視覺辨識課程

謝坤達

JUMBOKH@GMAIL.COM

0953313123

大綱

1. 視覺辨識介紹
2. 影像標示
3. 在雲端上實作深度學習物件辨識模型
4. 簡易 YOLO3 系統辨識
5. 臉部辨識實作

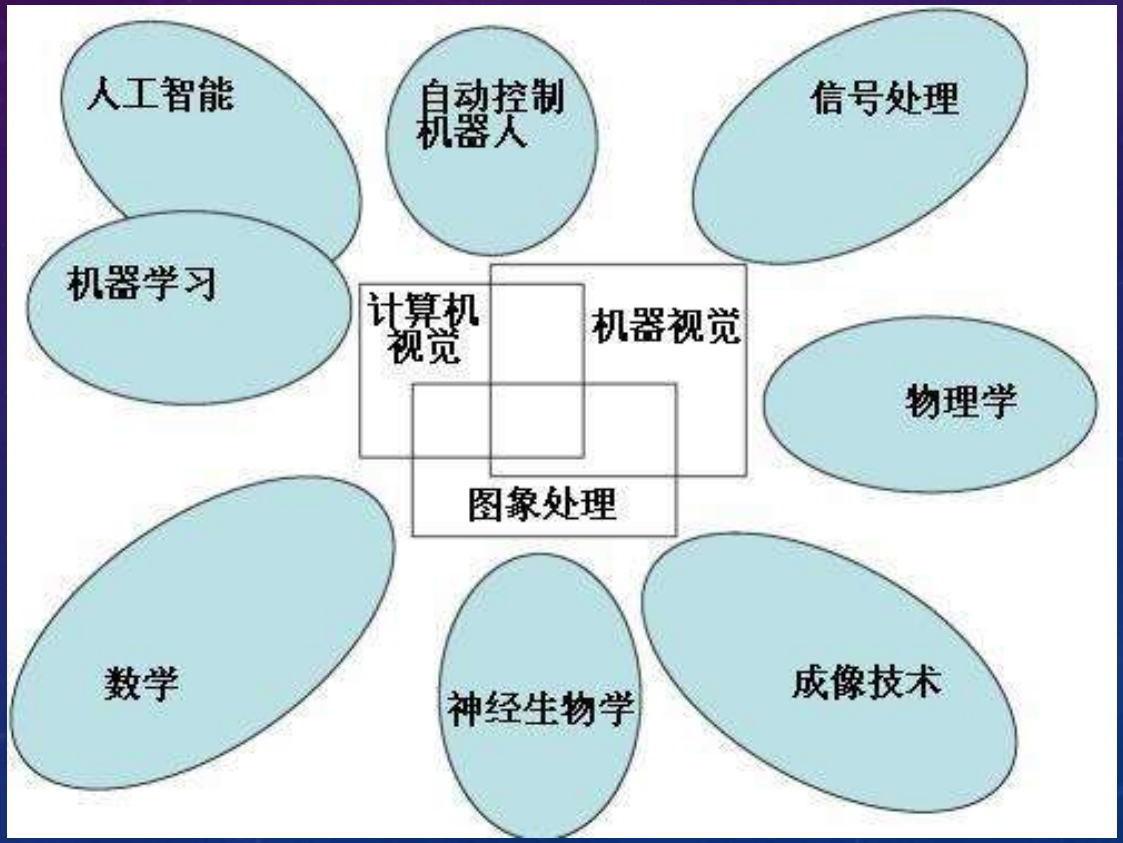
在雲端上實作深度學習物件辨識模型

- 何謂物件辨識
- 物件辨識演算法
- 訓練資料準備
- 實作

電腦視覺（COMPUTER VISION）

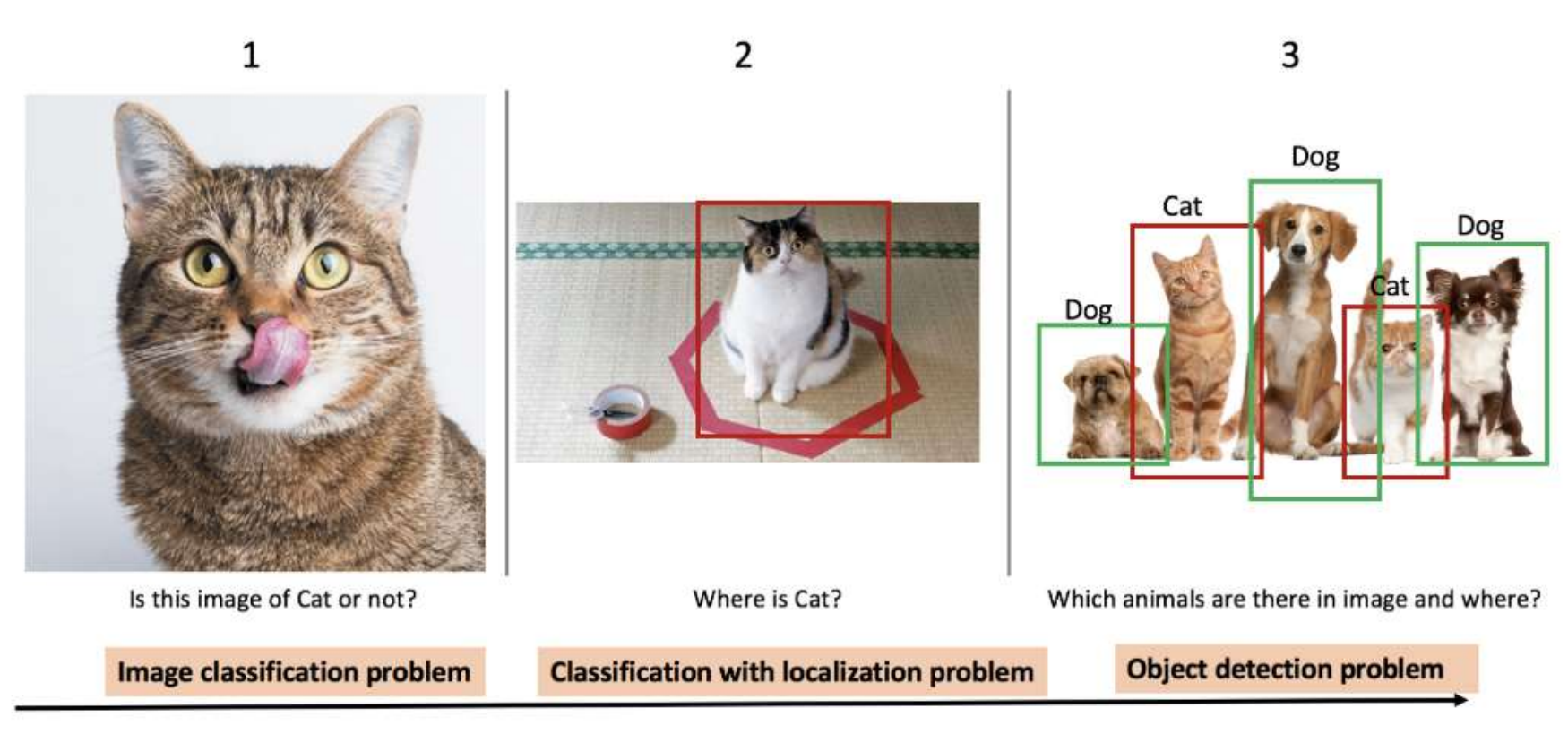
- 電腦視覺（Computer vision）是一門研究如何使機器「看」的科學，更進一步的說，就是指用攝影機和電腦代替人眼對目標進行辨識、跟蹤和測量等機器視覺，並進一步做圖像處理，用電腦處理成為更適合人眼觀察或傳送給儀器檢測的圖像

電腦視覺與其他領域的關係(維基百科)



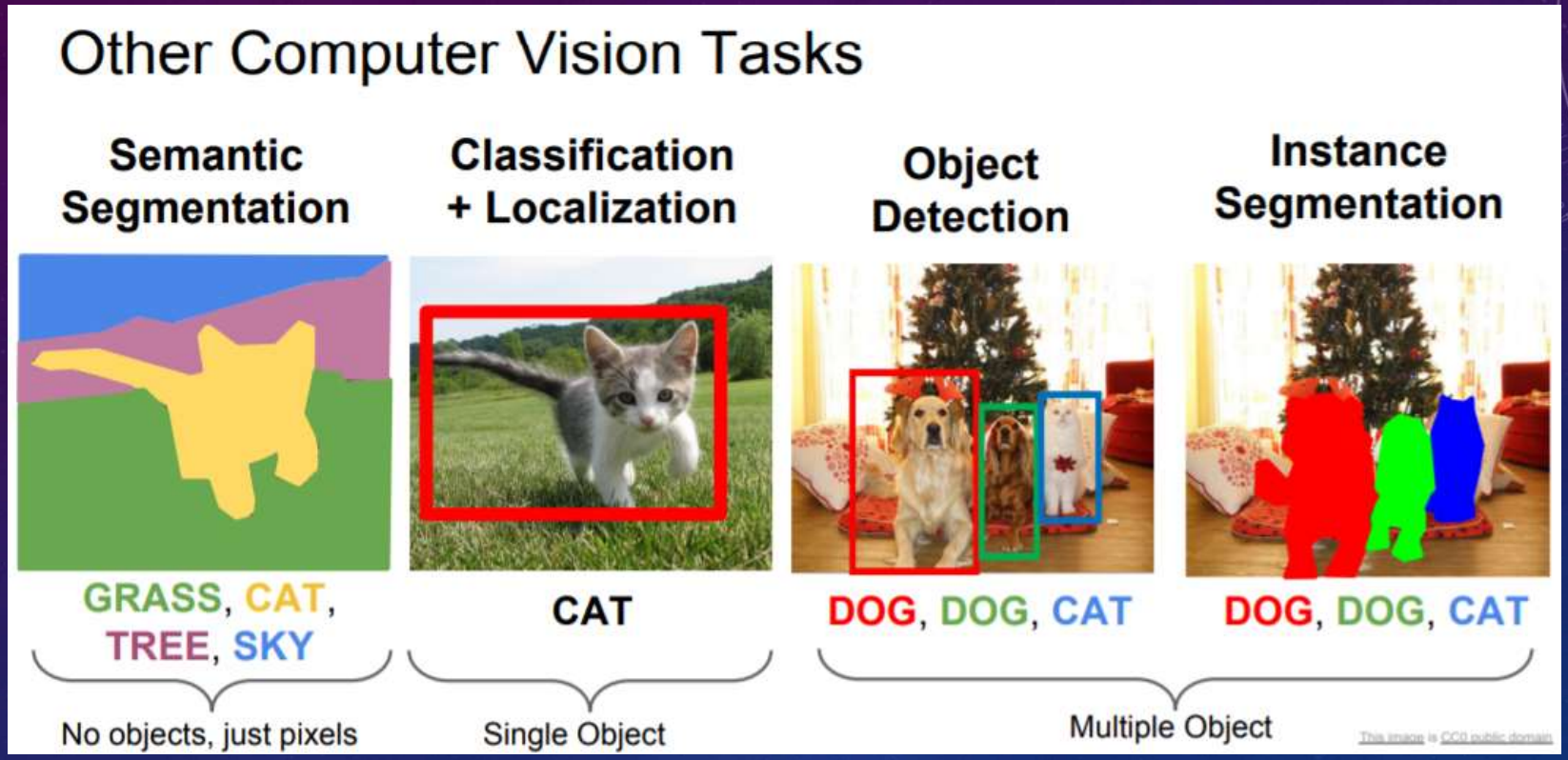
WHAT IS OBJECT DETECTION?

Object dection compares the image classification and localizations



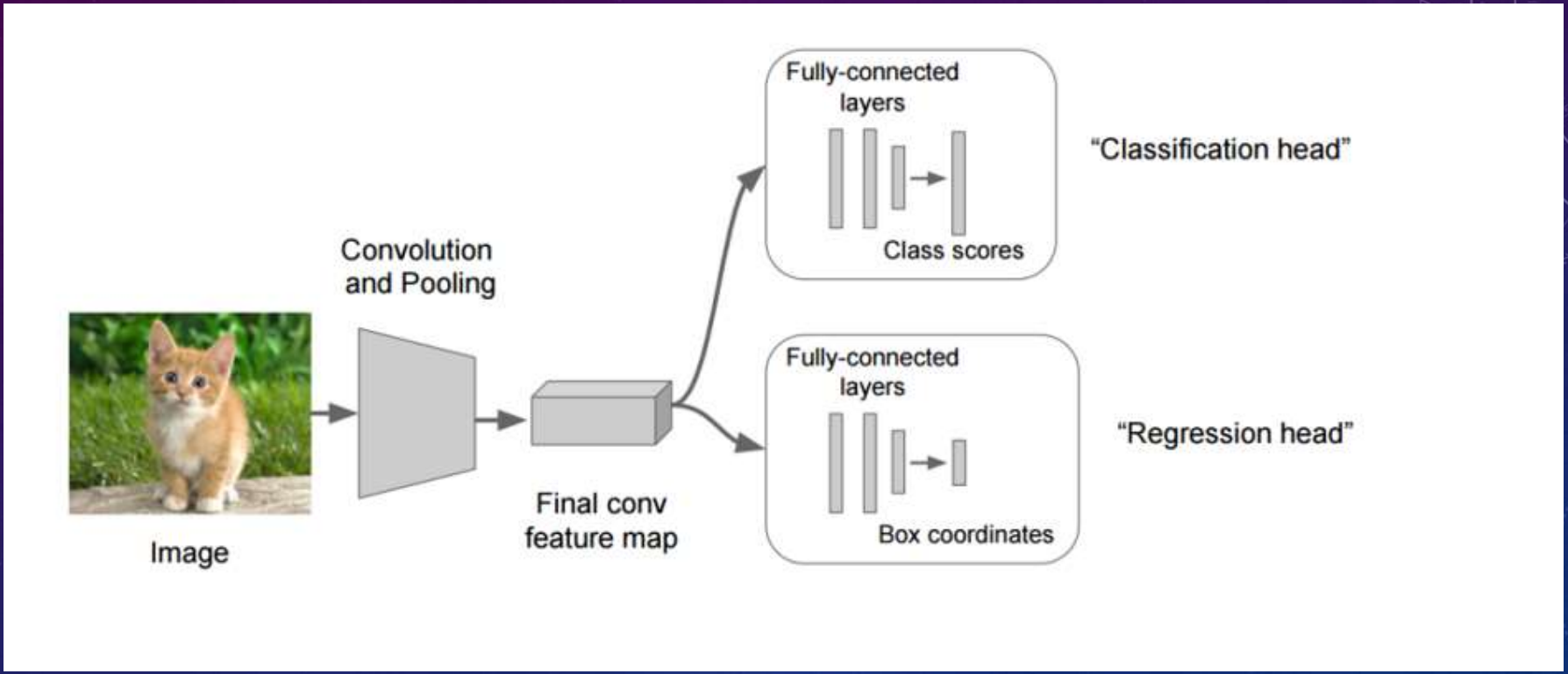
https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad

WHAT IS OBJECT DECTION?



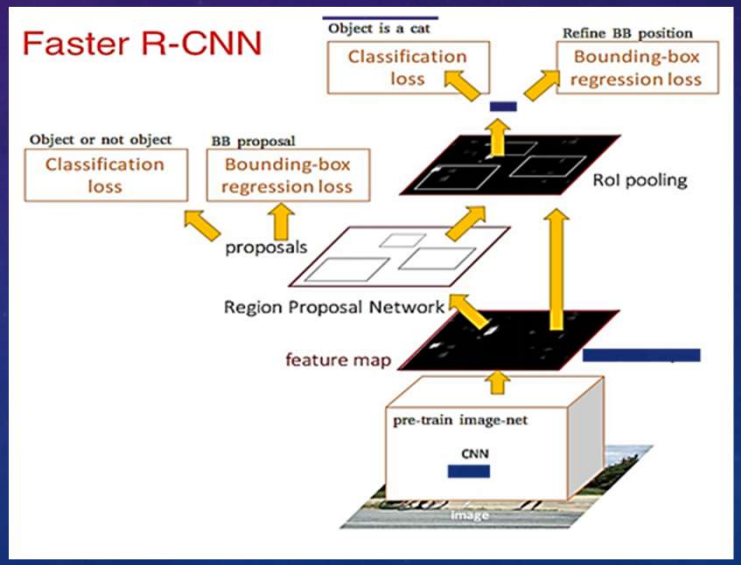
http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture11.pdf

OBJECT DECTION ALGORITHMS



OBJECT DECTION ALGORITHMS(CONT.)

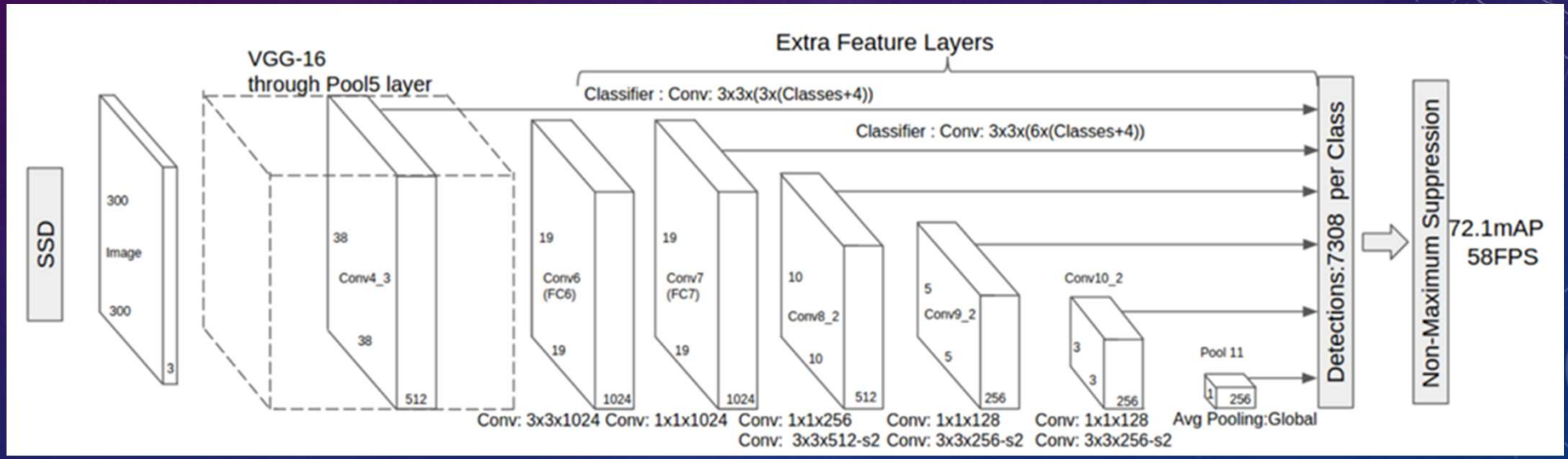
- R-CNN(Region with CNN) (See: https://arxiv.org/abs/1311.2524)
- Fast RCNN(See:https://arxiv.org/abs/1504.08083)
- Faster RCNN(See: https://arxiv.org/abs/1506.01497)
- R-FCN()
- SSD
- YOLOv1 to YOLOv3
- ReinaNet
- FPN



OBJECT DECTION ALGORITHMS(CONT.)

- **(參考: a.** 關於影像辨識，所有你應該知道的深度學習模型’
  - **b. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms)**
- Deep Learning for Object Detection: A Comprehensive Review https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9

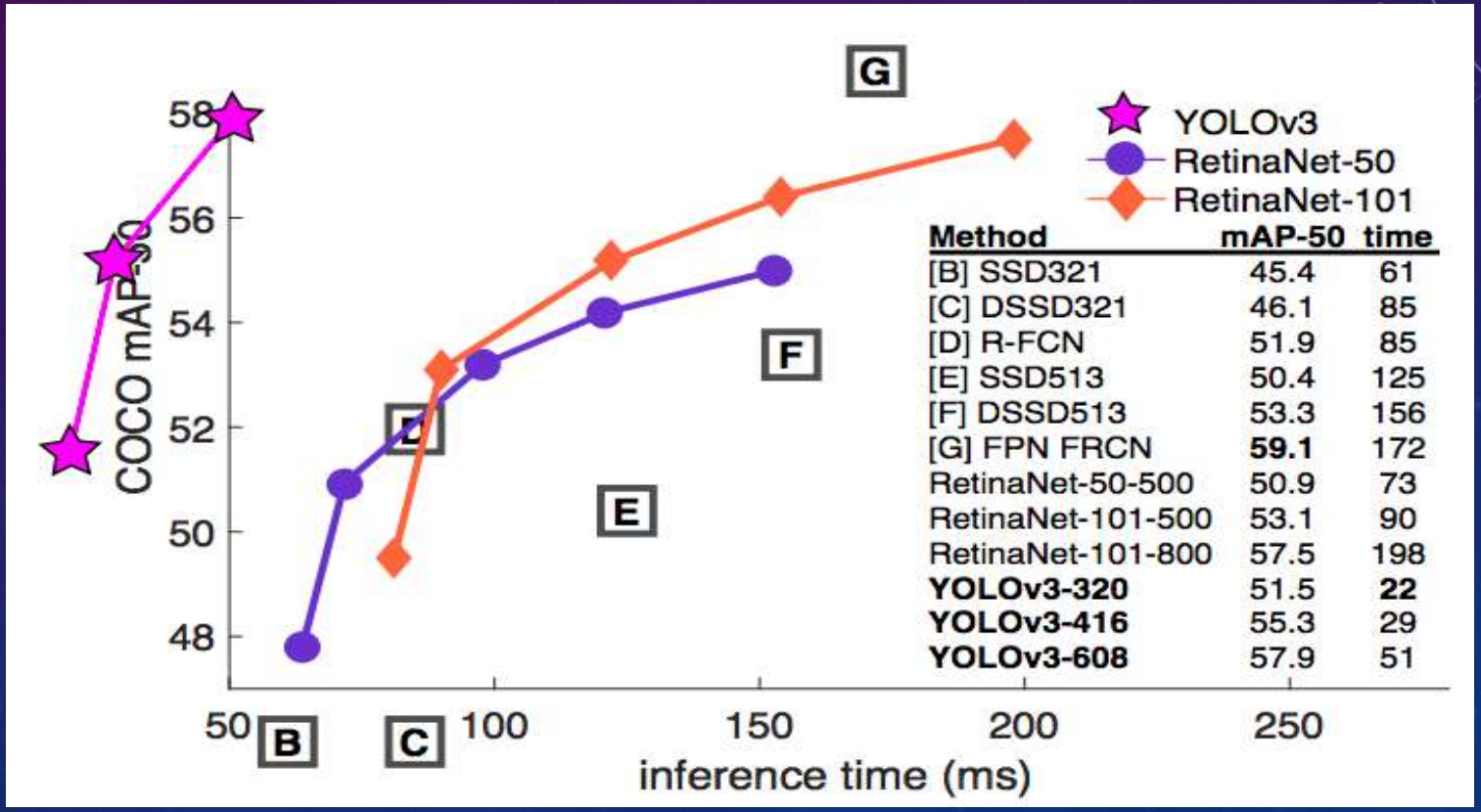
OBJECT DECTION ALGORITHMS(CONT.)



OBJECT DECTION ALGORITHMS(CONT.)

- one stage and two stages
  - (深度學習-什麼是one stage，什麼是two stage 物件偵測
  - Optimizing the Trade-off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction PDF)
- Comparison
- YOLOv3 is good as mAP and time
  - ✓What's new in YOLO v3?
- https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9
  (https://pjreddie.com/media/files/papers/YOLOv3.pdf

YOLO VS RETINANET PERFORMANCE ON COCO 50 BENCHMARK



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

ACCURACY AND SPEED TRADEOFF ON VOC 2007 (SOURCE: YOLOV2 PAPER)

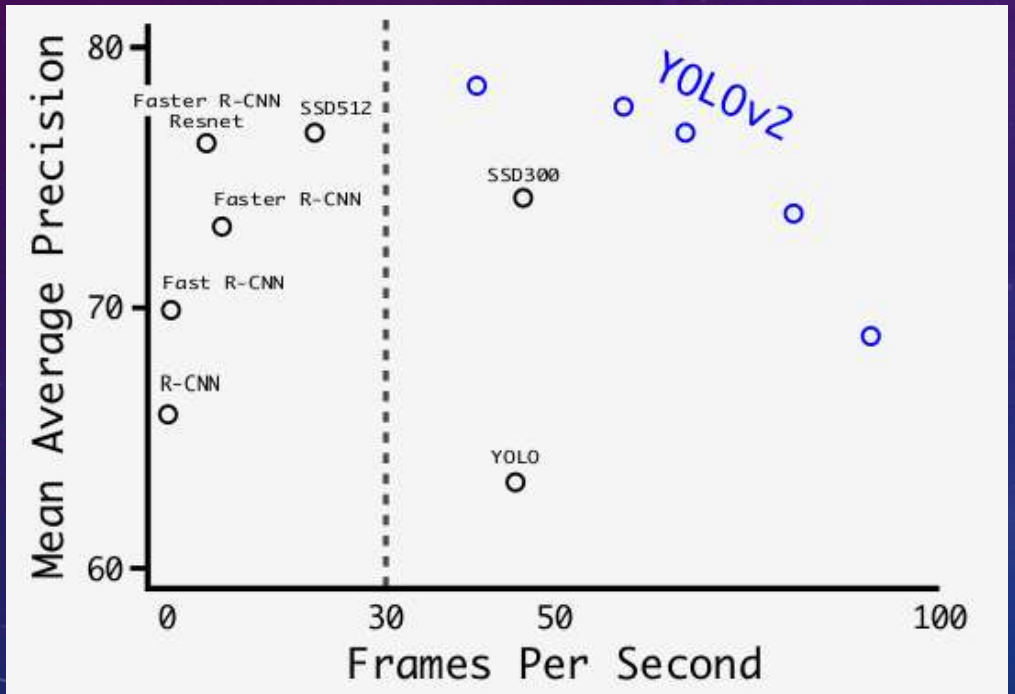
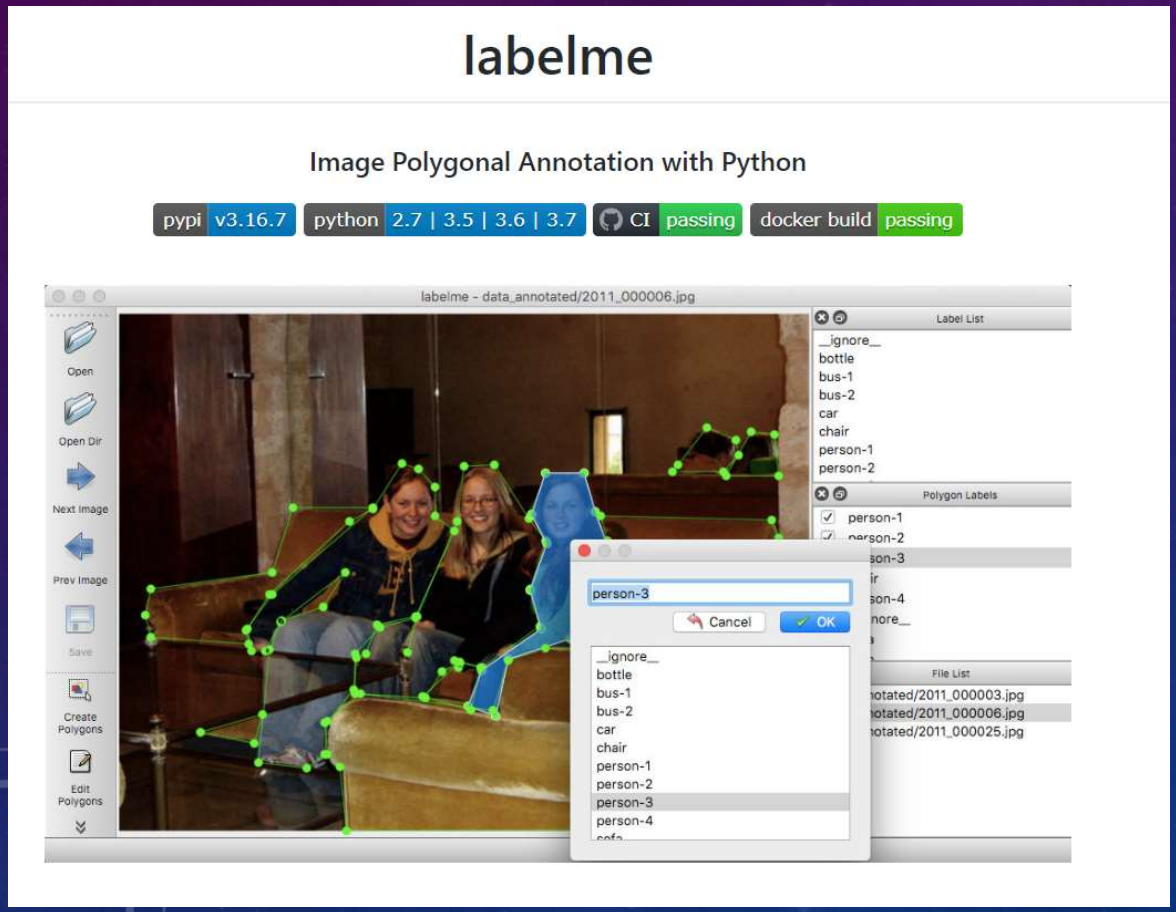


Figure 4: Accuracy and speed on VOC 2007.

- ✓ YOLO3: A Huge Improvement
  https://mc.ai/yolo3-a-huge-improvement/

訓練資料準備

https://github.com/wkentaro/labelme



https://tzutalin.github.io/labelImg/

- # python2
- conda create --name=labelme python=2.7
- source activate labelme
- # conda install -c conda-forge pyside2
- conda install pyqt
- pip install labelme
- # if you'd like to use the latest version. run below:
- # pip install git+https://github.com/wkentaro/labelme.git

- # python3
- conda create --name=labelme python=3.6
- source activate labelme
- # conda install -c conda-forge pyside2
- # conda install pyqt
- # pip install pyqt5 # pyqt5 can be installed via pip on python3
- pip install labelme
- # or you can install everything by conda command
- # conda install labelme -c conda-forge

其他安装文章

- https://www.itread01.com/content/1544810780.html

實作：訓練 XML標示

實作程式碼

- **Colab example**: tensorflow-object-detection-training-colab.ipynb https://colab.research.google.com/github/Tony607/object_detection_demo/blob/master/tensorflow_object_detection_training_colab.ipynb?hl=en
- Make a copy and test!

IMPLEMENTATIONS: TRAINING

```
from imageai.Detection.Custom import DetectionModelTrainer
trainer = DetectionModelTrainer()
trainer.setModelTypeAsYOLOv3()
trainer.setDataDirectory(data_directory="hololens")
trainer.setTrainConfig(object_names_array=["hololens"],
batch_size=8, num_experiments=5,
train_from_pretrained_model="pretrained-yolov3.h5")
trainer.setTrainConfig(object_names_array=["hololens"],
batch_size=8, num_experiments=5)
trainer.trainModel()
```

MODEL EVALUATION
IT SHOWS THE MAP OF OUR TRAINED MODELS.

```
from imageai.Detection.Custom import DetectionModelTrainer


trainer = DetectionModelTrainer()
trainer.setModelTypeAsYOLOv3()
trainer.setDataDirectory(data_directory="hololens")
trainer.evaluateModel(model_path="hololens/models",
json_path="hololens/json/detection_config.json", iou_threshold=0.5,
object_threshold=0.3, nms_threshold=0.5)
```

實作

- If I have many items?
- Object_names_array=[“Peanut”, “Hazelnut”]

EXTRAS

- ✓Run SSD, Faster RCNN and FCN
- ✓https://medium.com/swlh/how-to-train-an-object-detection-model-easy-for-free-f388ff3663e
- ✓(Or https://www.dlology.com/blog/how-to-train-an-object-detection-model-easy-for-free/)

結論

- The impact of the object detection
- YOLO3 is agreat framework so far
- Implementation is quite easy now
- Labeling job is quite laborious, we are working on some tricks

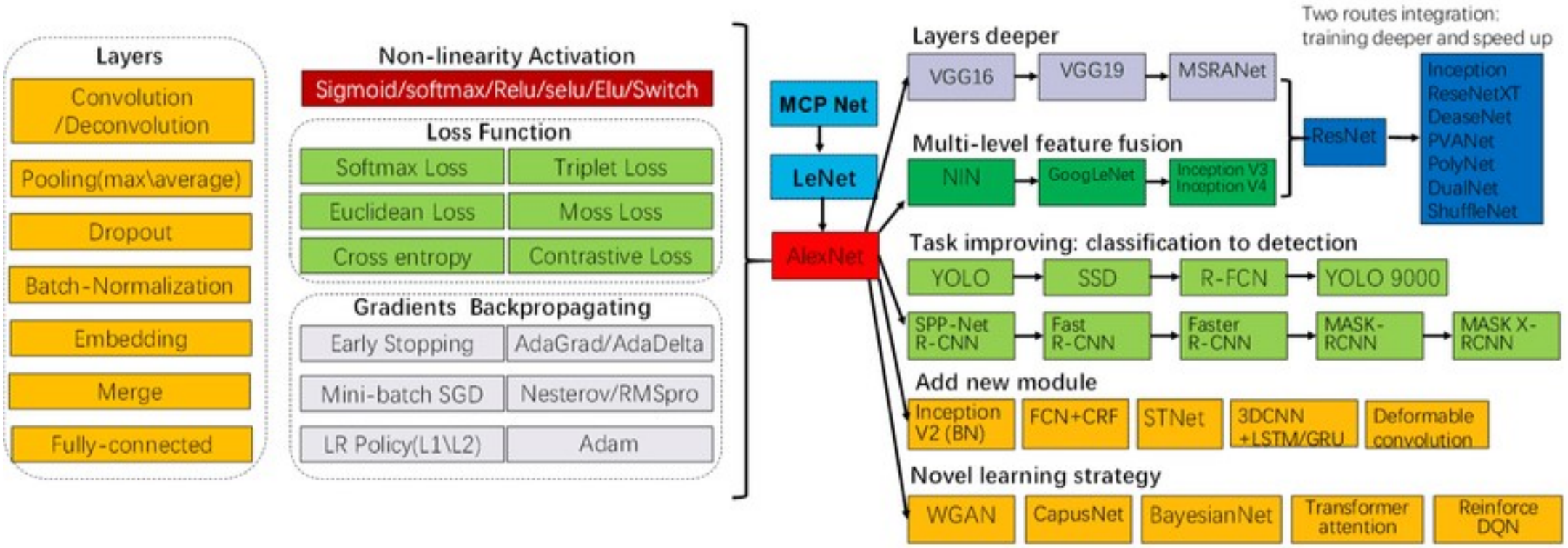
REFERENCE

- **Reference:**
- **ImageAI (v2.1.4)**
- https://github.com/OlafenwaMoses/ImageAI
- **Train Object Detection AI with 6 lines of code (Part_I)** https://medium.com/deepquestai/train-object-detection-ai-with-6-lines-of-code-6d087063f6ff
- **Object Detection with 10 lines of code (Part_II)**
- https://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606
  - **Evolution of Object Detection and Localization Algorithms** https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad
  - **Object Localization and Detection** https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html https://medium.com/%E8%B3%87%E6%96%99%E9%9A%A8%E7%AD%86/machine-learning-103-d81ef2ad3597

Outline

- TensorFlow Object Detection API
  - TensorFlow Object Detection API 安裝
  - 建立自己的訓練資料集
  - 訓練自己的 Object Detection AI 模型
- Computer Vision Tasks

The series of Typical CNN



Data Source : https://www.researchgate.net/figure/The-series-of-components-and-structural-variants-of-typical-convolutional-neural-network_fig12_323591839?_sg=7k4jLu1uyjdmnTR3j4JUpktAj8VfKSU8-EXjFN1Vq87dYPAE8qrJ7jVKmnHxp-c-YzfiPFNlD566Y196B0mxRCq4WmPygeLDqJYlENXtWg

So far: Image Classification

This image is CC0 public domain

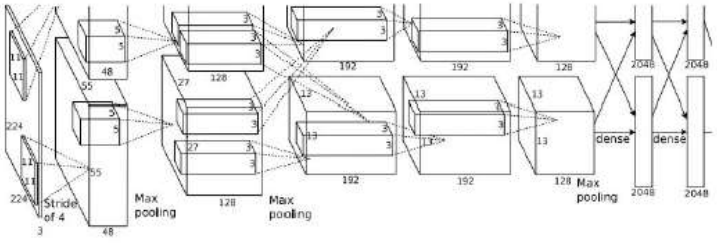


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

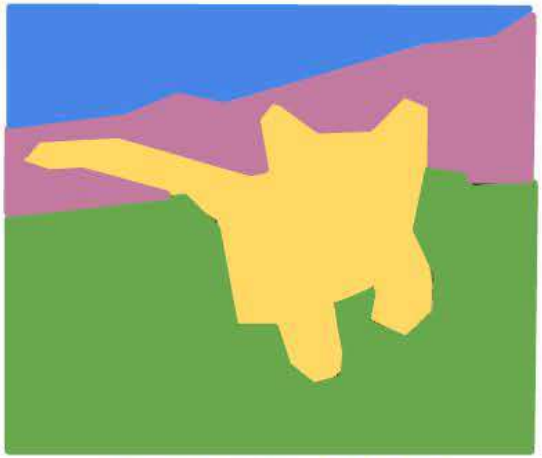
Vector:
4096

Fully-Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Other Computer Vision Tasks

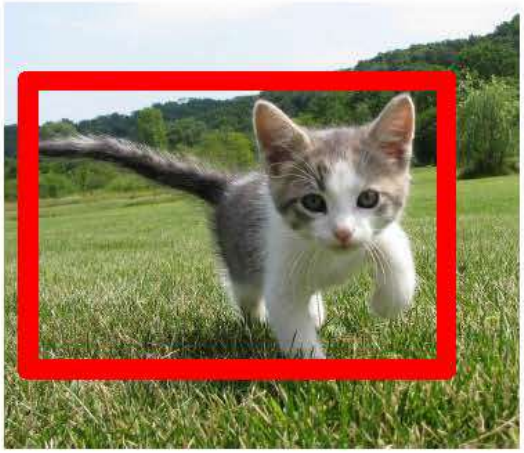
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

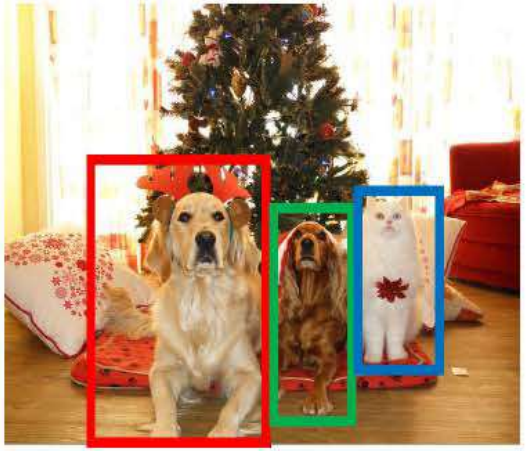
Classification + Localization



CAT

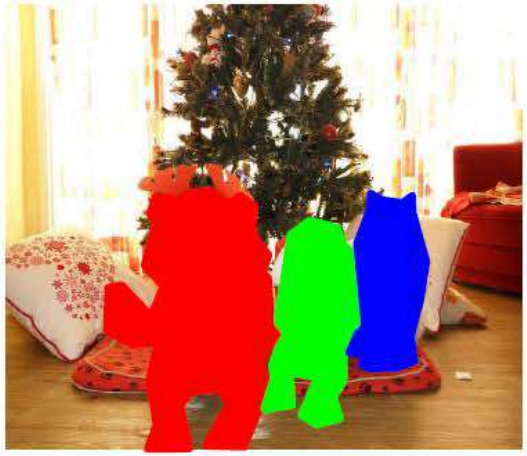
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation



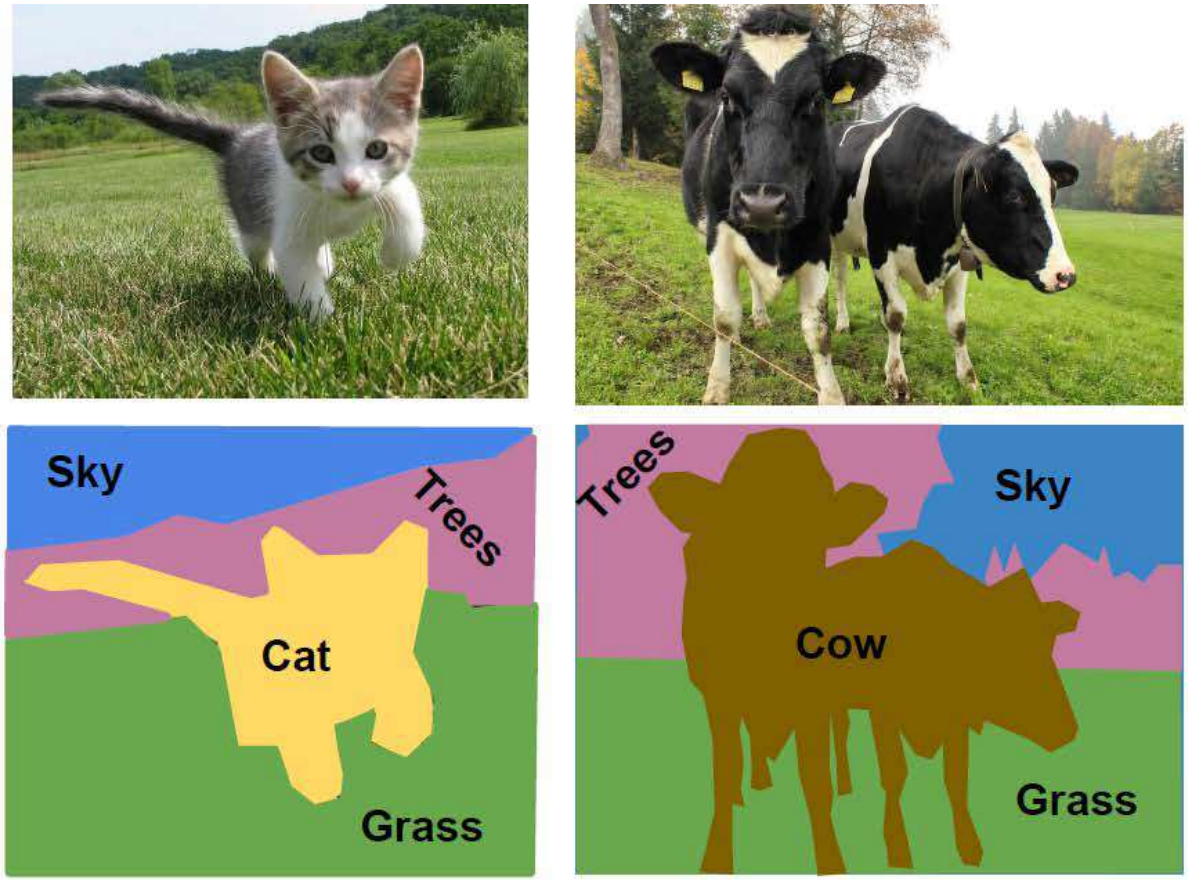
DOG, DOG, CAT

Multiple Object

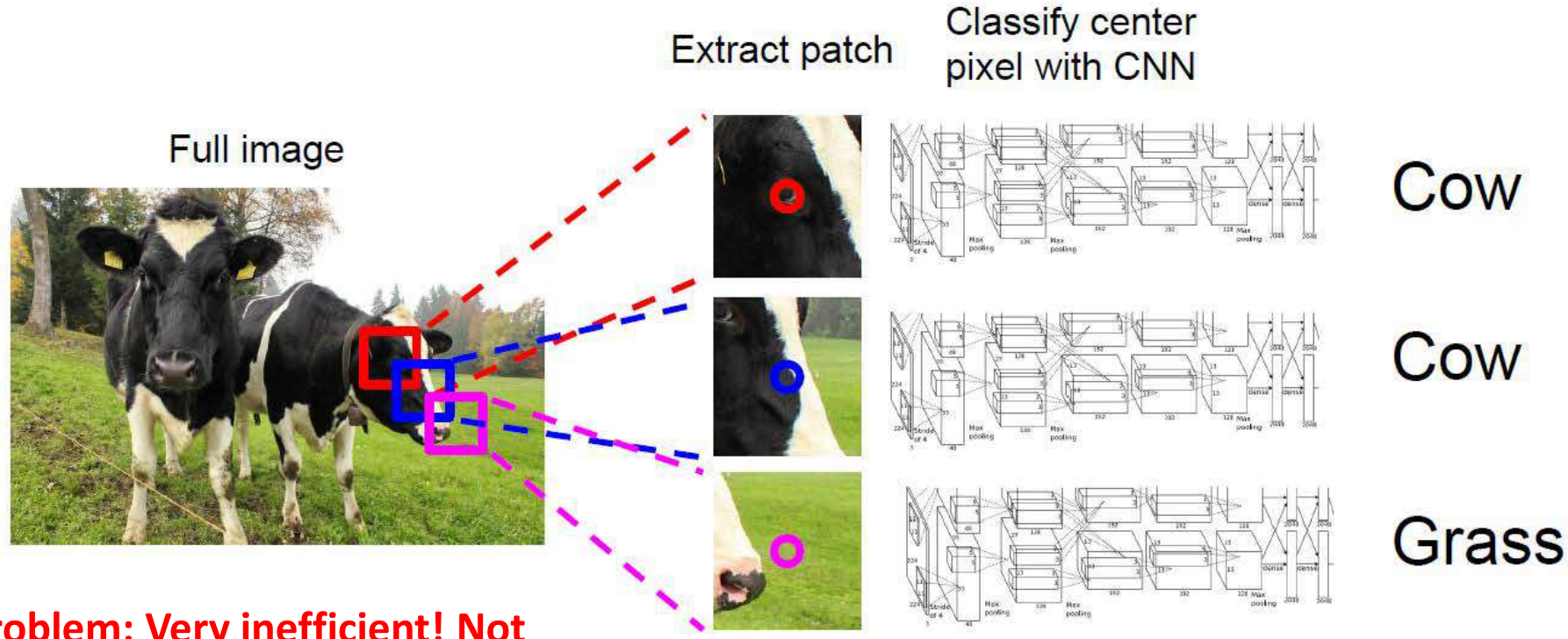
Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels



Semantic Segmentation Idea: Sliding Window

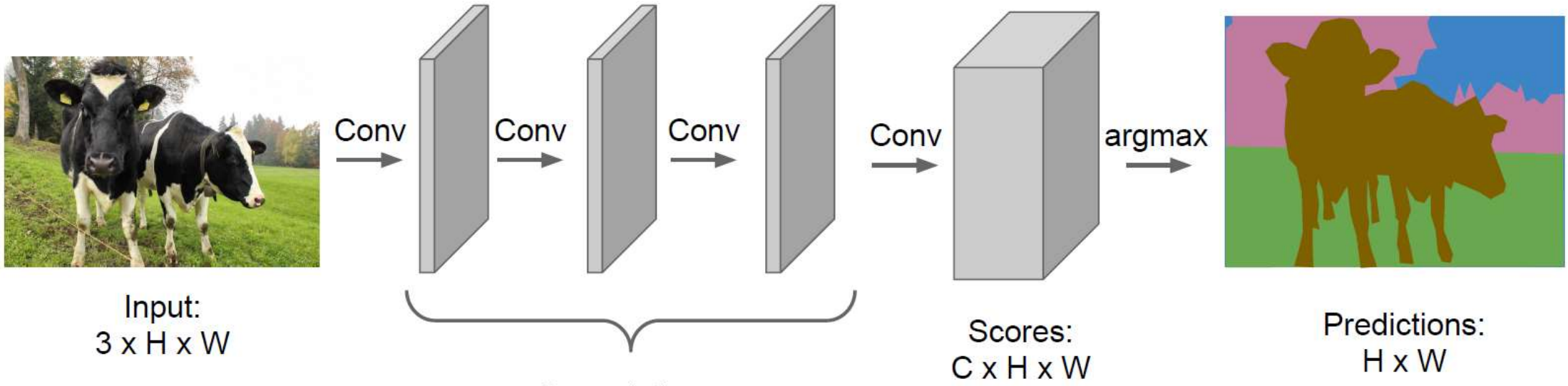


Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

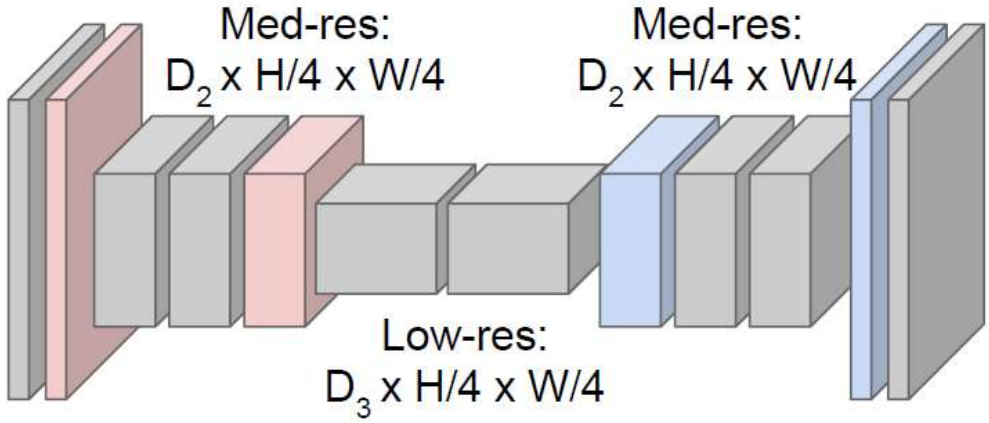


Problem: convolutions at original image resolution will be very expensive ...

Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

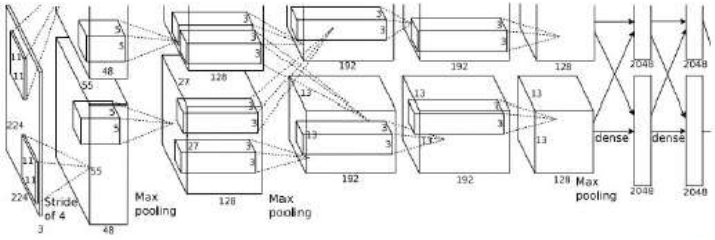
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Vector: 4096

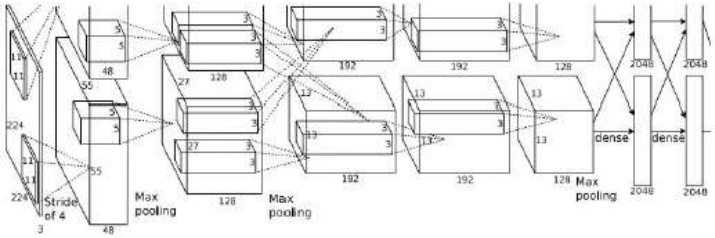
Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

Treat localization as a regression problem!

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

→ **Softmax Loss**

Vector: 4096

Fully Connected: 4096 to 4

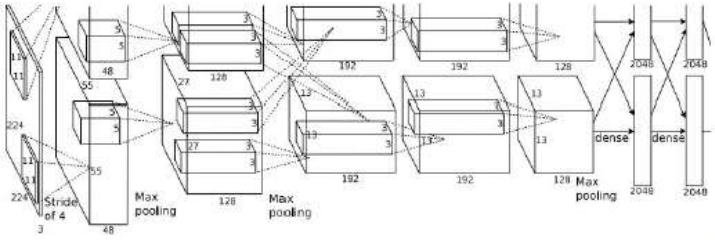
Box Coordinates (x, y, w, h) → **L2 Loss**

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

Treat localization as a regression problem!

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

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

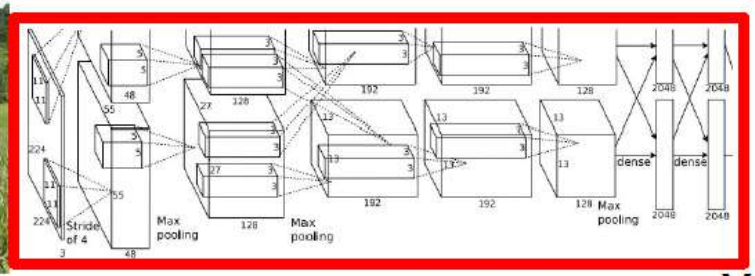
Multitask Loss

+ → **Loss**

Treat localization as a regression problem!

Classification + Localization

This image is CC0 public domain



Often pretrained on ImageNet
(Transfer learning)

Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

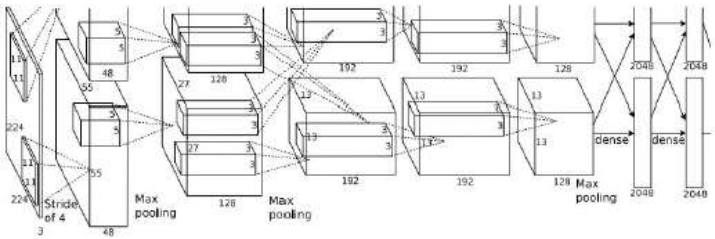
L2 Loss

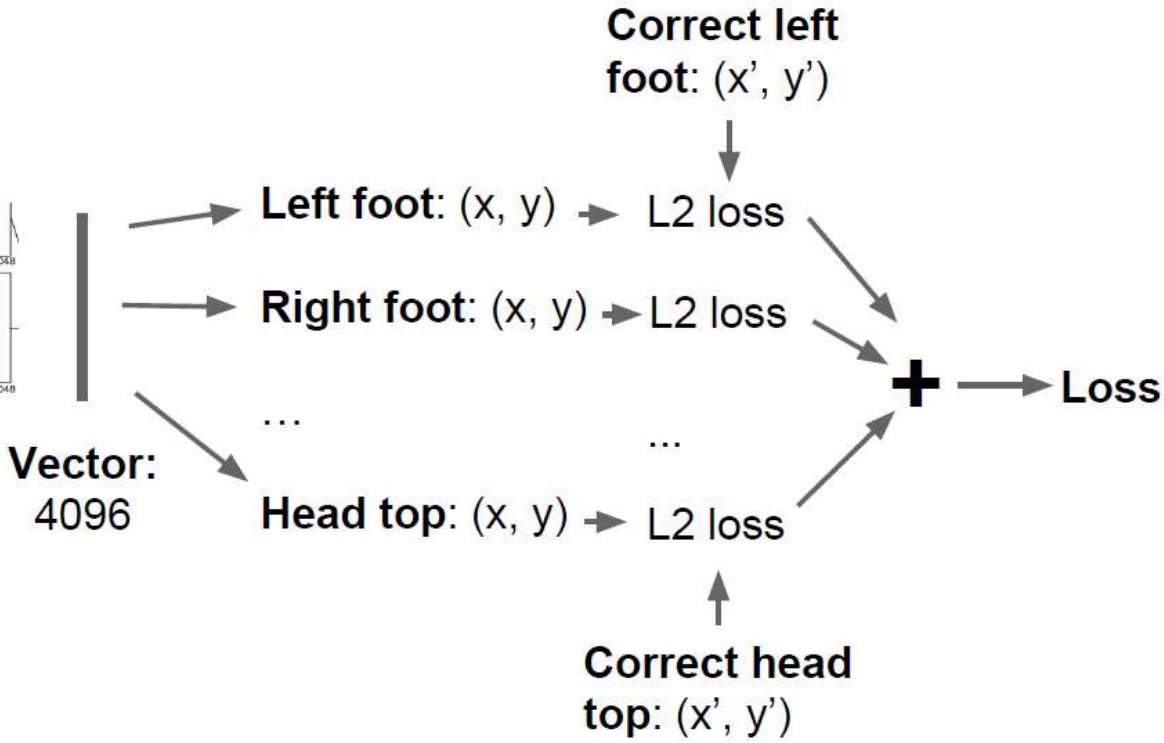
Correct box:
(x', y', w', h')

+ → **Loss**

Treat localization as a regression problem!

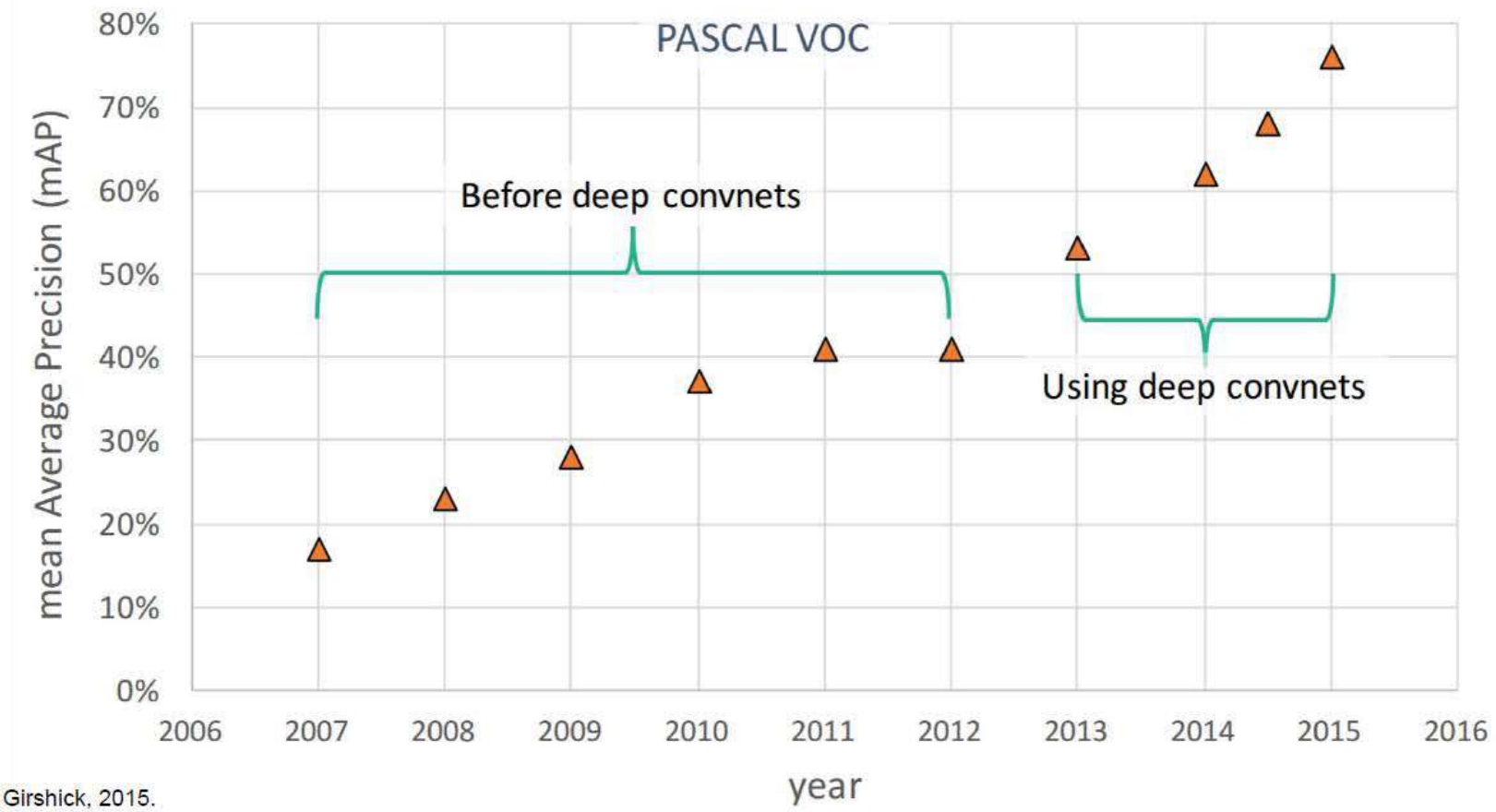
Aside: Human Pose Estimation



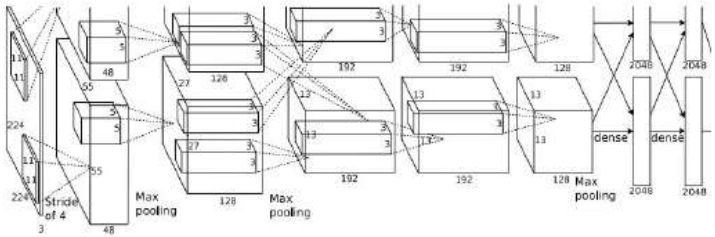


Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

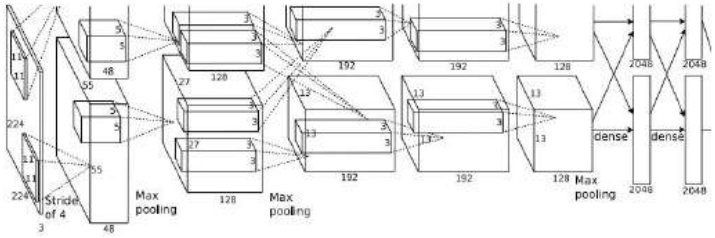
Object Detection: Impact of Deep Learning



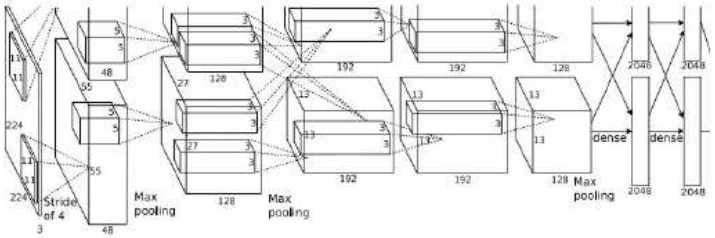
Object Detection as Regression?



CAT: (x, y, w, h)

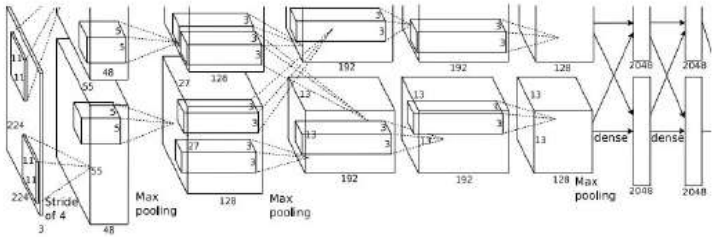


DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



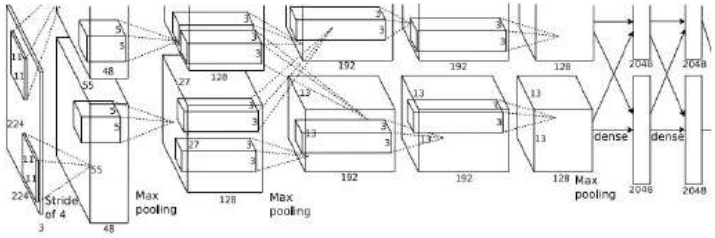
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Object Detection as Regression?



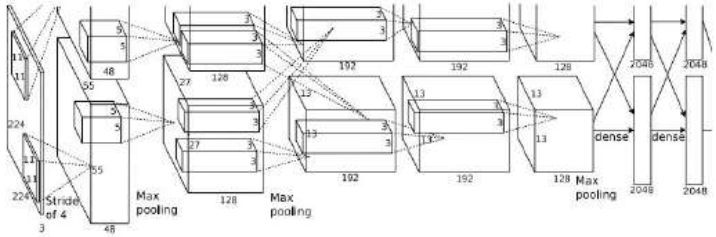
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

16 numbers

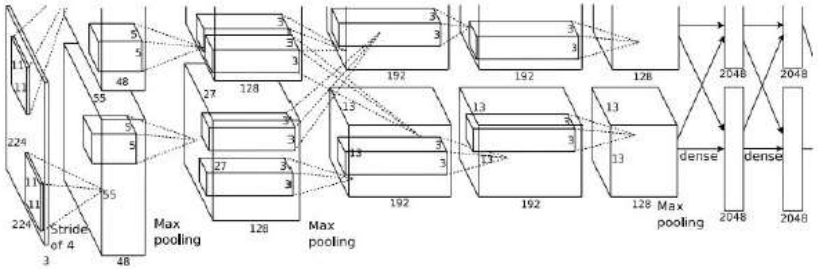


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Object Detection as Classification: Sliding Window

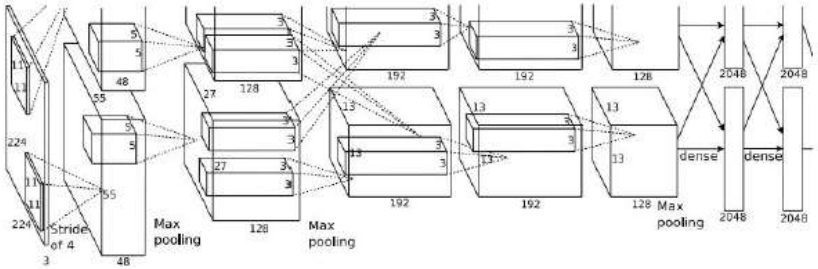
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object Detection as Classification: Sliding Window

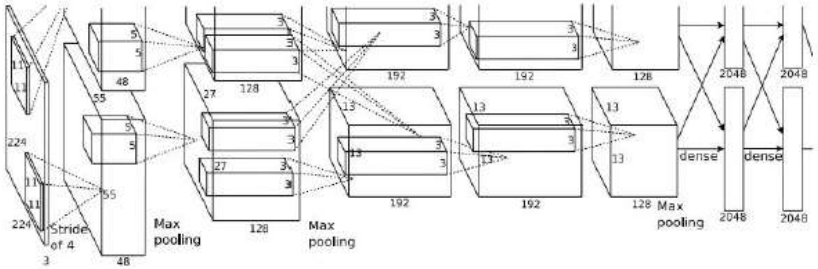
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

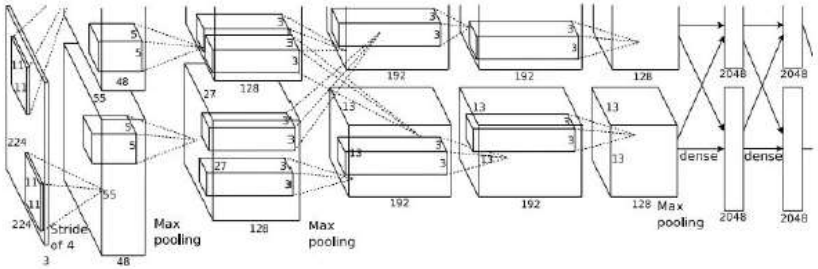
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

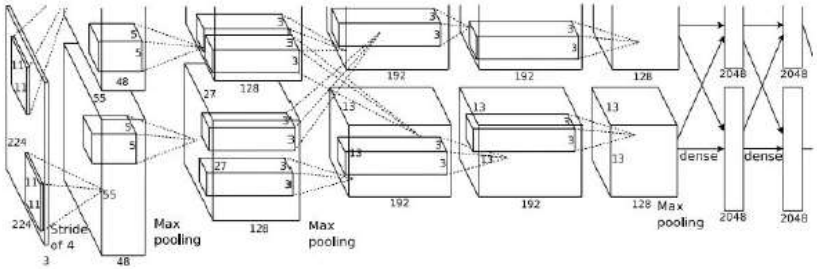
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

Object Detection as Classification: Sliding Window

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

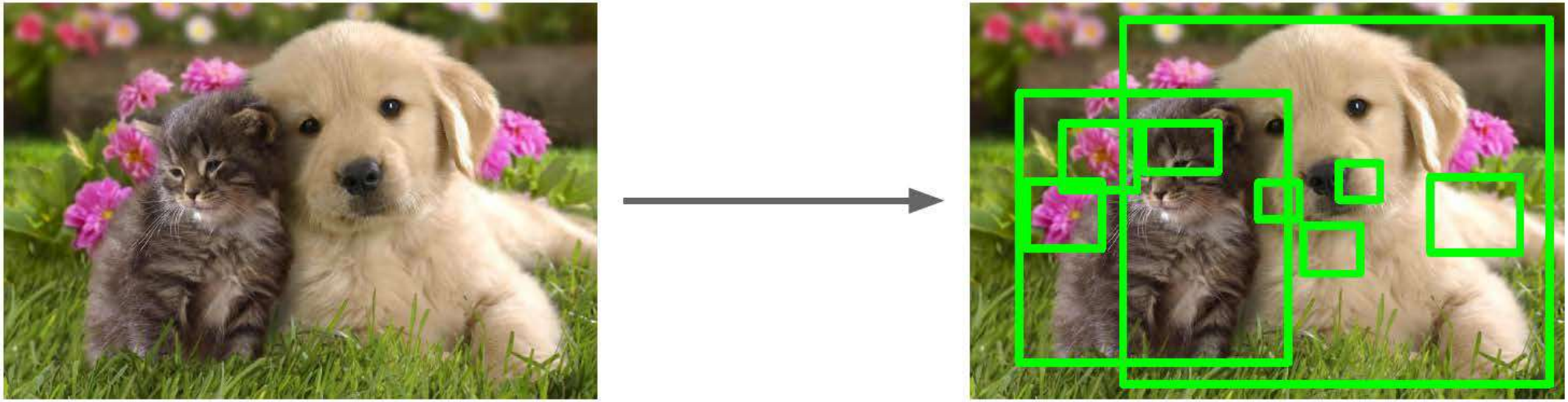


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations and scales, very computationally expensive!

Region Proposals

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 1000 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

R-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.

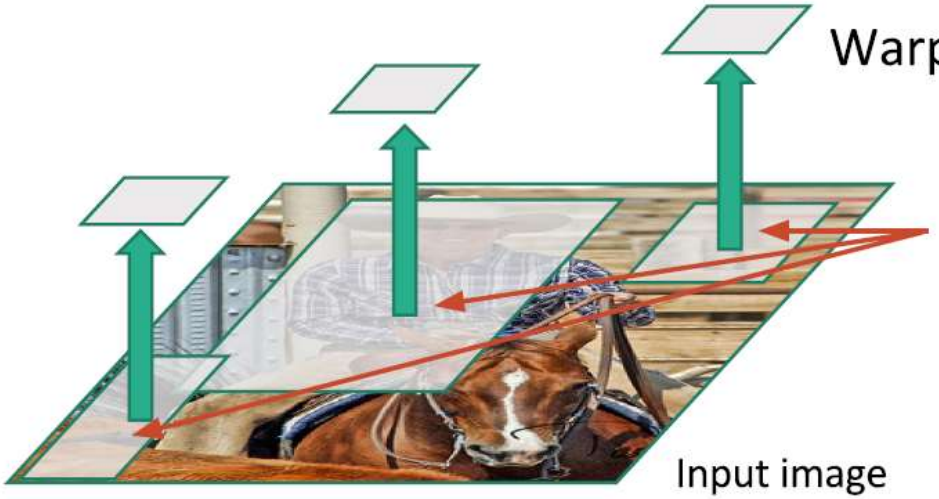
R-CNN

Input image

Regions of Interest (RoI) from a proposal method (~2k)

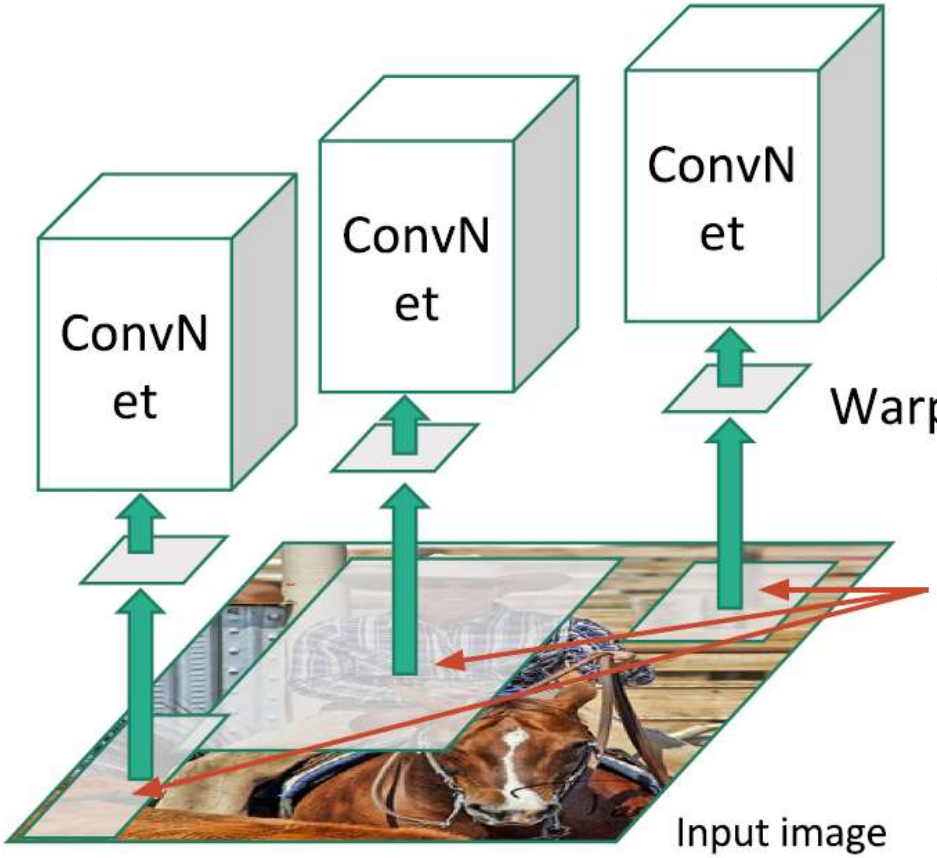
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



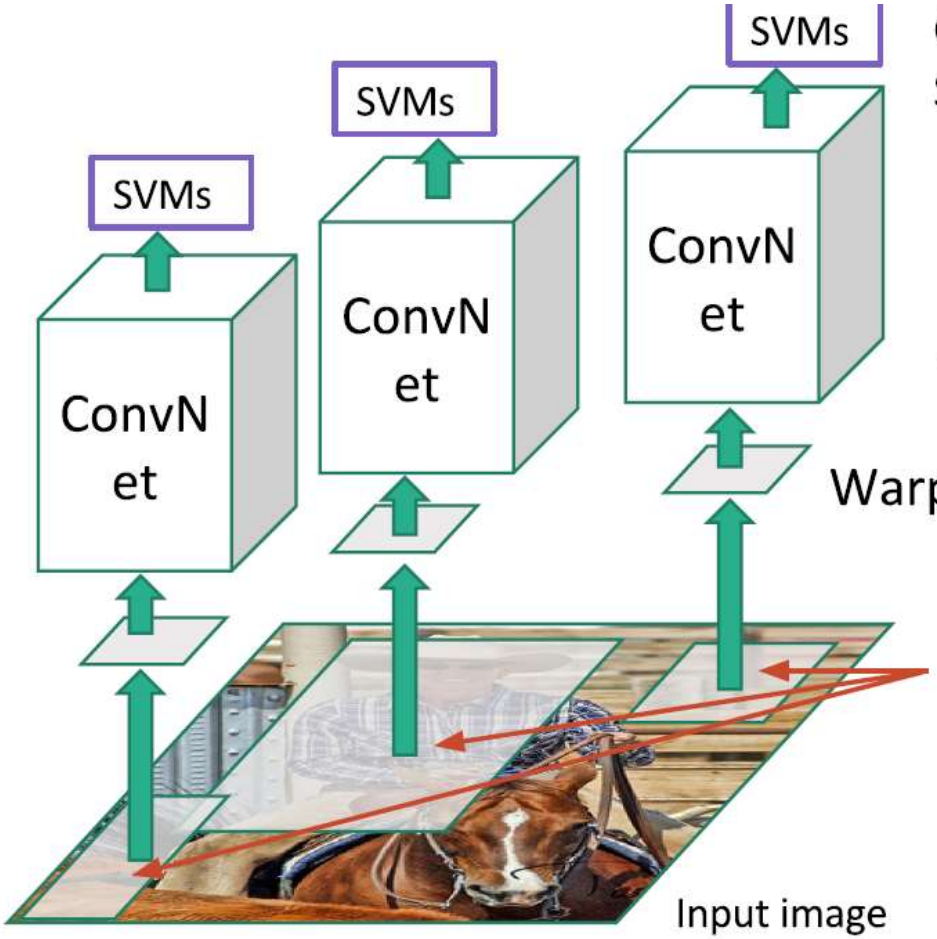
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



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



Classify regions with SVMs

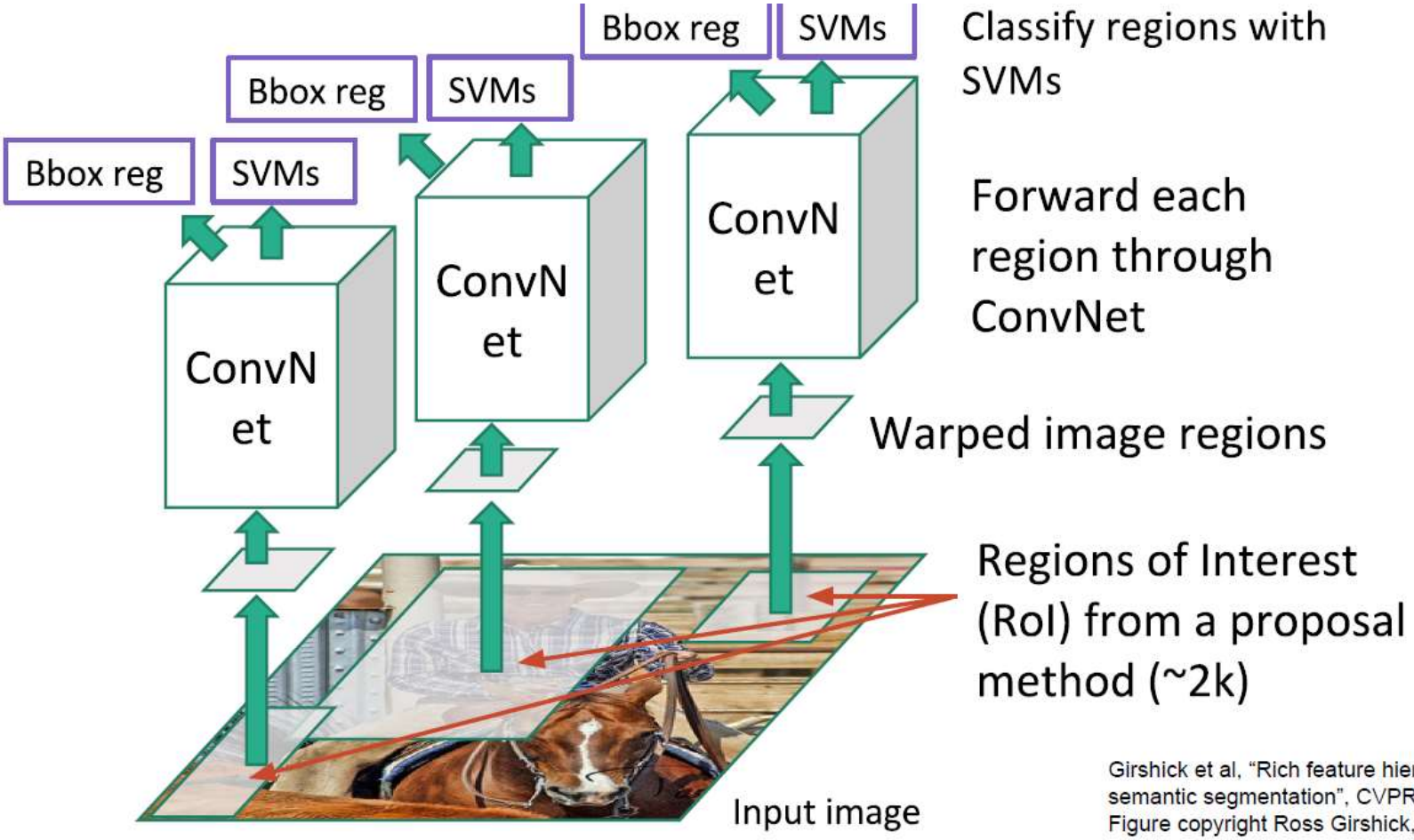
Forward each region through ConvNet

Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

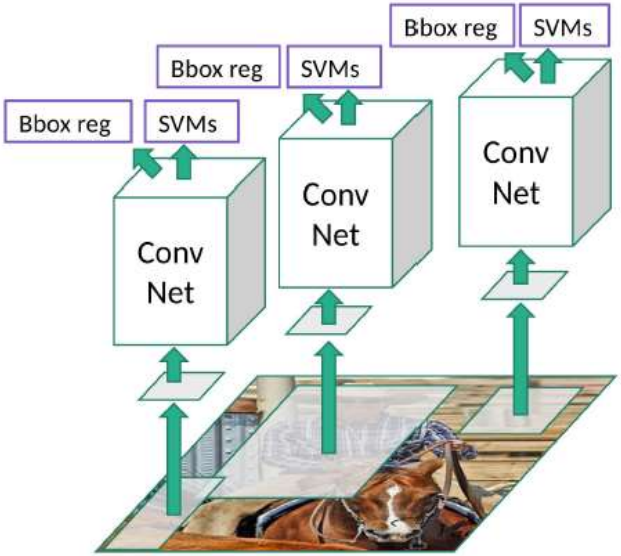
R-CNN



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: Problems

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - Fixed by SPP-net [He et al. ECCV14]



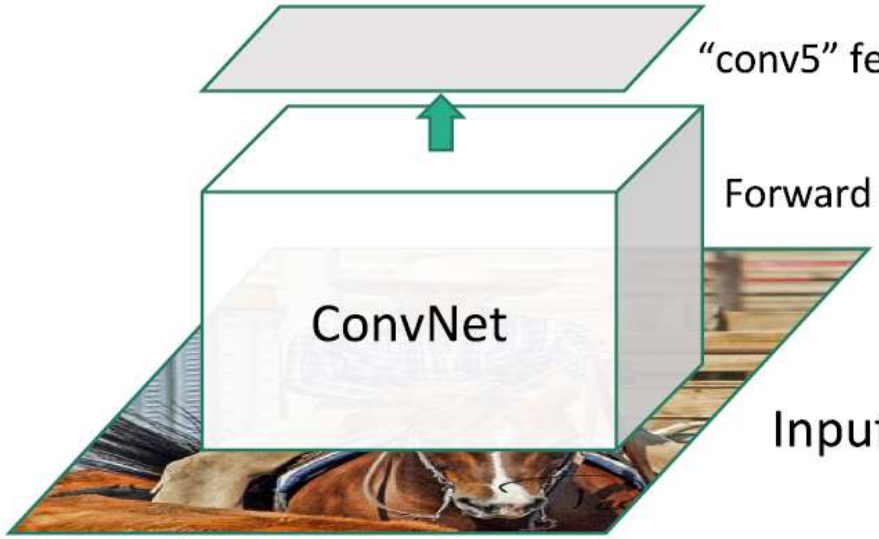
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Slide copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

Input image

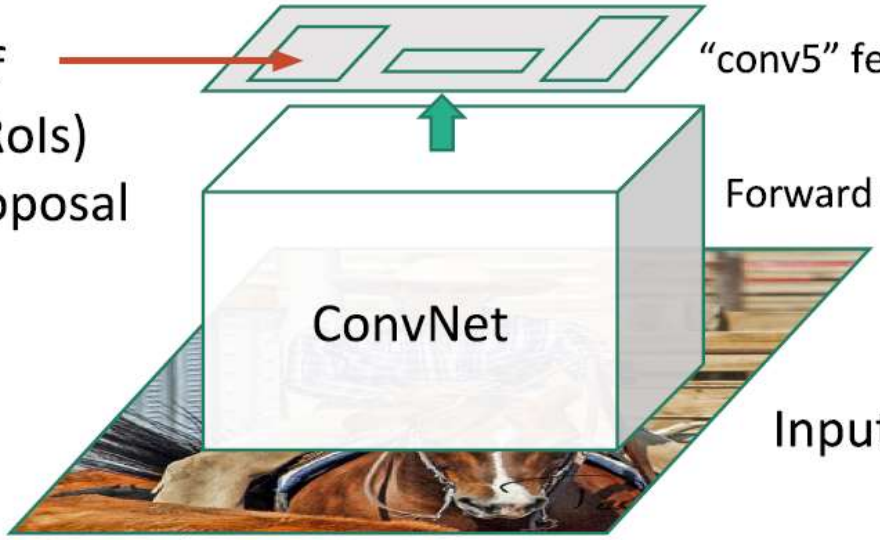
Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



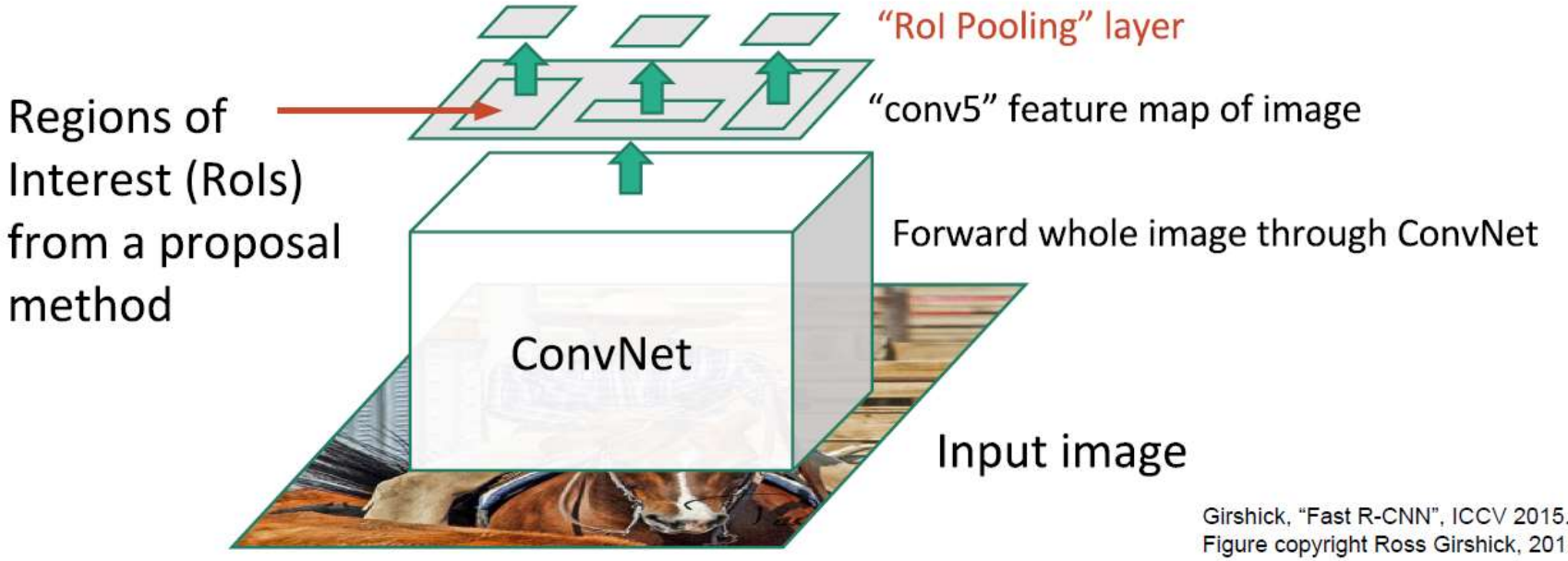
Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



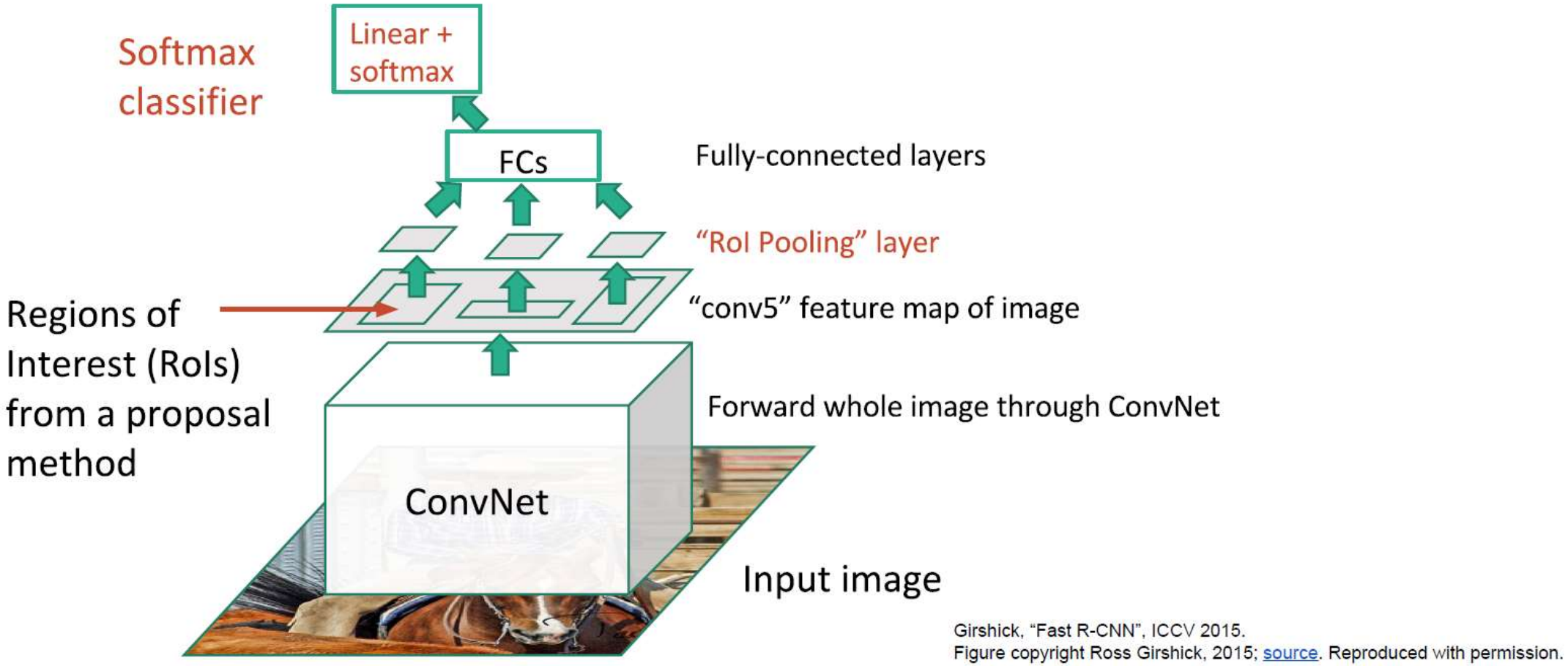
Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



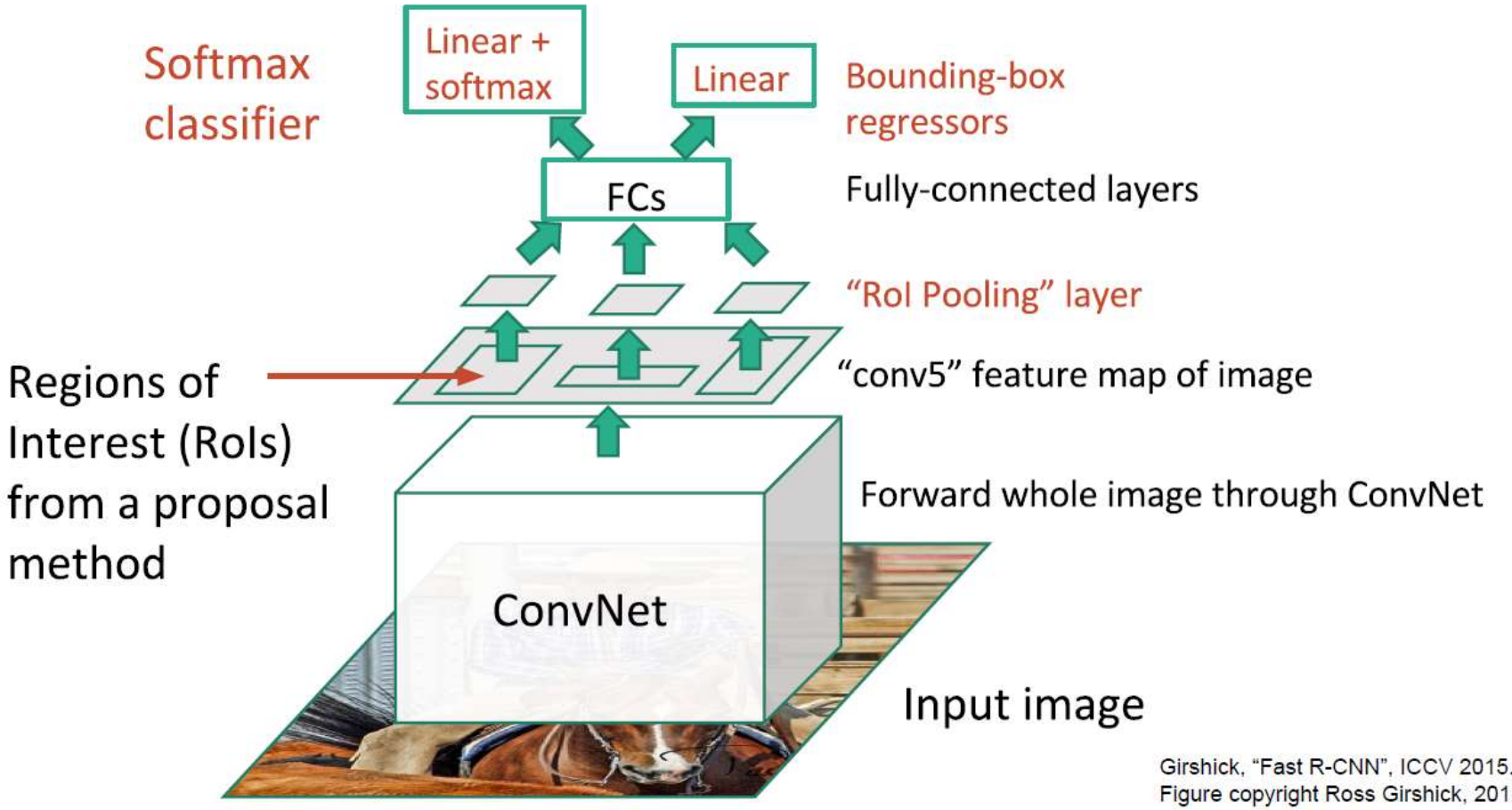
Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



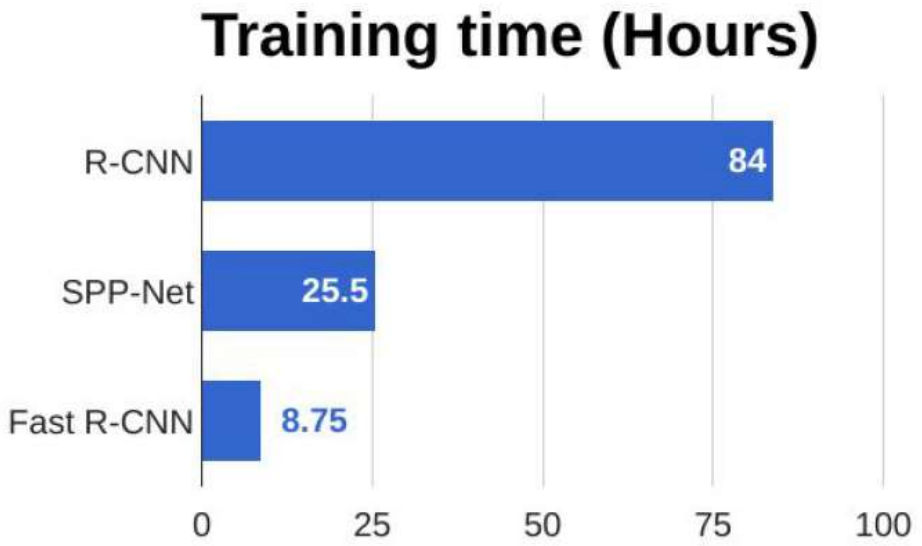
Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

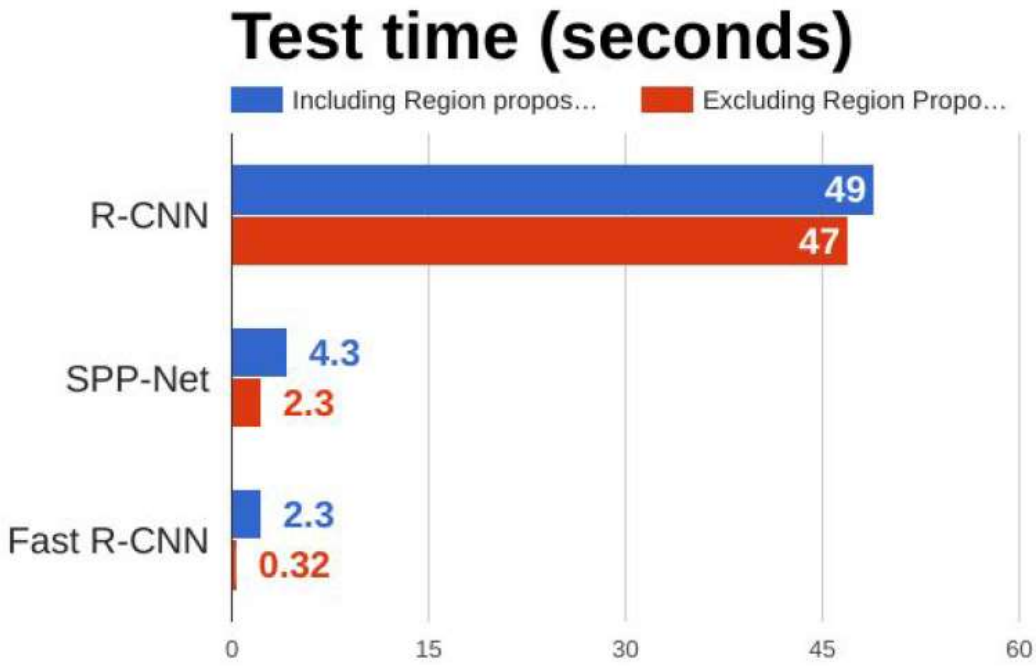
Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN vs SPP vs Fast R-CNN





Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Faster R-CNN

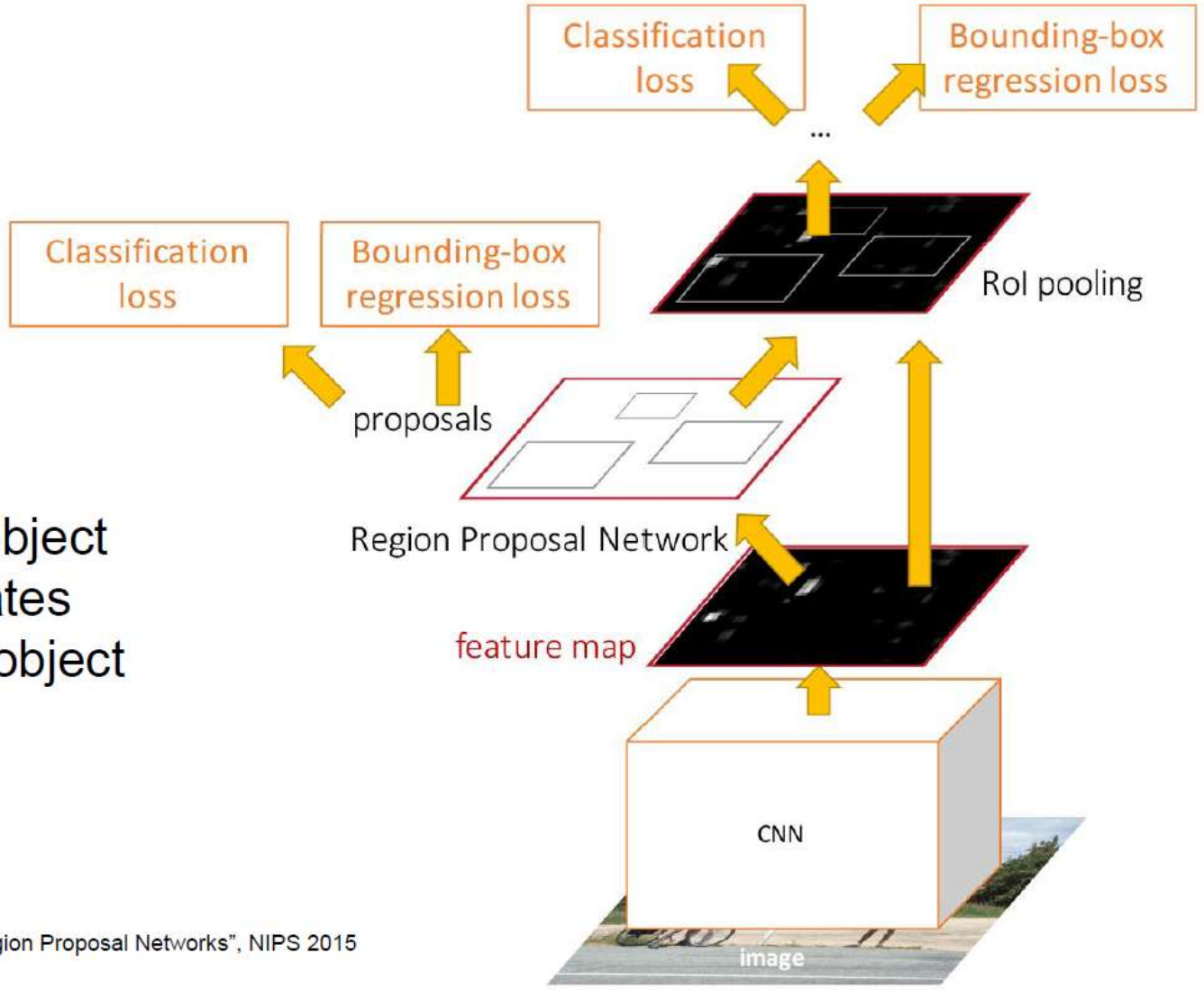
Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

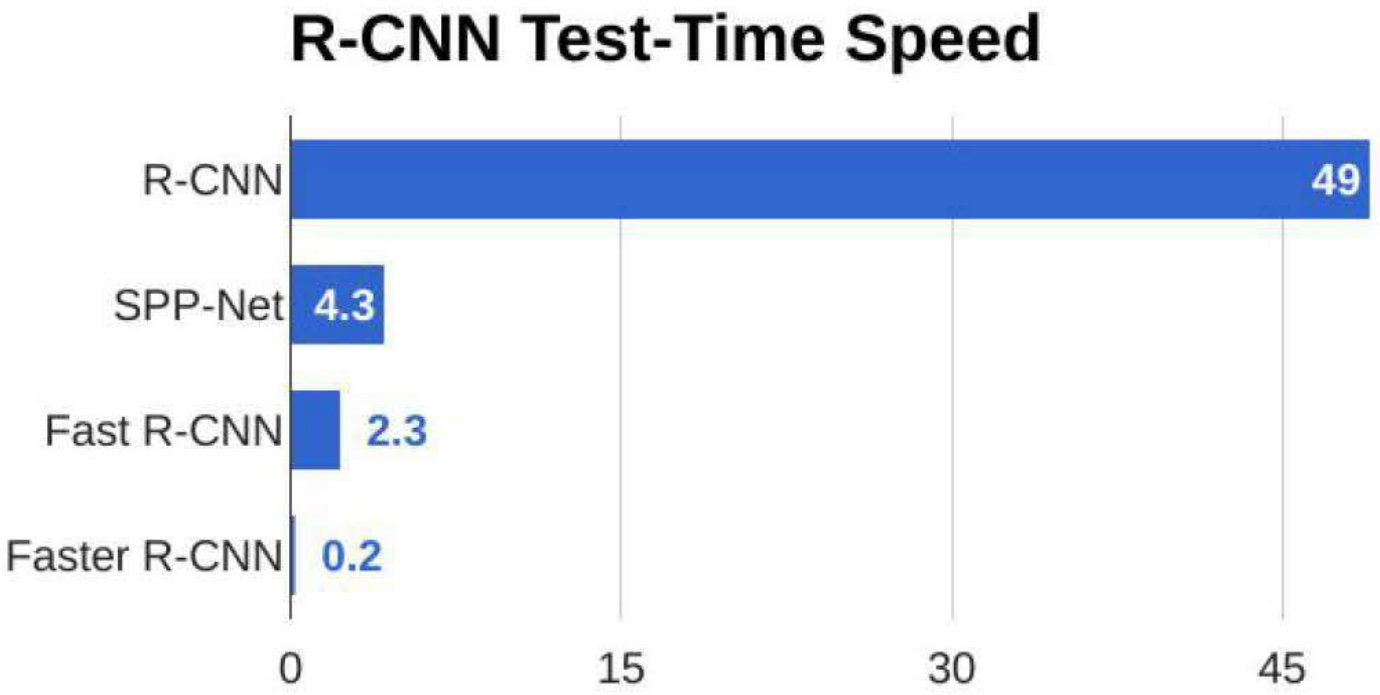
Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



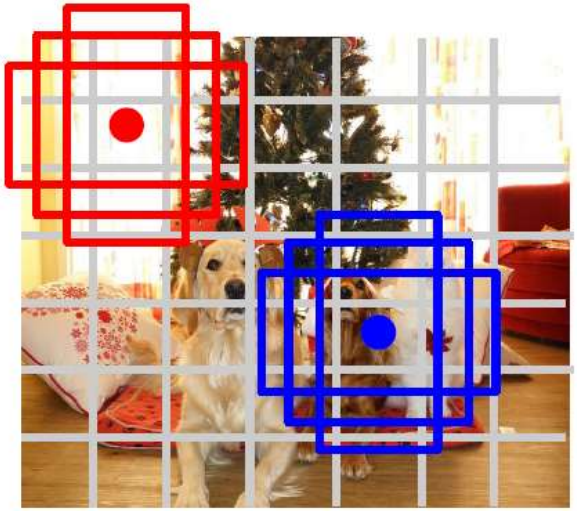
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Faster R-CNN


R-CNN Test-Time Speed
R-CNN
SPP-Net
Fast R-CNN
Faster R-CNN
49
4.3
2.3
0.2
0
15
30
45

Faster R-CNN: RPN Network

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:

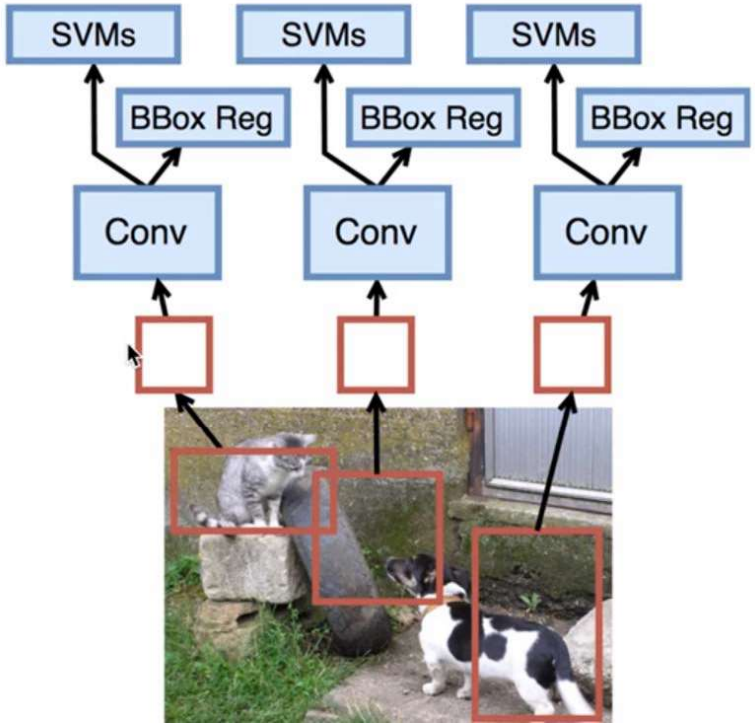
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

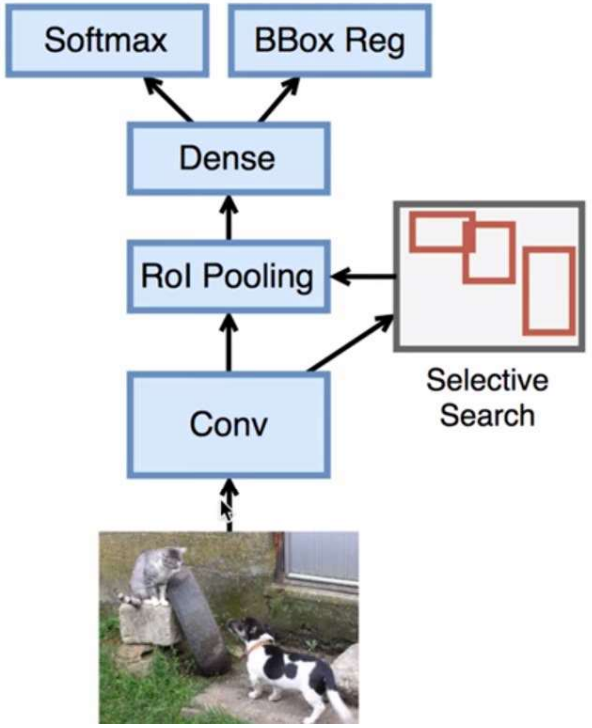
Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

R-CNN vs Fast R-CNN vs Faster R-CNN

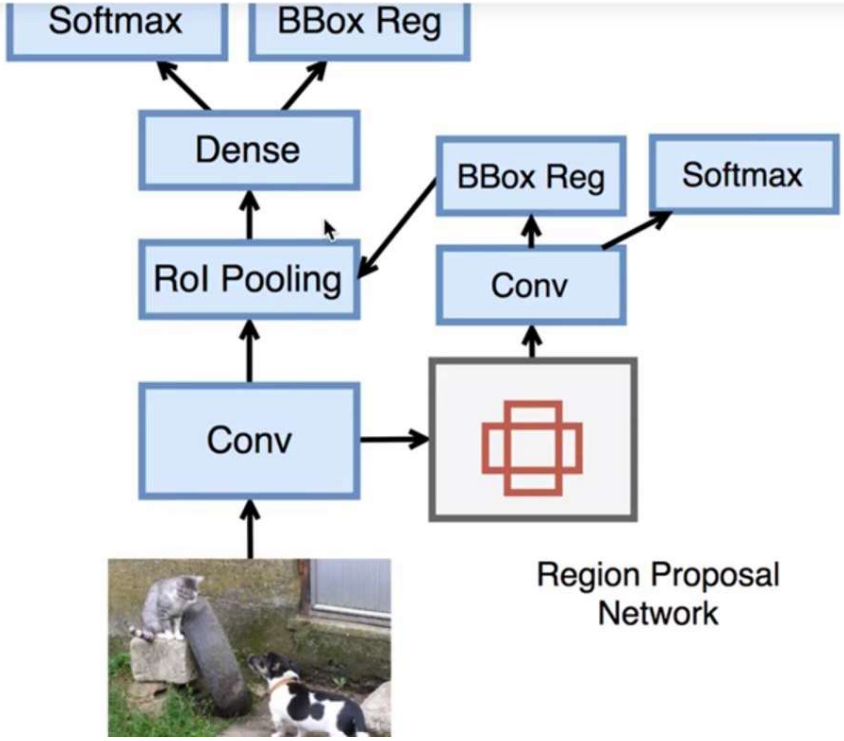
R-CNN



Fast R-CNN



Faster R-CNN



Object Detection: Lots of variables ...

Base Network
VGG16
ResNet-101
Inception V2
Inception V3
Inception
ResNet
MobileNet

Object Detection architecture
Faster R-CNN
R-FCN
SSD

Image Size
 # Region Proposals
...

Takeaways
Faster R-CNN is slower but more accurate

SSD is much faster but not as accurate

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

R-FCN: Dai et al, "R-FCN: Object Detection via Region-based Fully Convolutional Networks", NIPS 2016
Inception-V2: Ioffe and Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", ICML 2015
Inception V3: Szegedy et al, "Rethinking the Inception Architecture for Computer Vision", arXiv 2016
Inception ResNet: Szegedy et al, "Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning", arXiv 2016
MobileNet: Howard et al, "Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv 2017

TensorFlow Models API Installation

1. Download TensorFlow Models from GitHub
   https://github.com/tensorflow/models

2. Download Protobuf from GitHub (If OS is Windows, Version >= 3.5)
   https://github.com/protocolbuffers/protobuf/releases

3. 編譯 TensorFlow Models Protobuf Library (將 proto檔 編譯為 python 檔)
   batch command : (in Tensorflow API "research" folder)
   for /f %i in ('dir /b object_detection\protos*.proto') do protoc-3.7.1-win64\bin\protoc.exe object_detection\protos\%i --python_out=.

4. Copy TensorFlow API Folder to C:\

5. 建立 TensorFlow API OS Environment Variables
   變數名稱：PYTHONPATH
   變數值：C:\TensorFlowAPI\research;
   C:\TensorFlowAPI\research\slim;
   C:\TensorFlowAPI\research\object_detection

6. Restart PC

7. 測試 TensorFlow API 是否安裝成功 ==> C:\TensorFlowAPI\research\object_detection\builders>python model_builder_test.py
   Ran 16 tests in 0.125s
   OK (skipped=1)

Custom Training Dataset

1. 使用 labelImg 圈選 類別物件，並產生 xml 檔。

2. 使用 xml_to_csv.py (created by Datitran, and modify by Yannis) 把xml轉成csv檔

3. 使用 generate_tfrecord.py (created by Datitran, and modify by Yannis) 生成 train.record 與 eval.record

4. 指定標籤名稱 (建立 pbtxt 文件)

Training your AI Models

1. 在此選擇 faster_rcnn_resnet101_coco.config 做修改
   ==> Config 範例檔放置於 models\research\object_detection\samples\configs

2. 從 models/research/object_detection/legacy 複製 train.py 與 eval.py

3. 執行 object detection model training
   python train.py --logtostderr --pipeline_config_path=faster_rcnn_resnet101_coco.config --train_dir=training_log

4. 執行 object detection model evaluation
   python eval.py --logtostderr --pipeline_config_path=faster_rcnn_resnet101_coco.config --checkpoint_dir=training_log --eval_dir=eval_log

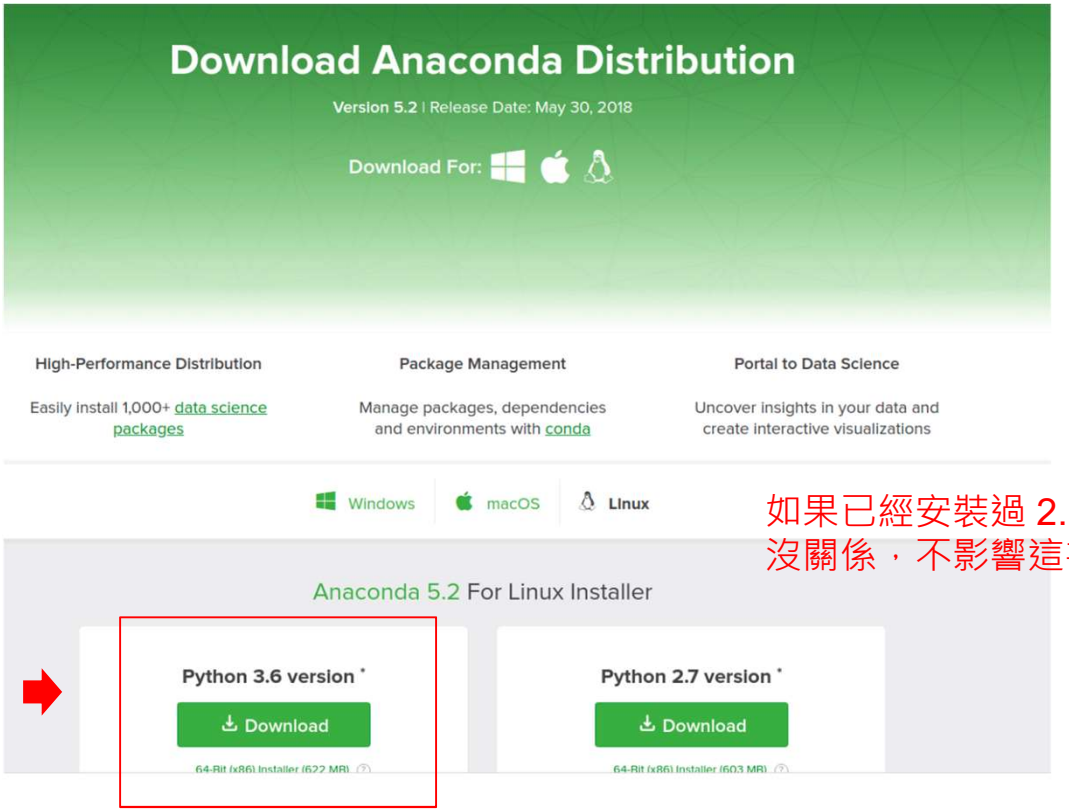
5. 執行 tensorboard
   tensorboard --port=6006 --logdir=TensorFlowAPI_CustomDatasets

YOLO事前準備事項

http://gg.gg/bcc2e

下載 Anaconda

https://www.anaconda.com/download/#linux



選擇 3.6 version

如果已經安裝過 2.7 版本也沒沒關係，不影響這次教學體驗

```
$ sudo apt-get update && sudo apt-get upgrade
$ sudo apt-get install virtualenv
$ virtualenv venv2 –python=2.7
$ source venv2/bin/activate
(venv2) pi@raspberrypi:
(venv2) $ pip install numpy matplotlib
(venv2) $ pip install -U tensorflow keras scikit-image
six tqdm pydot
```

下載模型權重與資料集

常見深度網路相關模型與資料集， 一共 1.1GB 請大家事前下載完成

https://storage.googleapis.com/jiawei-keras-model-dataset/keras-model-weight.zip

Terminal 執行

conda create -n py27 python=2.7

source activate py27

conda install jupyter

pip install -U tensorflow keras opencv-python scikit-image six tqdm pydot

Terminal 執行 (windows)

conda create -n py36 python=3.6

source activate py36

conda install jupyter

pip install -U tensorflow keras opencv-python scikit-image six tqdm pydot

Terminal 執行

unzip keras-model-weight.zip

cd keras-model-weight

cp -R datasets models keras.json ~/.keras

jupyter notebook

Terminal 執行 (windows)

unzip keras-model-weight.zip

進入目錄 keras-model-weight

複製資料 keras-model-weight 到 C:\Users\<你的用户名>

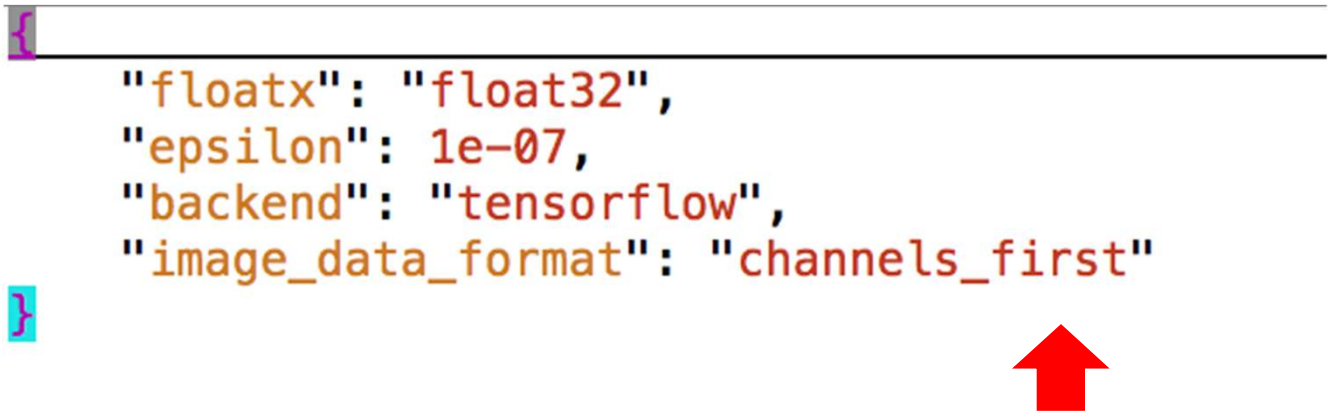
把原本的 .keras 整個資料夾刪除

把 keras-model-weight 改名叫 .keras

jupyter notebook

更改 keras.json

把 channels_last 改成 channels_first



下載模型權重與資料集

Yolo v2 相關模型與資料集， 一共 1.4GB 請大家事前下載完成

https://storage.googleapis.com/jiawei-yolo2-dataset/yolo-dataset.zip

Terminal 執行 (前四行上午執行過了不用管它)

conda create -n py27 python=2.7

source activate py27

conda install jupyter

pip install tensorflow keras opencv-python scikit-image six tqdm pydot

pip install git+https://github.com/aleju/imgaug

windows

下載 https://github.com/aleju/imgaug
解壓縮，進入 imgaug 資料夾
注意 要在 py36 環境
python setup.py install

Terminal 執行

unzip yolo-dataset.zip

cd yolo-dataset/keras-yolo2

jupyter notebook