KENTECH
Korea Institute of Energy Technology

Visionary Course – Energy AI
Week 08

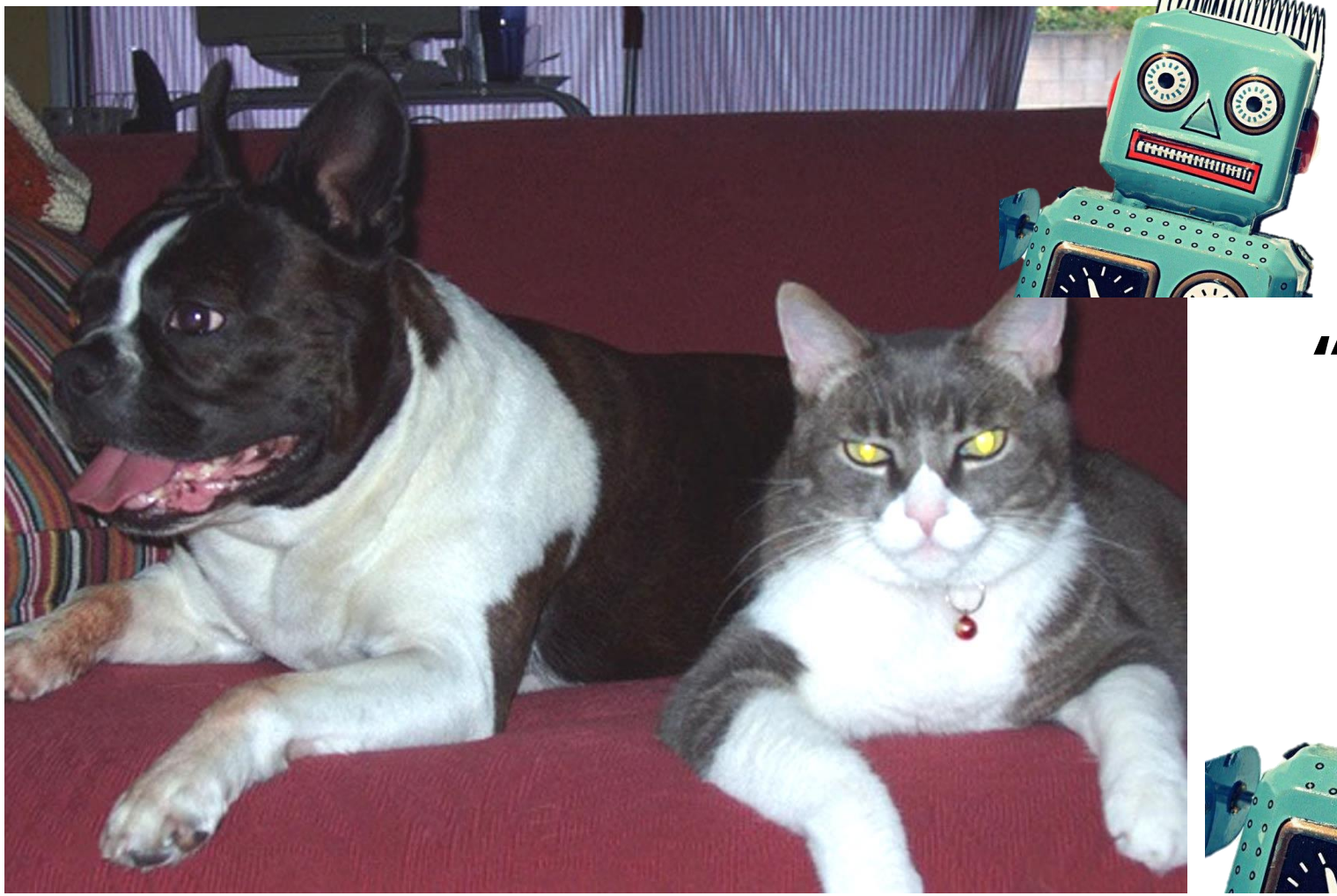
Apr. 22, 2022

Seokju Lee

KENTECH
Korea Institute of Energy Technology

Week 08a – Object Detection

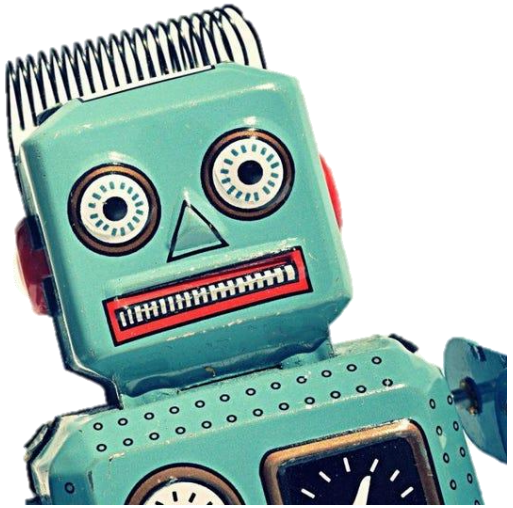
Limitation of Image Classification: Dog or Cat?

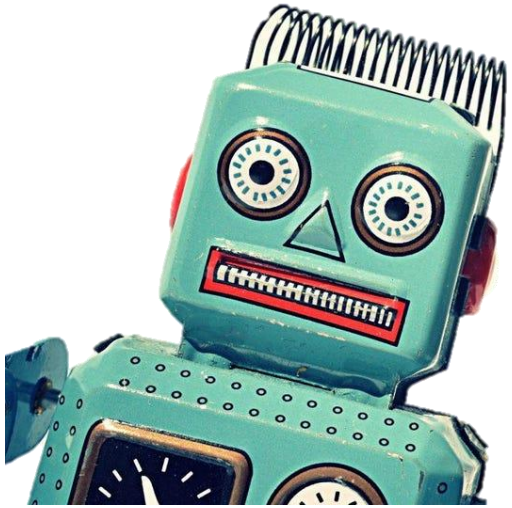


"Sofa?"

"Dog?"

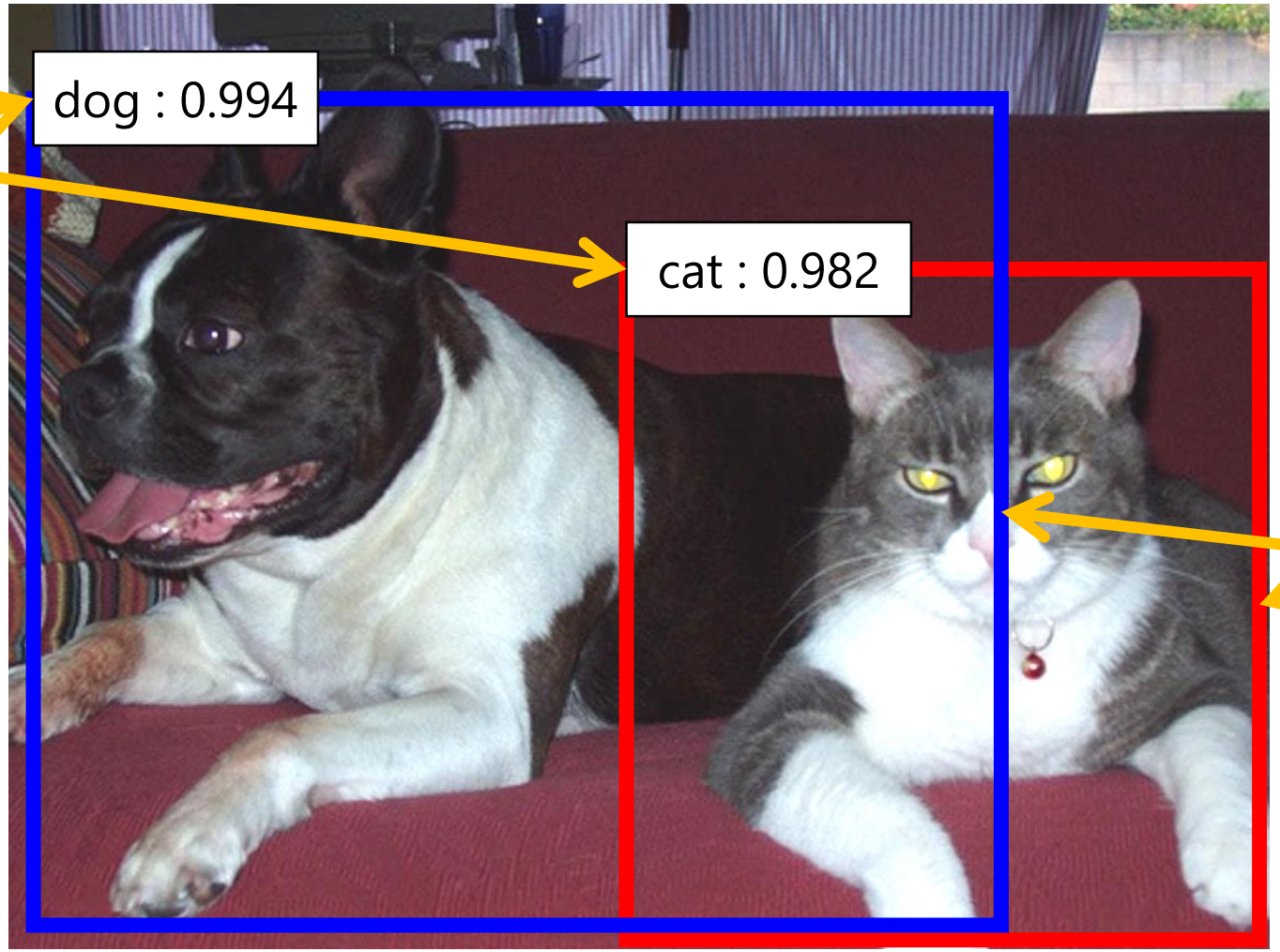
"Cat?"





[1] S. Ren, et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *NIPS* (2015).

Object Detection: What and Where?

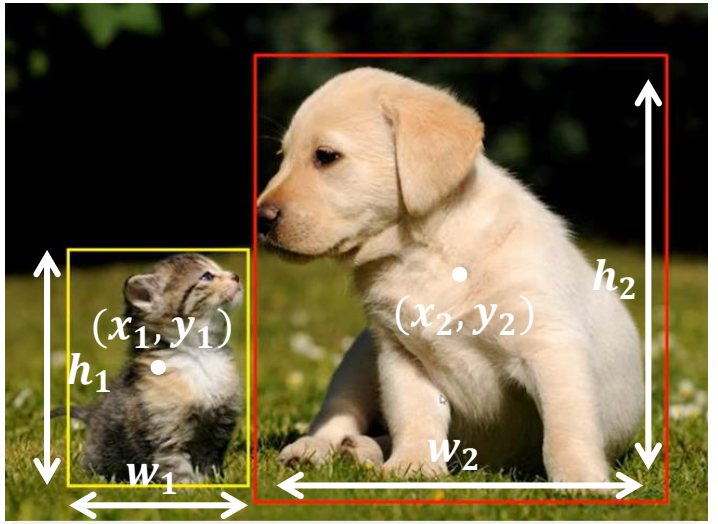


[1] S. Ren, et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *NIPS* (2015).

Object Detection: Input & Output

: Task of assigning **labels** & **bounding boxes** to all objects in the image.

Classify **which class** the object belongs to.



Find the coordinates of the bounding box **where** the object is located, and classify **which class** it belongs to.

Images	Class (=label)
I_1	cat
I_2	cat
I_3	dog

Classification

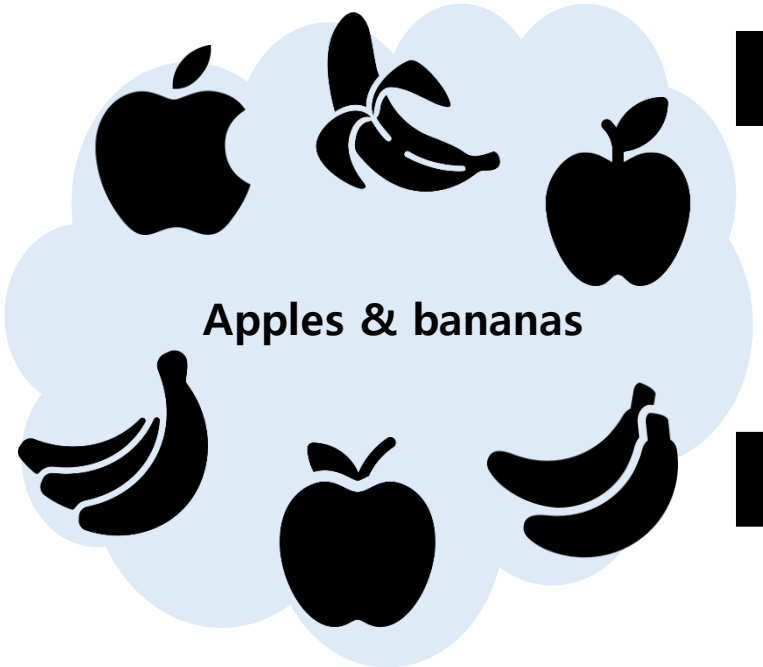
Images	Class (=label)	x	y	w	h
I_1	cat	60	210	100	180
I_1	dog	200	50	340	360
I_2	car	46	250	100	80

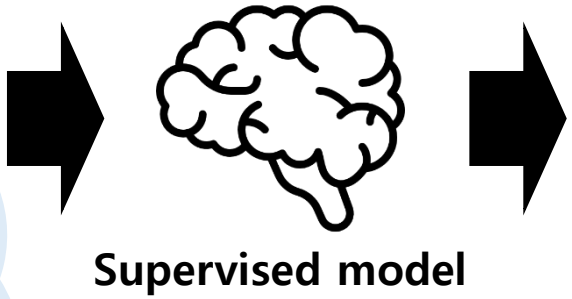
Classification + Regression

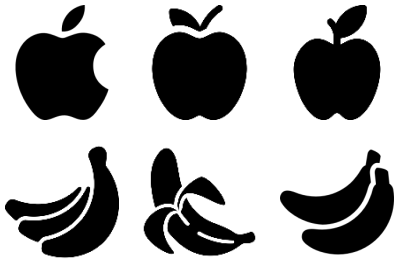
*Back to Basics: Supervised & Unsupervised Learning

Let's review – **Two** major approaches of machine learning?

1) **Supervised learning** requires **labels** for training
2) **Unsupervised learning** does **not** require **labels** for training

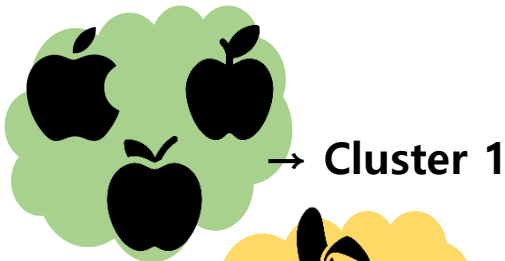




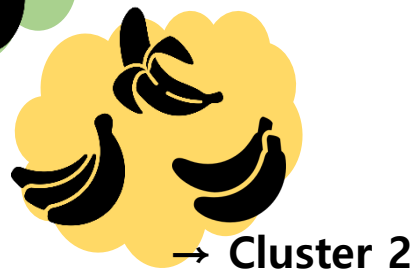


= "Apple"

= "Banana"

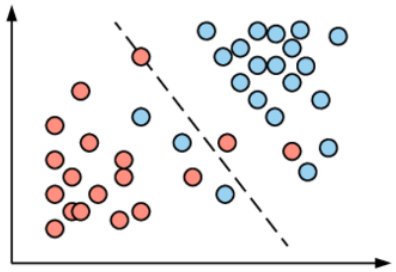


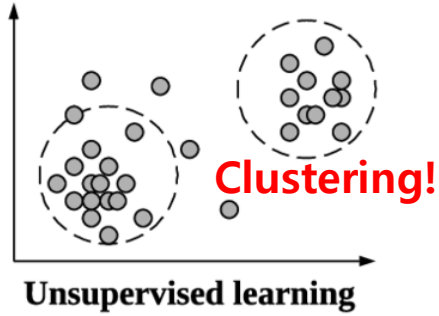
→ Cluster 1



→ Cluster 2

→ **Mathematical representation :**



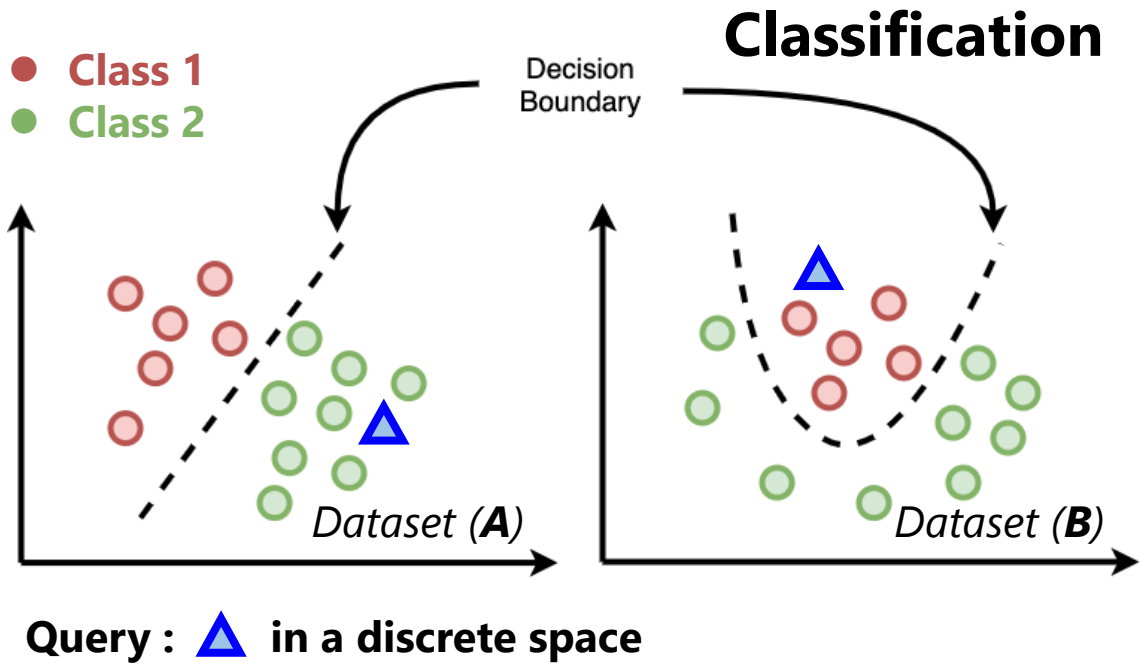


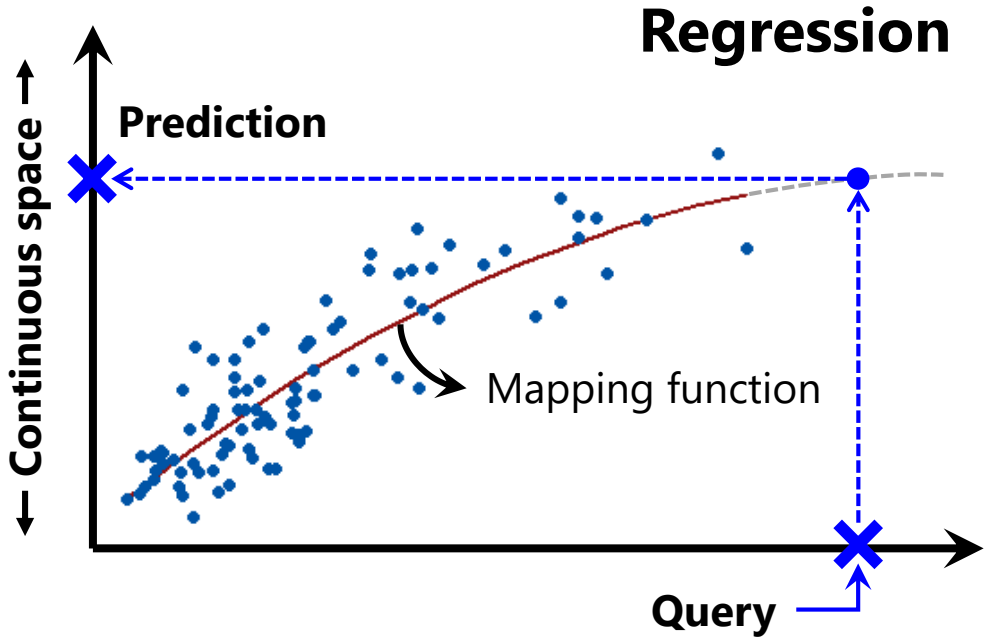
↑ Images from edX (IBM ML0101EN)

*Back to Basics: Classification & Regression

Supervised learning problem can be represented into **two tasks**.

1) **Classification** is the task of predicting a **discrete** class label.
2) **Regression** is the task of predicting a **continuous** quantity.





Object Detection: How to Design Models?

R-CNN (2013.11) → **OverFeat** (ICLR' 14) → MultiBox (CVPR' 14) → SPP-Net (ECCV' 14) → MR-CNN (ICCV' 15) → DeepBox (ICCV' 15) → AttentionNet (ICCV' 15) →

Fast R-CNN (ICCV' 15) → DeepProposal (ICCV' 15) → **Faster R-CNN** (NIPS' 15) → **OHEM** (CVPR' 16) → **YOLO v1** (CVPR' 16) → G-CNN (CVPR' 16) → AZNet (CVPR' 16) →

Inside-OutsideNet(ION) (CVPR' 16) → HyperNet (CVPR' 16) → CRAFT (CVPR' 16) → MultiPathNet(MPN) (BMVC' 16) → **SSD** (ECCV' 16) → GBDNet (ECCV' 16) →

CPF (ECCV' 16) → MS-CNN (ECCV' 16) → **R-FCN** (NIPS' 16) → PVANET (NIPSW' 16) → DeepID-Net (PAMI' 16) → NoC (TPAMI' 16) → DSSD (arXiv' 17) → TDM (CVPR' 17) → **YOLO v2** (CVPR' 17) →

Feature Pyramid Net(**FPN**) (CVPR' 17) → RON (CVPR' 17) → DCN (ICCV' 17) → DeNet (ICCV' 17) → CoupleNet (ICCV' 17) → **RetinaNet** (ICCV' 17) → DSOD (ICCV' 17) →

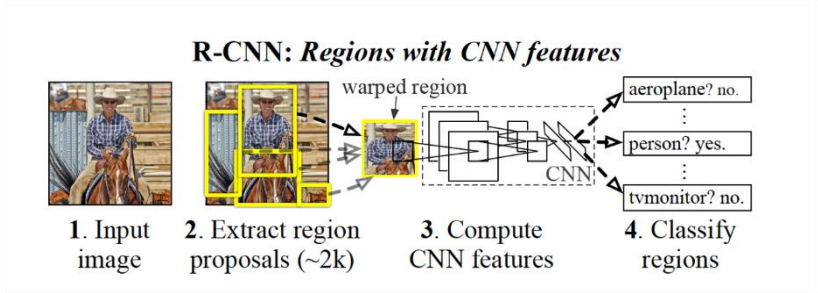
Mask R-CNN (ICCV' 17) → SMN (ICCV' 17) → **YOLO v3** (arXiv' 18) → SIN (CVPR' 18) → STDN (CVPR' 18) → **RefineDet** (CVPR' 18) → MLKP (CVPR' 18) → Relation-Net (CVPR' 18) →

Cascade R-CNN (CVPR' 18) → RFBNet (ECCV' 18) → CornerNet (ECCV' 18) → PFPNet (ECCV' 18) → Pelee (NIPS' 18) → HKRM (NIPS' 18) → R-DAD (AAAI' 19) → **M2Det** (AAAI' 19) ···

https://github.com/hoya012/deep_learning_object_detection

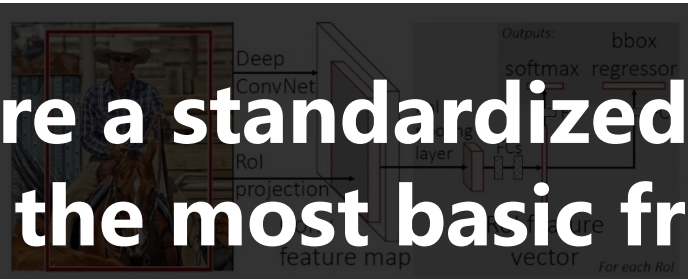
Object Detection: How to Design Models?

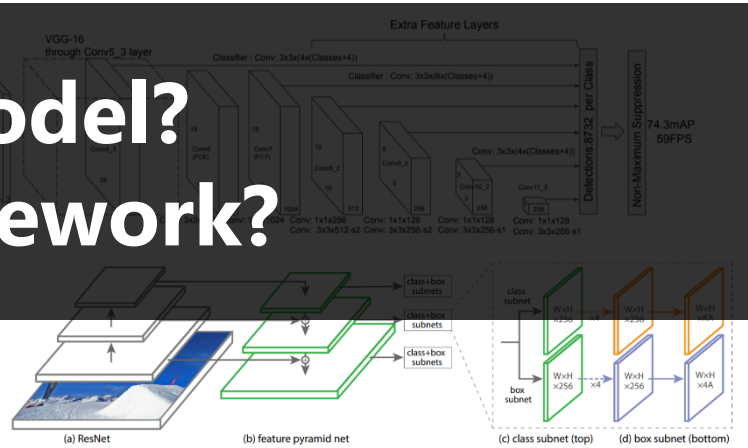
- R-CNN [Girshick et al. 2014]
- SPP-net [He et al. 2014]
- Fast R-CNN [Girshick. 2015]
- Faster R-CNN [Ren et al. 2015]
- R-FCN [Dai et al. 2016]
- YOLO [Redmon et al. 2016]
- SSD [Liu et al. 2016]
- Feature Pyramid Networks + Faster R-CNN [Lin et al. 2017]
- RetinaNet [Lin et al. 2017]
- Training with Large Minibatches (MegDet) [Peng, Xiao, Li, et al. 2017]
- Cascade R-CNN [Cai & Vasconcelos 2018]
- DETR [Carion et al. 2020]
- …

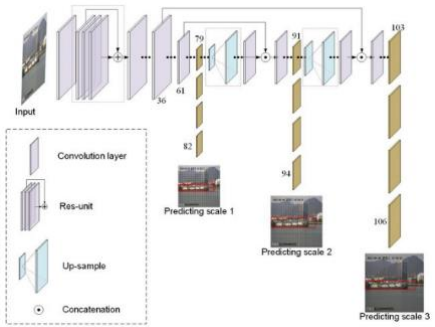


→ **Is there a standardized model?**

→ **What is the most basic framework?**







Object Detection: What is the Meta-Architecture?

Meta- means?

: High-level or symbolic abstraction.

Meta-architecture means?

: Abstract of deep neural architecture. Representative/generic structure of networks.

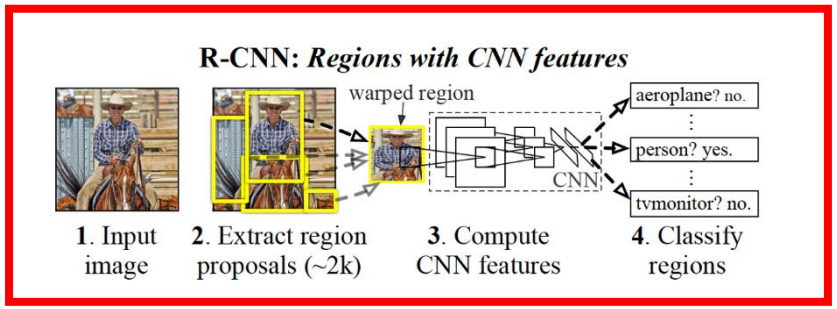
Meta-architecture for object detection?

→ Two-stage detector

→ One-stage detector

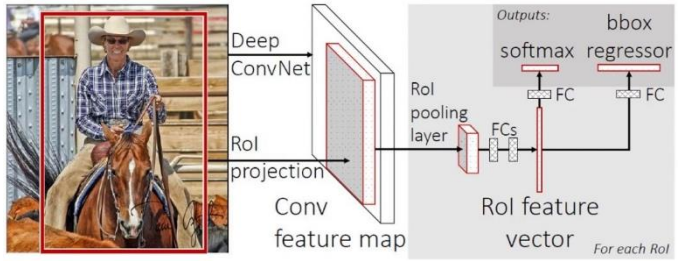
Object Detection: Two-Stage vs. One-Stage

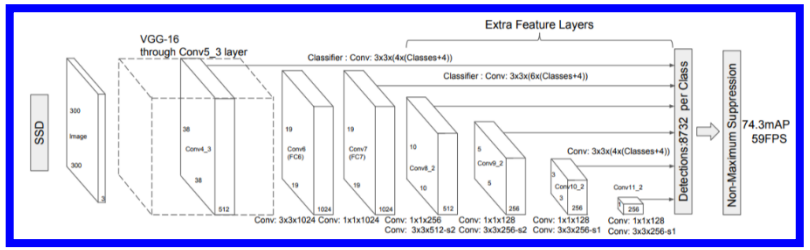
- **R-CNN [Girshick et al. 2014]**
- SPP-net [He et al. 2014]
- Fast R-CNN [Girshick. 2015]
- Faster R-CNN [Ren et al. 2015]
- R-FCN [Dai et al. 2016]
- YOLO [Redmon et al. 2016]
- **SSD [Liu et al. 2016]**
- Feature Pyramid Networks + Faster R-CNN [Lin et al. 2017]
- RetinaNet [Lin et al. 2017]
- Training with Large Minibatches (MegDet) [Peng, Xiao, Li, et al. 2017]
- Cascade R-CNN [Cai & Vasconcelos 2018]
- DETR [Carion et al. 2020]
- …

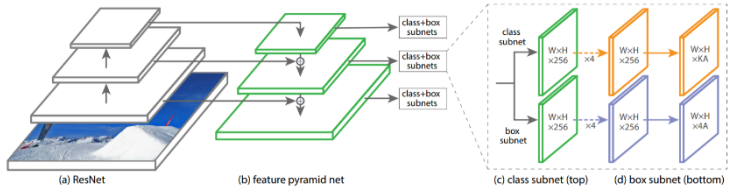


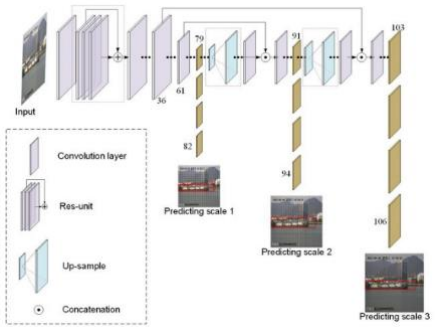
(1) Two-stage detector

(2) One-stage detector









Two-Stage Detector: R-CNN Series

References

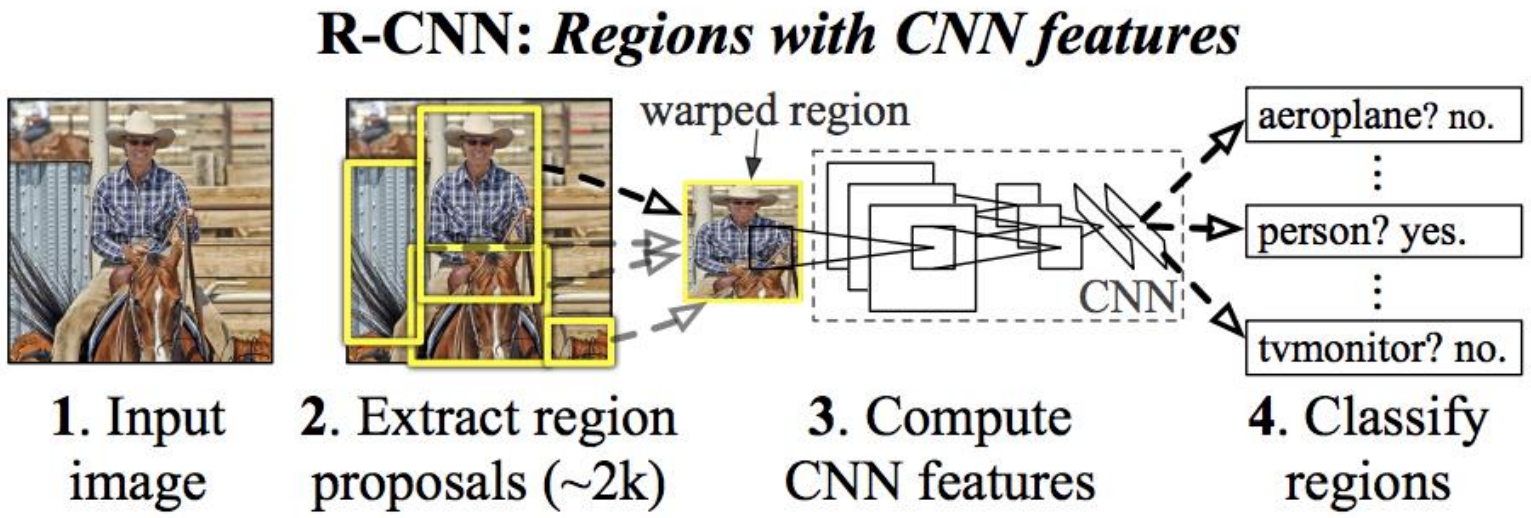
- B. Alexe, T. Deselaers, and V. Ferrari. "**Measuring the objectness of image windows.**" *T-PAMI*, 2012.
- I. Endres and D. Hoiem. "**Category independent object proposals.**" *ECCV*, 2010.
- J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. "**Selective search for object recognition.**" *IJCV*, 2013.
- A. Krizhevsky, I. Sutskever, and G. Hinton. "**ImageNet classification with deep convolutional neural networks.**" *NIPS*, 2012.
- X. Wang, M. Yang, S. Zhu, and Y. Lin. "**Regionlets for generic object detection.**" *ICCV*, 2013.
- R. Girshick, J. Donahue, T. Darrell, and J. Malik. "**Rich feature hierarchies for accurate object detection and semantic segmentation.**" *CVPR*, 2014.

R-CNN: Region-Based Convolutional Neural Networks

Key idea

→ Exploit strong **CNN representation** for **accurate** region classification

Hybrid approach: Traditional region proposal + CNN feature extraction



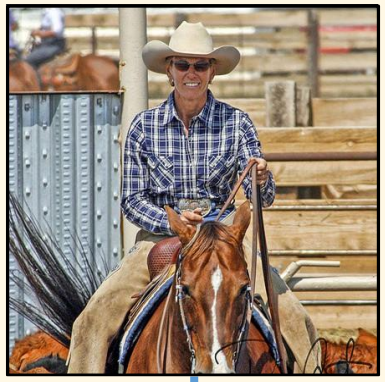
[1] R. Girshick, et al., "Rich feature hierarchies for accurate object detection and semantic segmentation." *CVPR* (2014).

R-CNN: Region-Based Convolutional Neural Networks

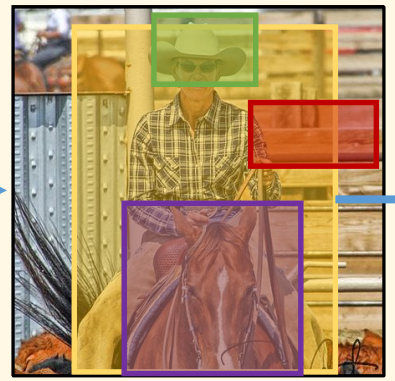
Per-image computation

Per-region computation for each $r_i \in r(I)$

I:



Selective search



Crop & warp

ConvNet(r_i)

1-vs-rest SVMs

Box regressor

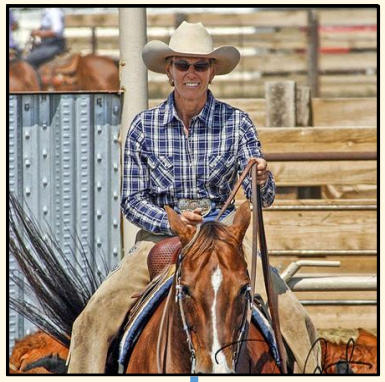
Figure credit: Ross Girshick

R-CNN: Region-Based Convolutional Neural Networks

Per-image computation

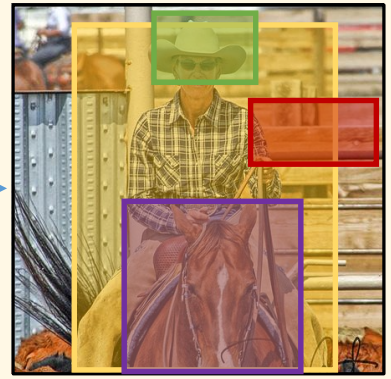
Per-region computation for each $r_i \in r(I)$

I:



Selective search

1



Use an off-the-shelf region/object/detection proposal algorithm (~2k proposals per image)

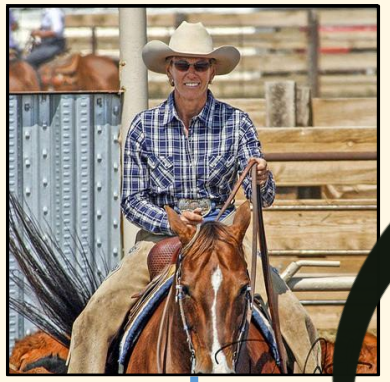
Figure credit: Ross Girshick

R-CNN: Region-Based Convolutional Neural Networks

Per-image computation

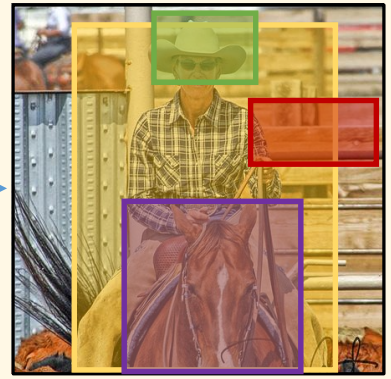
Per-region computation for each $r_i \in r(I)$

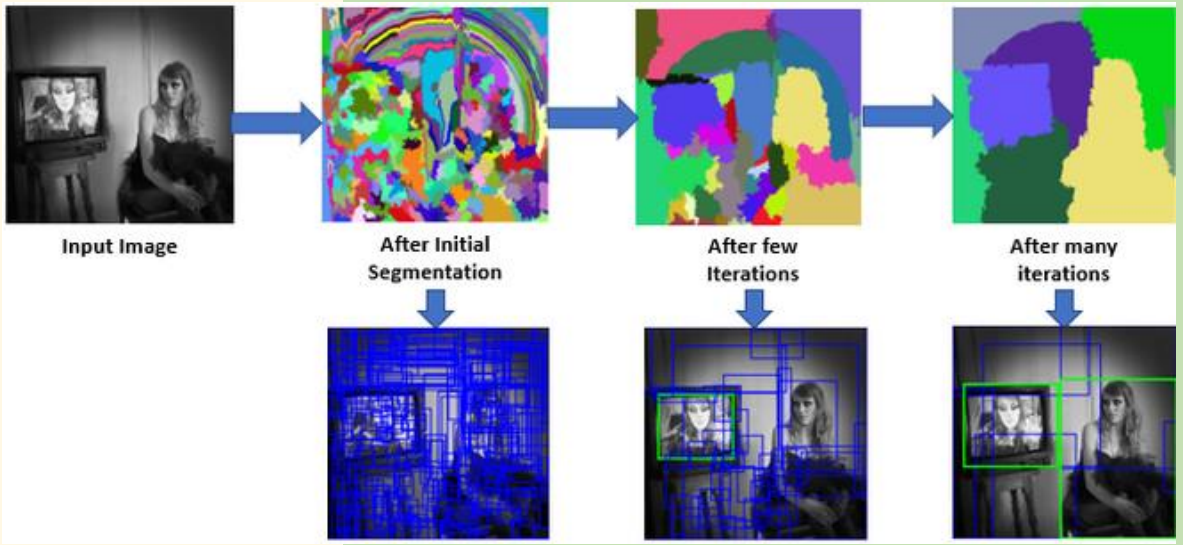
I:



Selective search

1





Selective search?
→ Region proposal based on different **similarity** metrics. (e.g., color, texture, size, and shape)

Region-of-Interest (RoI)

Use an off-the-shelf region/object/detection proposal algorithm (~2k proposals per image)

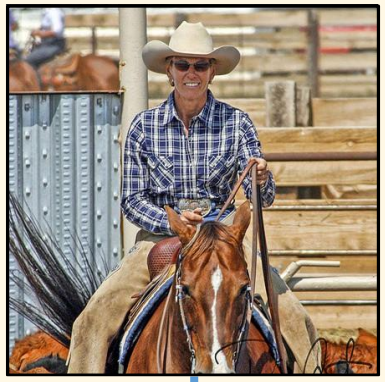
Figure credit: Ross Girshick

R-CNN: Region-Based Convolutional Neural Networks

Per-image computation

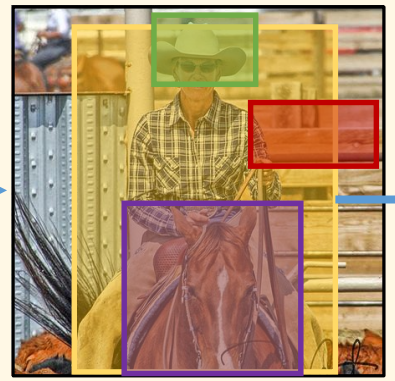
Per-region computation for each $r_i \in r(I)$

I:



Selective search

1



Crop & warp

2

Crop and warp each proposal image window to obtain a fixed-size network input

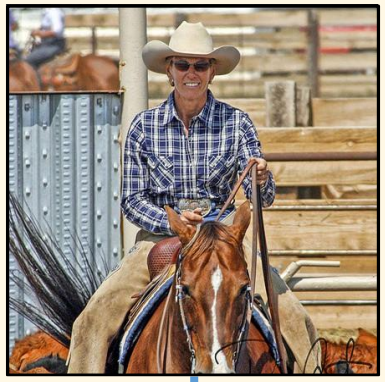
Figure credit: Ross Girshick

R-CNN: Region-Based Convolutional Neural Networks

Per-image computation

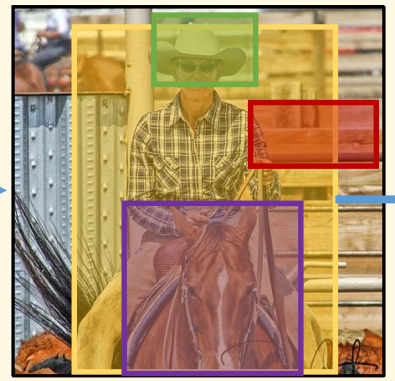
Per-region computation for each $r_i \in r(I)$

I:



Selective search

1



Crop & warp

2

ConvNet(r_i)

3

Forward propagate the fixed-size network input to get a feature representation

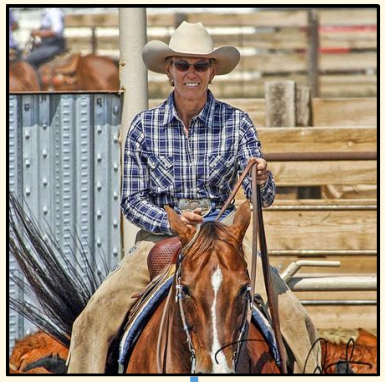
Figure credit: Ross Girshick

R-CNN: Region-Based Convolutional Neural Networks

Per-image computation

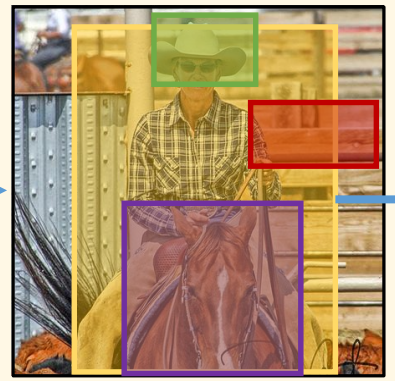
Per-region computation for each $r_i \in r(I)$

I:



Selective search

1



Crop & warp

2

ConvNet(r_i)

3

Classification

4

Object classification

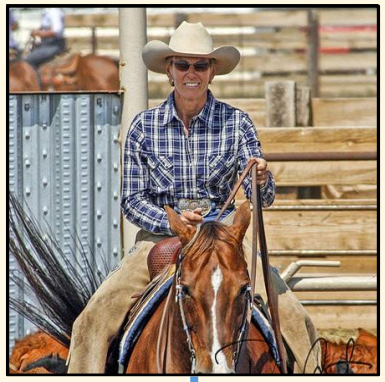
Figure credit: Ross Girshick

R-CNN: Region-Based Convolutional Neural Networks

Per-image computation

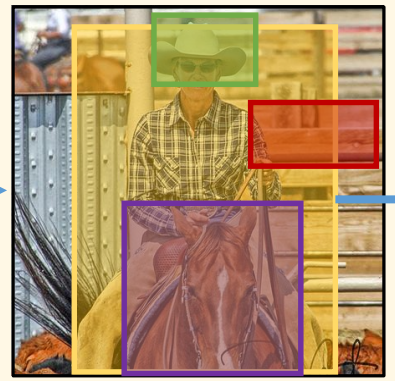
Per-region computation for each $r_i \in r(I)$

I:



Selective search

1



Crop & warp

2

ConvNet(r_i)

3

Classification

4

Box regressor

5

Refine proposal localization with bounding-box regression

Figure credit: Ross Girshick

The Problem with R-CNN: "Slow"

Per-image computation

Per-region computation for each $r_i \in r(I)$

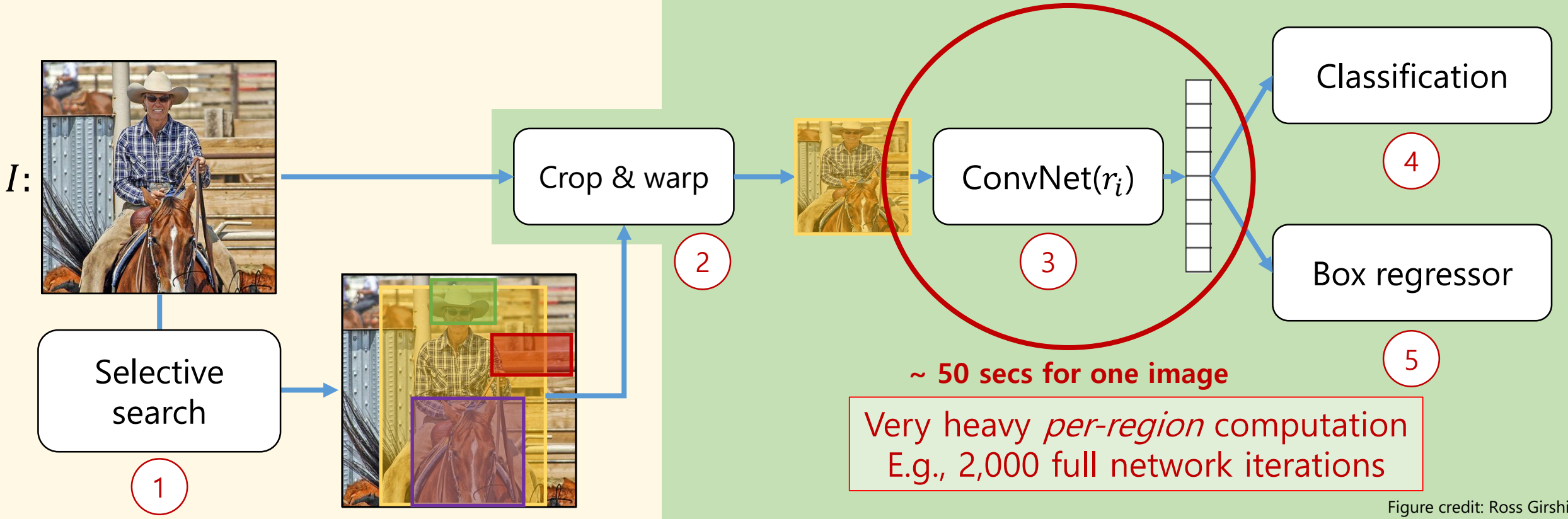


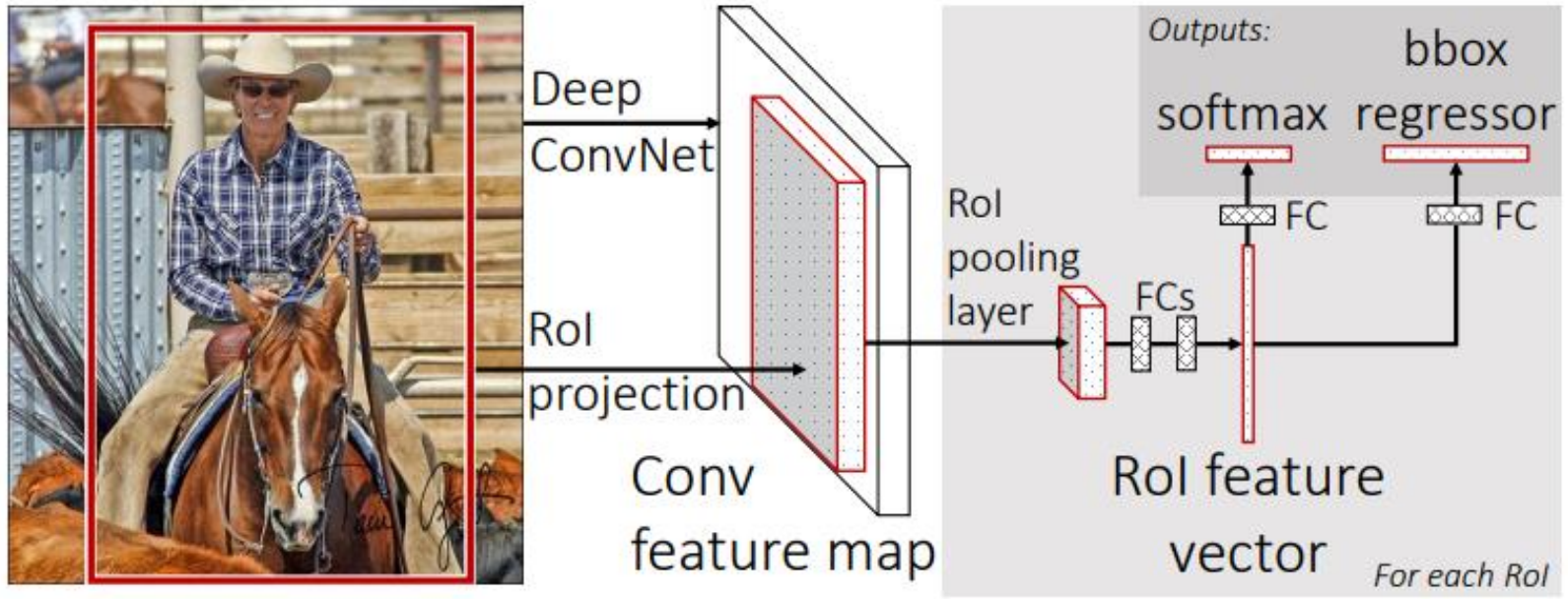
Figure credit: Ross Girshick

Solution: Fast R-CNN

Key idea

→ **Sharing** heavy per-region computation for **fast** inference (2 *fps*, x100 times to the R-CNN)

Region proposal + CNN feature extraction for the entire input image in the first stage



[1] R. Girshick, et al., "Fast R-CNN." *ICCV* (2015).

One-Stage Detectors

Key idea

→ Object detection from **single feedforward** of CNN

→ Eliminate proposal generation & per-region feature resampling and classification

→ Extremely fast (40~90 FPS with single GPU)

References

- YOLO: You Only Look Once [Redmon et al., CVPR'16]
- YOLO-v2 [Redmon et al., CVPR'17]
- YOLO-v3 [Redmon et al., arXiv'18]
- **SSD [Liu et al., ECCV'16]**
- RetinaNet [Lin et al., ICCV'17]

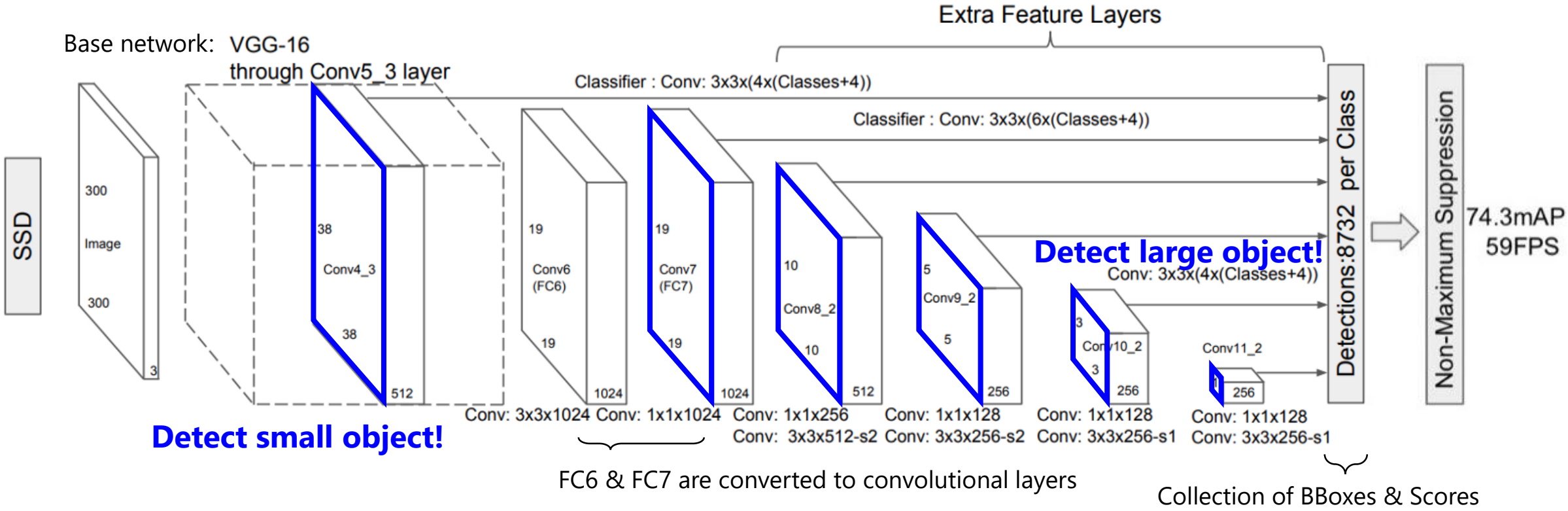
SSD: Single Shot Multibox Detector

Key idea

→ Fully convolutional layers to extract **spatial** features.

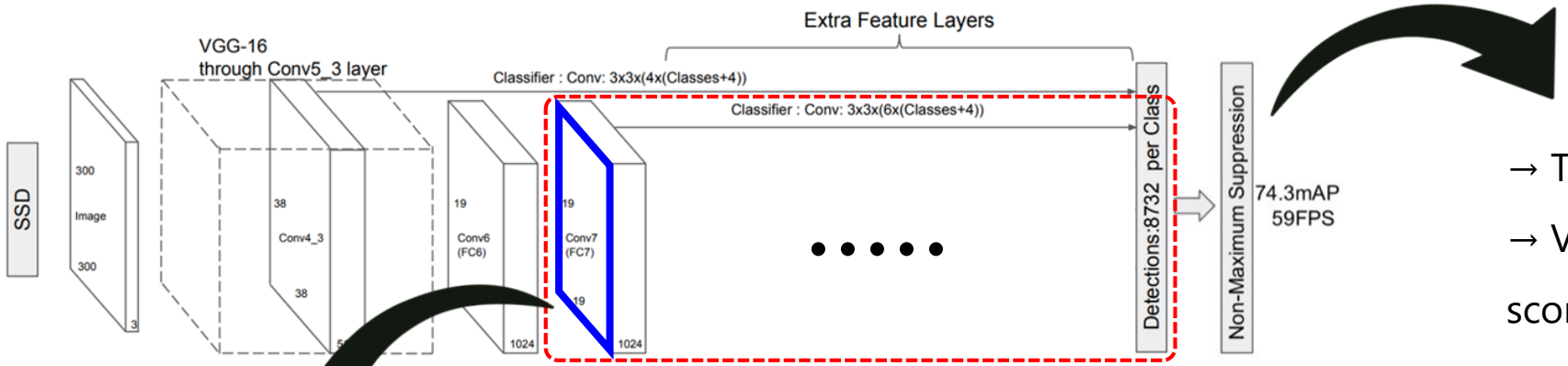
→ Aggregates **multi-scale** spatial features.

→ Meta-architecture for one-stage detector.



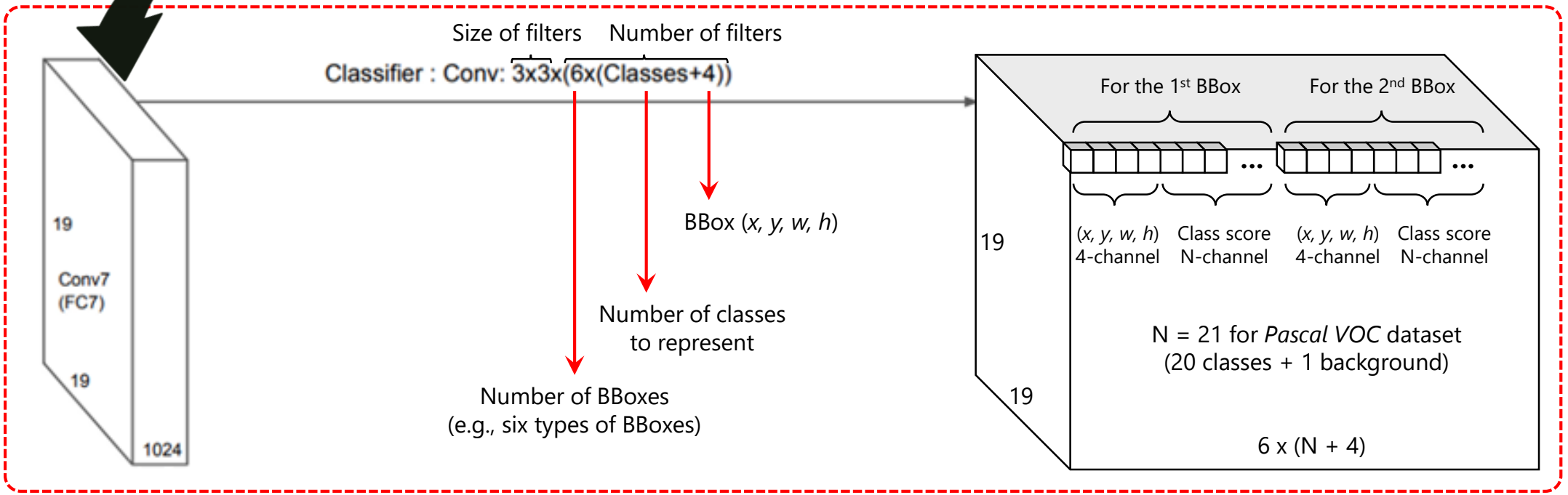
[1] Liu, et al., "Ssd: Single shot multibox detector." *ECCV* (2016).

SSD: Single Shot Multibox Detector



→ Total 8,732 BBoxes for each class

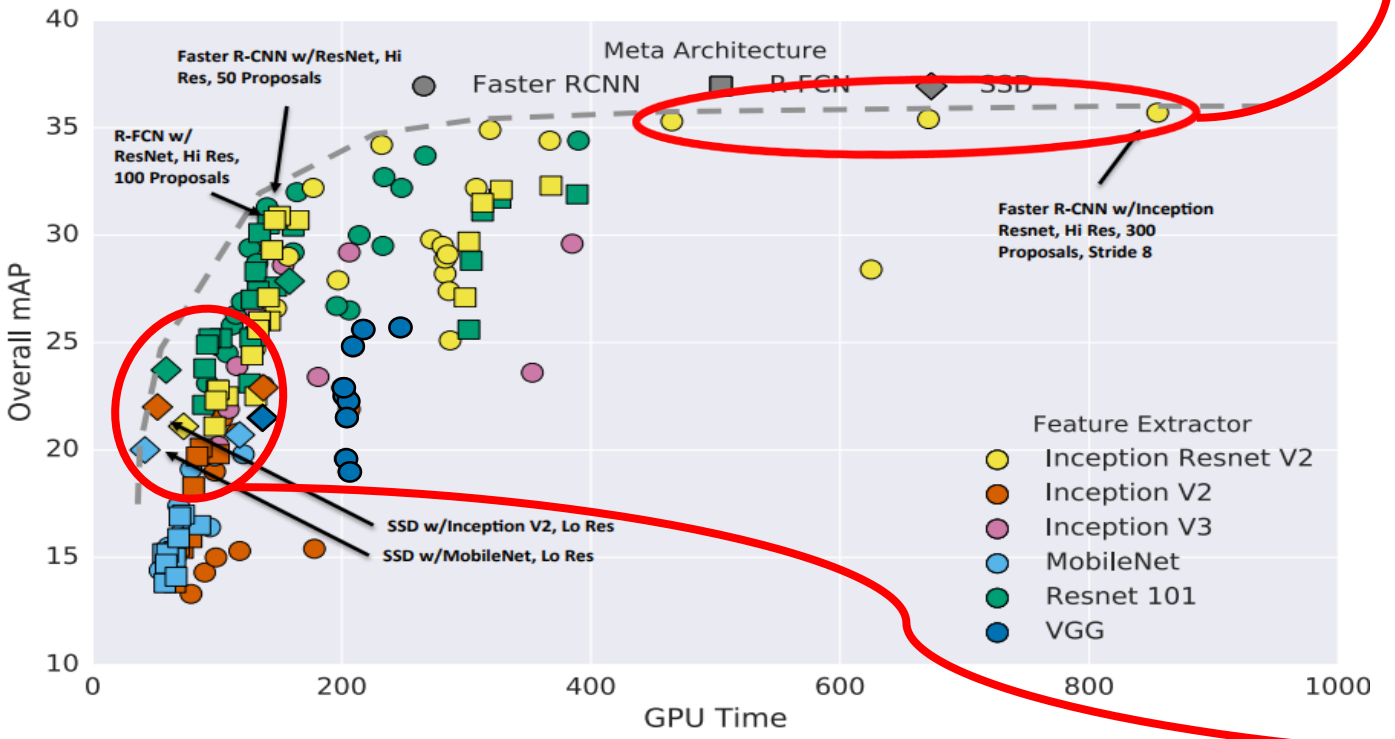
→ Visualize the BBoxes with the class score over 0.5 (threshold)

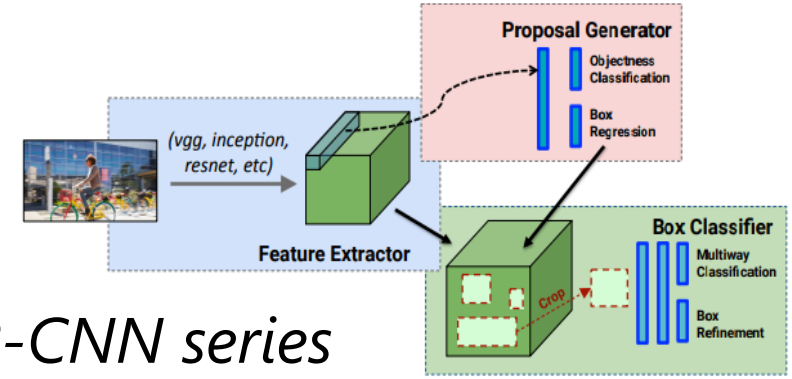


[1] Liu, et al., "Ssd: Single shot multibox detector." *ECCV* (2016).

Summary: What is the Best Model?

Speed & accuracy trade-offs





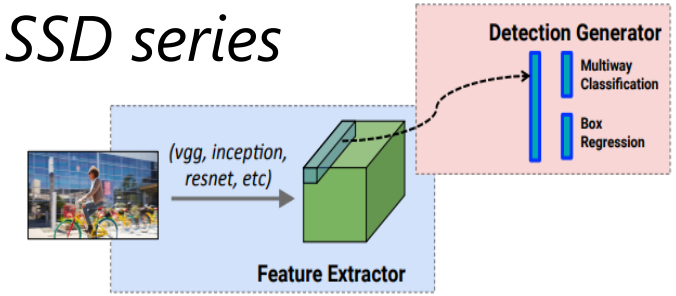
R-CNN series

Two-stage detectors

- Complex & Slow (~5FPS)
- More accurate

---------- vs. ----------

SSD series



One-stage detectors

- Simple & Fast (~55FPS)
- Less accurate

→ A lot more creative and high-performance architectures have been released so far, and they resemble **human** cognitive processes!

[1] Speed/accuracy trade-offs for modern convolutional object detectors. Huang et al.

KENTECH
Korea Institute of Energy Technology

Week 08a – Object Detection on Jetson Nano

Experiments

Before starting

*Your basic workspace is here: “cd ~/jetson-inference/build/aarch64/bin” Every code is pre-built in this path.

Live object detection with visualization

Q1. Run “python detectnet.py --flip-method=rotate-180”. What is on the screen? What is different from “imagenet.py” in the previous class?

Try using options for object detection

Q2.1. Run “python detectnet.py --overlay=line,labels,conf --flip-method=rotate-180”. What is different from the screen in Q1?

Q2.2. Control alpha (default alpha=120 e.g., 50 & 100) by running “python detectnet.py --alpha=50 --flip-method=rotate-180”. What does alpha mean?

Q2.3. Control threshold (default threshold =0.5 e.g., 0.3 & 0.7) by runing “python detectnet.py --threshold=0.3 --flip-method=rotate-180”. What does threshold mean?

Experiments

Try other object detection models

Q3.1. Detect objects and students. Observe the label and confidence.

Q3.2. Go to the linked page (https://github.com/dusty-nv/jetson-inference/blob/master/docs/detectnet-console-2.md#pre-trained-detection-models-available). Check the list of models. COCO dataset contains 91 classes as defined in this page (https://github.com/dusty-nv/jetson-inference/blob/master/data/networks/ssd_coco_labels.txt).

Q3.3. (optional) Check your installed model via “cd ~/jetson-inference/build/aarch64/bin/networks” “ls” You can see the models installed.

Q3.4. Deploy other detection networks via “python detectnet.py --network=ssd-inception-v2 --flip-method=rotate-180”. Try multiple different models and compare the label and confidence with the same object and angle. Which model is best?

Q3.5. Use facenet and find the maximum threshold to recognize your masked face.
Run “python detectnet.py --network=facenet --threshold=0.7 --flip-method=rotate-180”

Experiments

Try to detect objects on your own images

Q4.1. Find your preferable photo (yourselves, or your pet, or your favorite place, food,.., etc.) or take your photo instantly. Move the photo using browser to the ~/jetson-inference/build/aarch64/bin/images folder.

Q4.2. Try the models you are interested in. Run “python detectnet.py --network=facenet images/photo.jpg images/test/output_photo.jpg”

Q4.3. Observe the results which are saved in ~/jetson-inference/build/aarch64/bin/images/test folder.

Experiments

Some useful tips while debugging

*Sometimes, the python process does not respond. In this case, please terminate the process with `ctrl+c`. If it still does not respond at all, forcibly stop the process with `ctrl+z`, and check the running process name with the `ps -a` command, and then type `sudo pkill -9 [name-of-process]` command to kill the process. If you don't shut it down, it will remain as a 🙍zombie🙍 and keep occupying the processor (CPU or GPU) in the background.

*Sometimes, the best solution for resolving an issue is just rebooting the system.

"cd" = "change directory"
"ls" = "list segment (files & directories)"

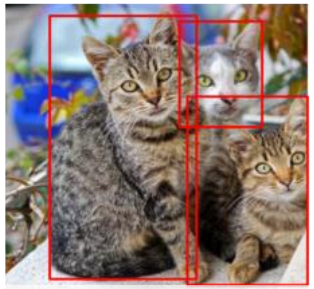
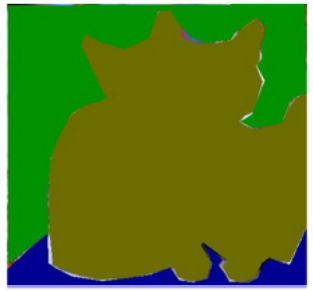
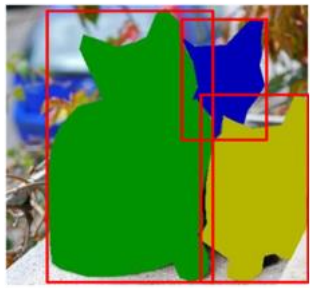
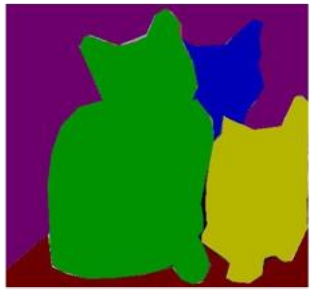
Computer Vision Tasks

*Slide by Kim, et al., "Video Panoptic Segmentation" (CVPR 2020)

Model Complexity ⇧ Output dimension ⇧

Image

Video

Image Classification	Object Detection	Semantic Segmentation	Instance Segmentation	Panoptic Segmentation
				
"Image-level"	"Box-level"	"Pixel-level"	"Pixel-level" + "Instance-level"	"Unknown pixel"
	Fast RCNN, CVPR 2015 **Faster RCNN** NeurIPS 2015 **YOLO,** CVPR 2016 **SSD,** ECCV 2016 **Cascade RCNN,** CVPR 2017 **CornerNet,** ECCV 2018 **HTC,** CVPR 2019	Dilated FCN, ICLR 2015 **DeepLab,** ICLR 2015 **PSPNet,** CVPR 2016 **Deform FCN,** ICCV 2017 **EncNet,** CVPR 2018 **DeepLabv2,** ECCV 2018 **DANet,** CVPR 2019 **Deformable V2,** ICCV 2019	DeepMask, NeurIPS 2015 MNC, CVPR 2016 FCIS, CVPR 2017 Cascade Mask-RCNN, PAMI 2019 **HTC,** CVPR2019 ⋮	
Karpathy et.al., CVPR 2014	**Kai et.al.,** CVPR 2016	**Xizhou et.al.,** CVPR 2017	**Linjie et.al.,** ICCV 2019	**Kim et.al.,** CVPR 2020
Video Classification	Video Object Detection	Video Semantic Segmentation	Video Instance Segmentation	Video Panoptic Segmentation