Deep Learning for Humanists @DHSI2023 – Day 2

Hoyeol Kim

These slides will be shared on GitHub

Tensor

Rank	Type	Examples
0	Scalar	[1]
1	Vector	[1,1]
2	Matrix	[[1,1], [1,1]]
3	3 tensor	[[[1,1], [1,1]], [[1,1], [1,1]]]
n	N tensor	N*[…]]]]…

Tensor

Sentences
Hi Adam
Hi Kate
Hi Setsuko

Unique Words	Index	One Hot Encoding Vector
Hi	0	[1,0,0,0]
Adam	1	[0,1,0,0]
Kate	2	[0,0,1,0]
Setsuko	3	[0,0,0,1]

Tensor

Unique Words	Vector Representations
Hi Adam	[[1,0,0,0], [0,1,0,0]]
Hi Kate	[[1,0,0,0], [0,0,1,0]]
Hi Setsuko	[[1,0,0,0], [0,0,0,1]]

Mini batch input will be in the following:

Hi Adam Hi Kate Hi Setsuko

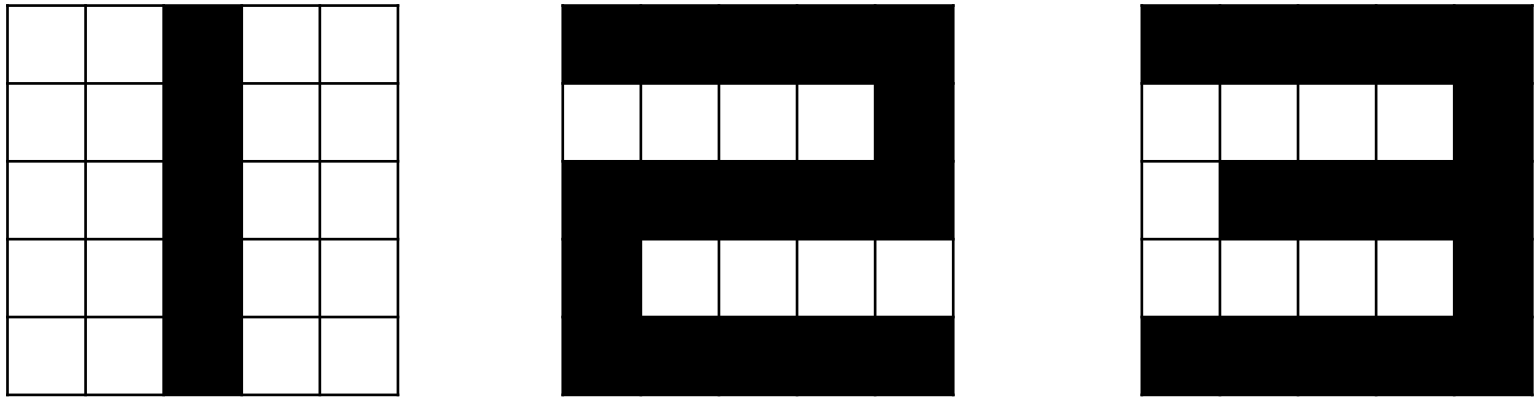
[[[1,0,0,0], [0,1,0,0]], [[1,0,0,0], [0,0,1,0]], [[1,0,0,0], [0,0,0,1]]]

(3, 2, 4) 3d tensor

Tensor in Grayscale Image

(3, 5, 5)

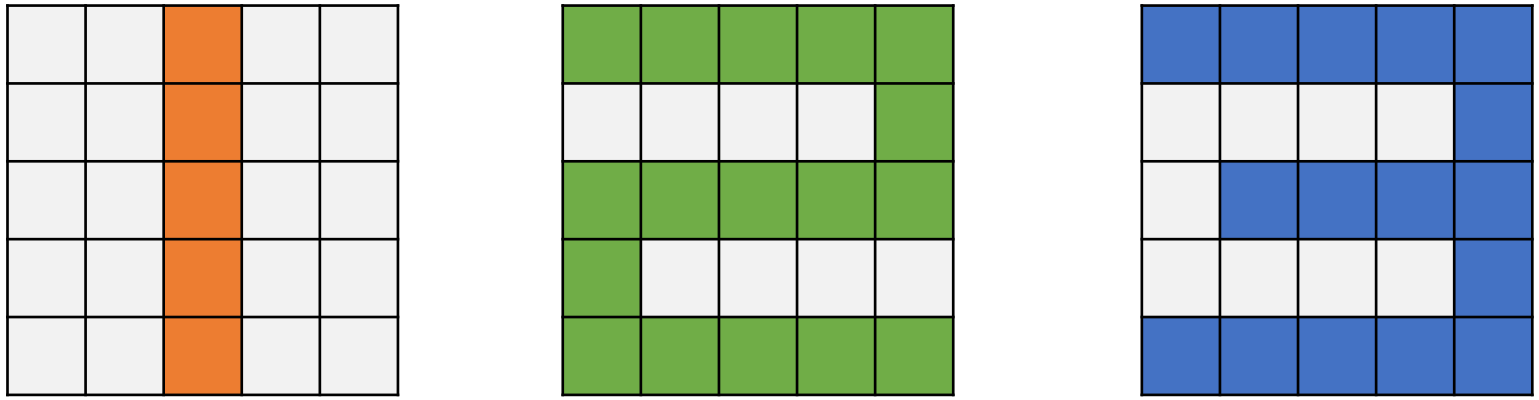
3 images 5 rows 5 columns



Tensor in RGB Color Image

(3, 5, 5, 3) 4d tensor

3 images 5 rows 5 columns red, green, blue



Tensor

PyTorch: torch.tensor()

https://pytorch.org/docs/stable/tensors.html

TensorFlow: tf.Tensor()

https://www.tensorflow.org/api_docs/python/tf/Tensor

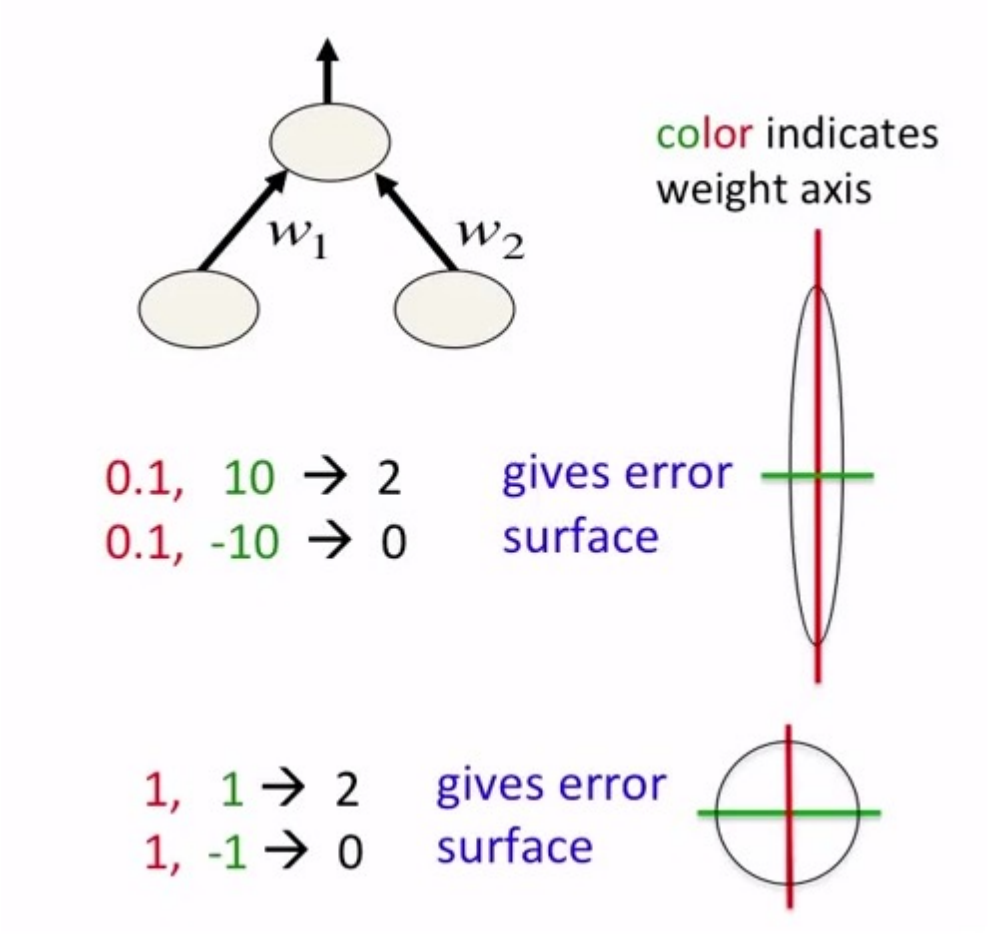
Albumentations: albumentations.pytorch.transforms.ToTensorV2()

https://albumentations.ai/docs/api_reference/pytorch/transforms/

TensorFlow: tf.convert_to_tensor()

https://www.tensorflow.org/api_docs/python/tf/convert_to_tensor

Normalization



- The predictions of neural networks would be inaccurate if features were big in scale.
- If values were very high, it would take a lot of computation time and memory. For example, each pixel normally ranges from 0 to 255.

Normalization

Content | StyleNet IN (ours) | StyleNet BN

Data Augmentation

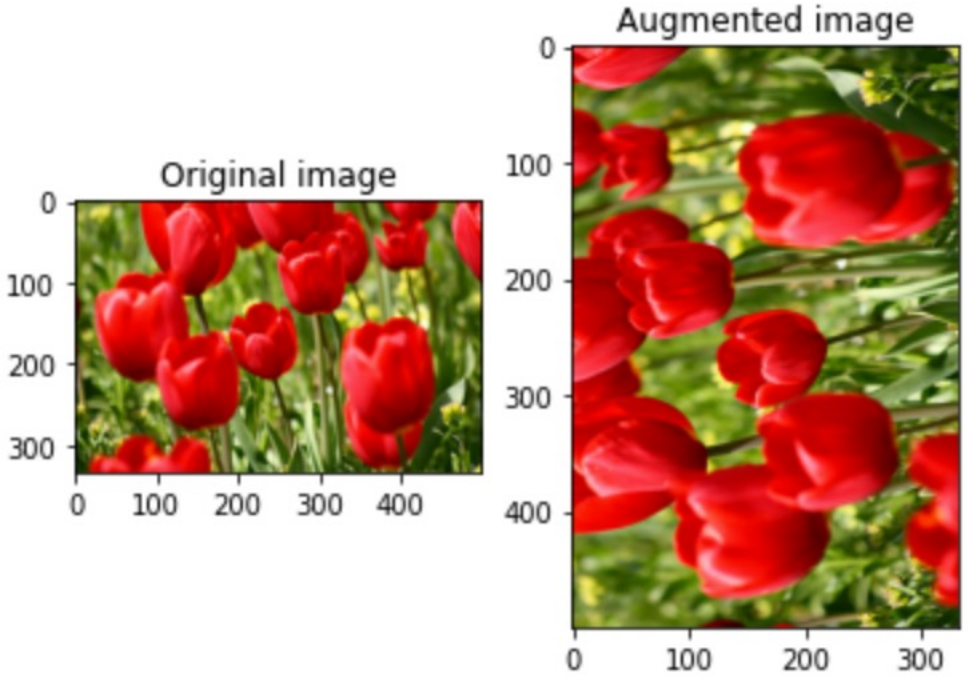
Data augmentation is used to increase the amount of data by adding modified copies of already existing data.

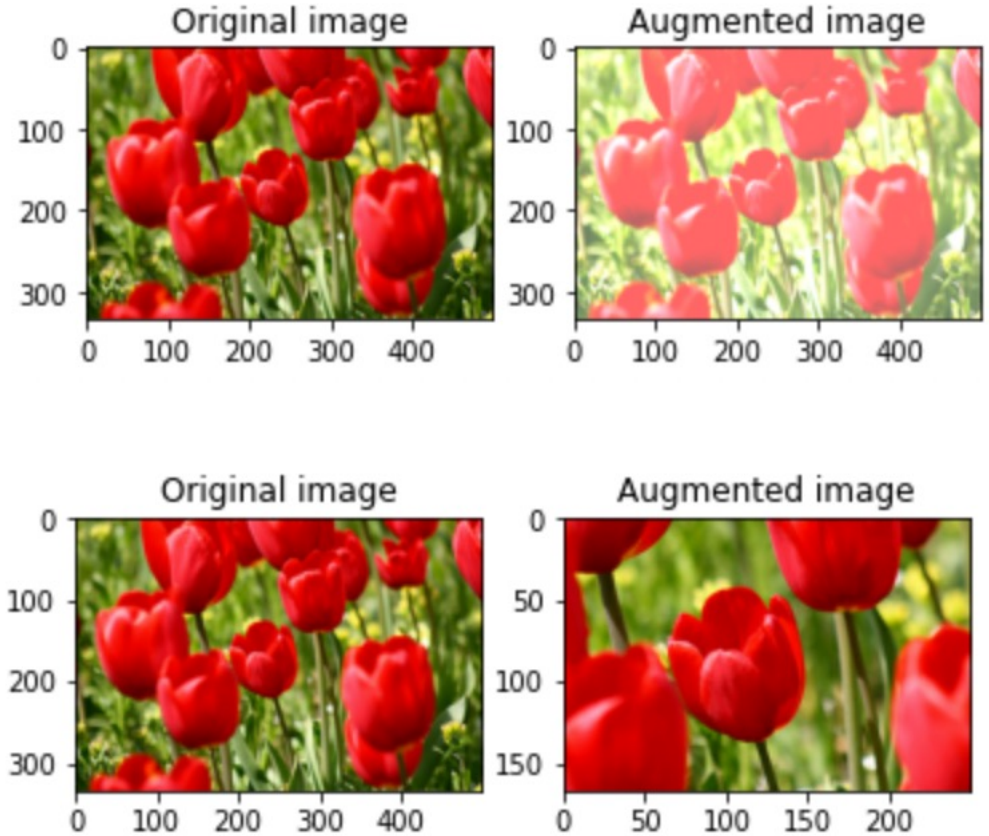
There are a variety of data augmentation techniques, such as rotation, flipping, and cropping.

Recommneded Source: https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

Data Augmentation

1) Rotation 2) RGB 3) Crop 4) Flip 5) Translation (moving objects in images)

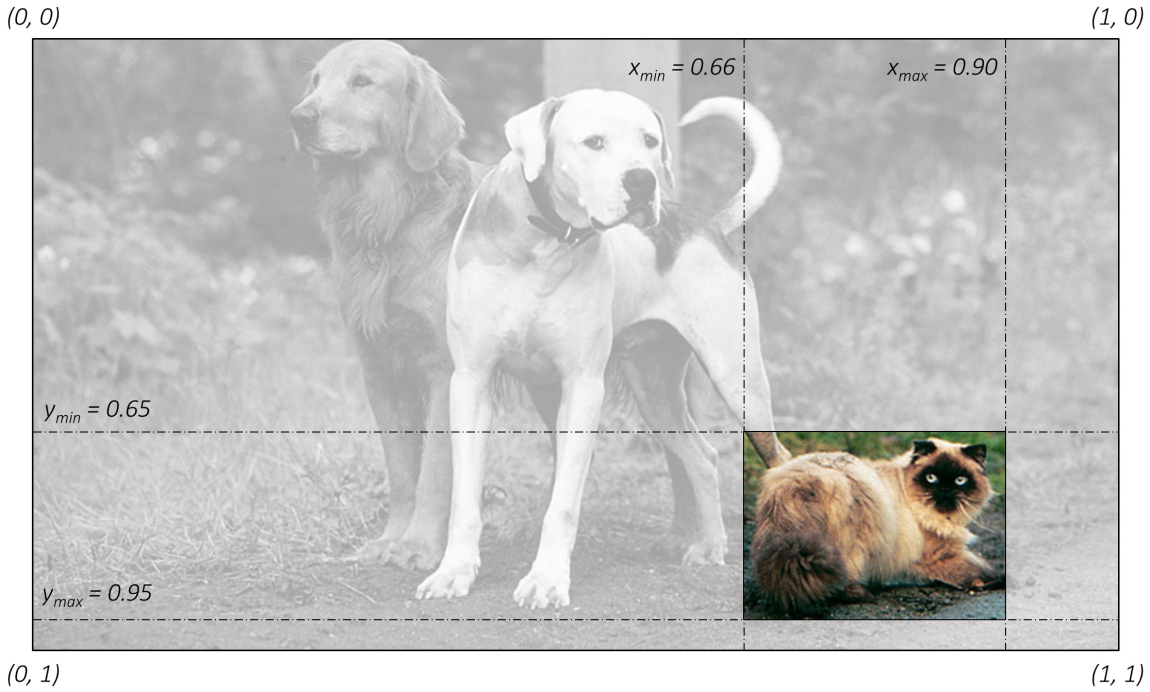




Source: https://www.tensorflow.org/tutorials/images/data_augmentation

Computer Vision Tasks

- Image classification
- Object detection
  - Face recognition
  - Edge detection
- Image restoration
  - Colorization
  - Creating styles
- Video motion analysis



Boundary Coordinates $(x_{min}, y_{min}, x_{max}, y_{max}) = (0.66, 0.65, 0.90, 0.95)$

Classification

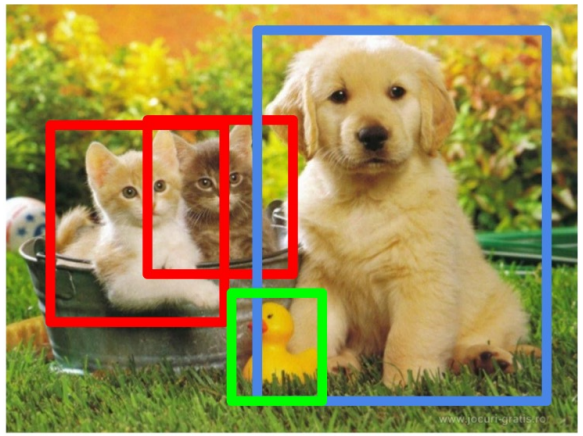
CAT

Classification + Localization

CAT

Single object

Object Detection



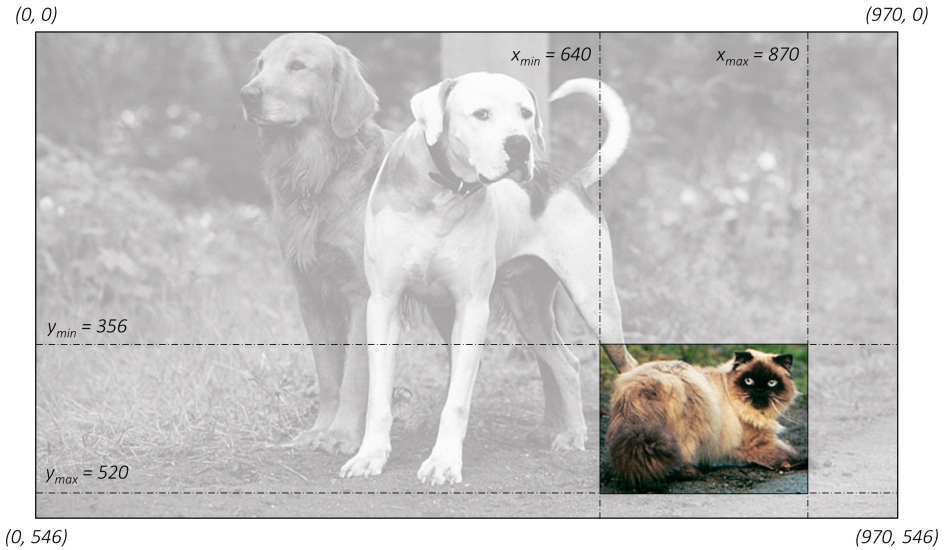
CAT, DOG, DUCK

Instance Segmentation

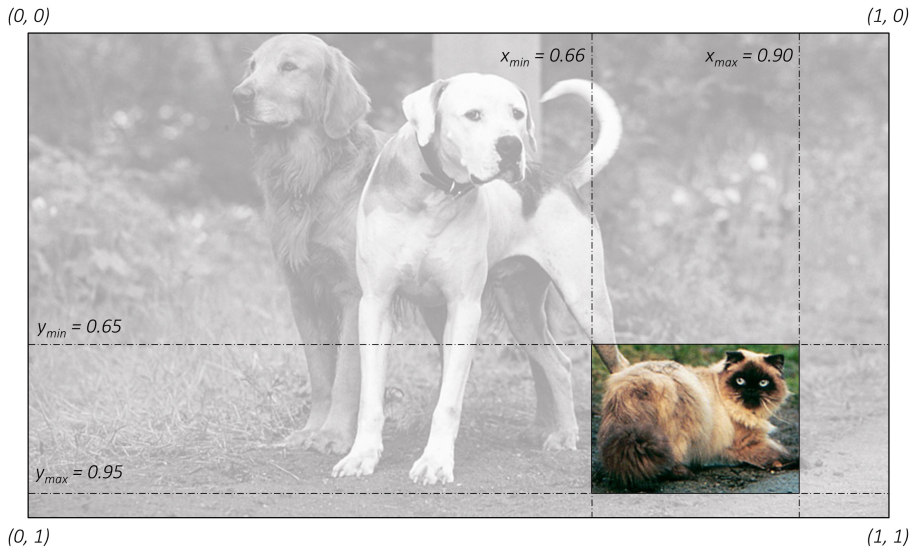
CAT, DOG, DUCK

Multiple objects

Object Detection

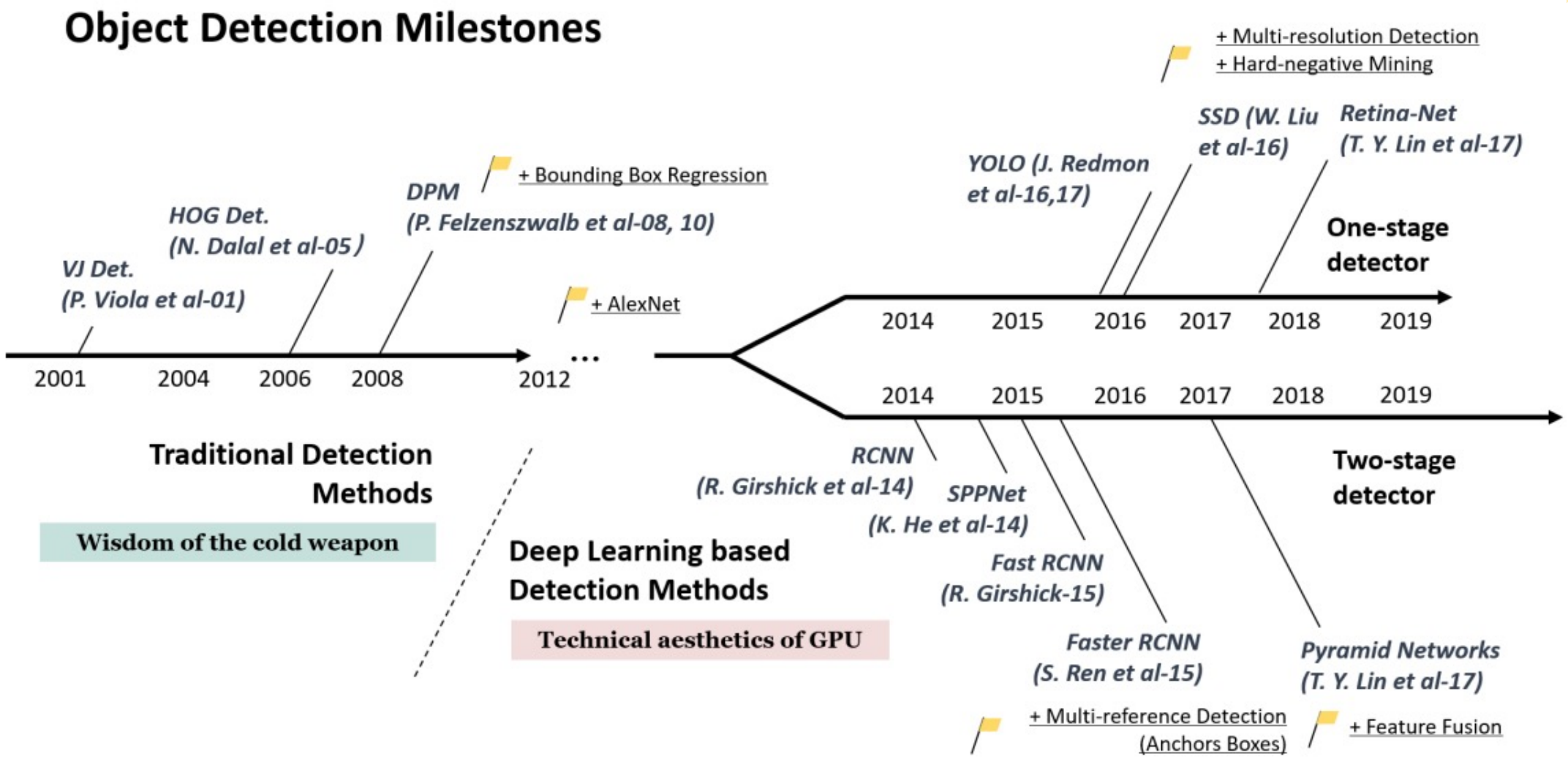


Boundary Coordinates $(x_{min}, y_{min}, x_{max}, y_{max}) = (640, 356, 870, 520)$



Boundary Coordinates $(x_{min}, y_{min}, x_{max}, y_{max}) = (0.66, 0.65, 0.90, 0.95)$

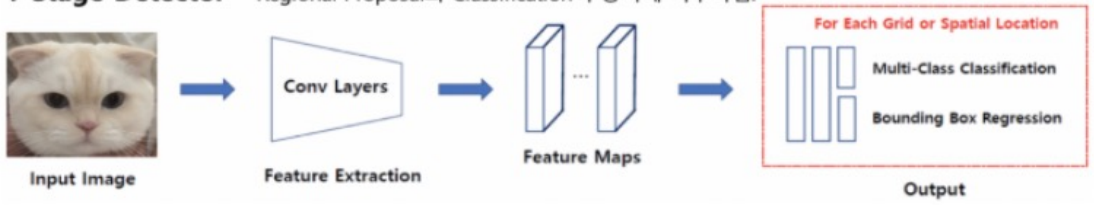
Object Detection Models



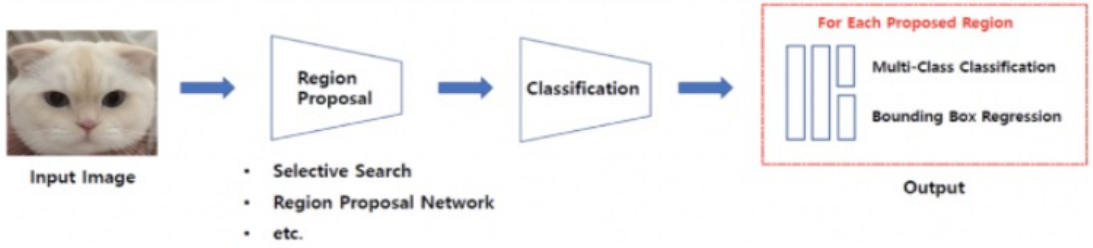
Source: https://arxiv.org/pdf/1905.05055.pdf

Object Detection Models

1-Stage Detector - Regional Proposal와 Classification이 동시에 이루어짐.

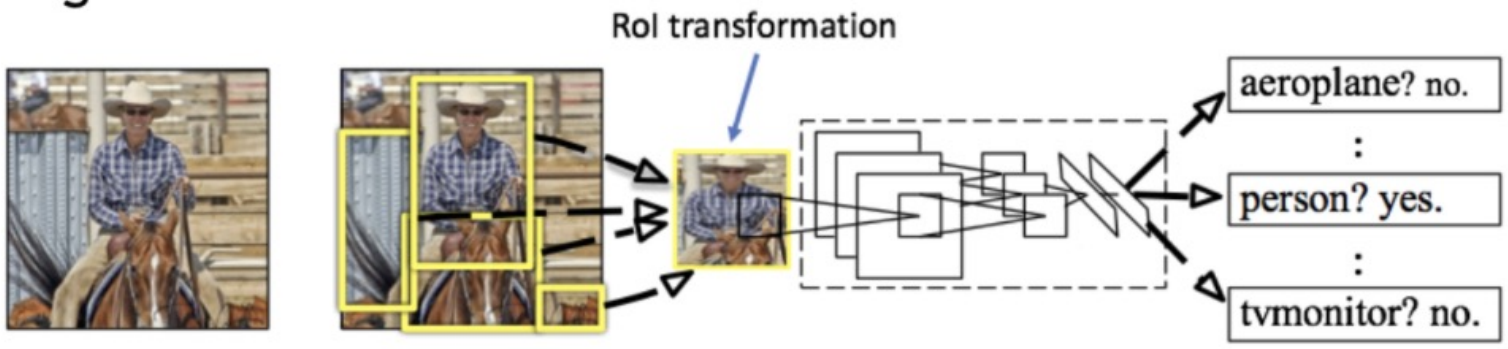


2-Stage Detector - Regional Proposal와 Classification이 순차적으로 이루어짐.



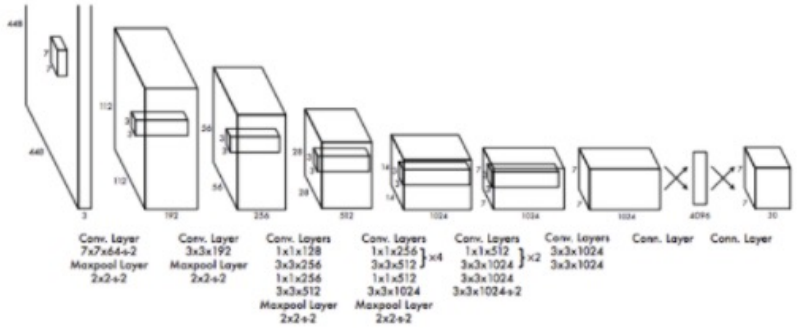
- Selective Search
- Region Proposal Network
- etc.

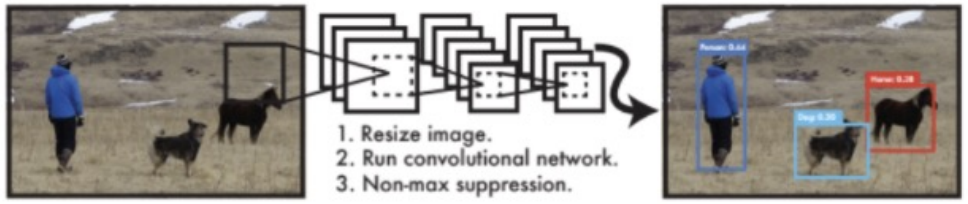
Many stage



Input image | Object / region proposals | Deep Learning region classifier | Region classification, box regression

One stage

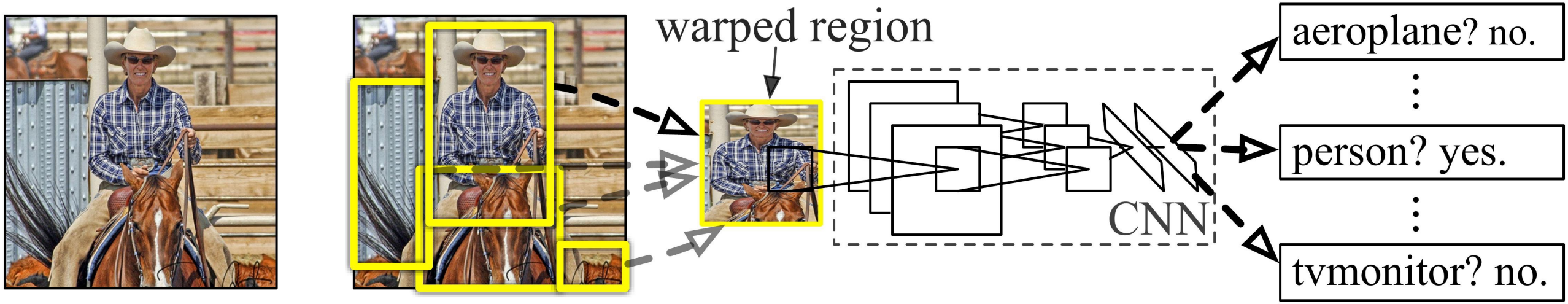




Redmond et al. You Only Look Once: Unified Real-time Object Detection. In CVPR 2016

R-CNN

R-CNN: *Regions with CNN features*



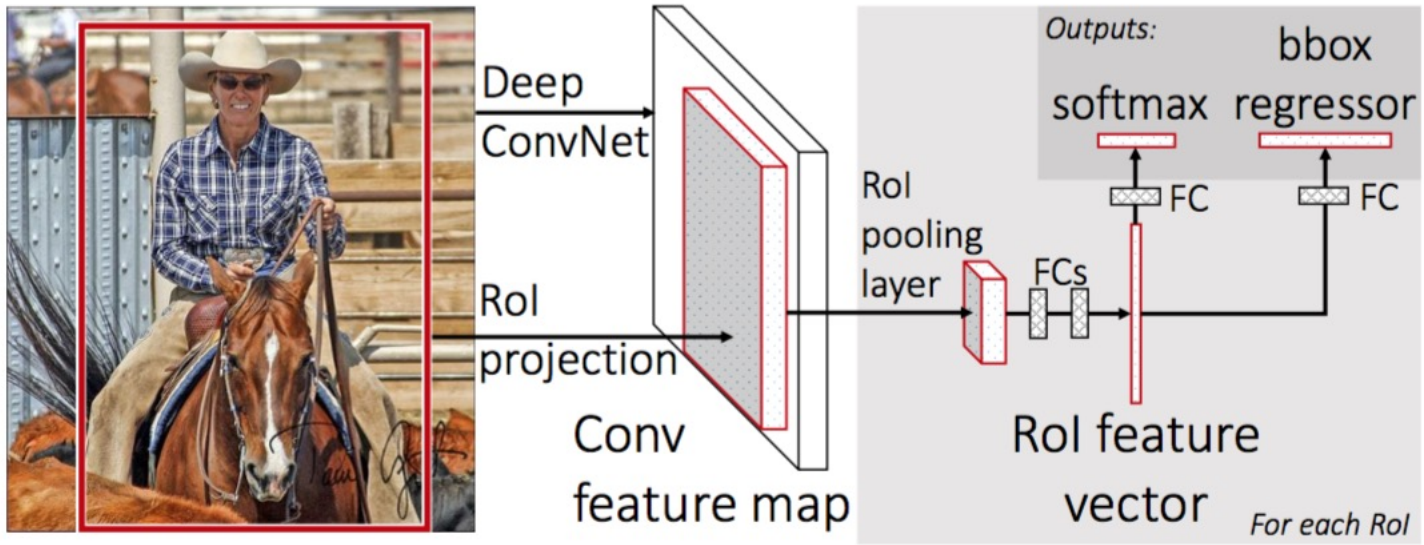
1. Input image

2. Extract region proposals (~2k)

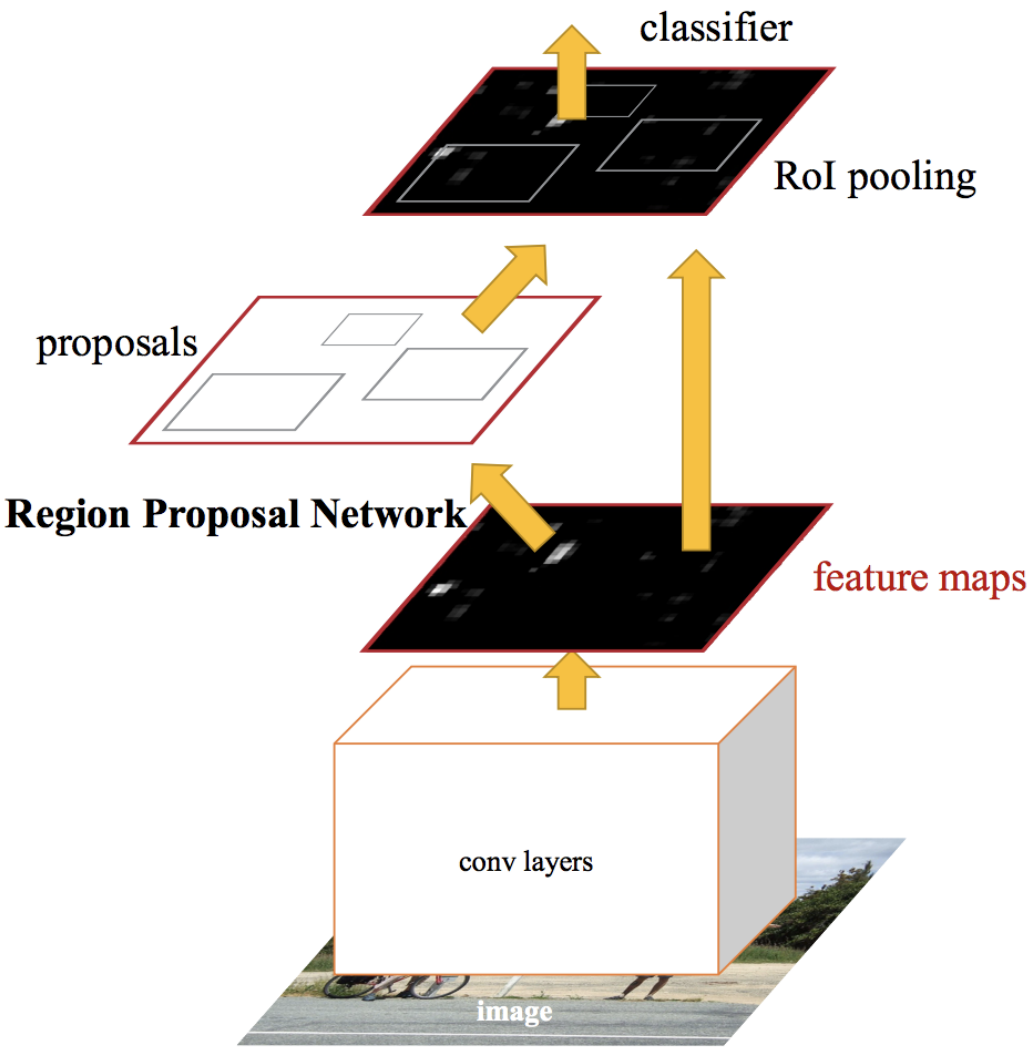
3. Compute CNN features

4. Classify regions

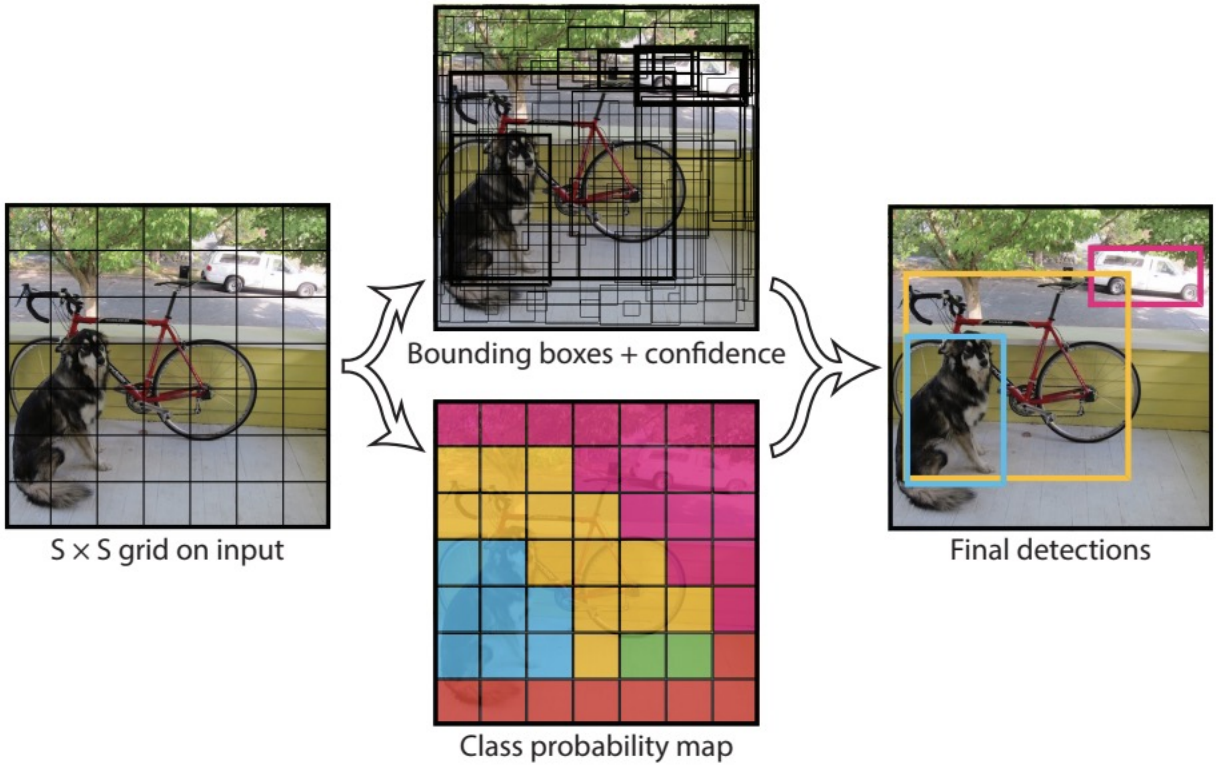
Fast R-CNN


Deep
ConvNet
RoI
projection
Conv
feature map
RoI
pooling
layer
FCs
Outputs:
softmax
bbox
regressor
FC
FC
RoI feature
vector
For each RoI

Faster R-CNN


classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image

YOLO (You Only Look Once)



RetinaNet

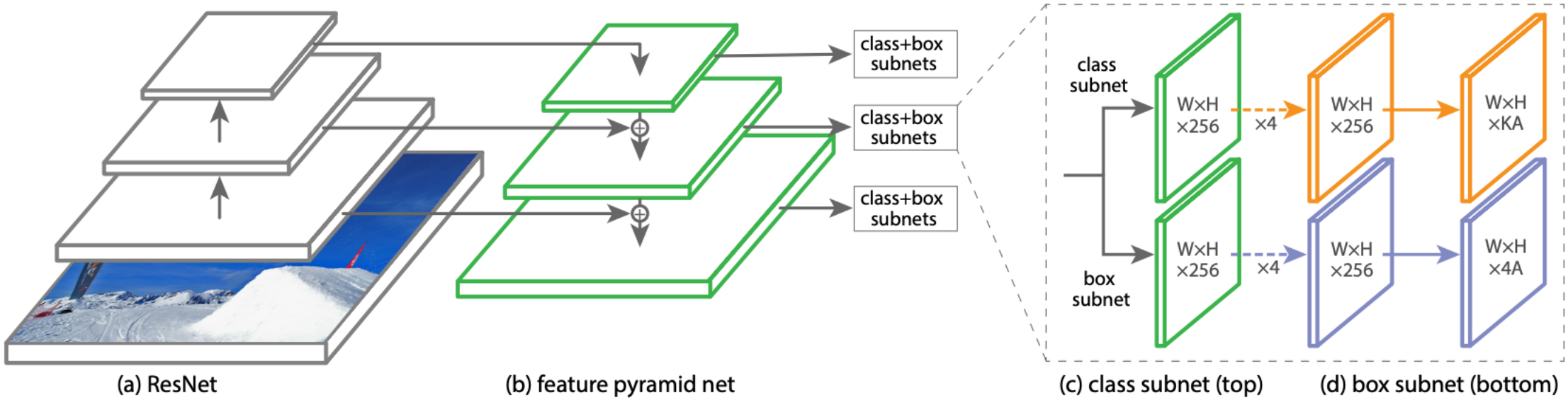


Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [20] backbone on top of a feedforward ResNet architecture [16] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN [20] while running at faster speeds.

Article: https://arxiv.org/pdf/1708.02002.pdf

Source: https://developers.arcgis.com/python/guide/how-retinanet-works/#:~:text=RetinaNet%20is%20one%20of%20the,with%20aerial%20and%20satellite%20imagery.

Object Detection: Dataset

LARXEL · UPDATED 2 YEARS AGO

1353 | New Notebook | Download (417 MB)

Face Mask Detection

853 images belonging to 3 classes.

Data | Code (159) | Discussion (5) | Metadata

About Dataset

Preview

About this Data

Masks play a crucial role in protecting the health of individuals against respiratory diseases, as is one of the few precautions available for COVID-19 in the absence of immunization. With this dataset, it is possible to create a model to detect people wearing masks, not wearing them, or wearing masks improperly.
This dataset contains **853** images belonging to the **3** classes, as well as their bounding boxes in the PASCAL VOC format.
The classes are:

Usability
8.75

License
CC0: Public Domain

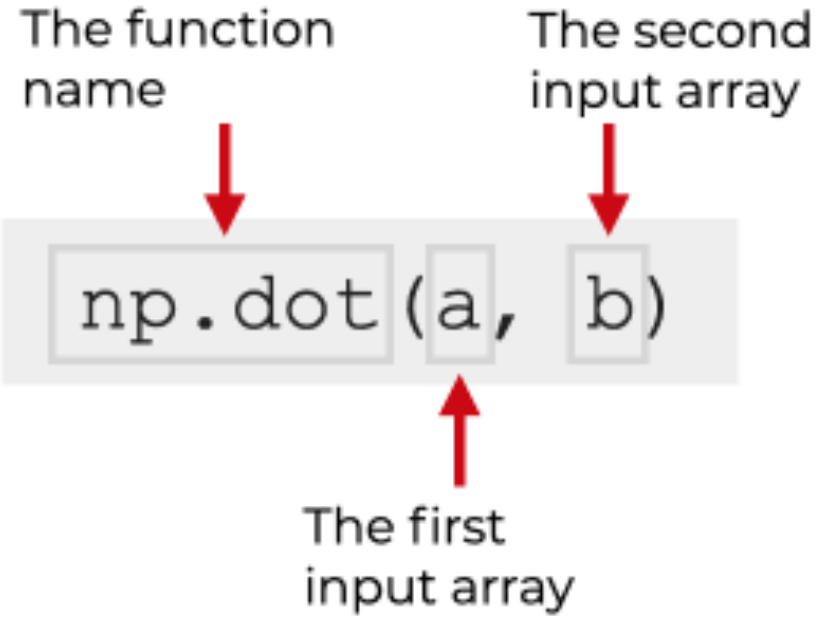
Expected update frequency
Never

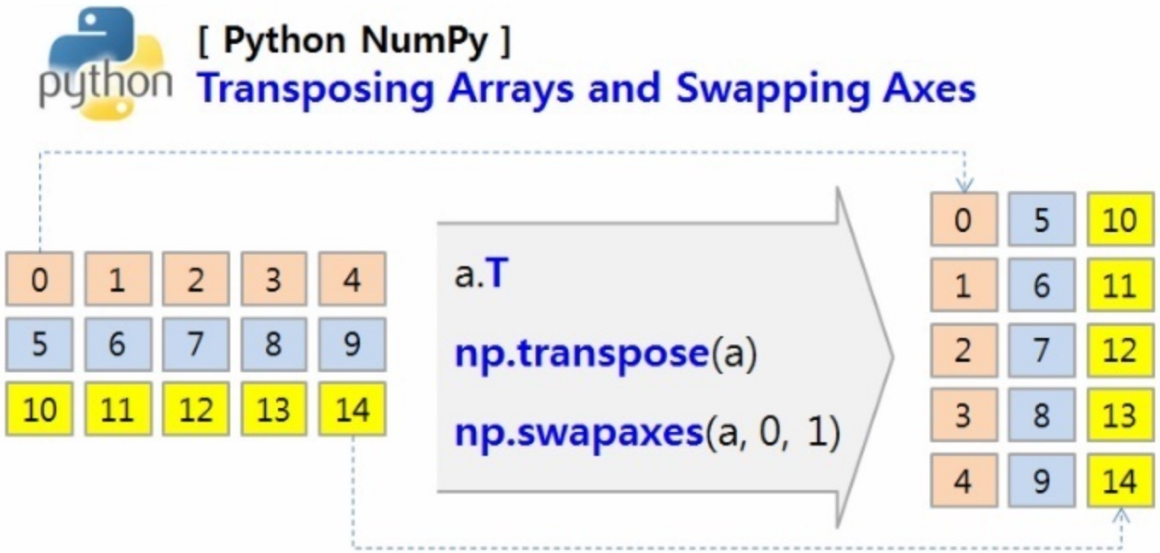
Kaggle dataset: kaggle.com/datasets/andrewmvd/face-mask-detection

NumPy: transpose() / dot()

```
numpy.dot(3, 4)
```

3×4





Any Questions?