DATA LAB

GUARDA AVANTI

Big Data, nuove competenze
per nuove professioni.

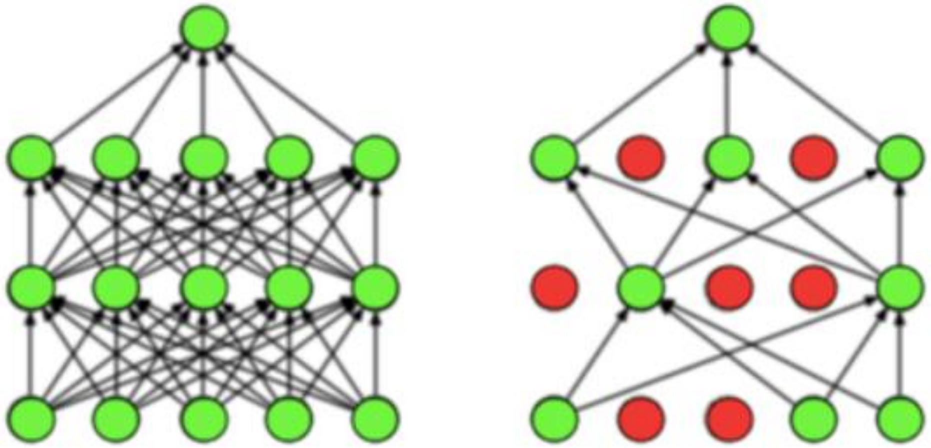
"Anticipare la crescita con le nuove competenze sui Big Data - Edizione 3" Operazione Rif. PA 2021-16029/RER approvata con DGR n° 927 del 21 giugno 2021 e co-finanziata dal Fondo Sociale Europeo PO 2014-2020 Regione Emilia-Romagna


Deep Learning
DATA LAB

Regularization Techniques in Deep Learning

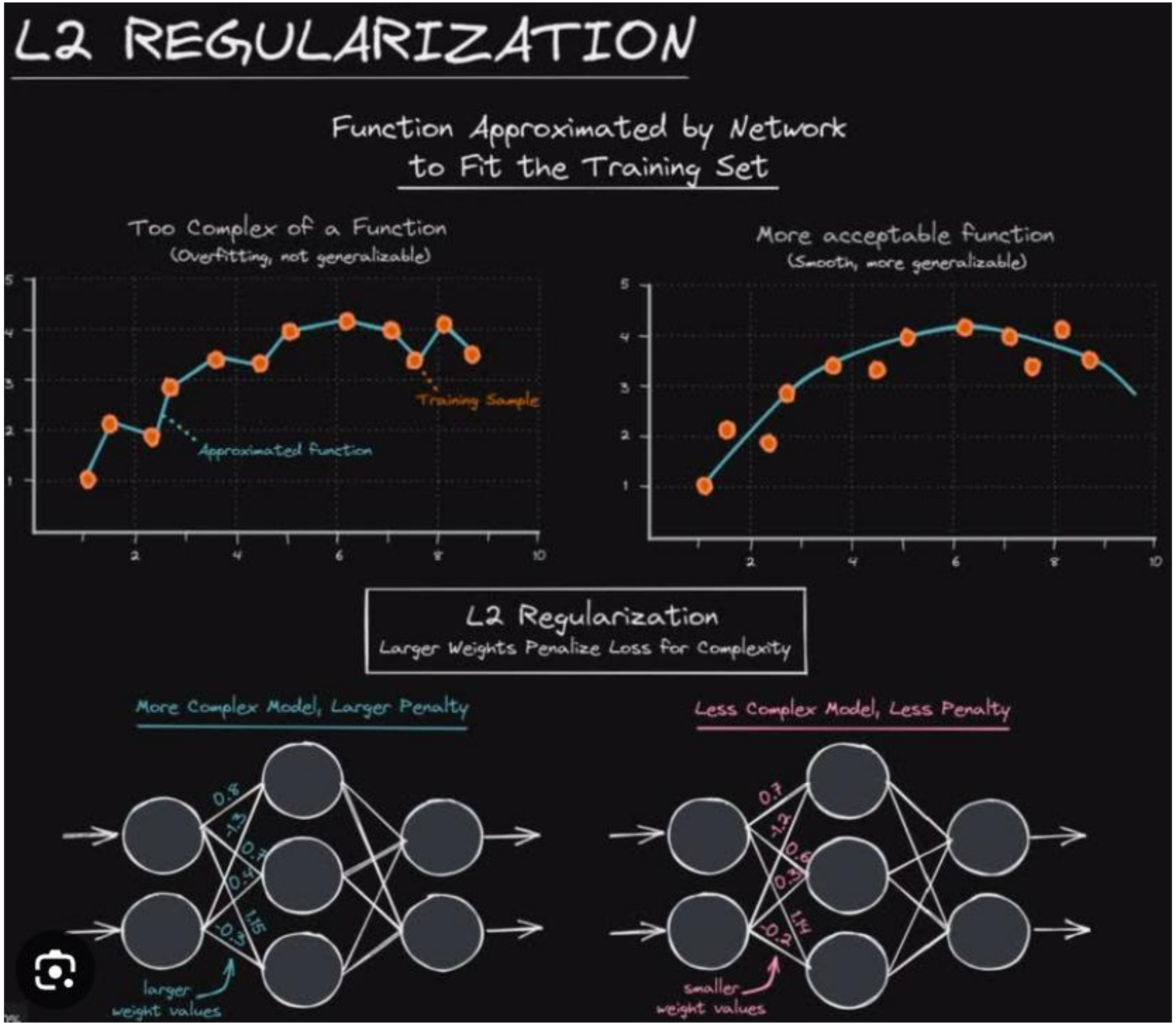
- L1 Regularization
- L2 Regularization
- Early Stopping
- Dropout Regularization
- Training knowledge Augmentation
- Batch standardization

DROPOUT

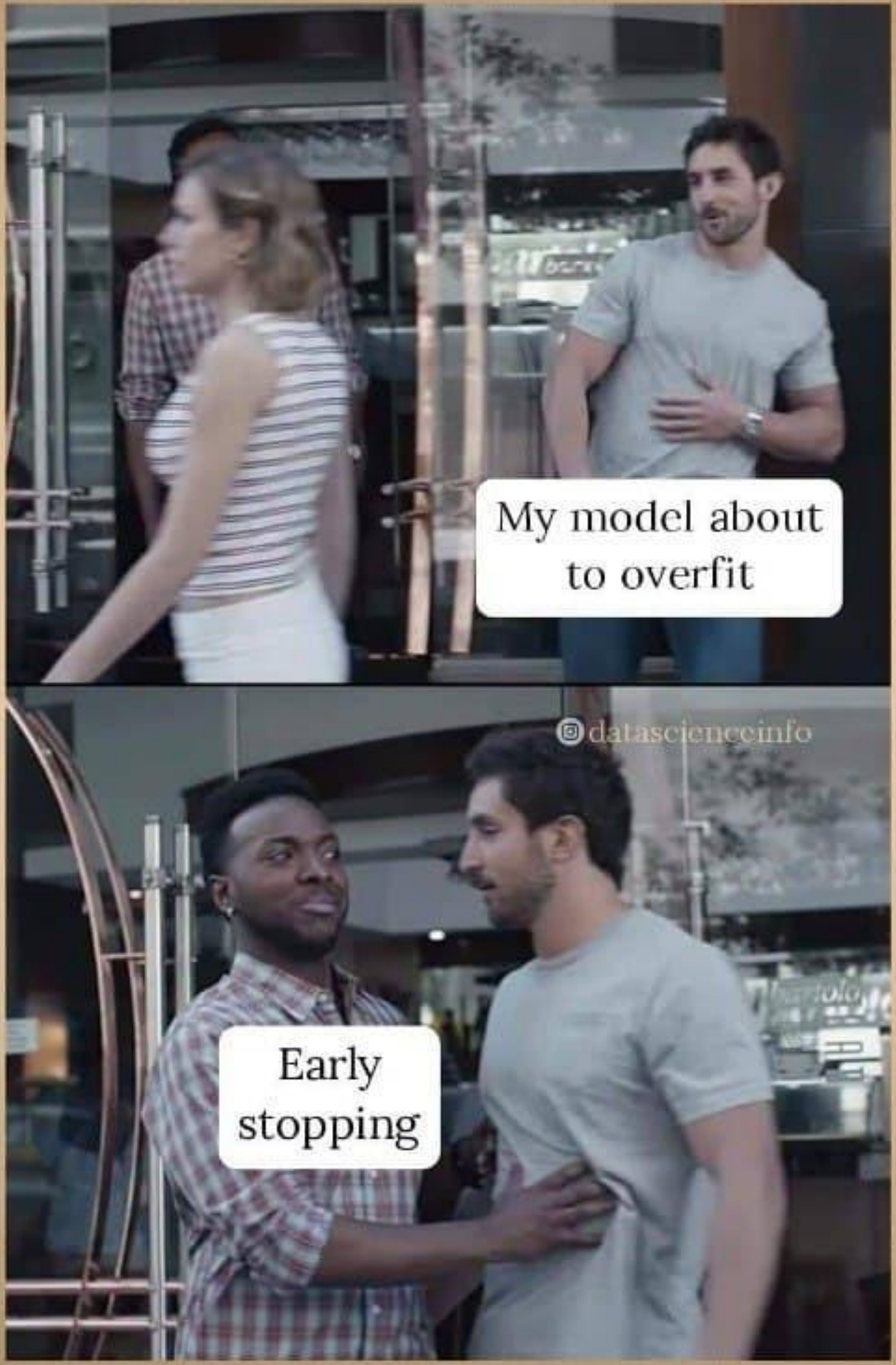


(a) Standard Neural Net

(b) After applying dropout.



EARLY STOPPING



BATCH NORMALIZATION

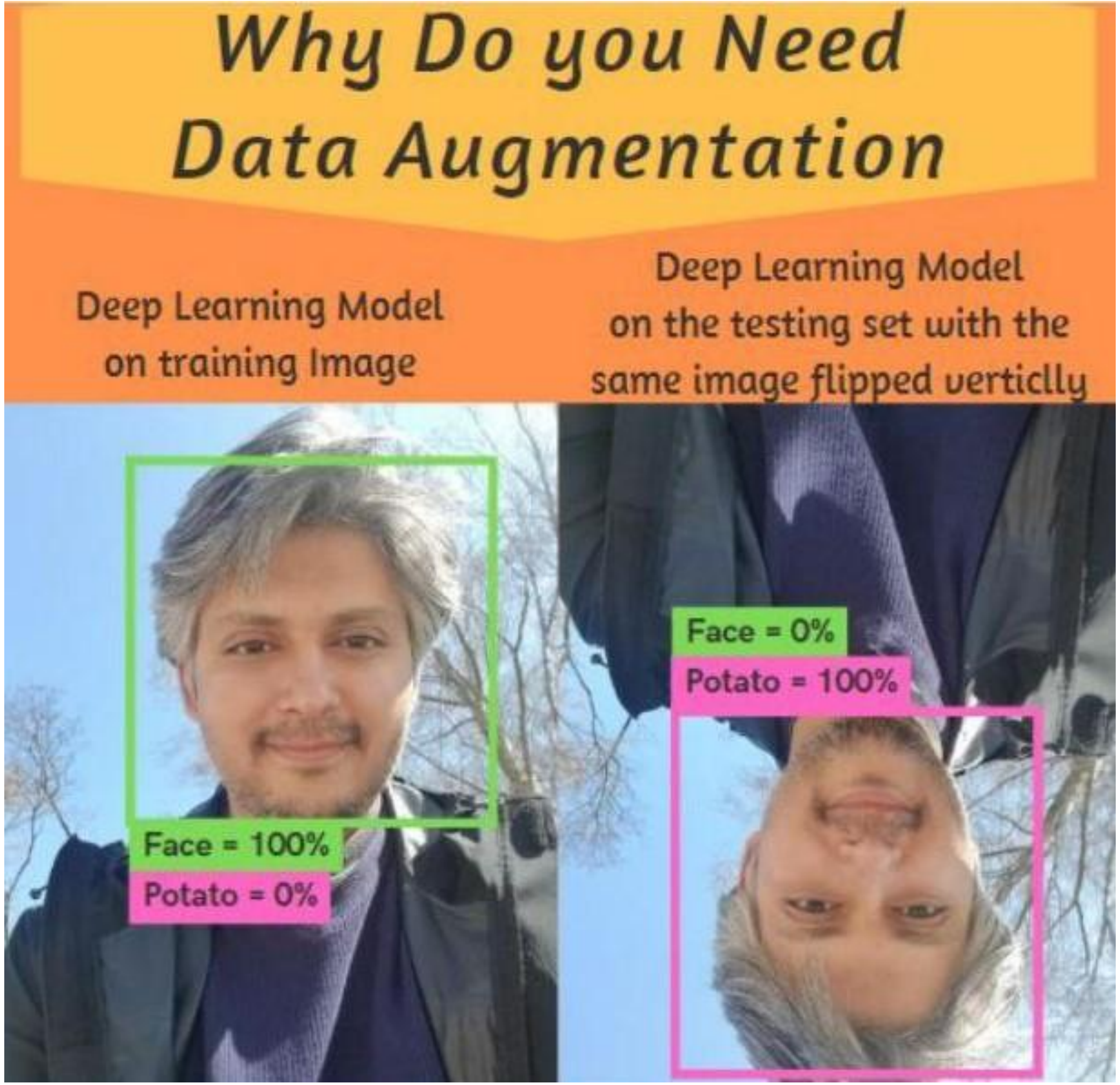
Batch normalization is a technique for improving the performance and stability of neural networks.
The idea is to normalize the inputs of each layer in such a way that they have a mean output activation of zero and standard deviation of one. This is analogous to how the inputs to networks are standardised. We know that normalising the inputs to a network helps it learn. But a network is just a series of layers, where the output of one layer becomes the input to the next. We normalise the output of one layer before applying the activation function, and then feed it into the following layer

Benefits:

- **Networks train faster:** The converge is more quickly.
- **Allows higher learning rates:** Gradient descent works with small learning rates for the network to converge. As networks get deeper, gradients get smaller during back propagation, and require even more iterations. So with batch normalization we are increasing the speed of the training.
- **Makes weights easier to initialize**
- **Simplifies the creation of deeper neural networks**
- **Provides some extra regularization:** In some cases work as well as dropout.
- **Makes more activation functions viable** regulating the values going into each activation function, nonlinearities that don't work well in deep networks tend to become viable again.

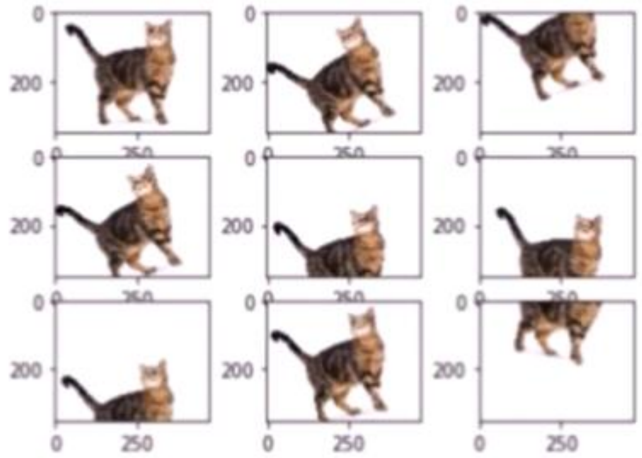
```
from keras.layers.normalization import BatchNormalization
model = Sequential()

model.add(Dense(units = 64, input_shape=(n_inputs,))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(units = 64))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(units = n_classes))
model.add(BatchNormalization())
model.add(Activation('softmax'))

model.compile(loss='binary_crossentropy', optimizer='adam')

model.fit(X_train, y_train, nb_epoch=50, batch_size=32)
```

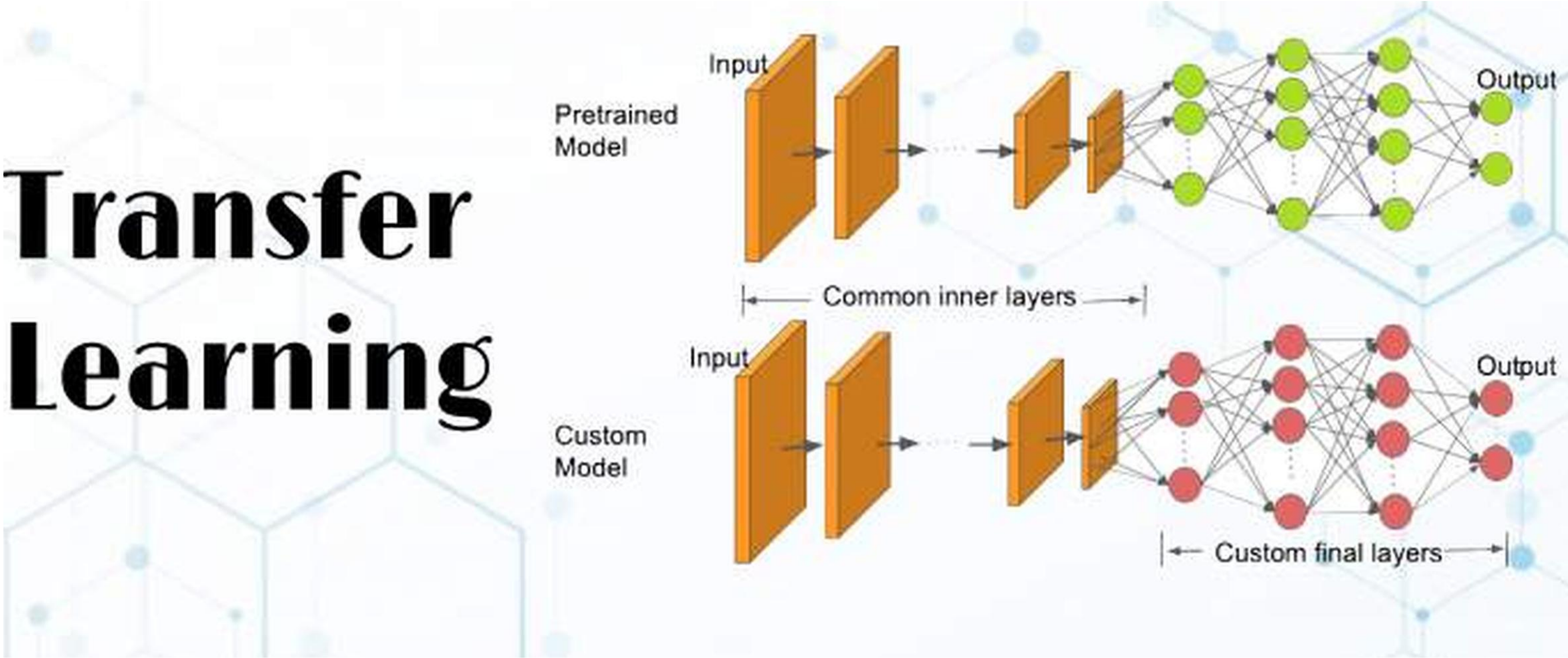

I SEE
BAD DATA


Why Do you Need
Data Augmentation
Deep Learning Model
on training Image
Deep Learning Model
on the testing set with the
same image flipped verticlly
Face = 100%
Potato = 0%
Face = 0%
Potato = 100%

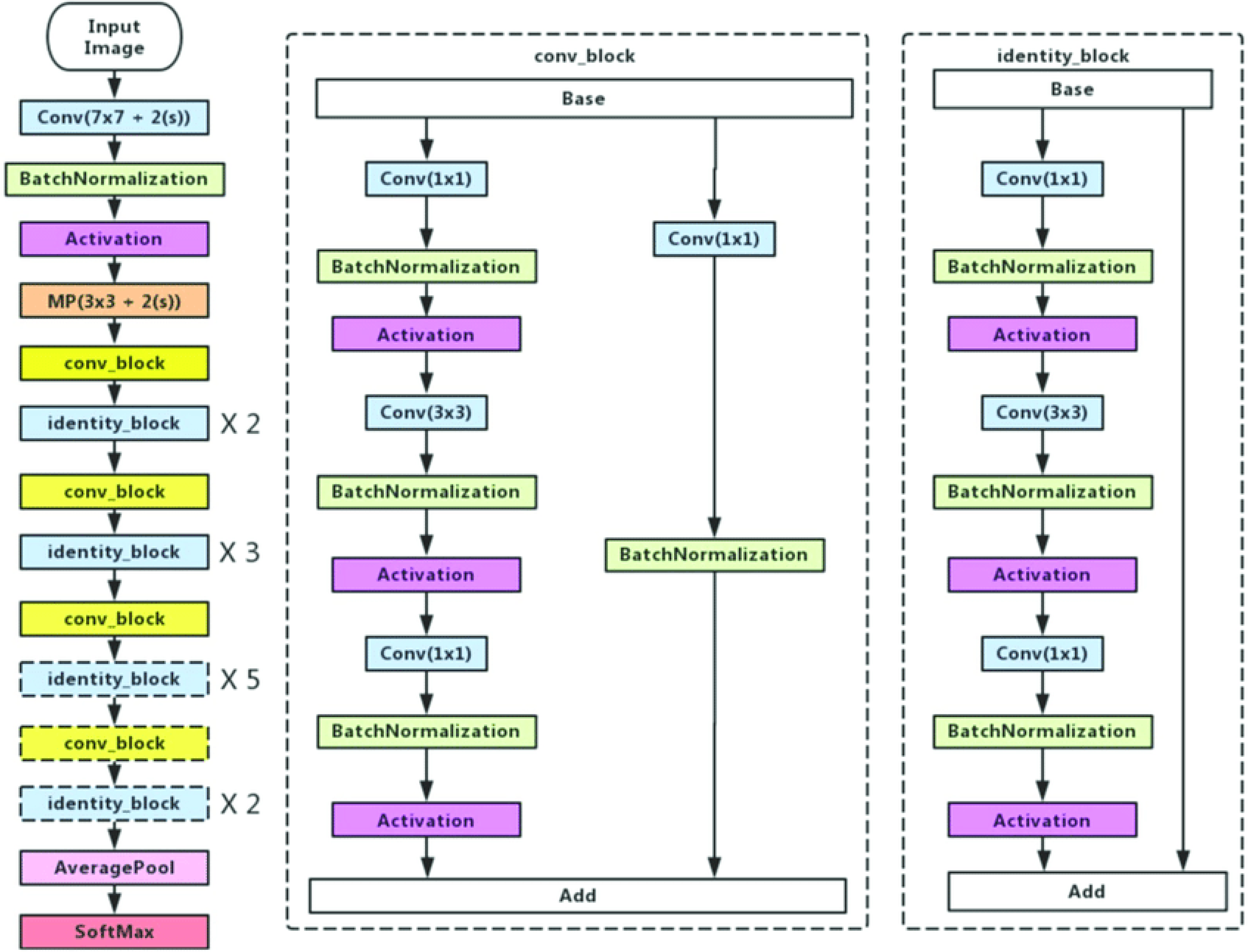
Data Augmentation

Flip: You can flip images horizontally and vertically.
Rotation: One key thing to note about this operation is that image dimensions may not be preserved after rotation.
Scale: The image can be scaled outward or inward.
Crop: Unlike scaling, we just randomly sample a section from the original image. We then resize this section to the original image size.
Translation: Translation just involves moving the image along the X or Y direction (or both)
Gaussian Noise: Over-fitting usually happens when your neural network tries to learn high frequency features (patterns that occur a lot) that may not be useful. Gaussian noise, which has zero mean, essentially has data points in all frequencies, effectively distorting the high frequency features.




Transfer
Learning
Input
Pretrained
Model
Output
Common inner layers
Input
Custom
Model
Output
Custom final layers

Transfer learning be like

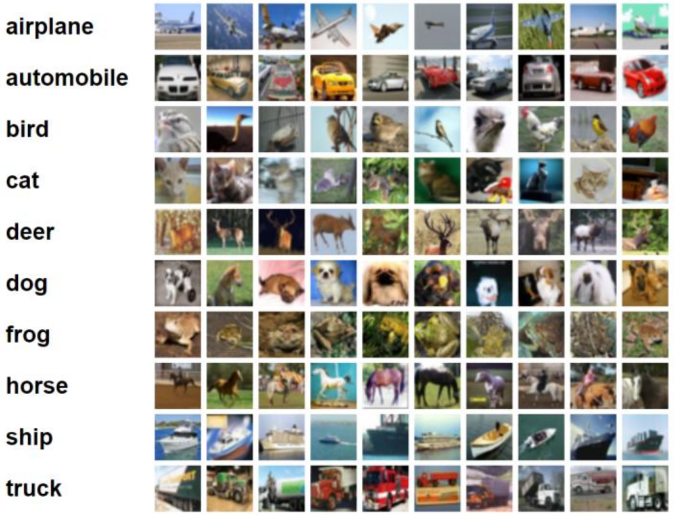

Input Image
Conv(7x7 + 2(s))
BatchNormalization
Activation
MP(3x3 + 2(s))
conv_block
identity_block
X 2
conv_block
identity_block
X 3
conv_block
identity_block
X 5
conv_block
identity_block
X 2
AveragePool
SoftMax
conv_block
Base
Conv(1x1)
BatchNormalization
Activation
Conv(3x3)
BatchNormalization
Activation
Conv(1x1)
BatchNormalization
Activation
Conv(1x1)
BatchNormalization
Add
identity_block
Base
Conv(1x1)
BatchNormalization
Activation
Conv(3x3)
BatchNormalization
Activation
Conv(1x1)
BatchNormalization
Activation
Add

The CIFAR-10 dataset

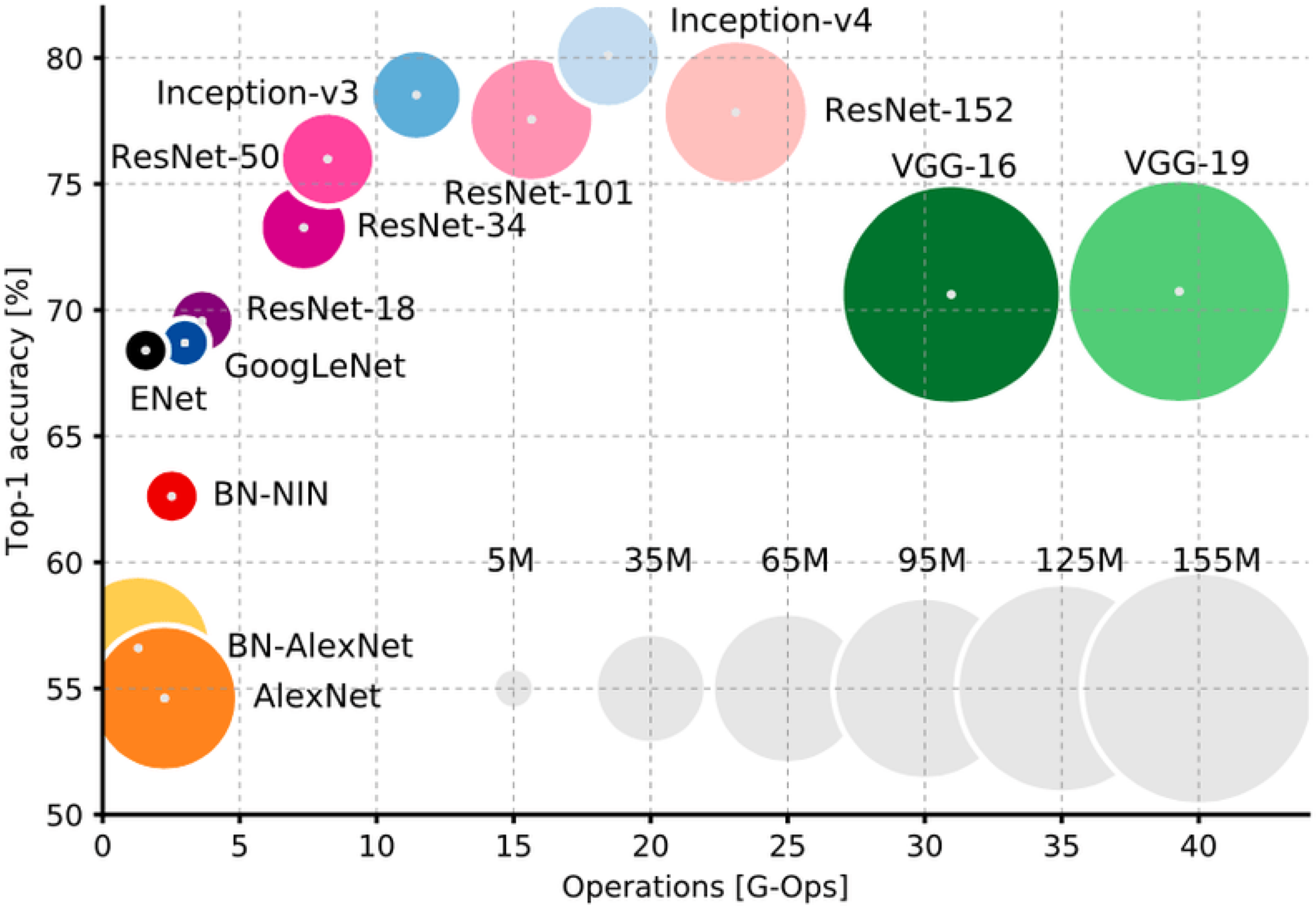
The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test bat
order, but some training batches may contain more images from one class than another. Between them, t

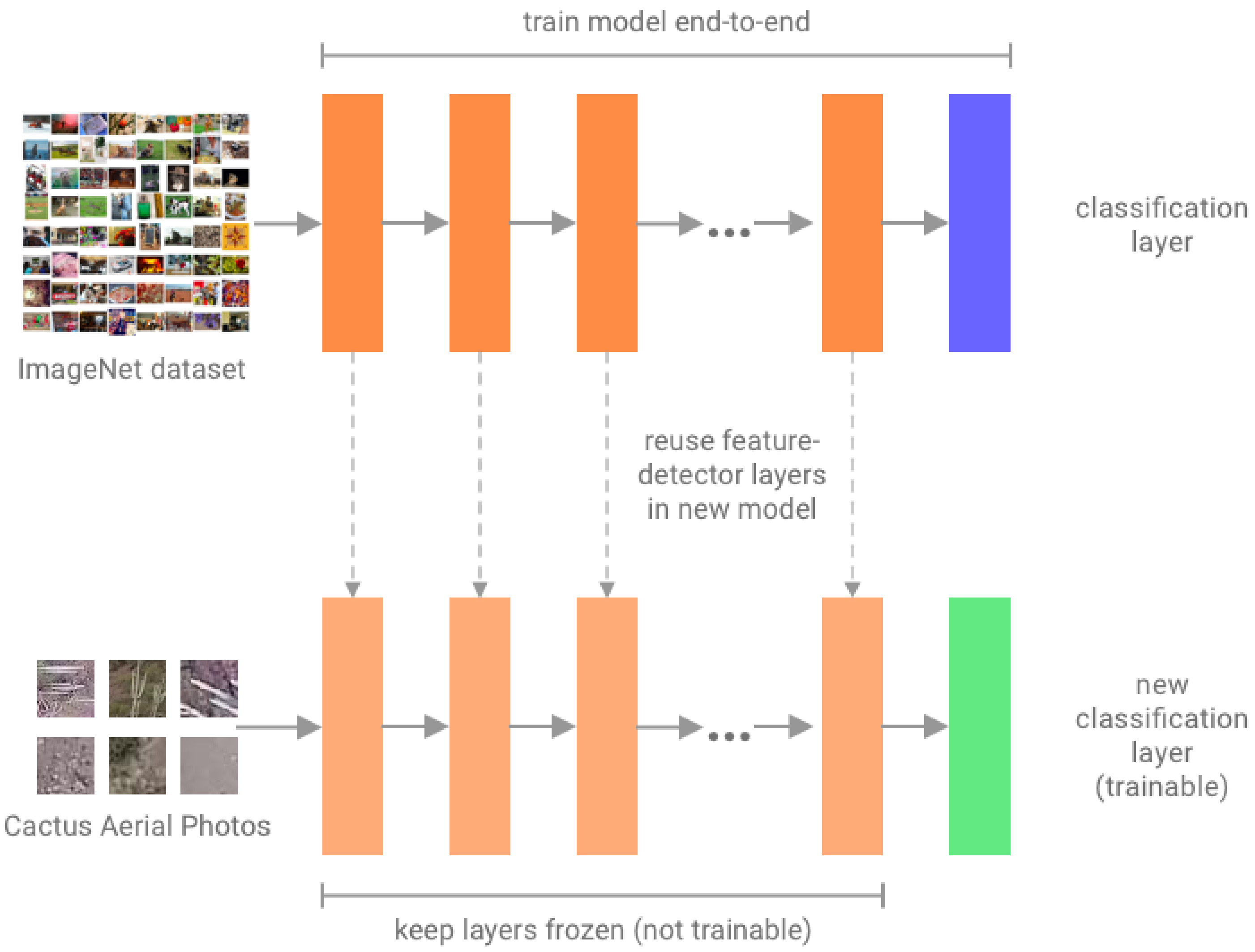
Here are the classes in the dataset, as well as 10 random images from each:

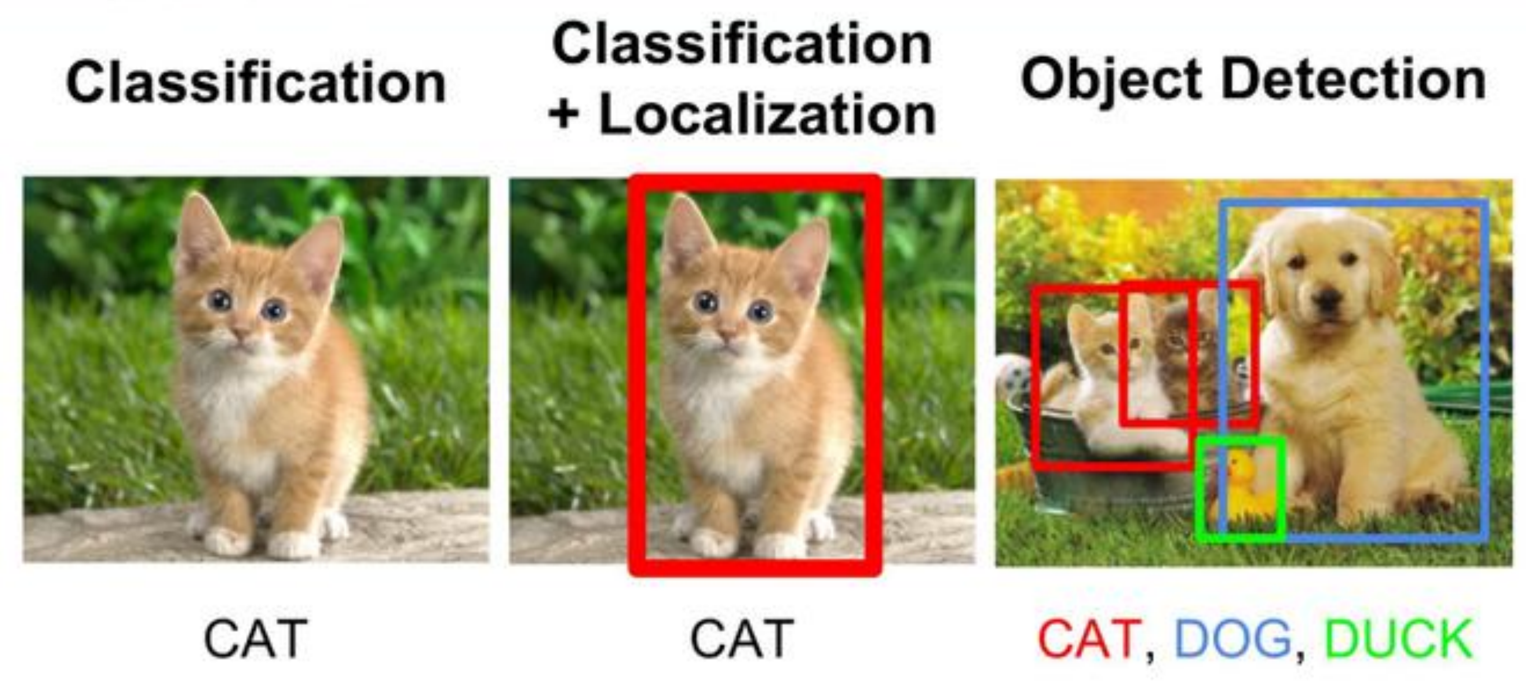


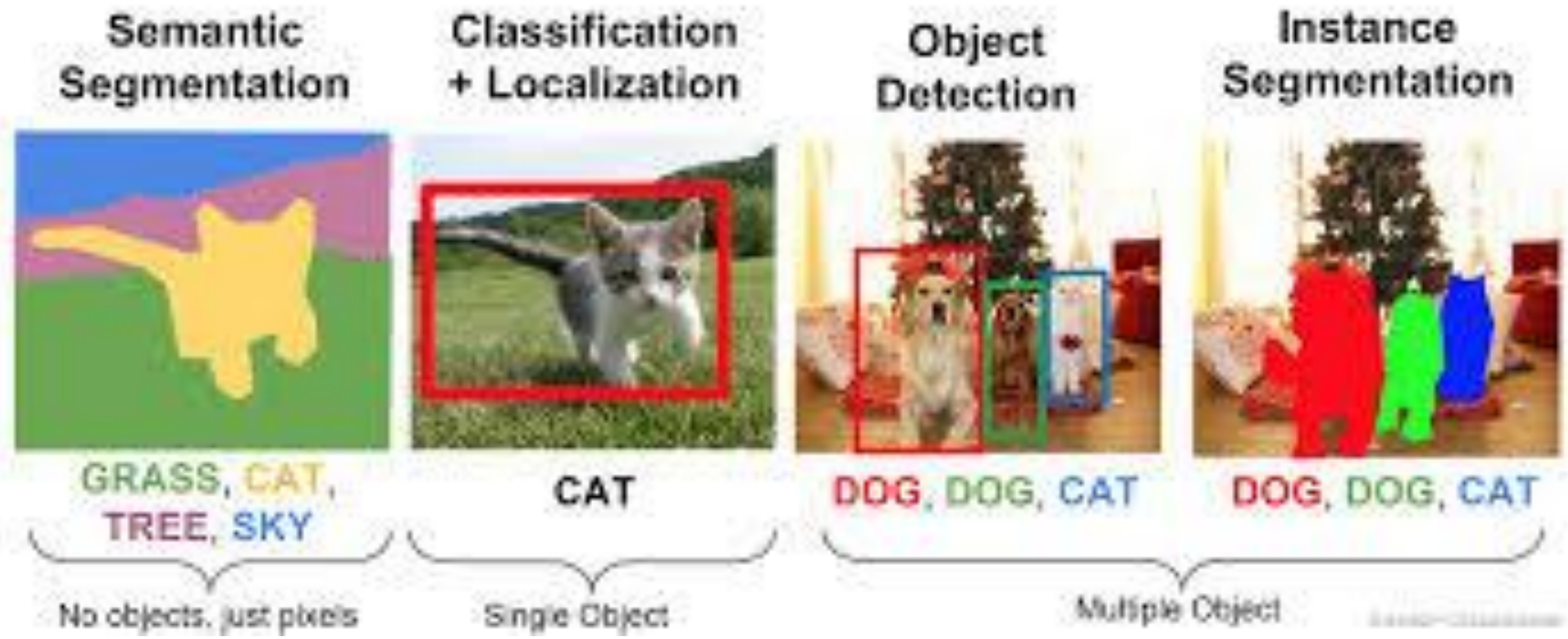
COCO Dataset | nk7260ynpa 20190406



Inception-v4
Inception-v3
ResNet-152
ResNet-50
ResNet-101
VGG-16
VGG-19
ResNet-34
ResNet-18
GoogLeNet
ENet
BN-NIN
5M
35M
65M
95M
125M
155M
BN-AlexNet
AlexNet
Top-1 accuracy [%]
80
75
70
65
60
55
50
0
5
10
15
20
25
30
35
40
Operations [G-Ops]

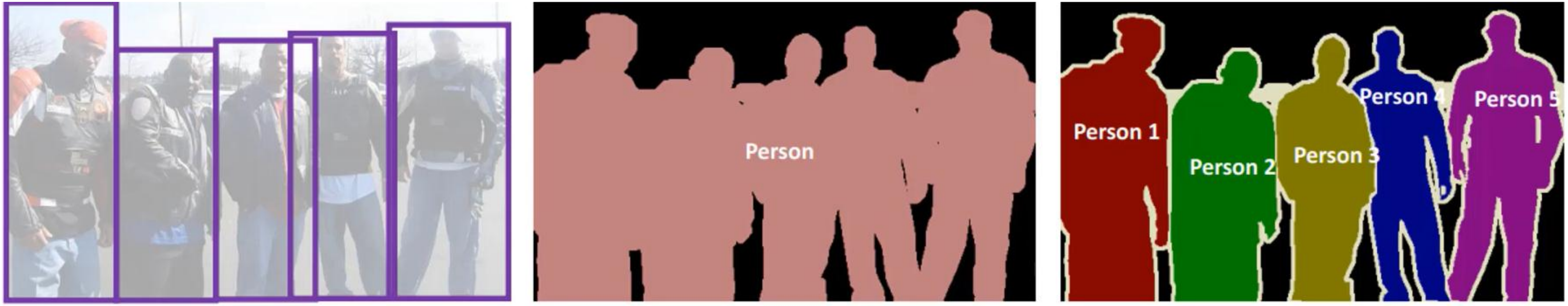

train model end-to-end
classification layer
ImageNet dataset
reuse feature-detector layers in new model
new classification layer (trainable)
Cactus Aerial Photos
keep layers frozen (not trainable)


Classification
Classification
+ Localization
Object Detection
CAT
CAT
CAT, DOG, DUCK


Semantic
Segmentation
Classification
+ Localization
Object
Detection
Instance
Segmentation
GRASS, CAT,
TREE, SKY
CAT
DOG, DOG, CAT
DOG, DOG, CAT
No objects, just pixels
Single Object
Multiple Object

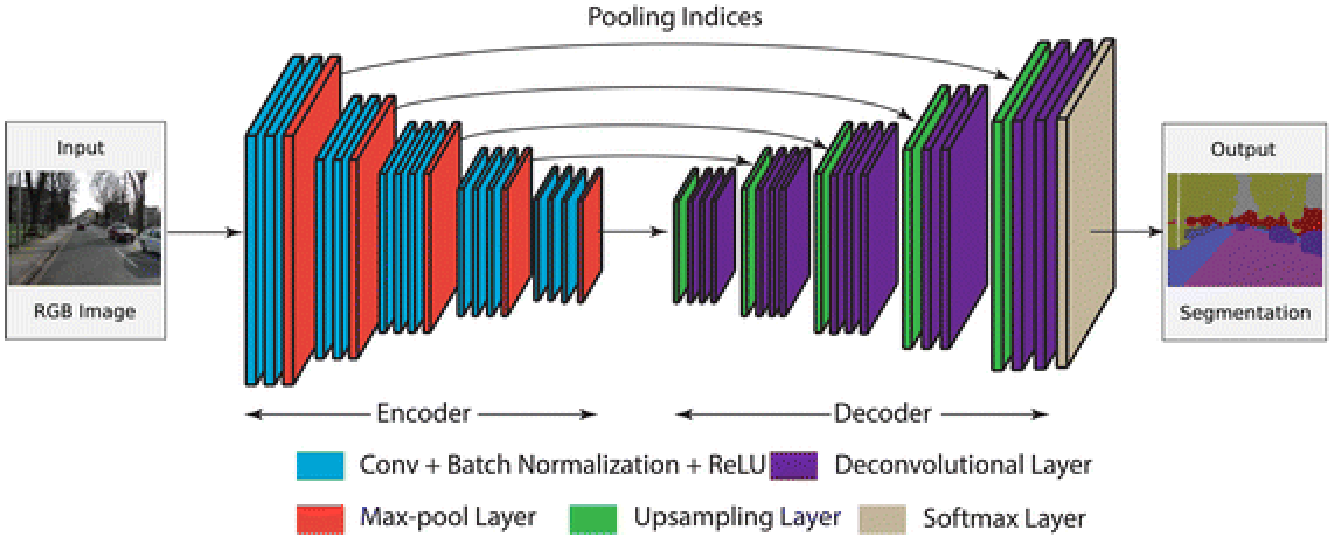
Object Detection

SEGMENTATION

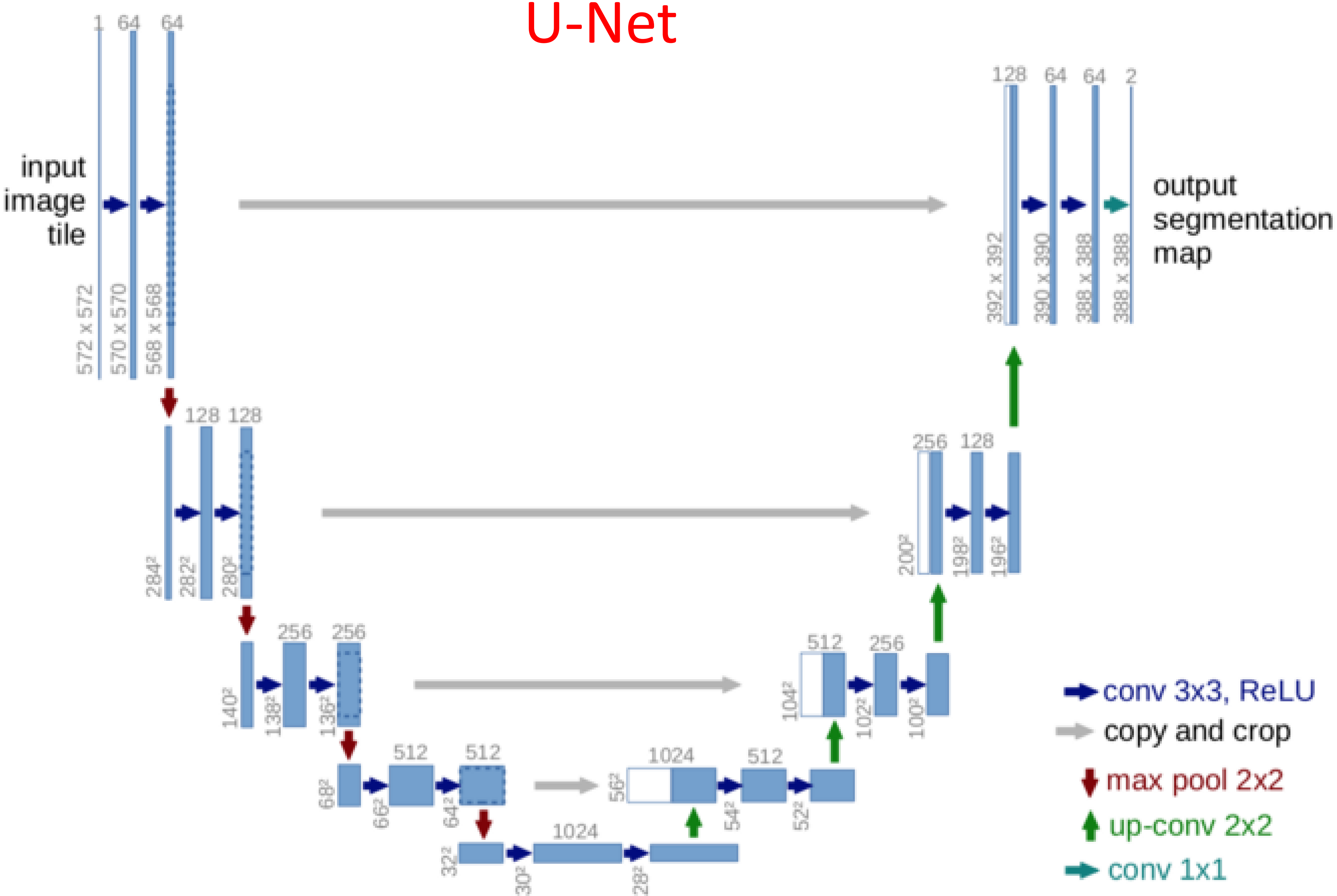


Object Detection Semantic Segmentation **Instance Segmentation**

CONVOLUTIONAL ENCODER-DECODER



U-Net



input
image
tile
output
segmentation
map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

DETECTRON 2