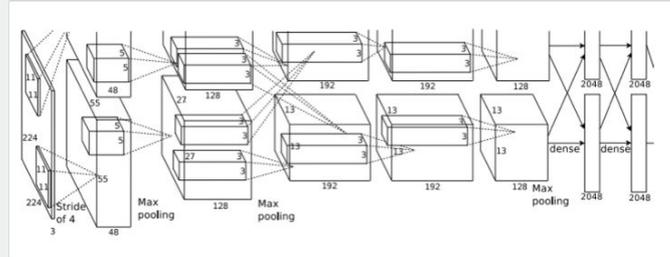
Apply a CNN to many different crops
of the image, CNN classifies each
crop as object or background



Hershey's: **No**
Flipz: **No**
Reese's: **No**
Background: **Yes**

Detecting Multiple Objects: Sliding Window

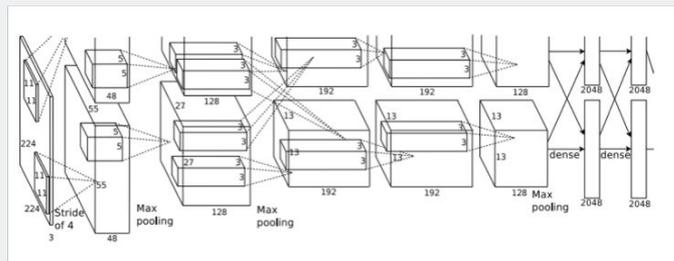
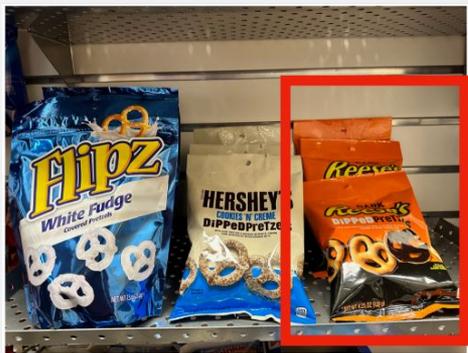
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Hershey's: **No**
Flipz: **Yes**
Reese's: **No**
Background: **No**

Detecting Multiple Objects: Sliding Window

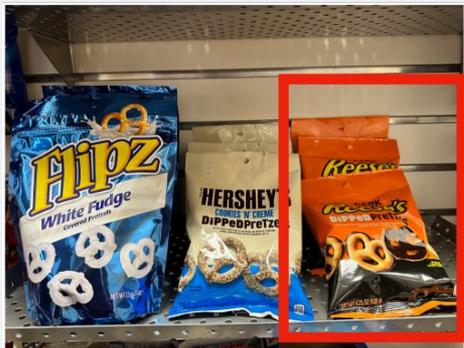
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Hershey's: **No**
Flipz: **No**
Reese's: **Yes**
Background: **No**

Detecting Multiple Objects: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Q: How many possible boxes are there in a HxW image?

Aha Slides (In-class participation)

<https://ahaslides.com/Z44PZ>



Detecting Multiple Objects: Sliding Window

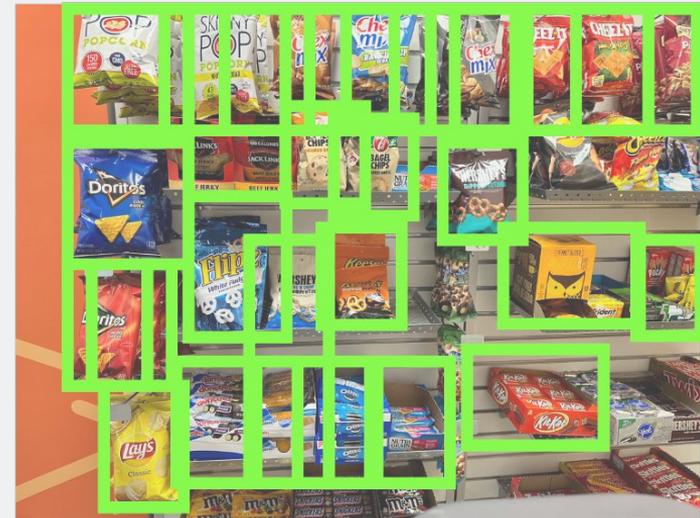
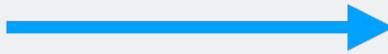
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

800 x 600 image has
~58M boxes. No way
we can evaluate them
all



Region Proposals

- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for “blob-like” image regions
- Relatively fast to run; e.g. **Selective Search** gives 2000 region proposals in a few seconds on CPU



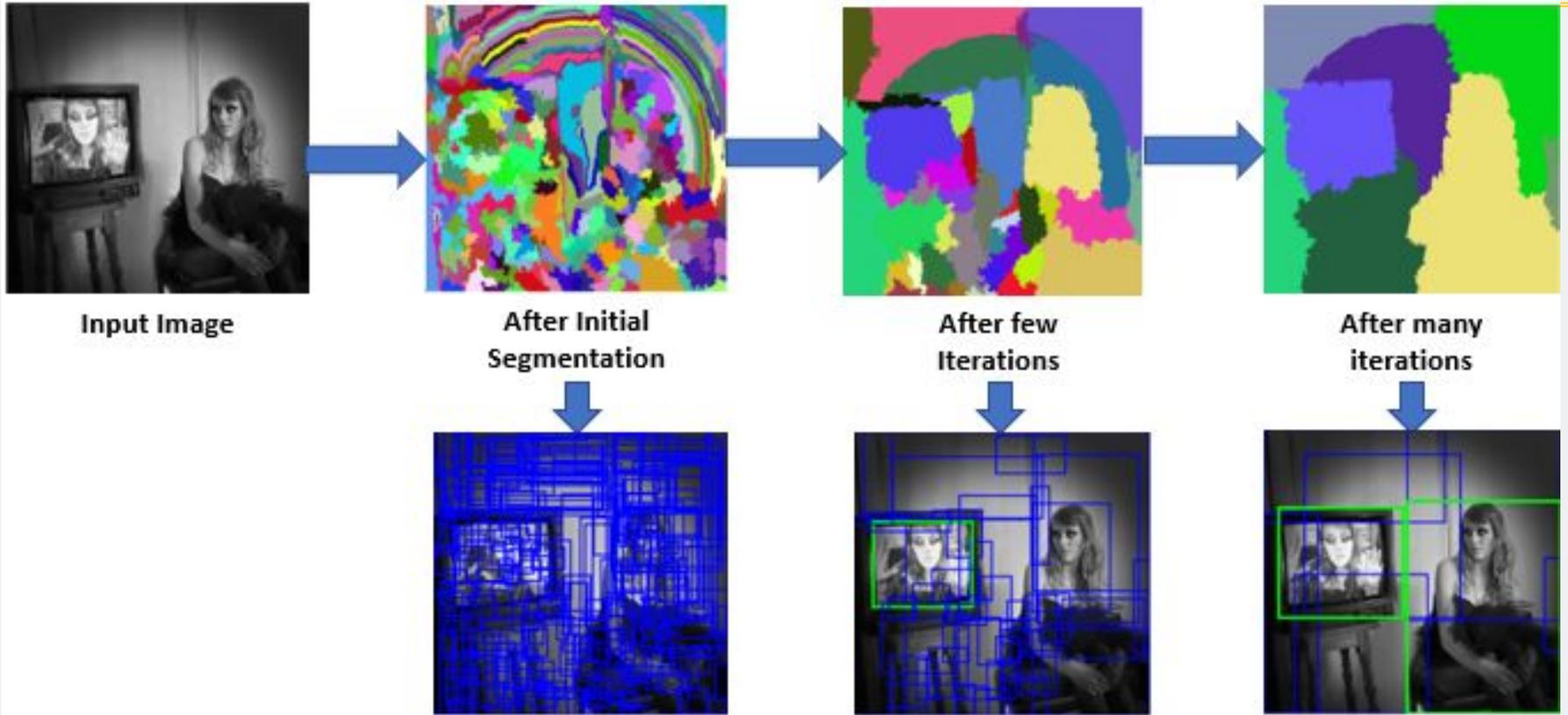
Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012

Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013

Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014

Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

Note on selective search



Felzenszwalb et al, "Efficient Graph-Based Image Segmentation"

<https://www.geeksforgeeks.org/selective-search-for-object-detection-r-cnn/>

Why do we still care about Region Proposals?

(CVPR 2024)

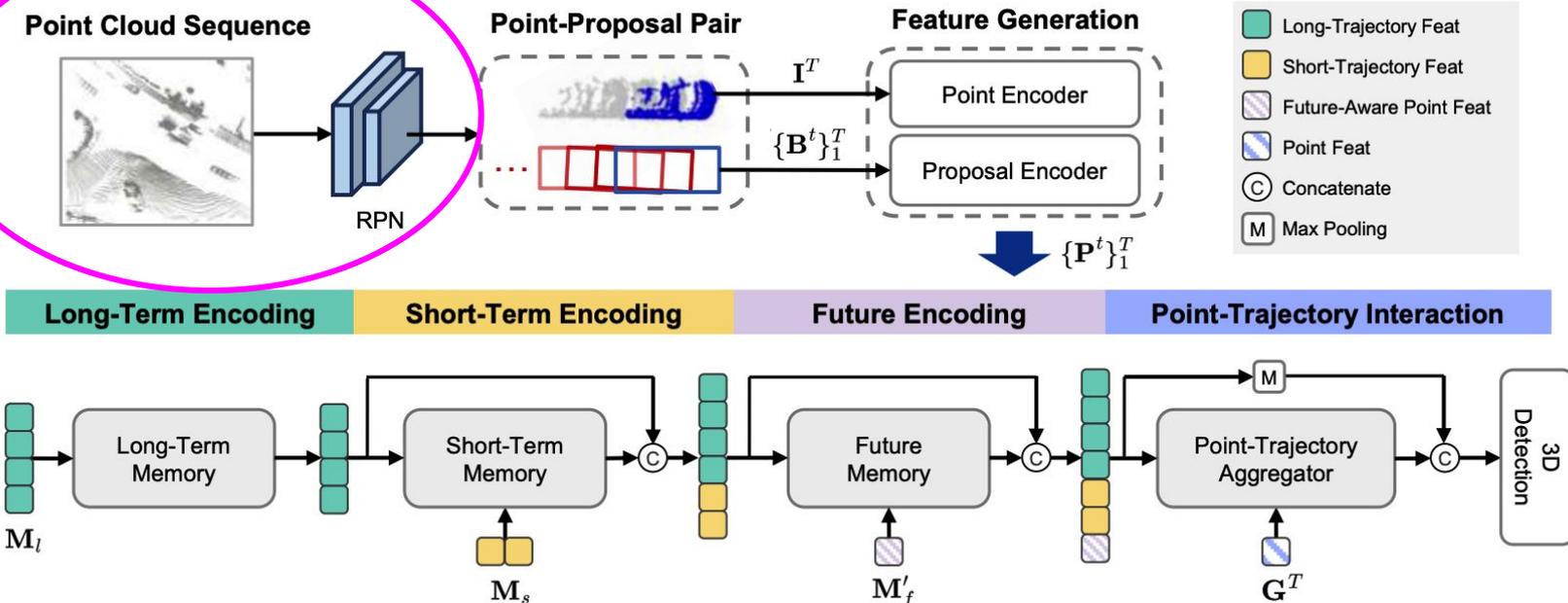
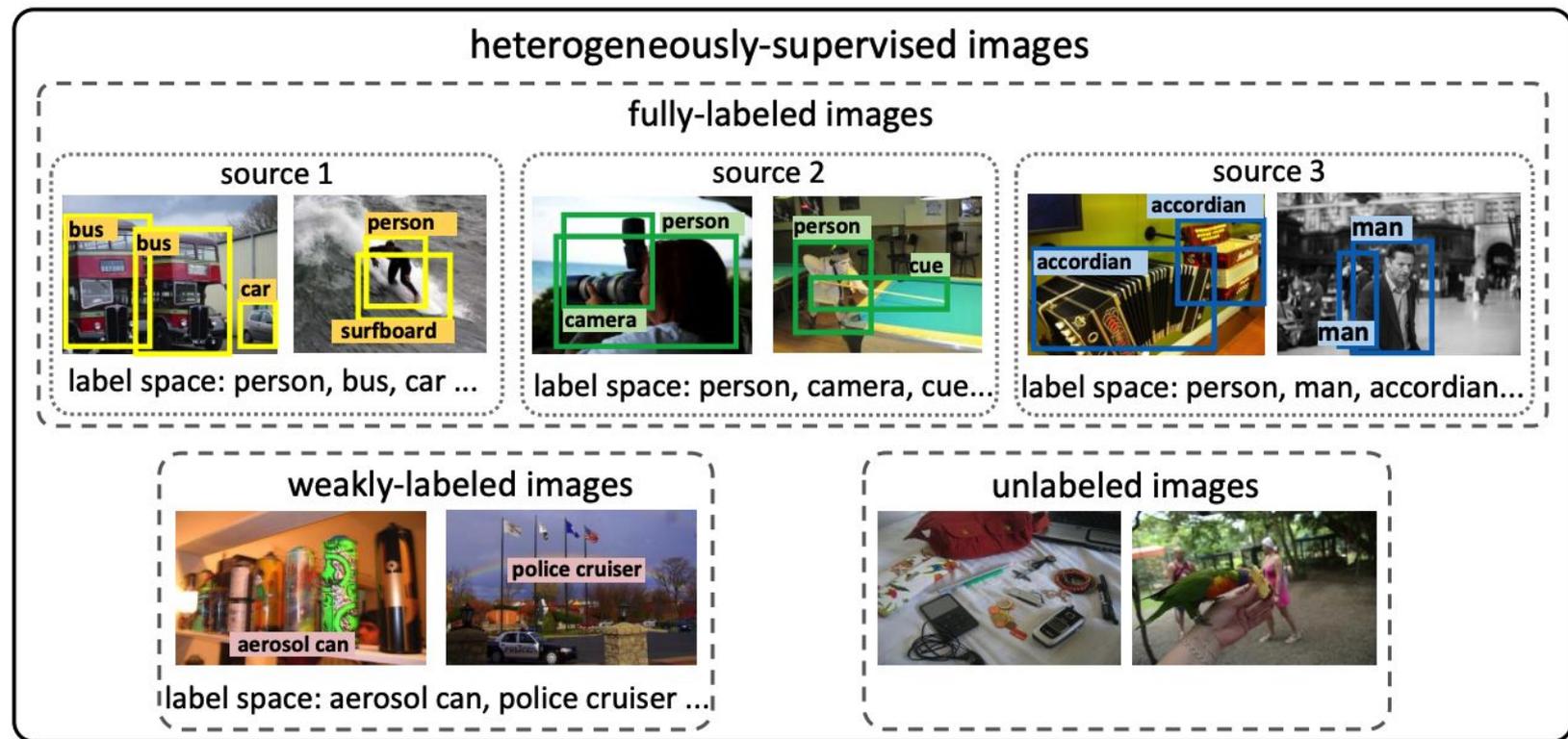


Figure 2. Overall framework of the proposed Point-Trajectory Transformer (PTT). First, we utilize a region proposal network (RPN) at timestamp T to generate proposals \mathbf{B}^T for each frame, sample the corresponding point-of-interest \mathbf{I}^T , and connect past T -frame 3D proposals to form proposal trajectories $\{\mathbf{B}^1, \dots, \mathbf{B}^T\}$. Then, we take the single-frame point cloud for each object and its previous multi-

https://openaccess.thecvf.com/content/CVPR2024/papers/Huang_PTT_Point-Trajectory_Transformer_for_Efficient_Temporal_3D_Object_Detection_CVPR_2024_paper.pdf

Why do we still care about Region Proposals?

(TPAMI
2024)



<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10552883>

“UniDetector”

Why do we still care about Region Proposals?

(TPAMI
2024,
cont'd)

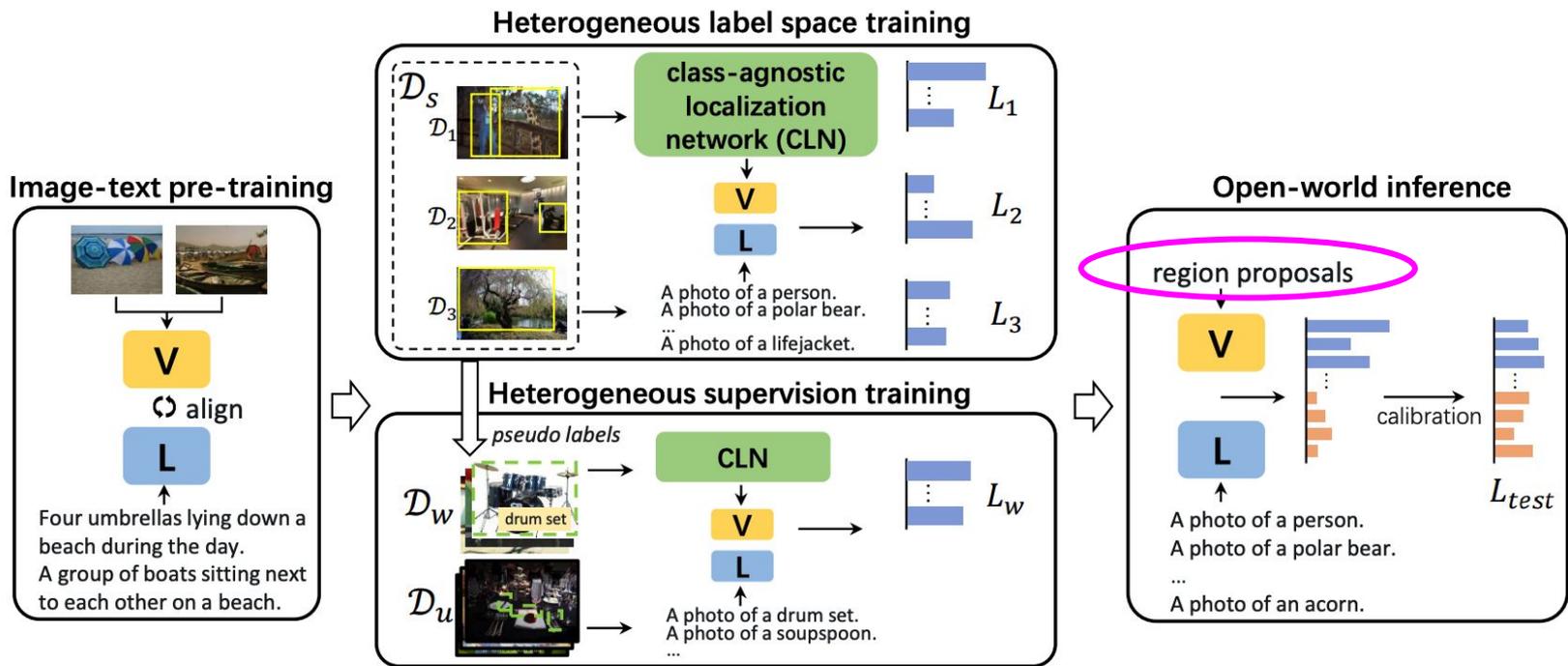


Fig. 2: **Overview of UniDetector.** It consists of four steps. With the image-text pre-training parameters, UniDetector is

<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10552883>

R-CNN: Region-Based CNN

R-CNN: Region-Based CNN

Input
image



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

Figure copyright Ross Girshick, 2015; source.
Reproduced with permission

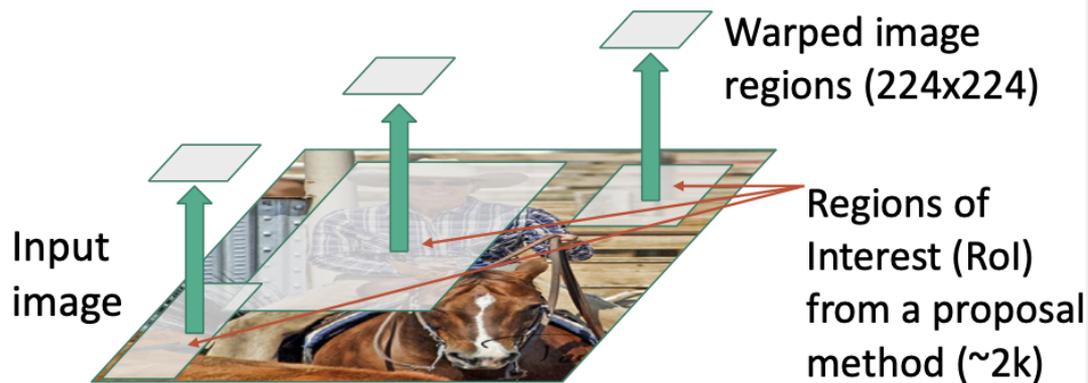
R-CNN: Region-Based CNN

R-CNN: Region-Based CNN



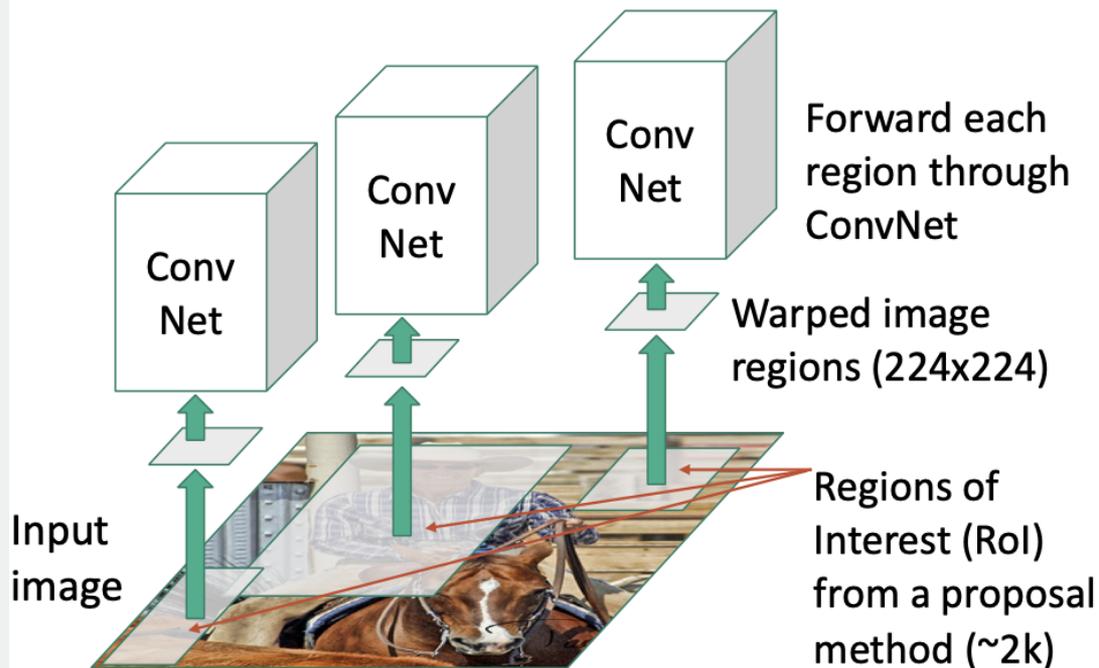
R-CNN: Region-Based CNN

R-CNN: Region-Based CNN



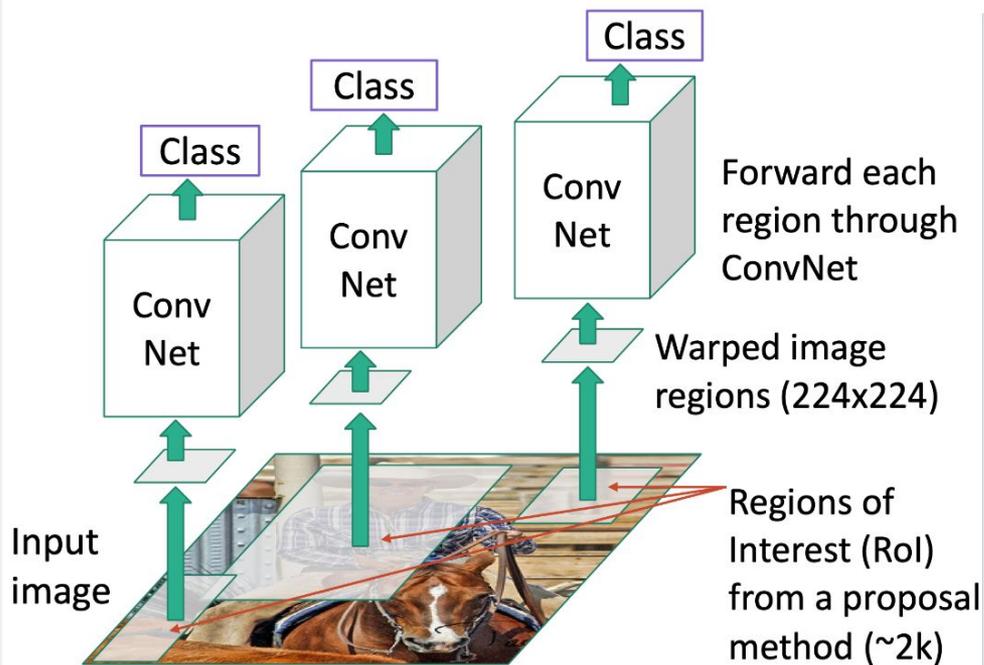
R-CNN: Region-Based CNN

R-CNN: Region-Based CNN



R-CNN: Region-Based CNN

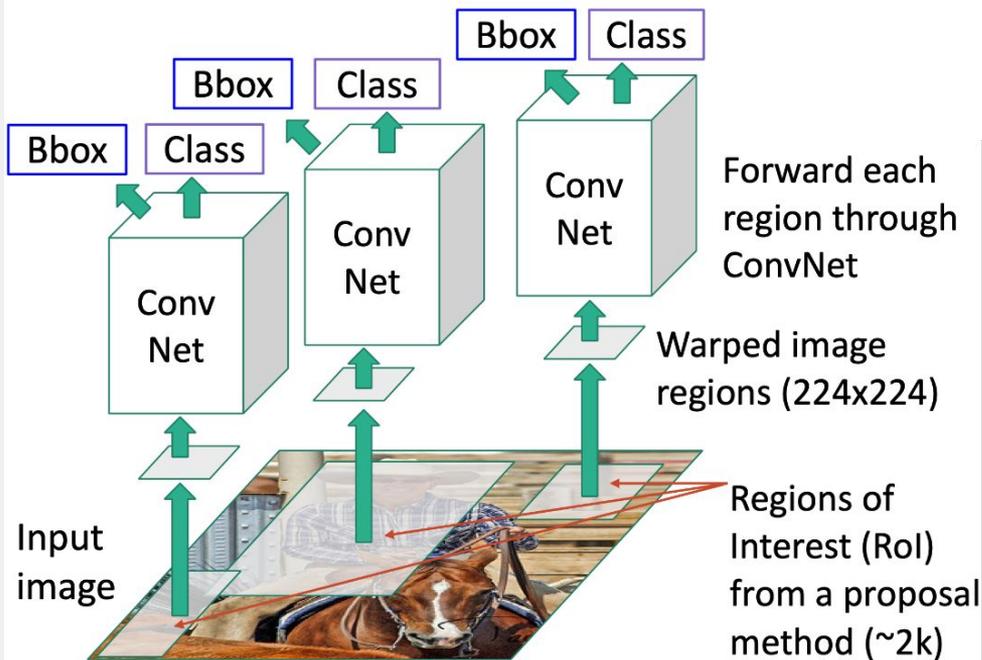
R-CNN: Region-Based CNN



Classify each region

R-CNN: Region-Based CNN

R-CNN: Region-Based CNN

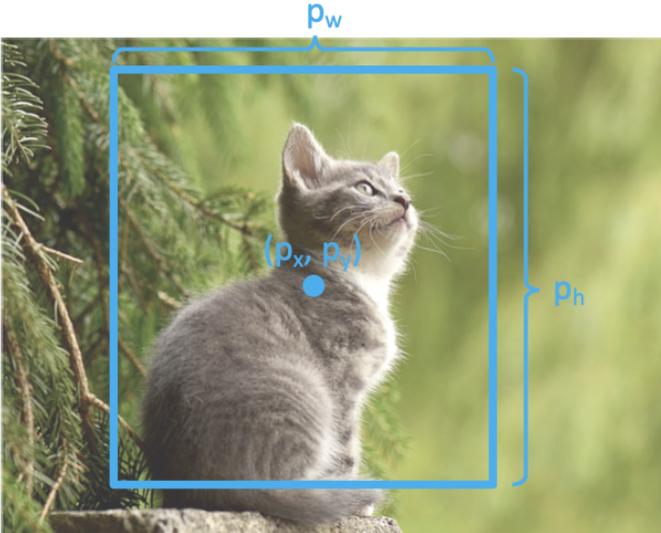


Classify each region

Bounding box regression:
Predict “transform” to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

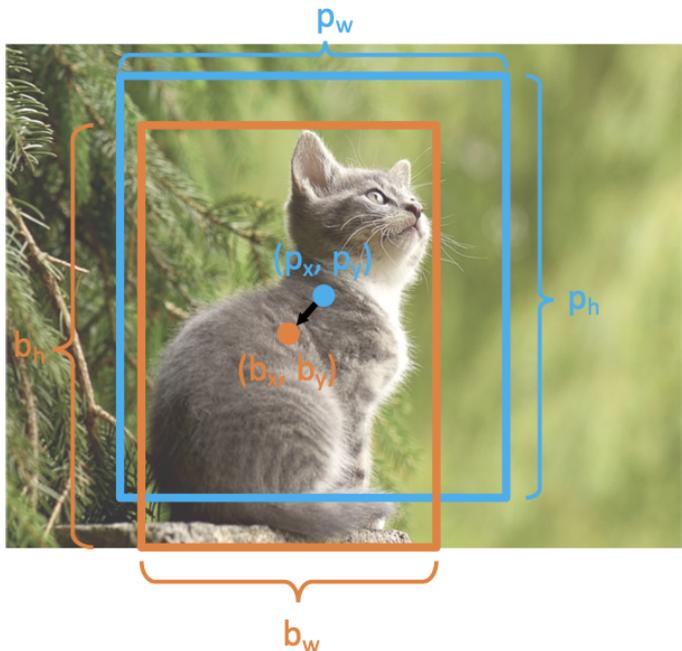
R-CNN: Box Regression

Consider a **region proposal** with center (p_x, p_y) , width p_w , height p_h



Model predicts a **transform** (t_x, t_y, t_w, t_h) to correct the region proposal

R-CNN: Box Regression



Consider a **region proposal** with center (p_x, p_y) , width p_w , height p_h

Model predicts a **transform** (t_x, t_y, t_w, t_h) to correct the region proposal

The **output box** is defined by:

$$b_x = p_x + p_w t_x$$

Shift center by amount relative to proposal size

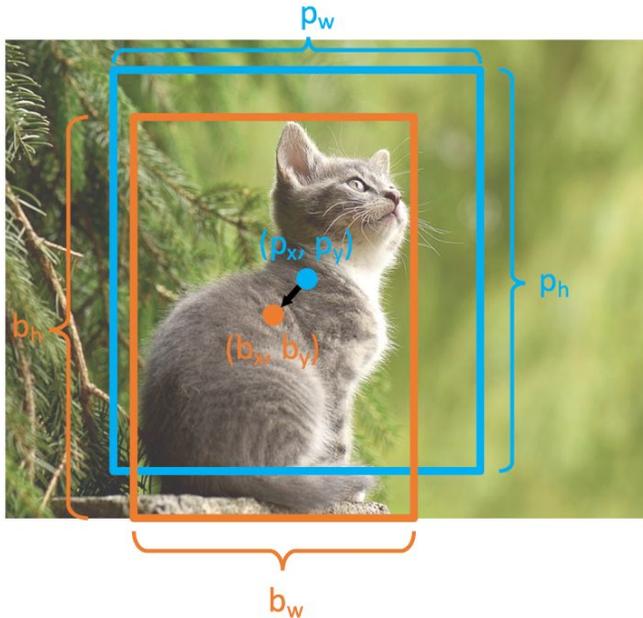
$$b_y = p_y + p_h t_y$$

$$b_w = p_w \exp(t_w)$$

Scale proposal; exp ensures that scaling factor is > 0

$$b_h = p_h \exp(t_h)$$

R-CNN: Box Regression



Consider a **region proposal** with center (p_x, p_y) , width p_w , height p_h

Model predicts a **transform** (t_x, t_y, t_w, t_h) to correct the region proposal

The **output box** is defined by:

$$b_x = p_x + p_w t_x$$

$$b_y = p_y + p_h t_y$$

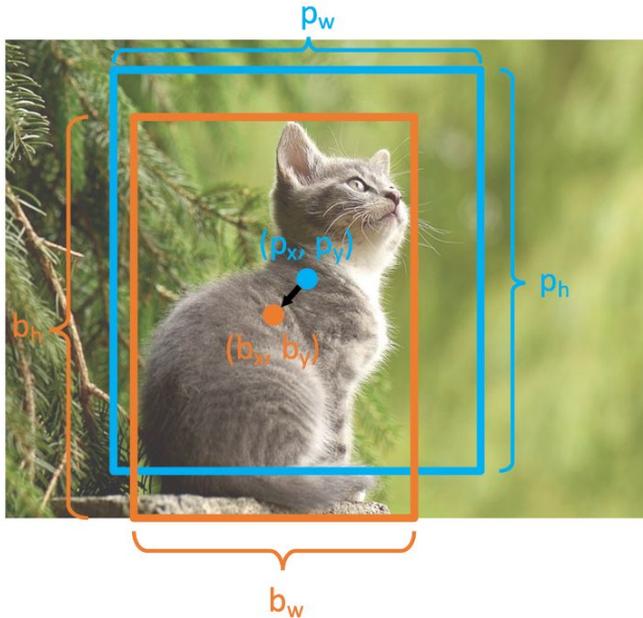
$$b_w = p_w \exp(t_w)$$

$$b_h = p_h \exp(t_h)$$

When transform is 0, output = proposal

L2 regularization encourages leaving proposal unchanged

R-CNN: Box Regression



Consider a **region proposal** with center (p_x, p_y) , width p_w , height p_h

Model predicts a **transform** (t_x, t_y, t_w, t_h) to correct the region proposal

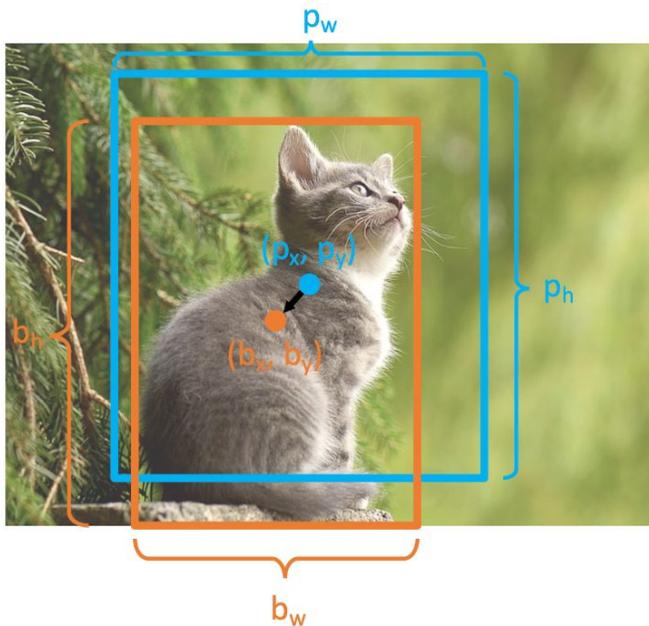
The **output box** is defined by:

$$\begin{aligned}b_x &= p_x + p_w t_x \\b_y &= p_y + p_h t_y \\b_w &= p_w \exp(t_w) \\b_h &= p_h \exp(t_h)\end{aligned}$$

Scale / Translation invariance:

Transform encodes *relative* difference between proposal and output; important since CNN doesn't see absolute size or position after cropping

R-CNN: Box Regression



Consider a **region proposal** with center (p_x, p_y) , width p_w , height p_h

Model predicts a **transform** (t_x, t_y, t_w, t_h) to correct the region proposal

The **output box** is defined by

$$b_x = p_x + p_w t_x$$

$$b_y = p_y + p_h t_y$$

$$b_w = p_w \exp(t_w)$$

$$b_h = p_h \exp(t_h)$$

Given **proposal** and **target output**, we can solve for the **transform** the network should output:

$$t_x = (b_x - p_x) / p_w$$

$$t_y = (b_y - p_y) / p_h$$

$$t_w = \log(b_w / p_w)$$

$$t_h = \log(b_h / p_h)$$

R-CNN: Training

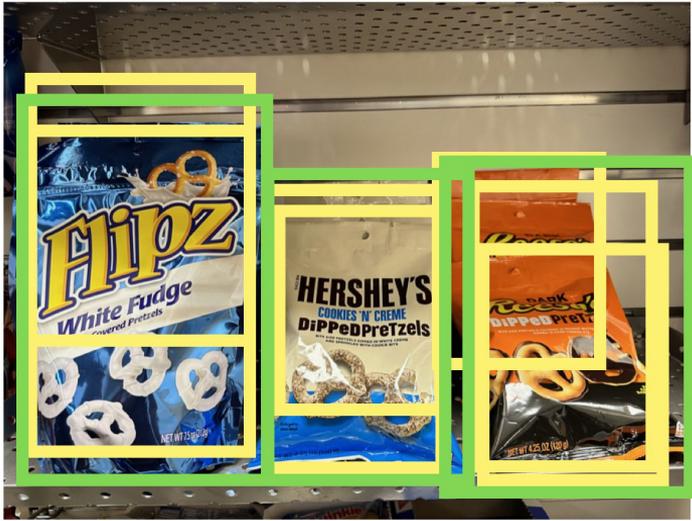
Input Image



Ground Truth

R-CNN: Training

Input Image

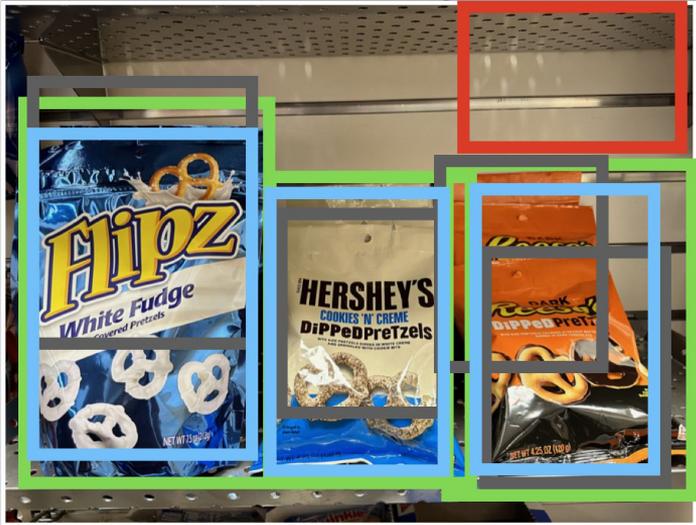


Ground Truth

Region Proposals

R-CNN: Training

Input Image



Ground Truth

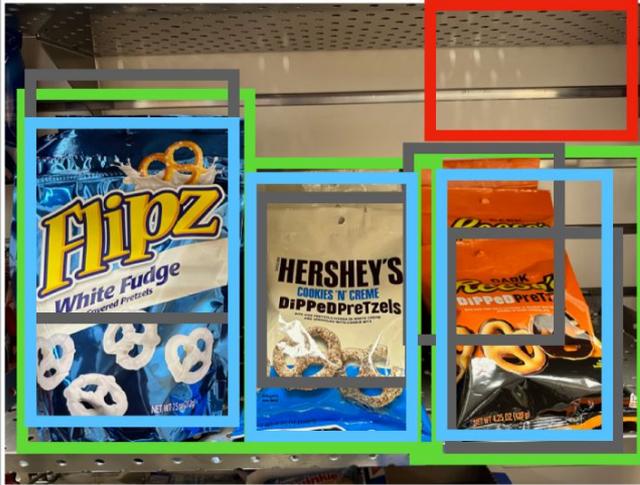
Positive

Neutral

Negative

R-CNN: Training

Input Image



Categorize each region proposal as **positive**, **negative** or neutral based on overlap with the Ground truth boxes:

Positive: > 0.5 IoU with a GT box

Negative: < 0.3 IoU with all GT boxes

Neutral: between 0.3 and 0.5 IoU with GT boxes

Ground Truth

Positive

Neutral

Negative

R-CNN: Training

Input Image



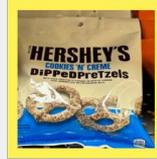
Ground Truth

Positive

Neutral

Negative

Run each region through CNN
Positive regions: predict class and transform
Negative regions: just predict class



Crop pixels from
each positive and
negative proposal,
resize to 224 x 224

R-CNN: Training

Input Image



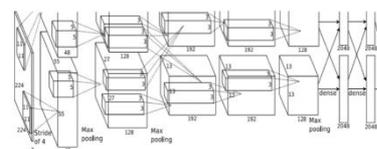
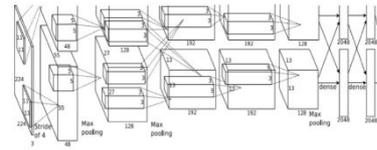
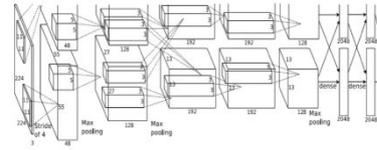
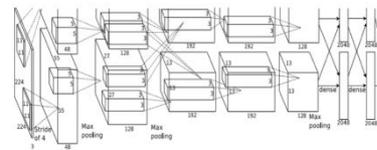
Ground Truth

Positive

Neutral

Negative

Run each region through CNN
Positive regions: predict class and transform
Negative regions: just predict class



Class target: Flipz
Box target: →



Class target: Hershey's
Box target: →



Class target: Reese's
Box target: →



Class target: Background
Box target: None

R-CNN: Test Time

Input Image



Region Proposals

Run proposal method:

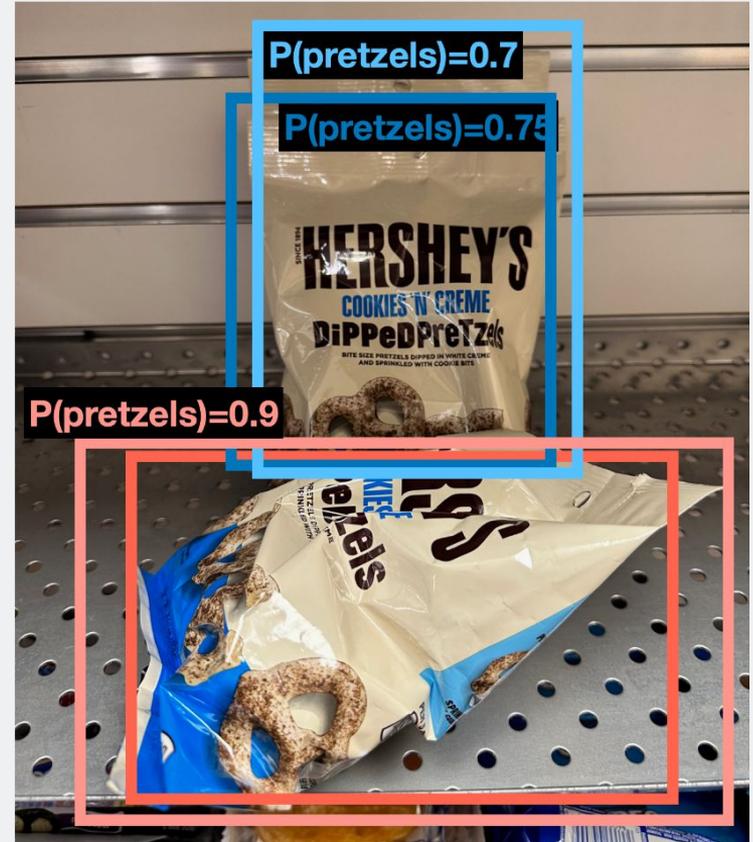
1. Run CNN on each proposal to get class scores, transforms
2. Threshold class scores to get a set of detections

2 Problems:

1. CNN often outputs overlapping boxes
2. How to set thresholds?

Overlapping Boxes

Problem: Object detectors often output many overlapping detections

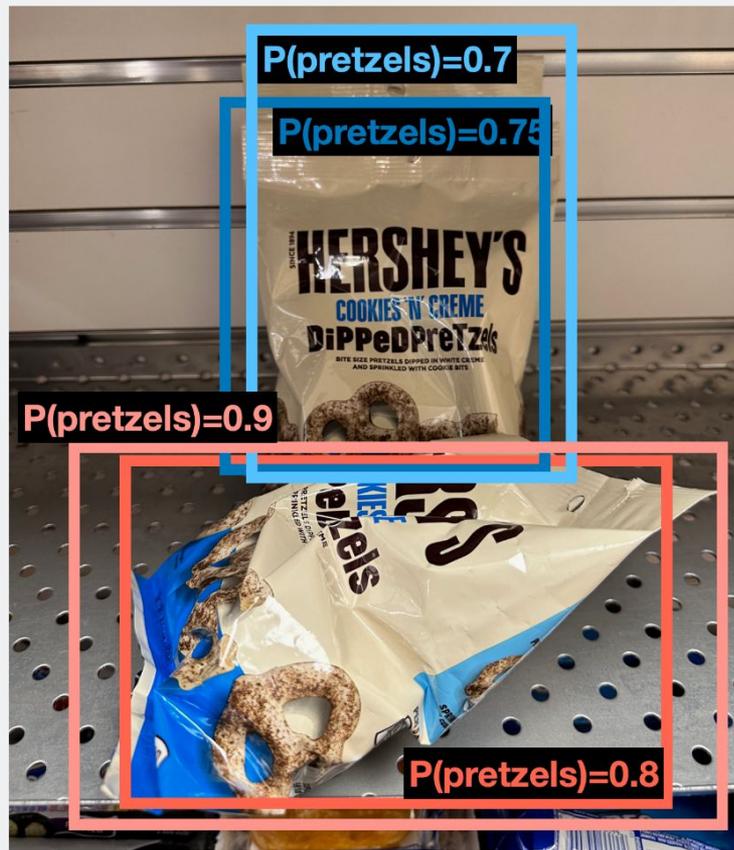


Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with $IoU > \text{threshold}$ (e.g. 0.7)
3. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with $IoU > \text{threshold}$ (e.g. 0.7)
3. If any boxes remain, GOTO 1

$$IoU(\text{red}, \text{red}) = 0.8$$

$$IoU(\text{red}, \text{blue}) = 0.03$$

$$IoU(\text{red}, \text{light blue}) = 0.05$$



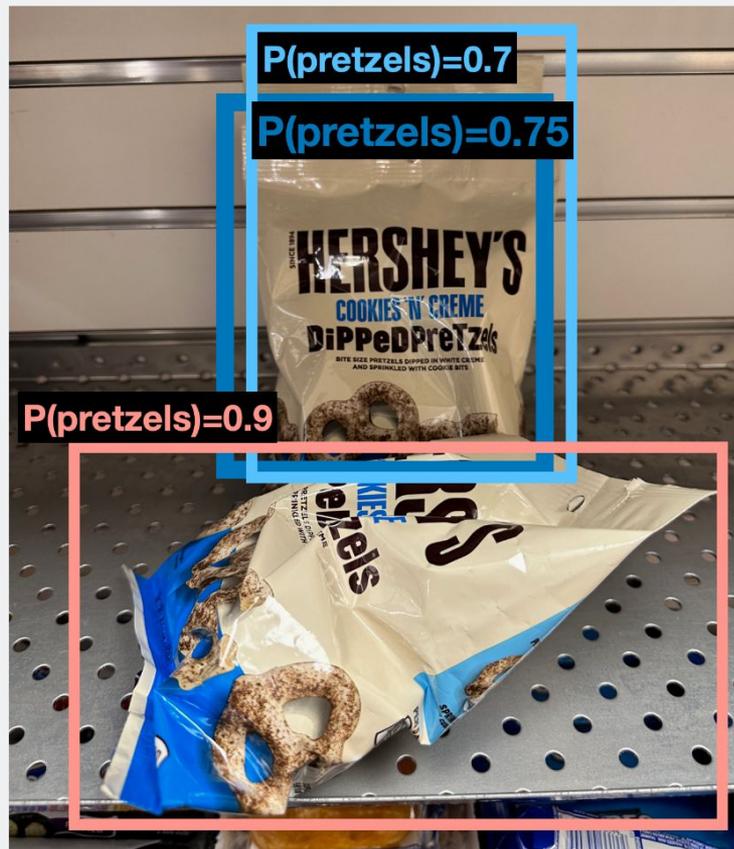
Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with $IoU > \text{threshold}$ (e.g. 0.7)
3. If any boxes remain, GOTO 1

$$IoU(\text{■}, \text{■}) = 0.85$$

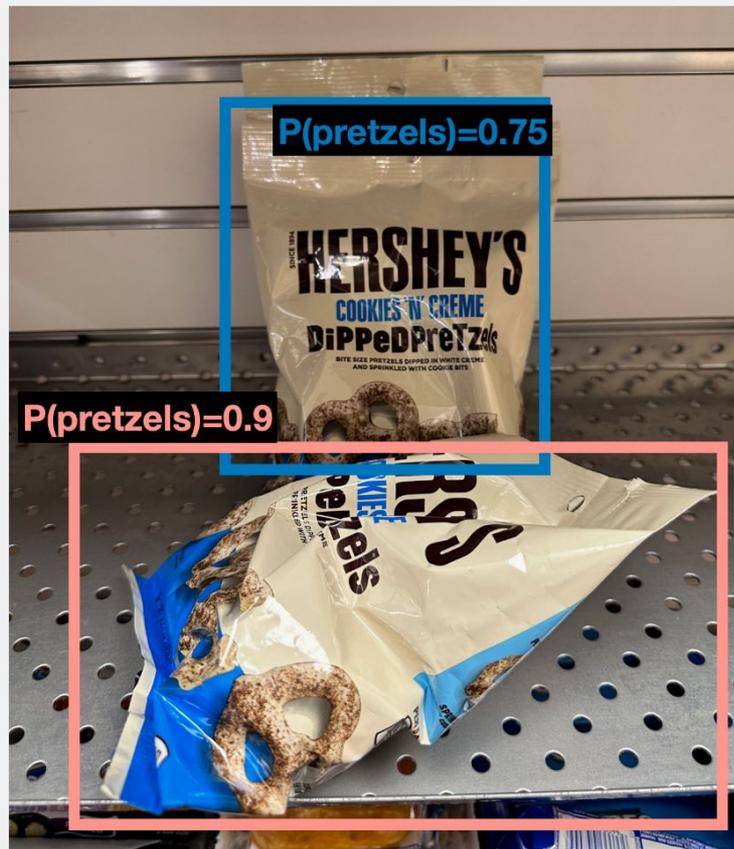


Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with $IoU > \text{threshold}$ (e.g. 0.7)
3. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with $IoU > \text{threshold}$ (e.g. 0.7)
3. If any boxes remain, GOTO 1

Problem: NMS may eliminate “good” boxes when objects are highly overlapping... no good solution



[Crowd image](#) is free for commercial use under the [Pixabay license](#)

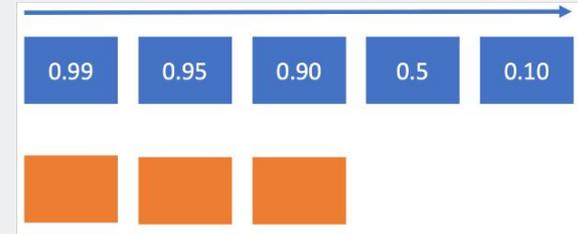
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)

All pretzel detections sorted by score



All ground-truth pretzel boxes

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative

All pretzel detections sorted by score

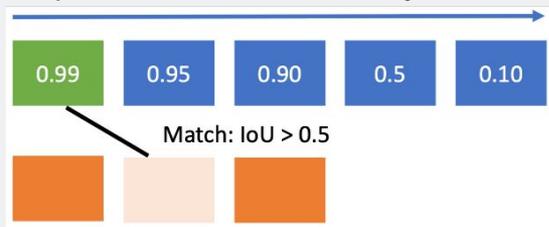


All ground-truth pretzel boxes

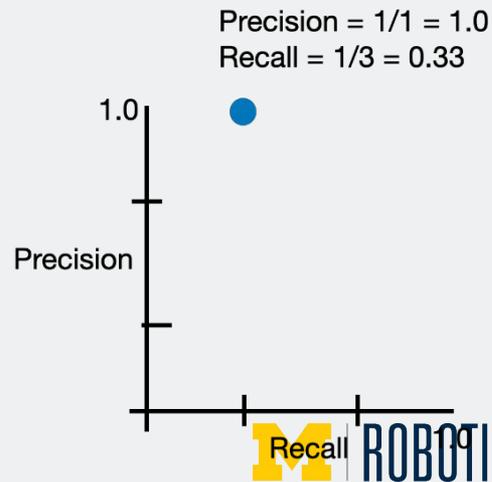
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve

All pretzel detections sorted by score



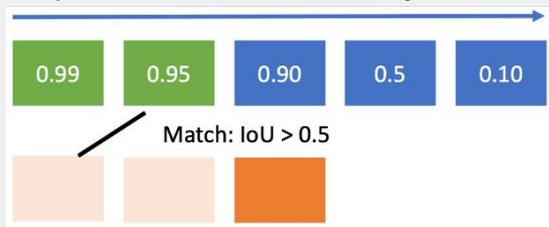
All ground-truth pretzel boxes



Evaluating Object Detectors: Mean Average Precision (mAP)

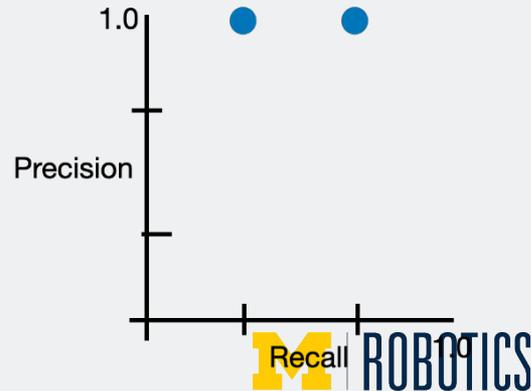
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve

All pretzel detections sorted by score



All ground-truth pretzel boxes

Precision = $2/2 = 1.0$
Recall = $2/3 = 0.67$



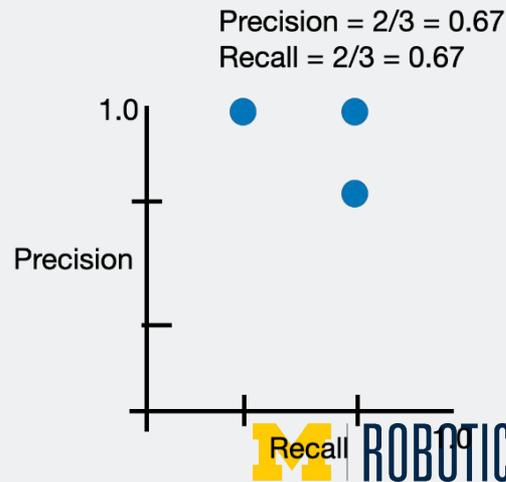
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve

All pretzel detections sorted by score



All ground-truth pretzel boxes



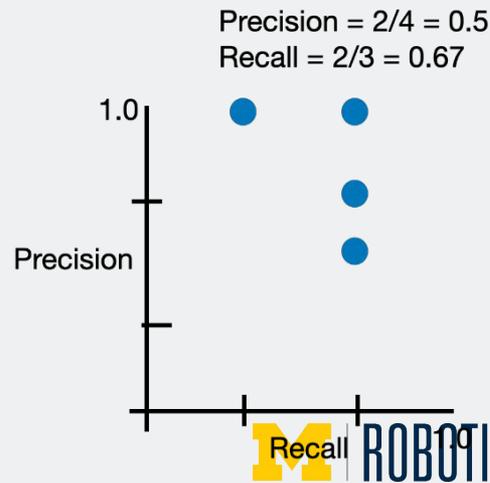
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve

All pretzel detections sorted by score



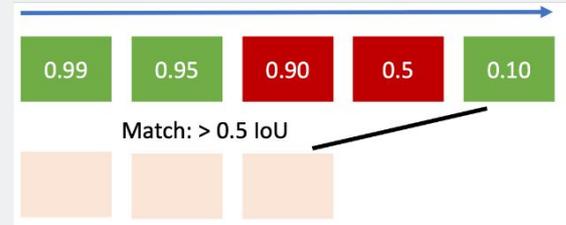
All ground-truth pretzel boxes



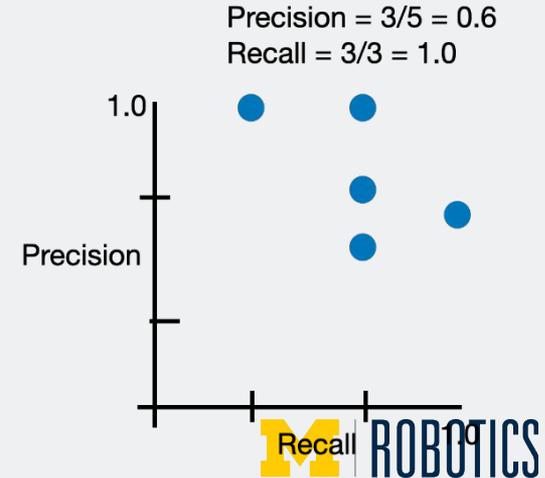
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve

All pretzel detections sorted by score



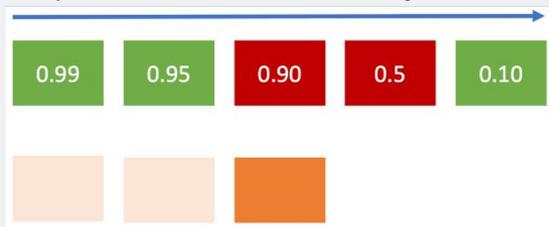
All ground-truth pretzel boxes



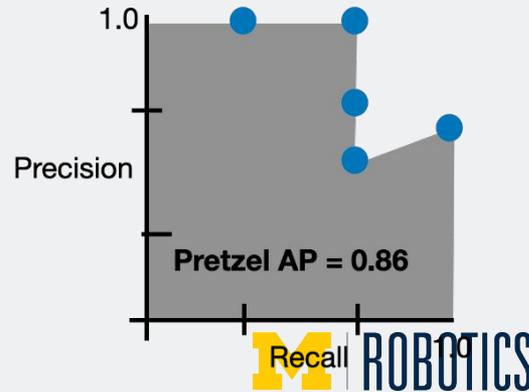
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve
 2. Average Precision (AP) = area under PR curve

All pretzel detections sorted by score



All ground-truth pretzel boxes

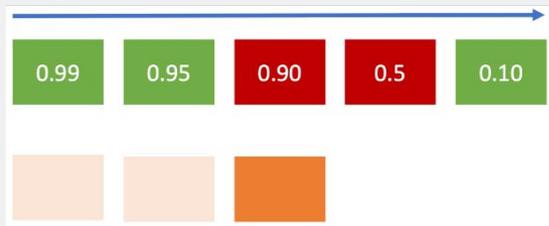


Evaluating Object Detectors: Mean Average Precision (mAP)

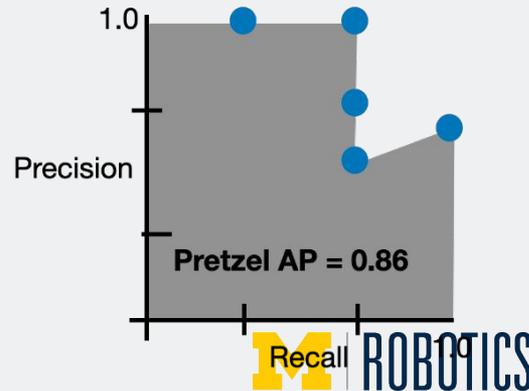
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve
 2. Average Precision (AP) = area under PR curve

How to get AP = 1.0: Hit all GT boxes with $\text{IoU} > 0.5$, and have no “false positive” detections ranked above any “true positives”

All pretzel detections sorted by score



All ground-truth pretzel boxes



Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve
 2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category

Flipz AP = 0.60
Hershey's AP = 0.85
Reese's AP = 0.81
mAP@0.5 = 0.75

Discussion Feb 16

- Questions / Feedback
- Clarification on R-CNN training
- Clarification on Canvas Quiz Questions
- Midterm practice session - Adi
- Office Hour