

# CNN for object detection

Dominik Lewy

## Agenda

1. What is object detection?
2. Traditional approach
3. R-CNN
4. Other methods
5. Transfer Learning
6. Experiments
7. Future plans

## Object Detection – what is it

True Image



Semantic Segmentation



**CAR, BACKGROUND**

- No objects, just pixels

# Object Detection – what is it

## Classification + Localization

CAR



- Single object

## Object Detection



- Multiple objects

## Object Detection – what is it

### Instance Segmentation



- Multiple objects

CAR

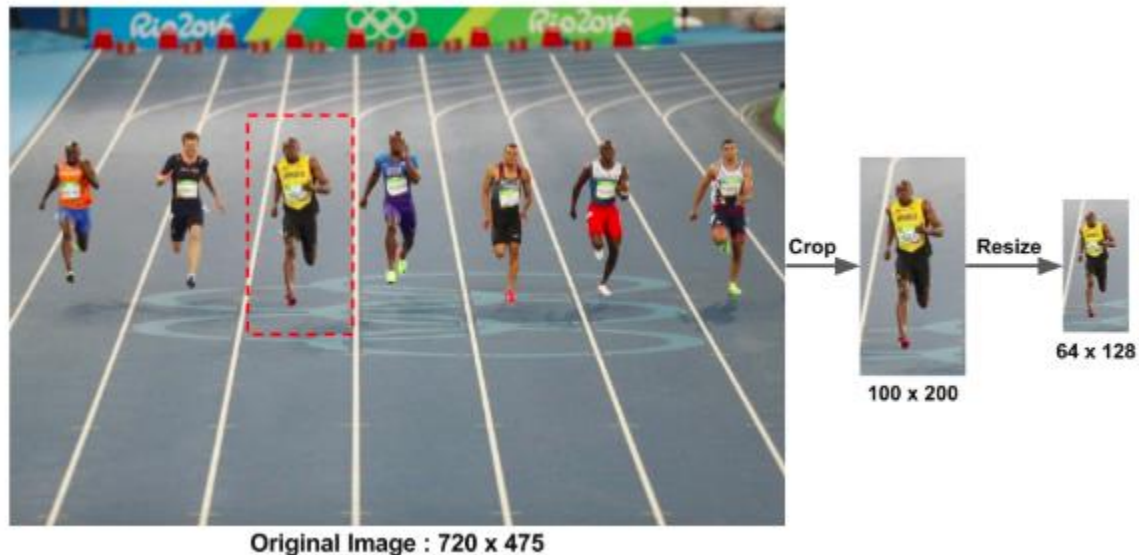
## Object Detection – history – HOG

### Histogram of Oriented Gradients

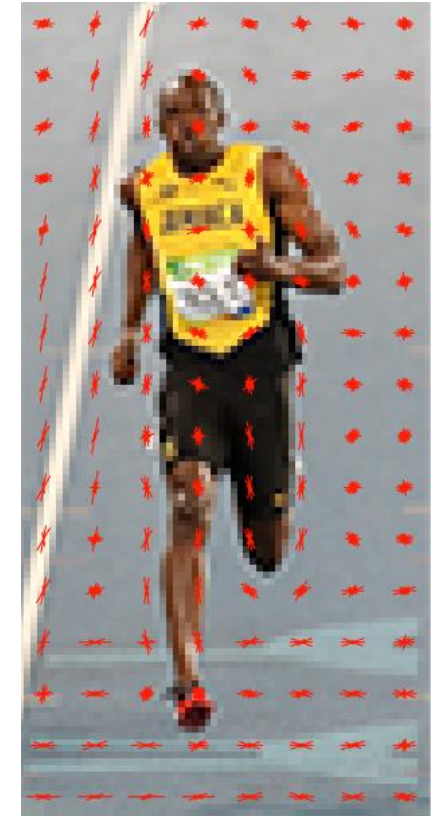
Method of feature selection consisting in the following steps:

1. Gradient calculation
2. Calculation of gradient magnitude and direction
3. Histogram creation
4. Block normalization

#### Input image



#### Histograms



## Object Detection – history – HOG

### 1. Gradient calculation

Gradient calculation in case of images is equivalent to filtering the image with the following filters, which are actually called Sobel filters.

#### Sobel filters

			-1
-1	0	1	0
			1

#### Gradients in x and y directions



Left : Absolute value of x-gradient. Center : Absolute value of y-gradient.  
Right : Magnitude of gradient.

## Object Detection – history – HOG

### 2. Calculation of gradient magnitude and direction

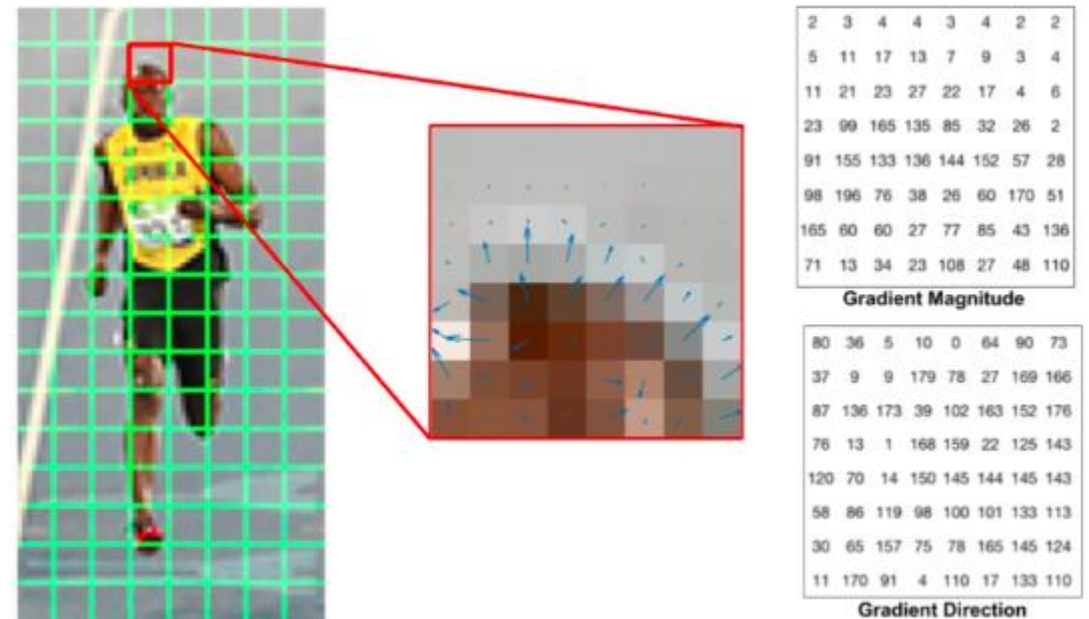
This step unifies the information coming from gradients in x and y directions providing the strength and the direction of the gradient.

#### Magnitude and direction equations

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan \frac{g_y}{g_x}$$

#### Gradients magnitude and direction



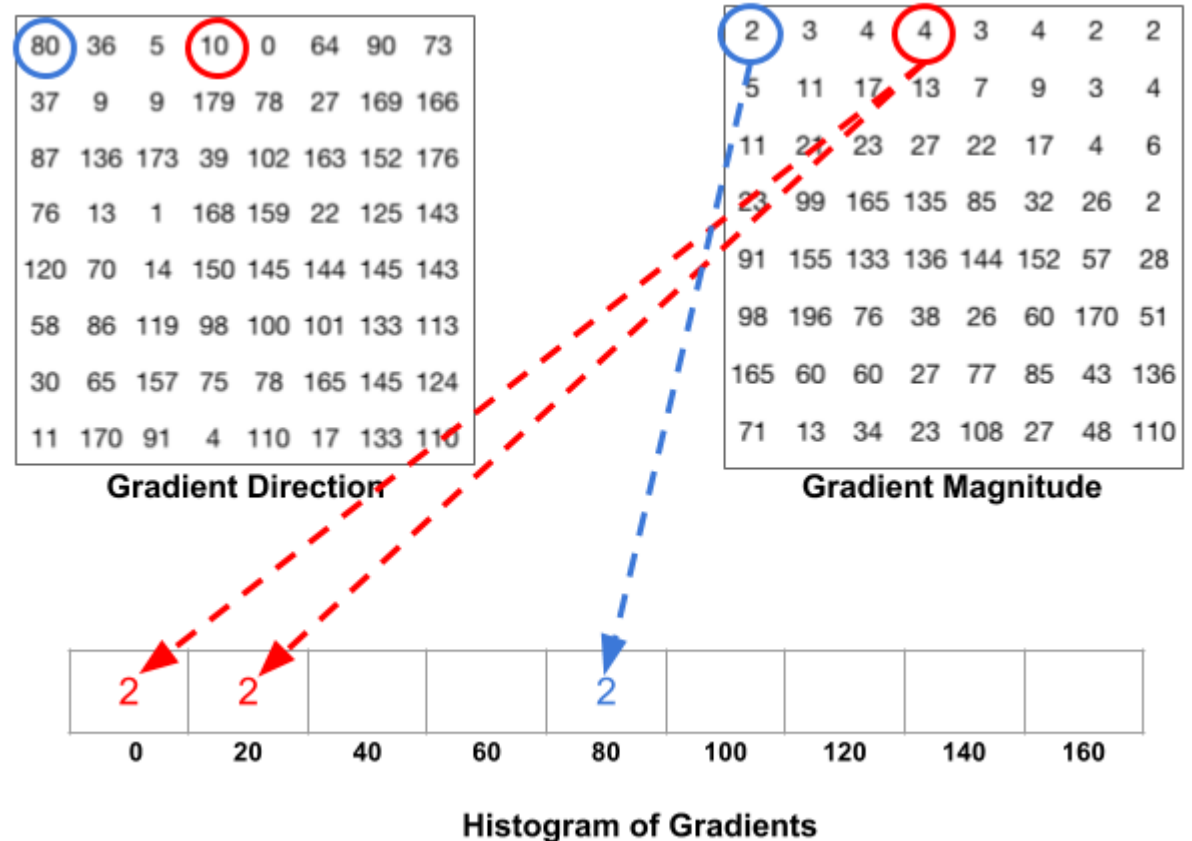


## Object Detection – history – HOG

### 3. Histogram creation

This step is the clue of the HOG method. Here the gradients of the image are transformed into a histogram. This step aims at:

- Making the representation more compact
- Denoising the representation



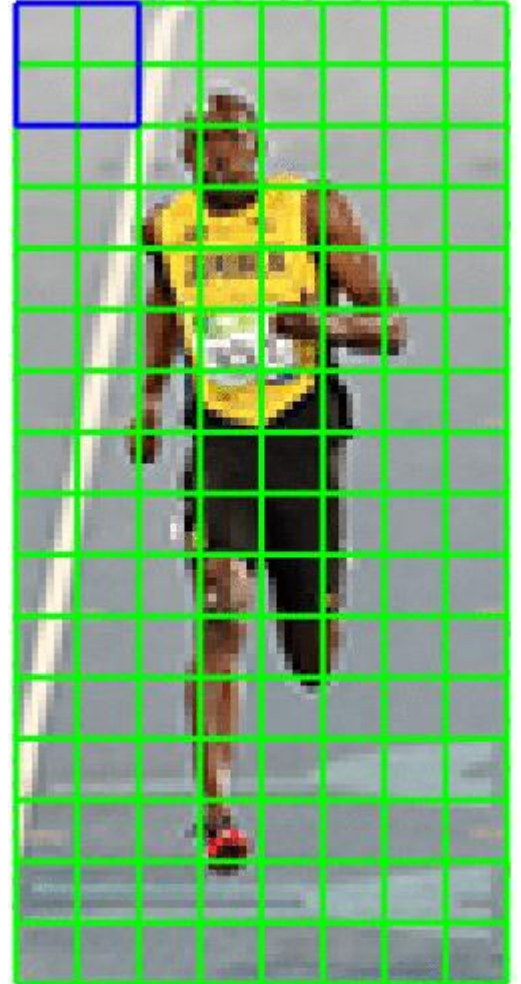
## Object Detection – history – HOG

### 4. Block normalization

Concatenating vectors for all of the histograms and normalizing them with the L2 norm (vector length).

### Summary/key takeaways:

- Both the cell size and block size are hyperparameters that can be optimized

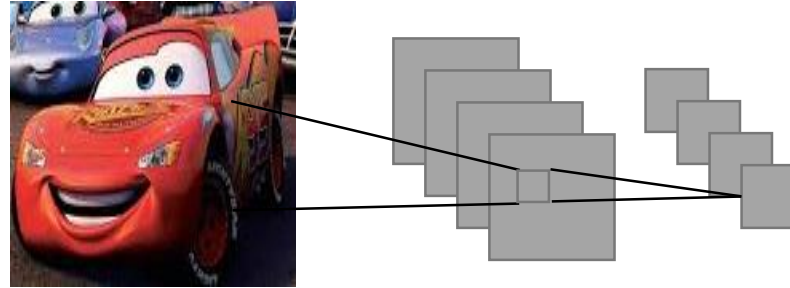


# Object Detection – regional convolutional neural networks

## 1. Input Image



## 3. CNN feature computation



## 2. Region of Interest proposal



## 4. Classification of regions

Airplane? No.

Car? Yes

Person? No

# CNN architectures

## History of CNN for classification - AlexNet

Architecture:

CONV1

MAX POOL1

NORM1

CONV2

MAX POOL2

NORM2

CONV3

CONV4

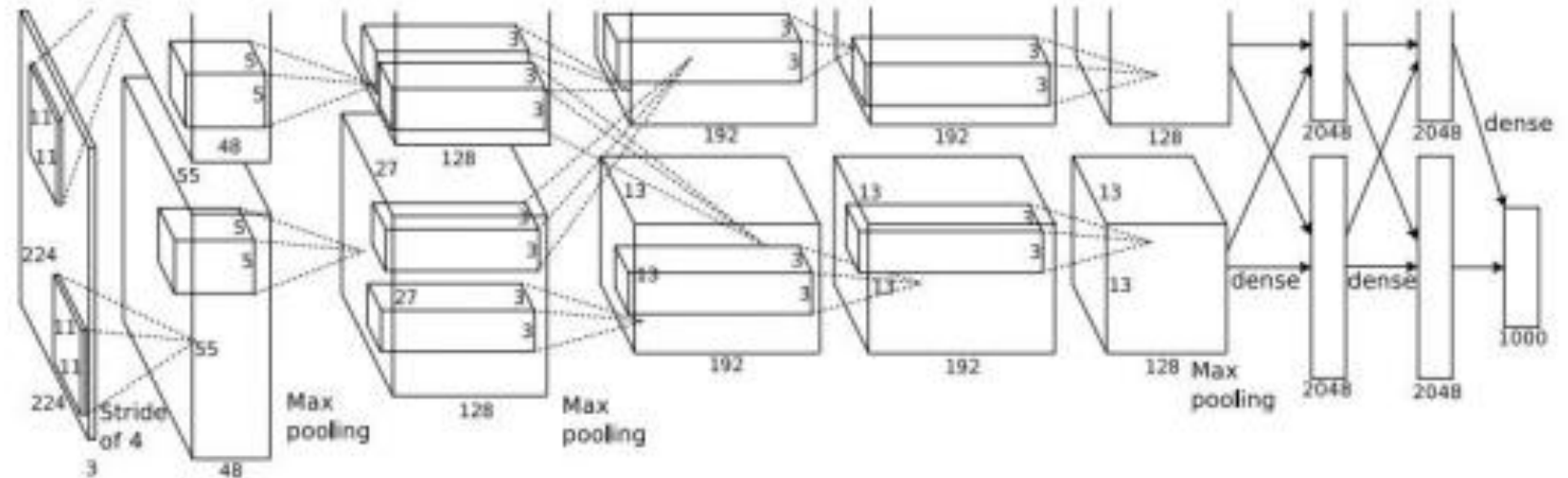
CONV5

Max POOL3

FC6

FC7

FC8



Interesting details:

- First use of ReLU
- Heavy data augmentation
- Dropout 0.5
- Batch size
- SGD Momentum
- Adaptive Learning rate
- L2 weight decay
- 7 Ensembles

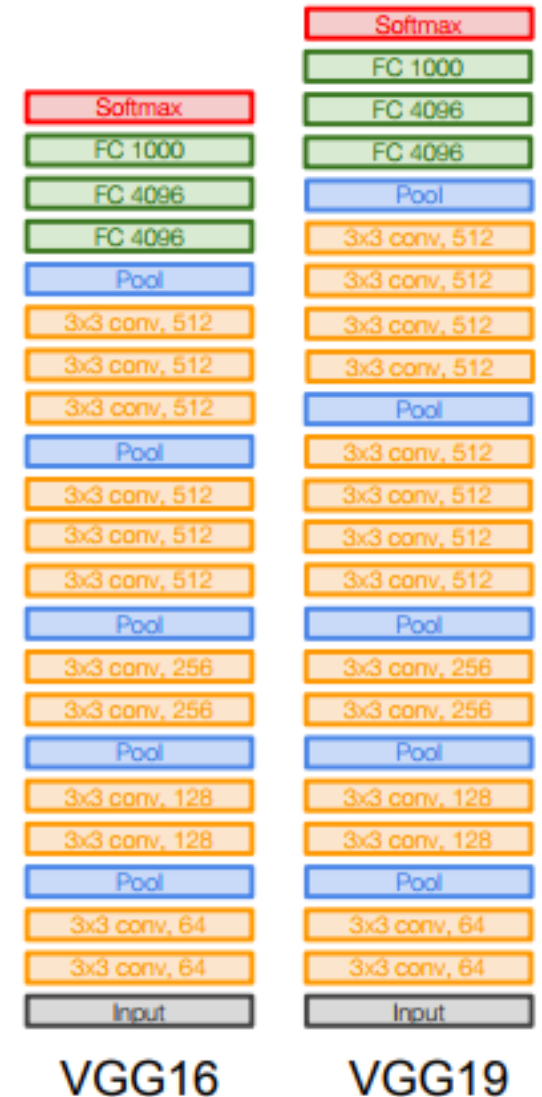
## History of CNN for classification - VGG Net

Goal:

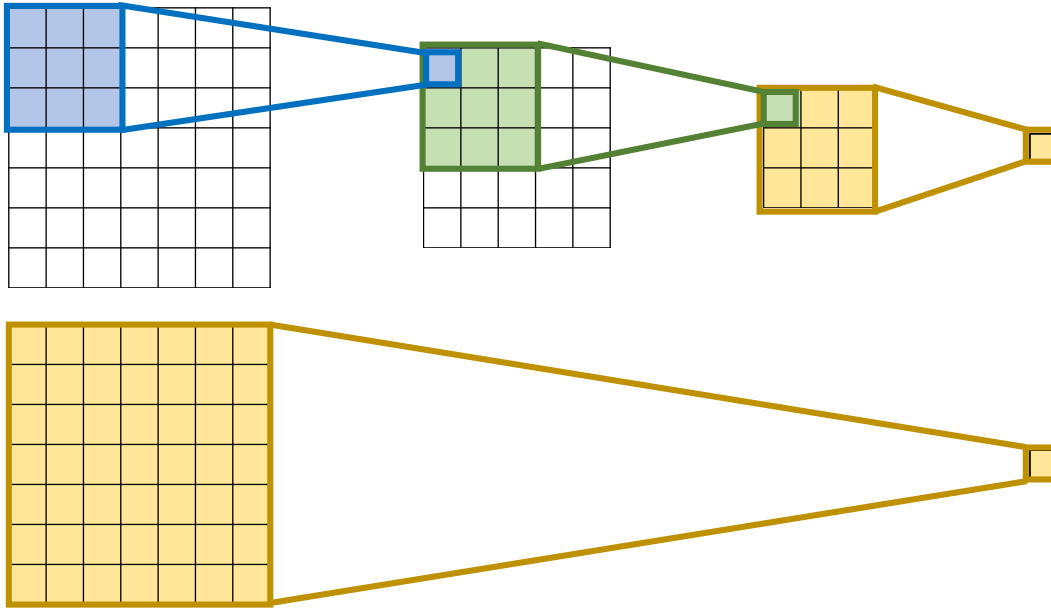
- Building deeper network with smaller filters

Motivation:

- Stacking multiple small filters can:
  - have the same effective receptive field as one bigger filter
  - have much less parameters
  - introduce more nonlinearities



## History of CNN for classification - VGG Net

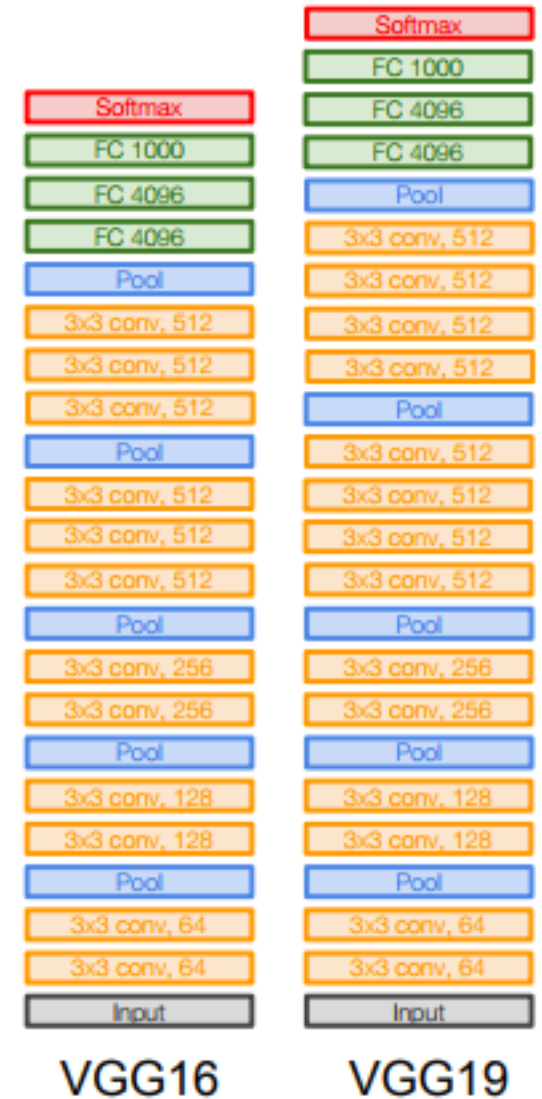


### Parameter number:

$$3 \cdot (3 \times 3) - 3 \cdot 3^2 = 27$$

$$(7 \times 7) - 7^2 = 49$$

This difference grows with the number of filters in each layer and the number of channels.



## History of CNN for classification - GoogleNet

Goal:

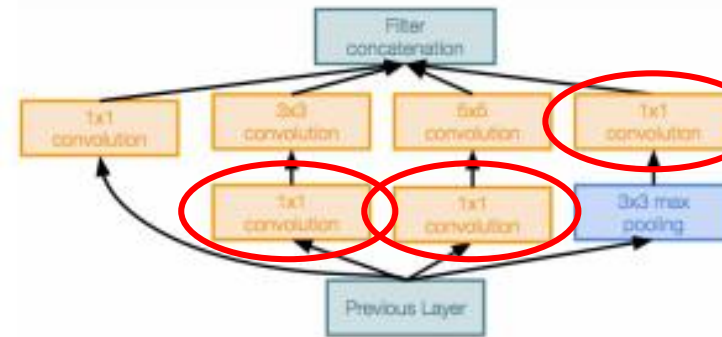
- Computational efficiency

How did they achieve it:

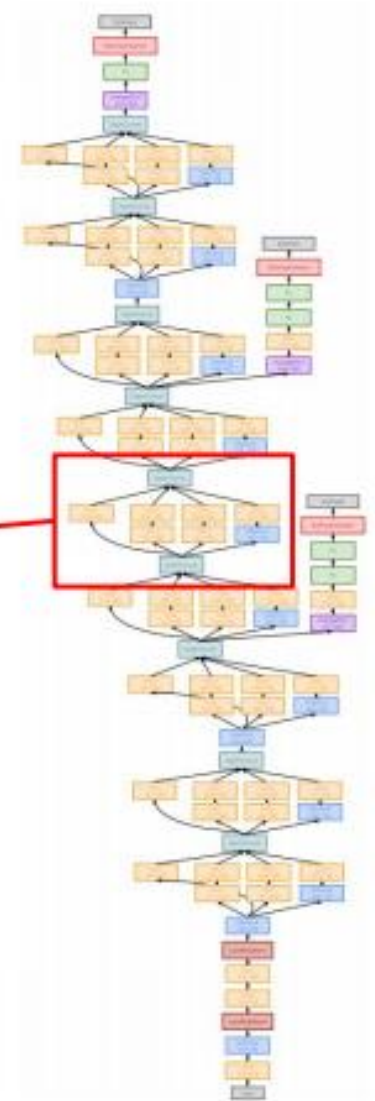
- Inception module
- No fully connected layers

Inception module:

- Various operations (convolutions and pooling) concatenated depth-wise (zero padding and stride adjusted to maintain spatial dimension)



Inception module

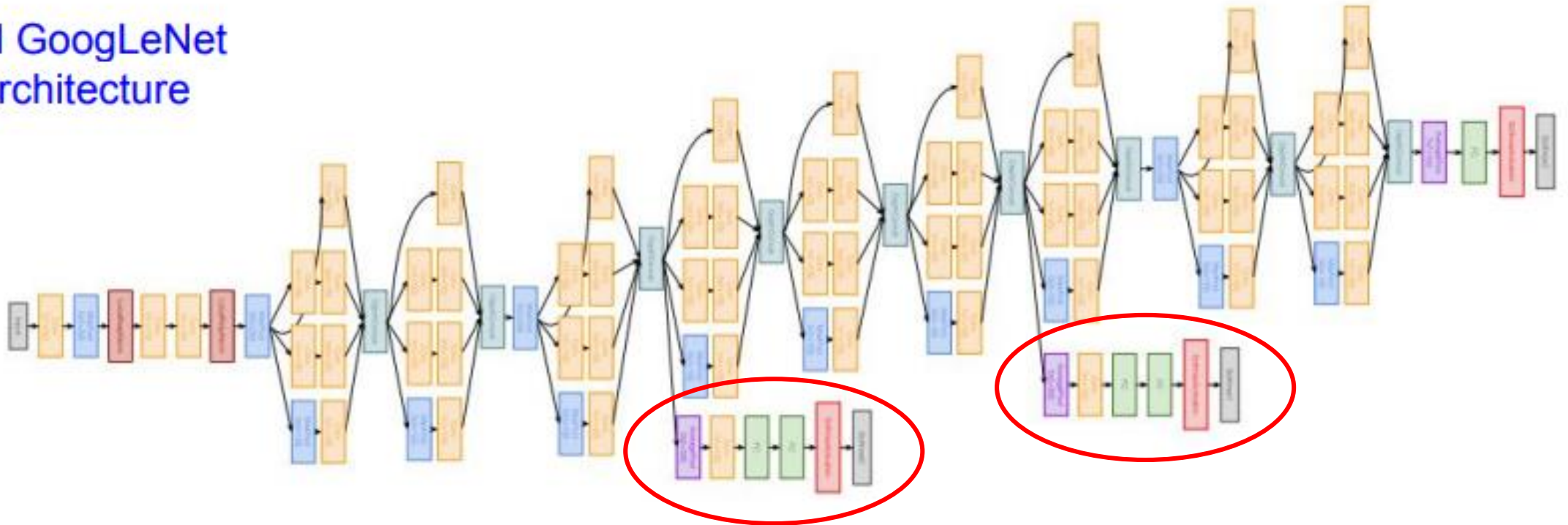




## History of CNN for classification - GoogLeNet

Additional auxiliary classification outputs help backpropagation.

### Full GoogLeNet architecture



- 12x less parameters than AlexNet

## History of CNN for classification - ResNet

### Properties:

- Extremely deep (152 layer)

### Motivation:

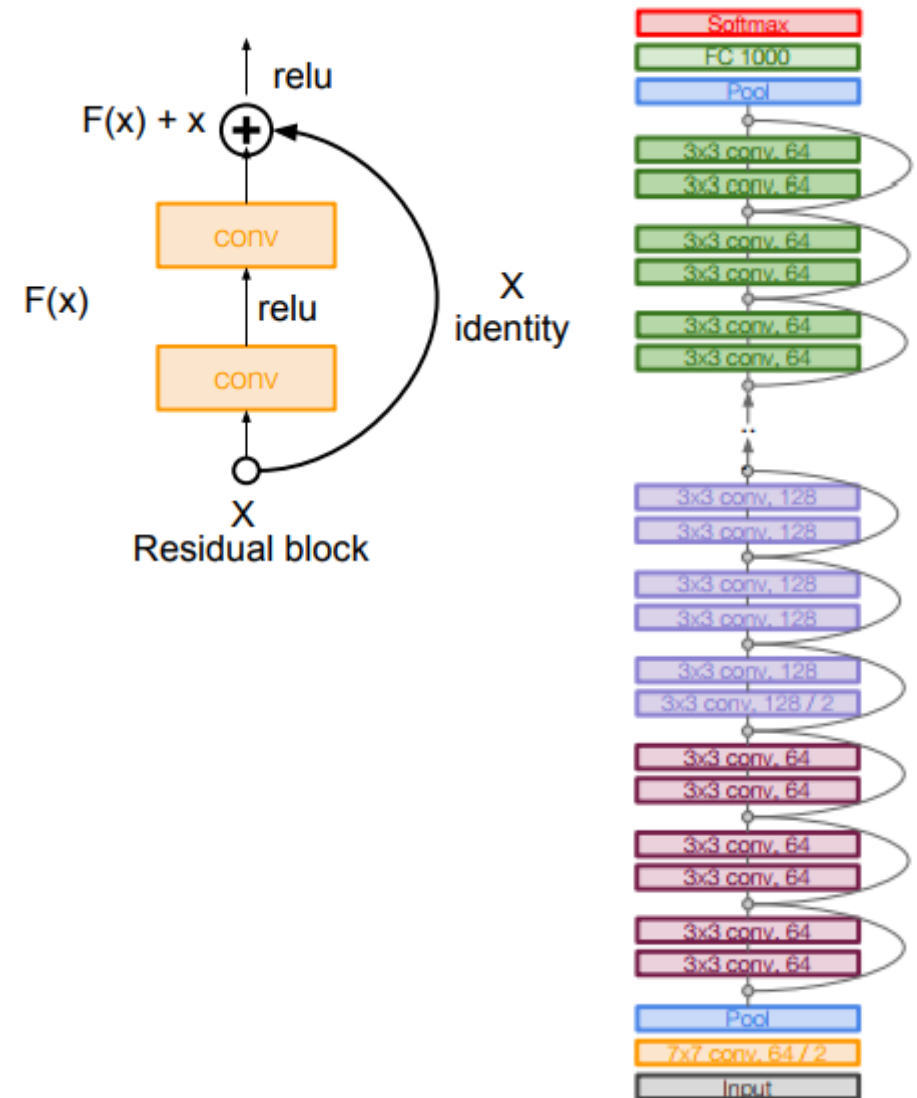
- Underlying assumption that deep model should perform at least as good as shallower models (or even better as by depth we increase model complexity and we expect at least training error to be lower)
- Experimentally it was not confirmed – probably an optimization problem

### Solution:

- Residual connection – instead of calculating the full activation map we only calculate the delta (residual)

$$H(x) = F(x) + x$$

- How do we need to modify the input to resolve our problem



# Region of Interest proposal methods

Object Detection – Region of Interest proposal

Sliding window



**BACKGROUND**

## Object Detection – Region of Interest proposal

### Sliding window



CAR

### Main problems:

- The search space has to be reduced by using a regular grid, fixed scales, and fixed aspect ratios
- Most of the boxes selected this way do not contain any object

## Object Detection – Region of Interest proposal

### Selective Search

#### Motivation:

- Hierarchical relation of object in an image
- Differences in a single aspect of the image and similarity in others
  - Texture
  - Color
- Enclosing of some objects by other

#### Solution:

- Using a data-driven approach to steer the sampling



(a)

(b)



(c)

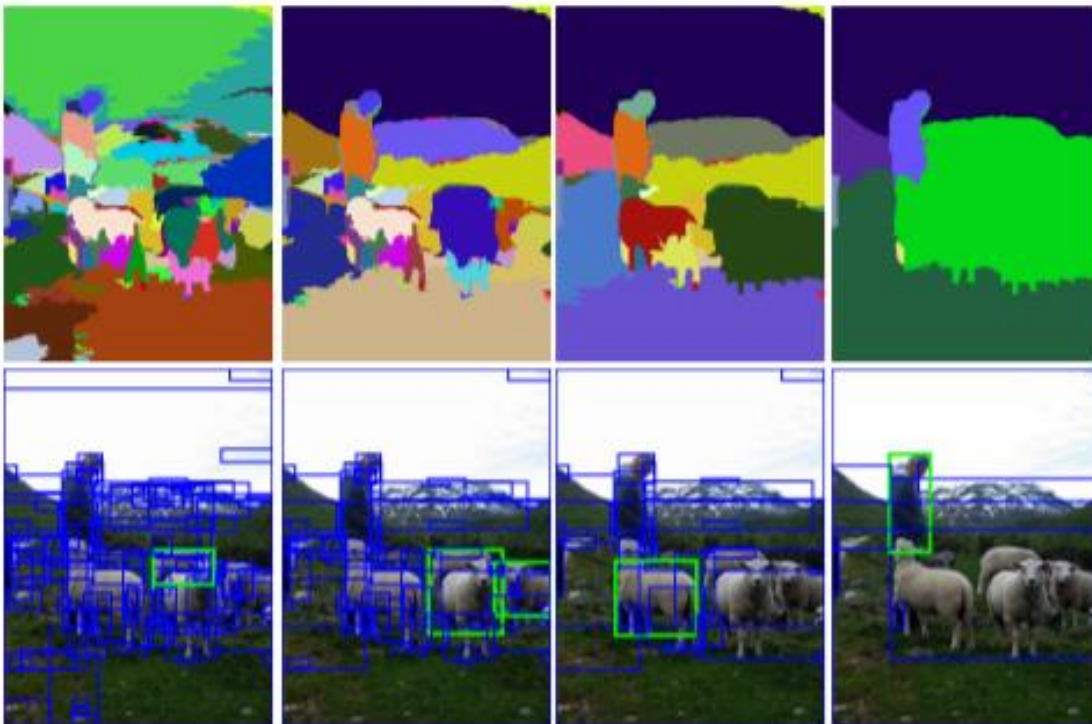
(d)

## Object Detection – Region of Interest proposal

### Selective Search

#### Design consideration:

- Capture all scales
- Diversification
- Fast to compute



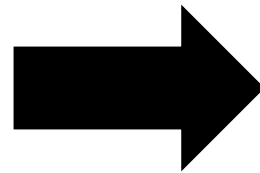
## Object Detection – Region of Interest proposal

### Selective Search

#### Complementary Similarity Measures

- Color
- Texture
- Size
- Fill

Full Score = C+T+S+F





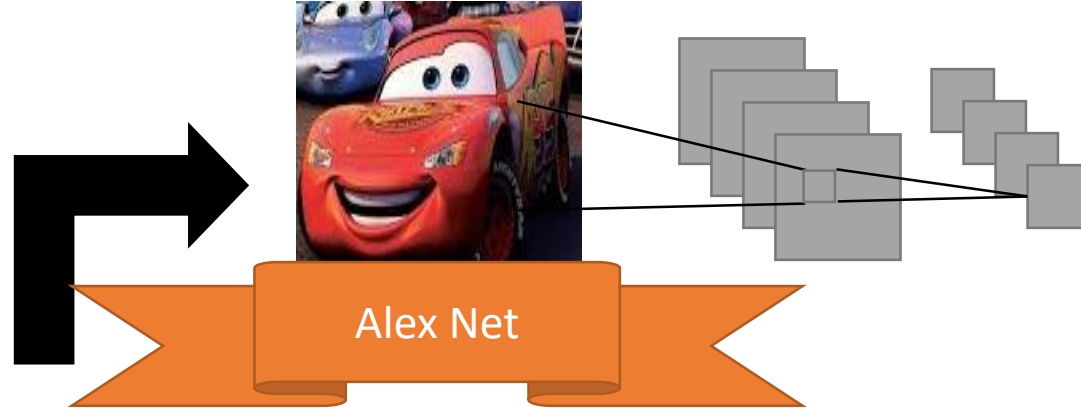
# R-CNN architectures

# Object Detection – overview – Regional CNN

## 1. Input Image



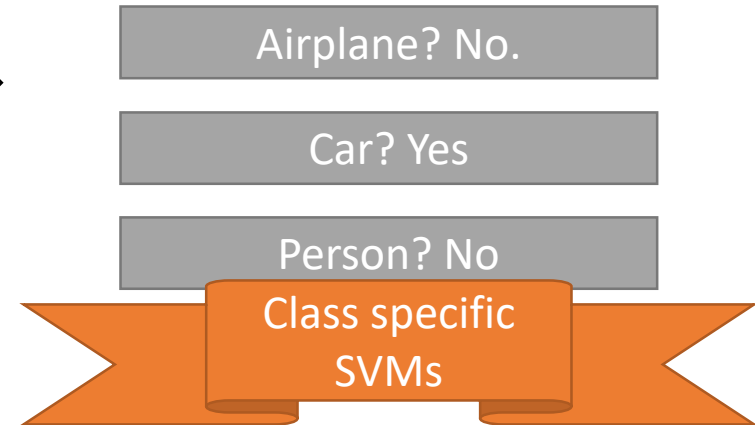
## 3. CNN feature computation



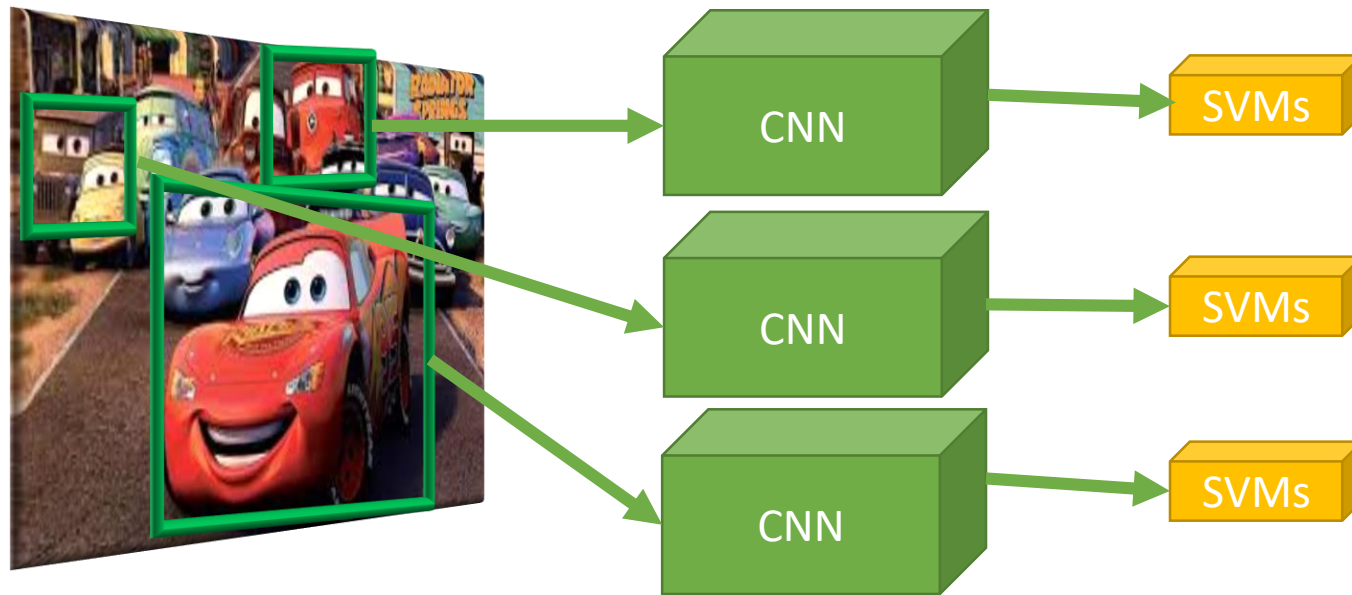
## 2. Region of Interest proposal



## 4. Classification of regions



# Object Detection – overview – Regional CNN

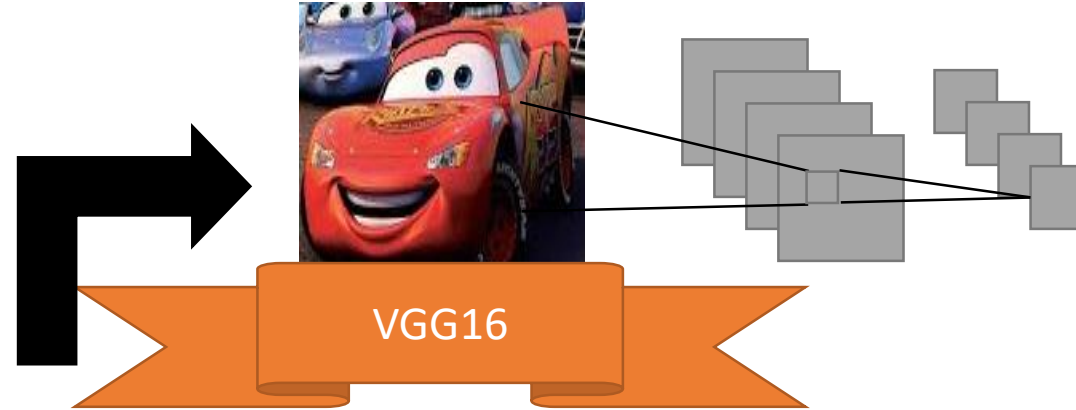


# Object Detection – overview – Fast Regional CNN vol.2

## 1. Input Image



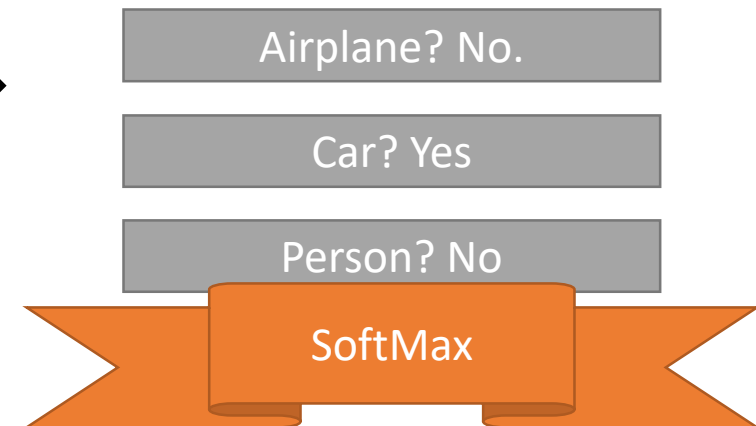
## 2&3. CNN feature computation



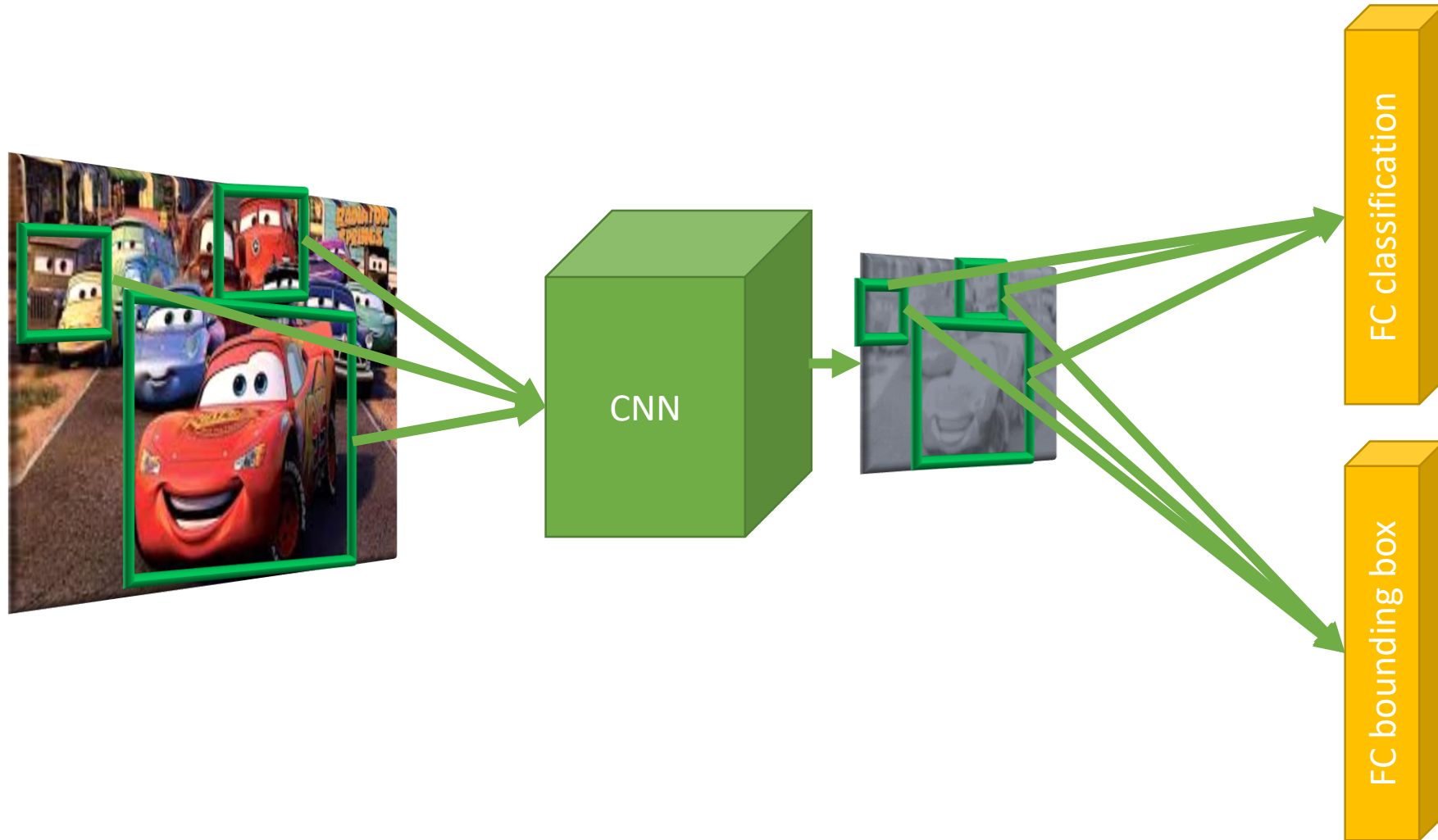
## 2&3. Region of Interest proposal



## 4. Classification of regions

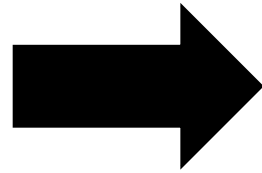


# Object Detection – overview – Fast Regional CNN vol.2

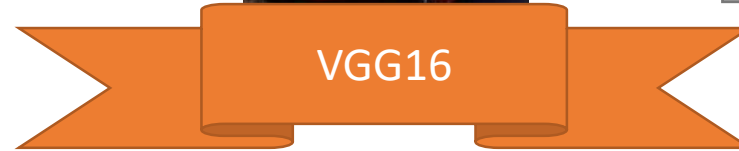
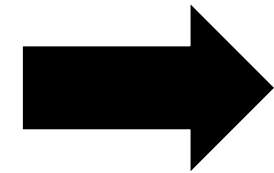
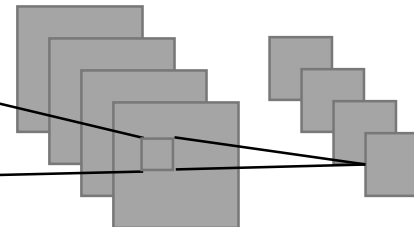


Object Detection – overview – Faster Regional CNN vol.3

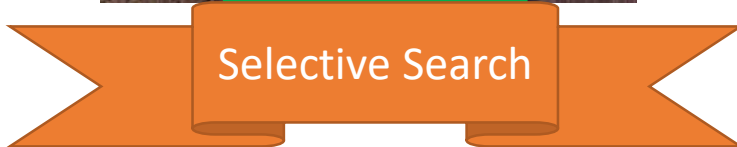
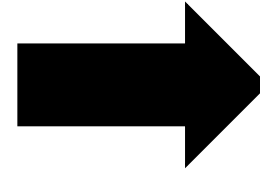
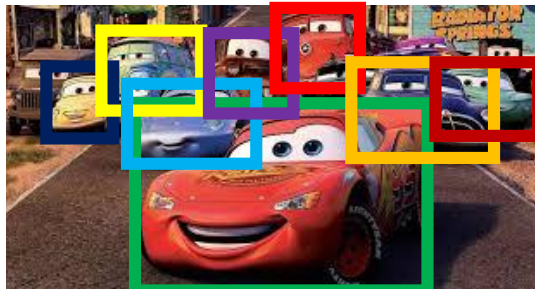
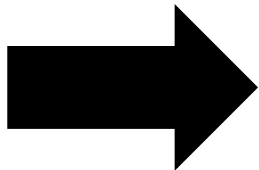
**1. Input Image**



**2. CNN feature computation**



**3. Region of Interest proposal**



**4. Classification of regions**

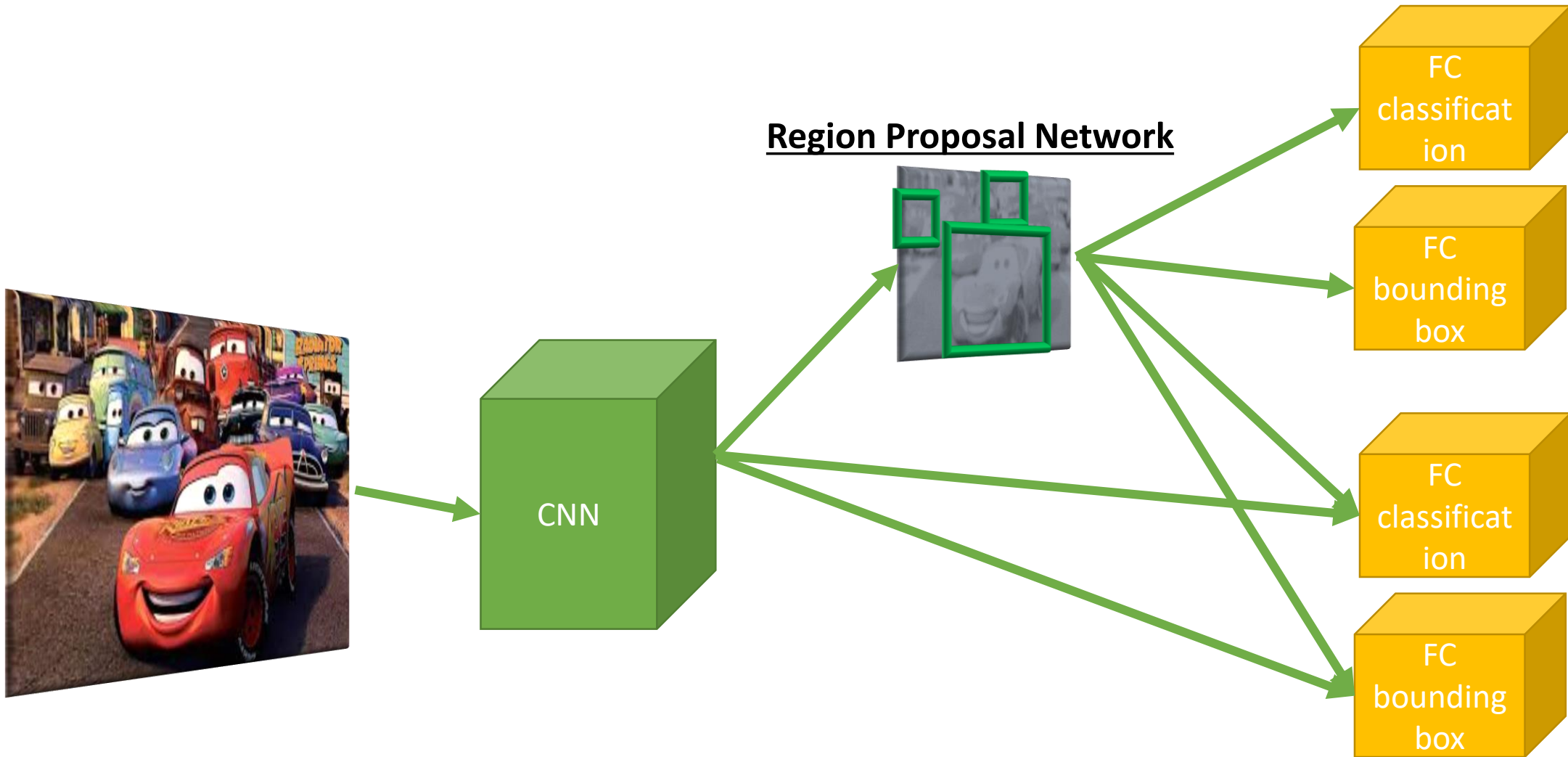
Airplane? No.

Car? Yes

Person? No

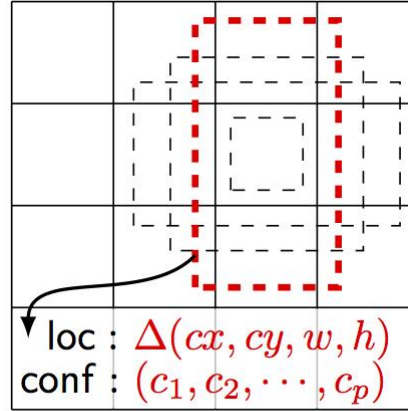
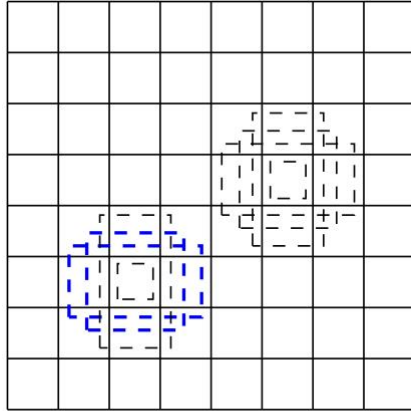
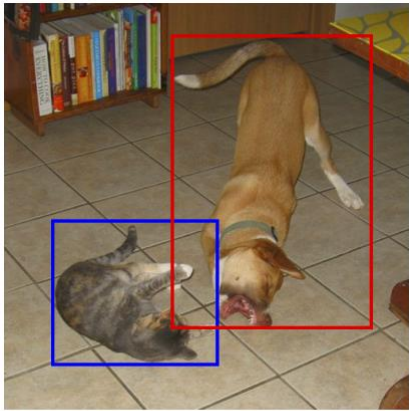


## Object Detection – overview – Fast Regional CNN vol.2



# Object Detection – overview – SSD

## Single Shot MultiBox Detector



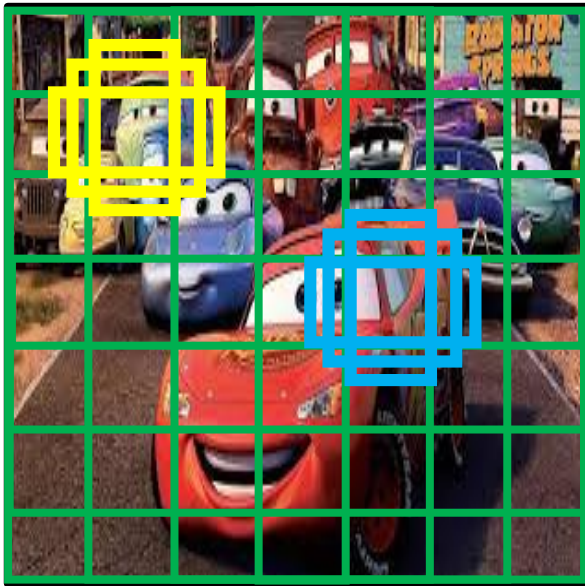
(a) Image with GT boxes

(b)  $8 \times 8$  feature map(c)  $4 \times 4$  feature map



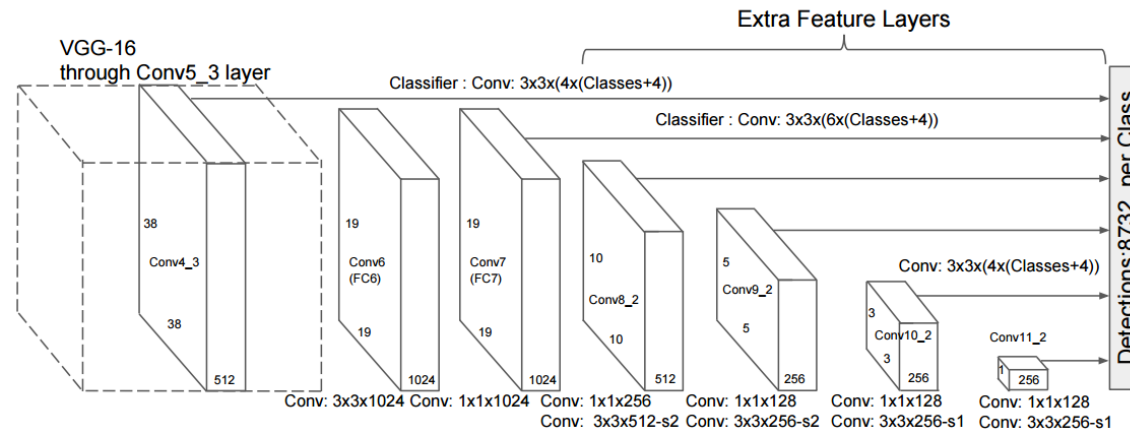
# Object Detection – overview – SSD

## 1. Input Image



Grid + anchor boxes

## 2. CNN feature computation



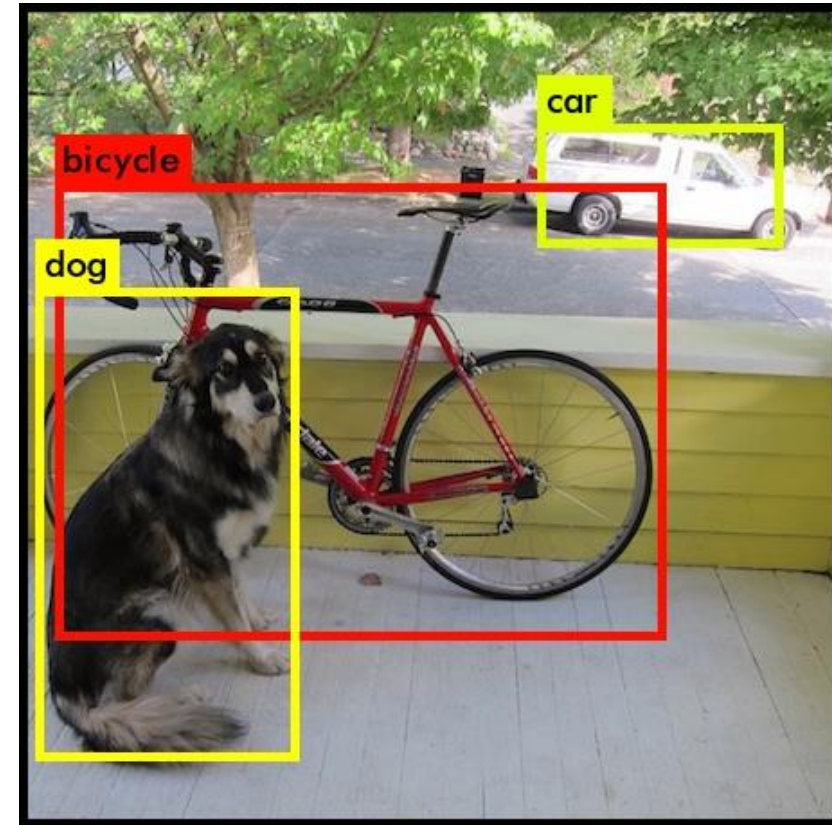
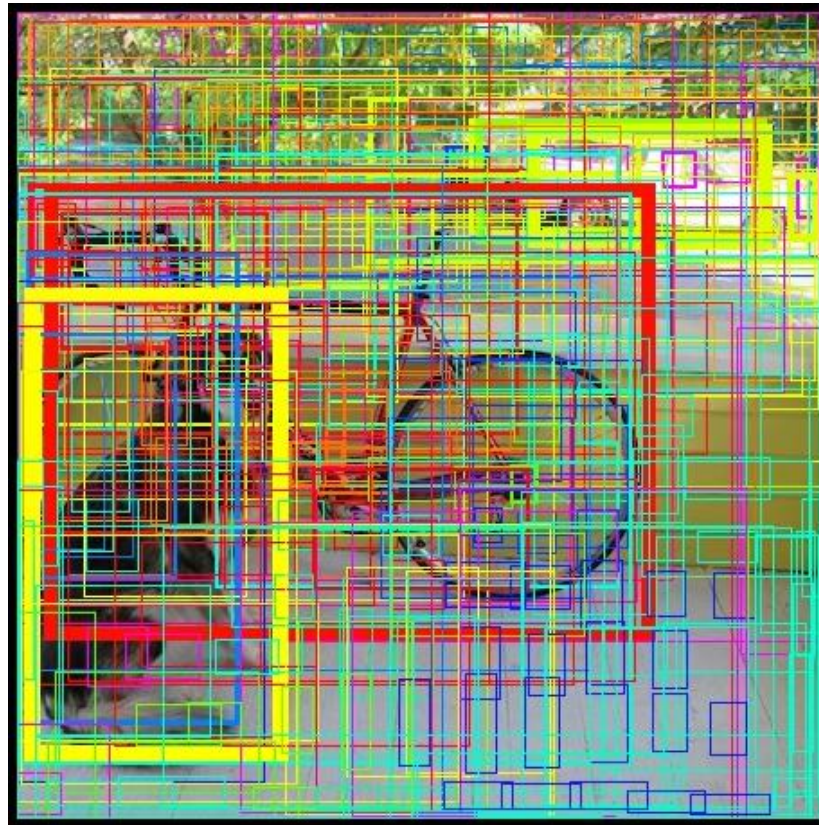
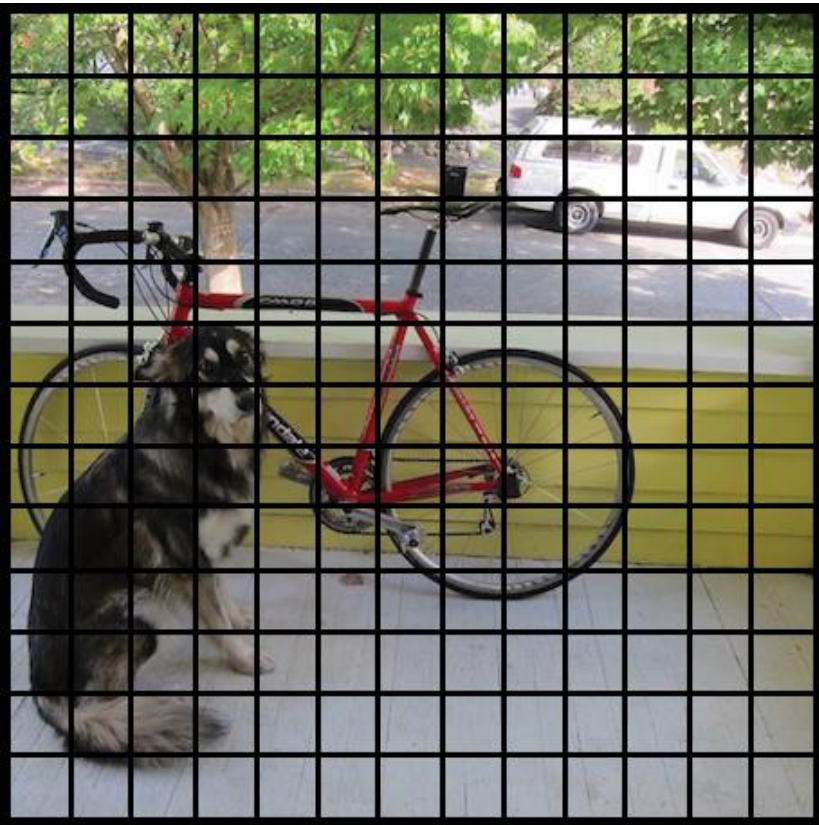
SSD

## 3. Classification of regions



$$7 \times 7 \times (5 * B + C)$$

## Object Detection – overview – YOLO



• **YOLO = You Only Look Once**

$$S \times S \times (5 + C)$$

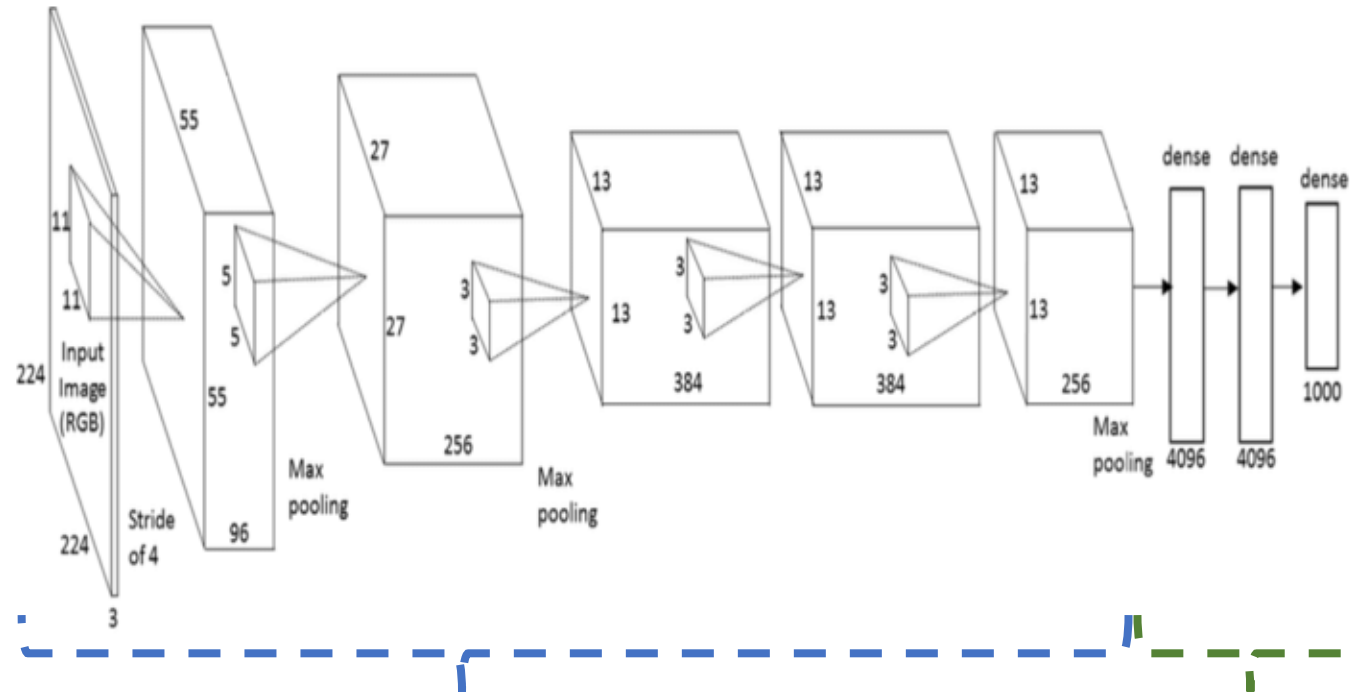
$x, y, w, h, \text{conf}$

# Transfer Learning

## Object Detection – Transfer Learning

- **ConvNet as fixed feature extractor.** Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer (this layer's outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset. In an AlexNet, this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. We call these features **CNN codes**. It is important for performance that these codes are ReLUd (i.e. thresholded at zero) if they were also thresholded during the training of the ConvNet on ImageNet (as is usually the case). Once you extract the 4096-D codes for all images, train a linear classifier (e.g. Linear SVM or Softmax classifier) for the new dataset.
- **Fine-tuning the ConvNet.** The second strategy is to not only replace and retrain the classifier on top of the ConvNet on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation. It is possible to fine-tune all the layers of the ConvNet, or it's possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a ConvNet contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset. In case of ImageNet for example, which contains many dog breeds, a significant portion of the representational power of the ConvNet may be devoted to features that are specific to differentiating between dog breeds.
- **Black-box off the shelf model.** Using model that was already trained on a different data to resolve the same problem on the data at hand.

# Object Detection – Transfer Learning



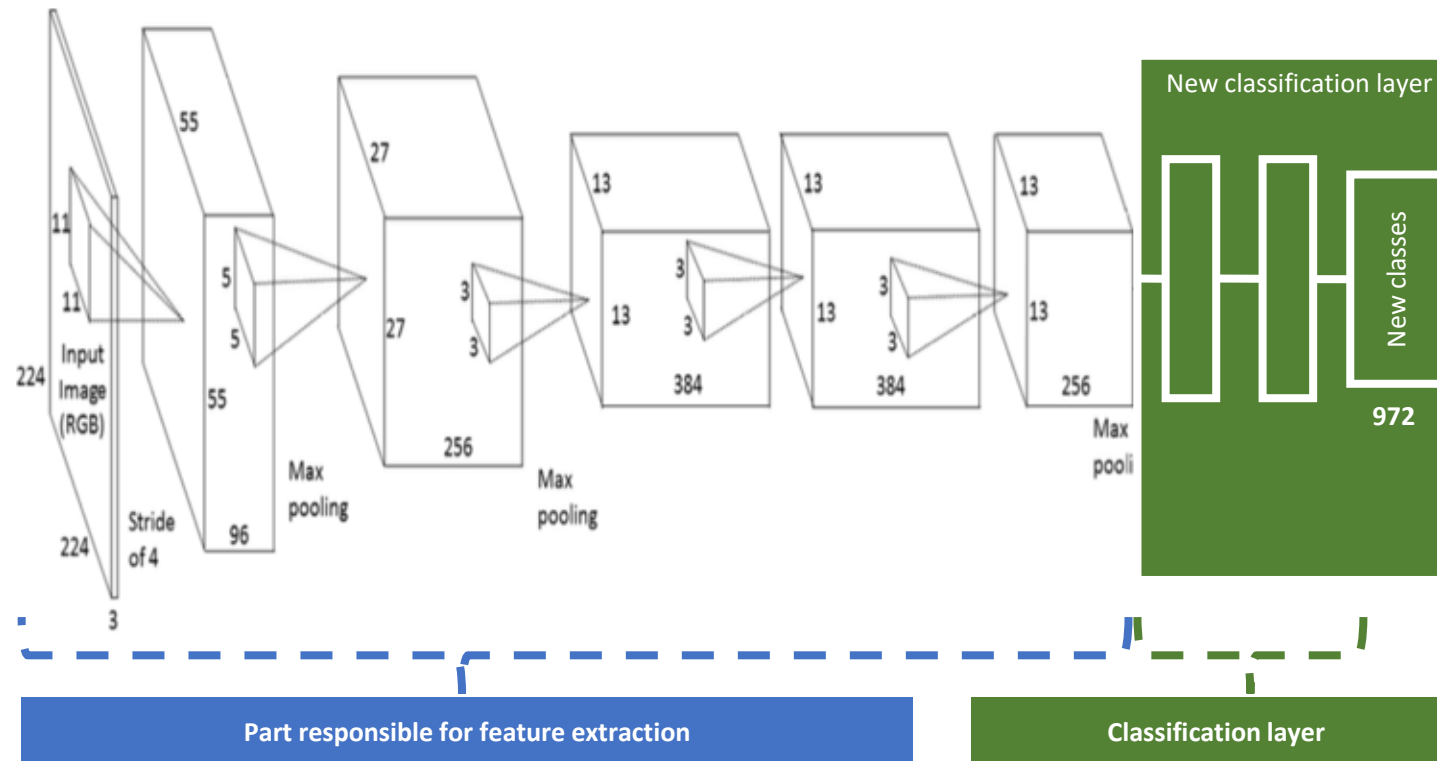
### Part responsible for feature extraction

At initial layers the net extracts features like edges and colors. The deeper into the net we will go the more high level feature will be extracted. Examples of high level features: faces, wheels or text.

### Categorization layer

This is a problem specific layer that uses the features extracted earlier to solve current problem.

# Object Detection – Transfer Learning



- Using information extracted from millions of images
- Possibility to fine-tune this part to extract specific features connected to problem at hand

- Retrained for the problem at hand

# Experiment

# Object Detection – Experiment Design

## 1. DATA PREPARATION



- Haar cascade classifier

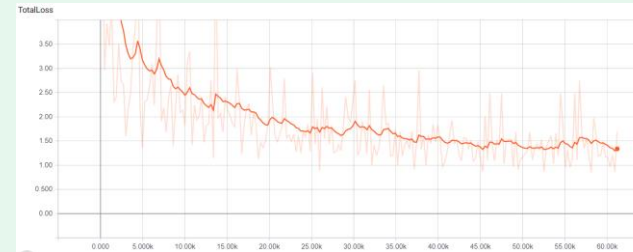


Generate TFRecord format  
&  
Create Label Map

## 2. CONFIGURATION & TRAINING



Pipeline configuration

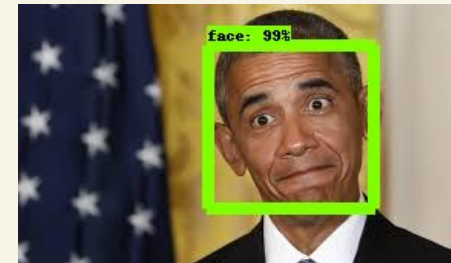


Training

## 3. VALIDATION

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Graph export

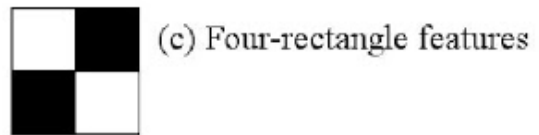
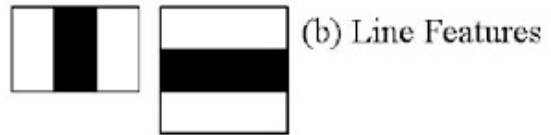


Results



## Object Detection – Experiment Data preparation

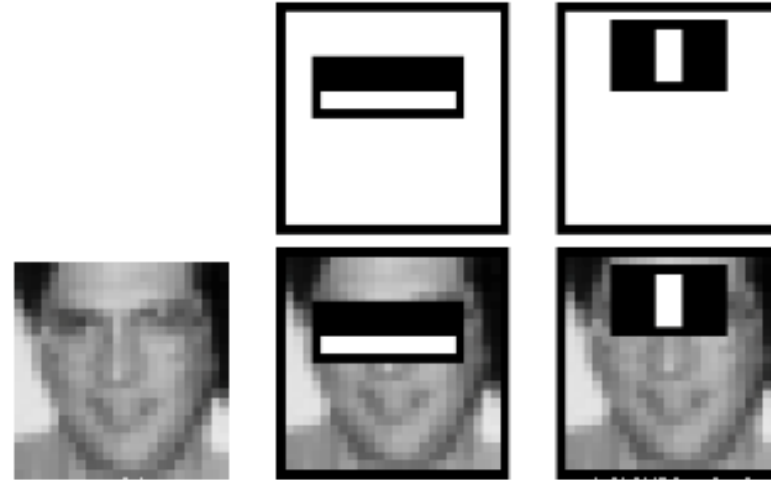
- HAAR FEATURES



### RAW DATA



### RELEVANT FEATURES



### LABELED DATA



## Object Detection – Experiment Configuration & training

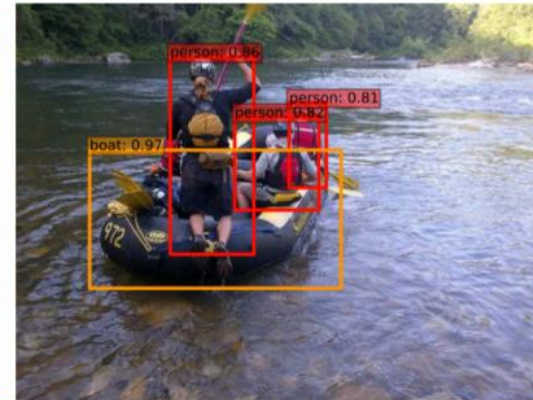
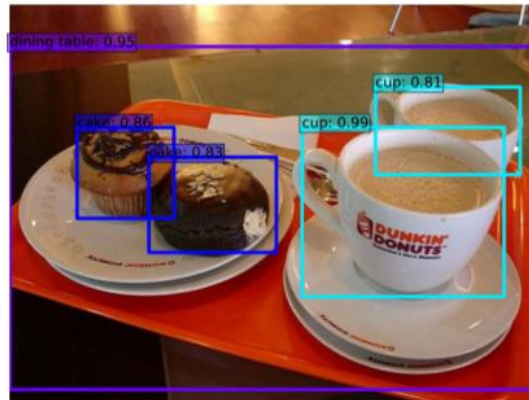
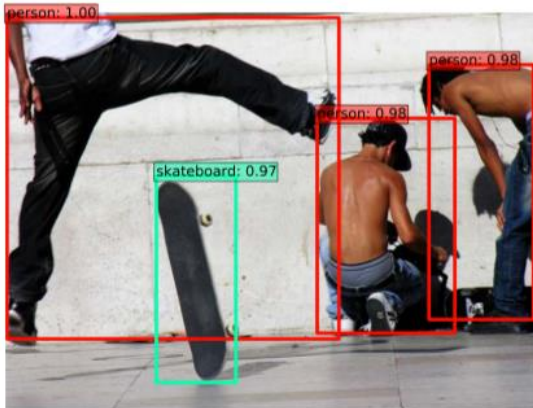
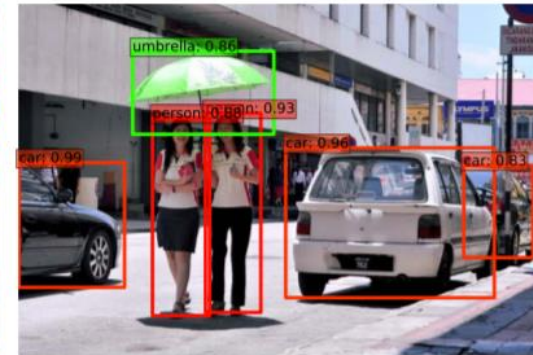
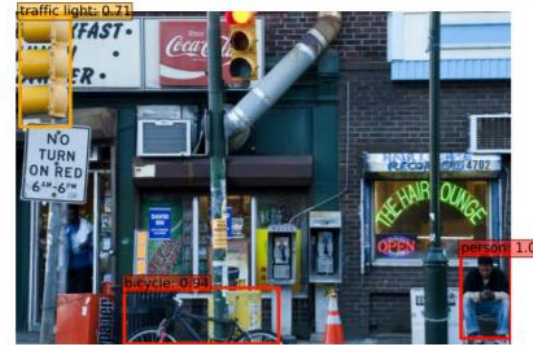
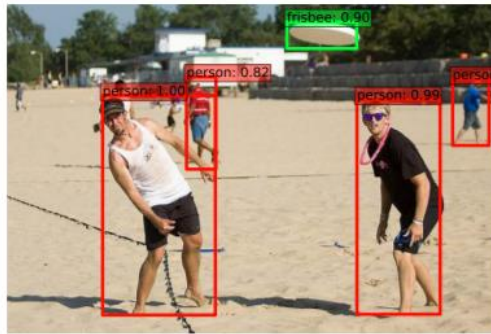
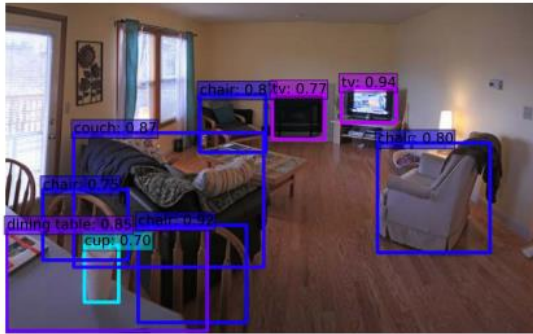
### • COCO dataset – Common Objects in Context



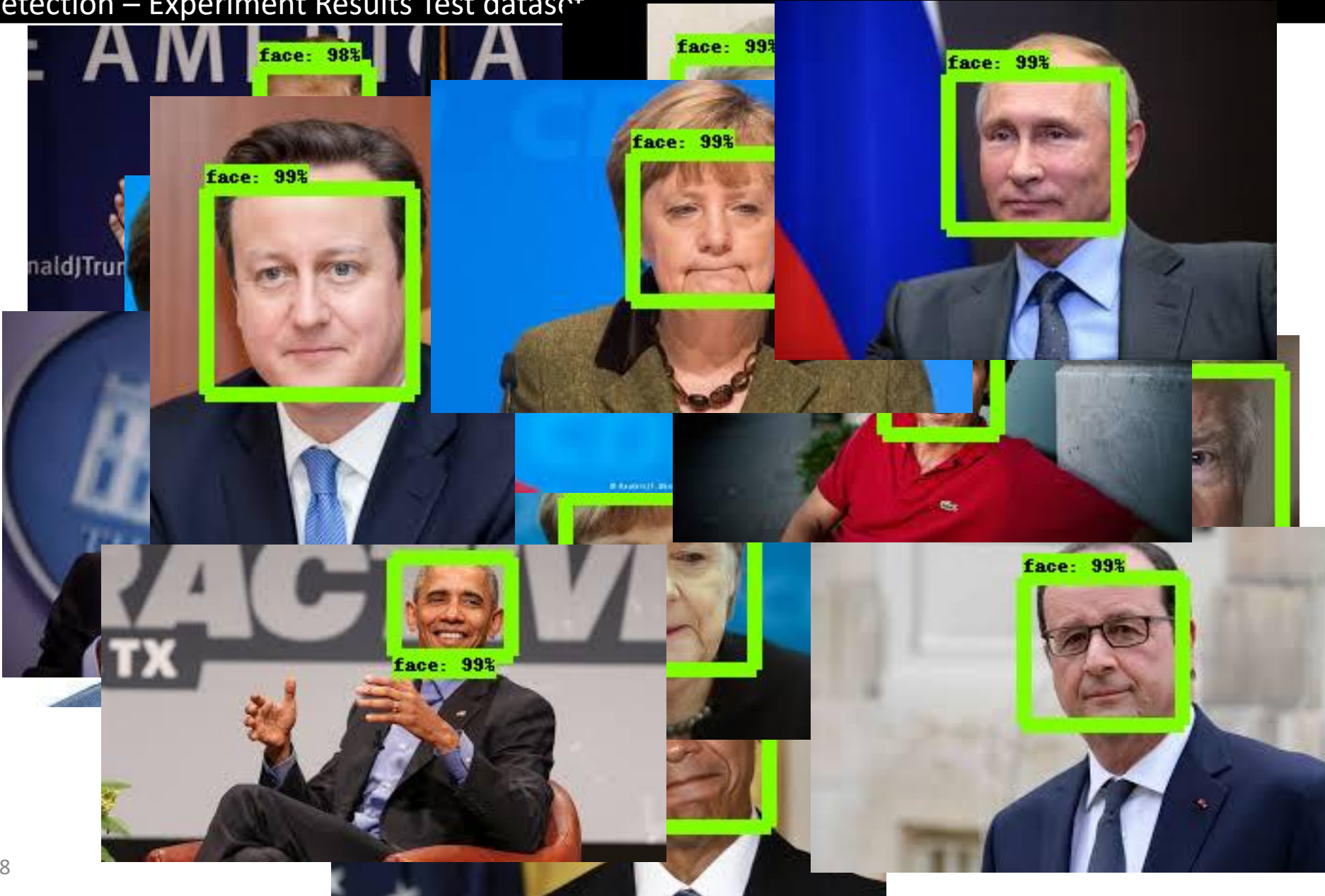
- 123,287 images
- 886,284 instances (labeled objects)



# Object Detection – Experiment SSD results on COCO dataset



# Object Detection – Experiment Results Test datasr+

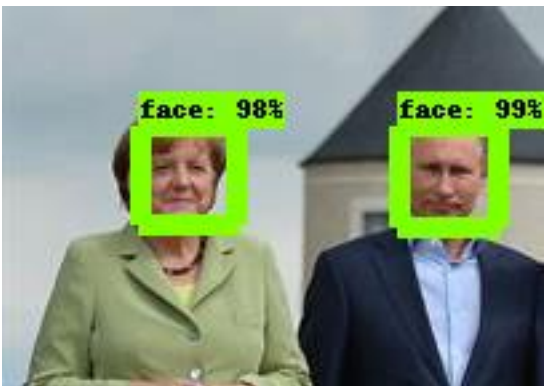
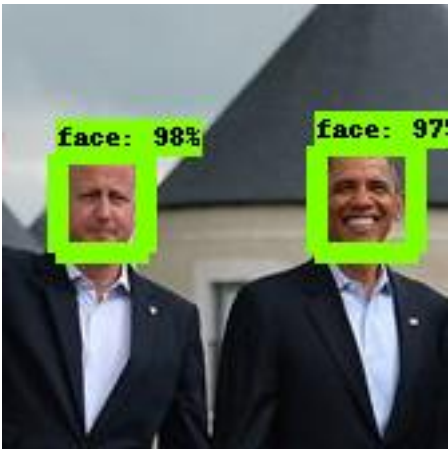




## Object Detection – Experiment Learnings

### Observations

- Scale of the class in the training images
- Different angels/views of the training class



# Future plans

## Object Detection – Transfer Learning

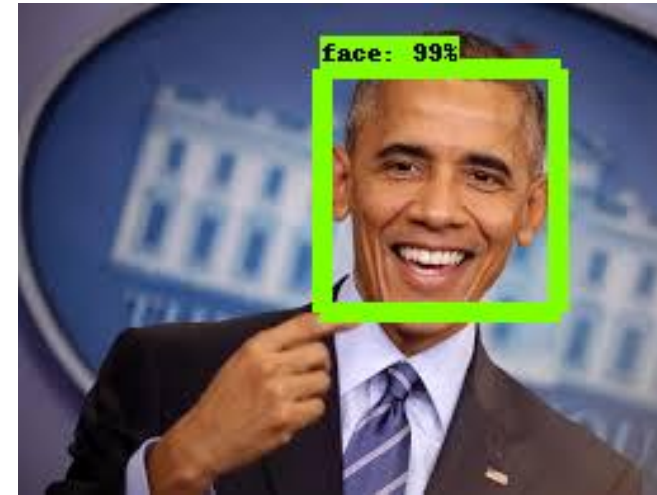
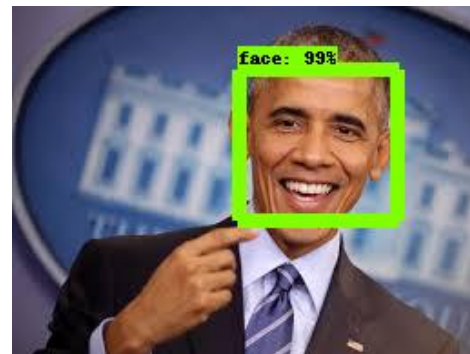
Adding new class to a previously trained object detection network.

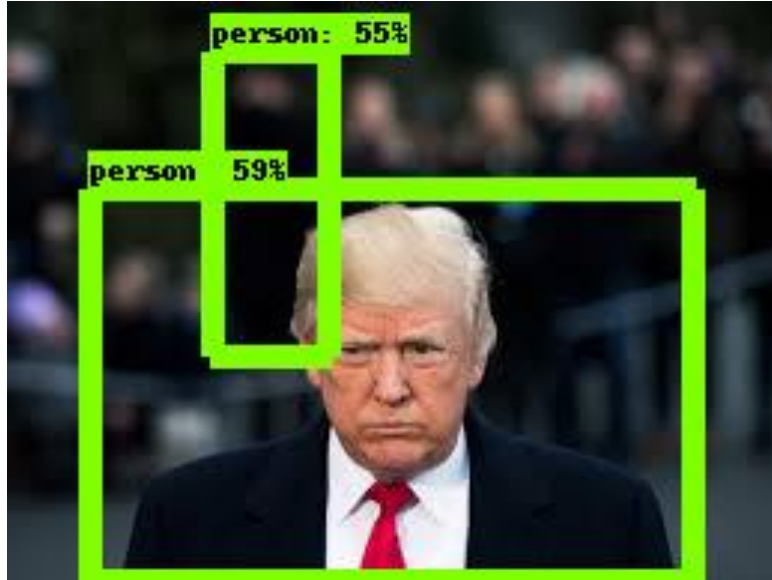
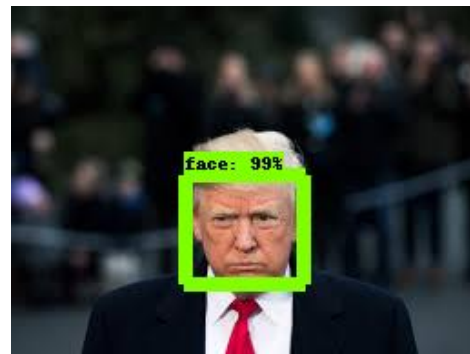
### Experiment design

#### **Comparison of 3 different object detection networks:**

- Original SSD network trained on COCO dataset
- SSD network from point 1 retrained with images of faces and chairs
- SSD network from point 1 retrained with images of faces

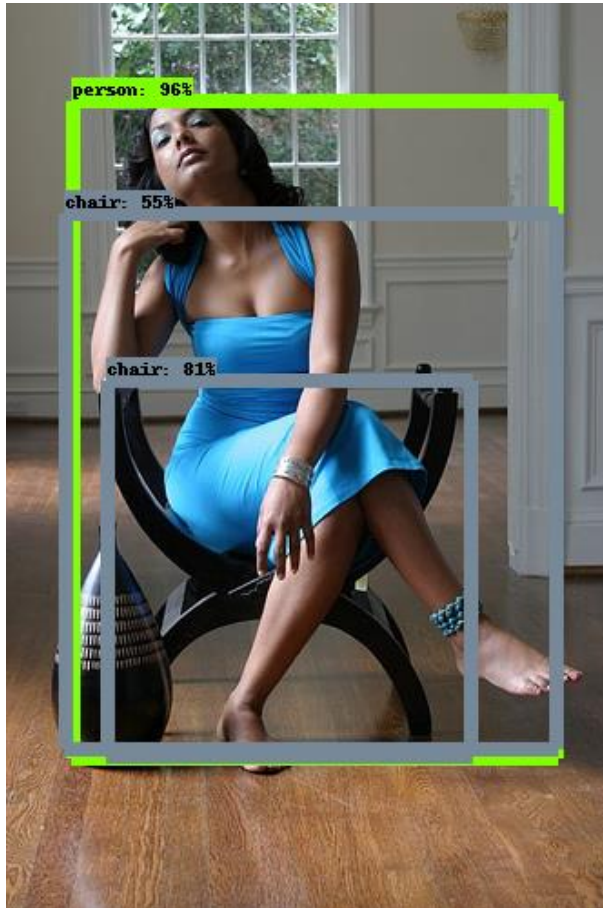


**Detection comparison****Original SSD****Face+Chair retrained SSD****ONLY Face retrained SSD**

**Detection comparison****Original SSD****Face+Chair retrained SSD****ONLY Face retrained SSD**

## Detection comparison

### Original SSD



### Face+Chair retrained SSD

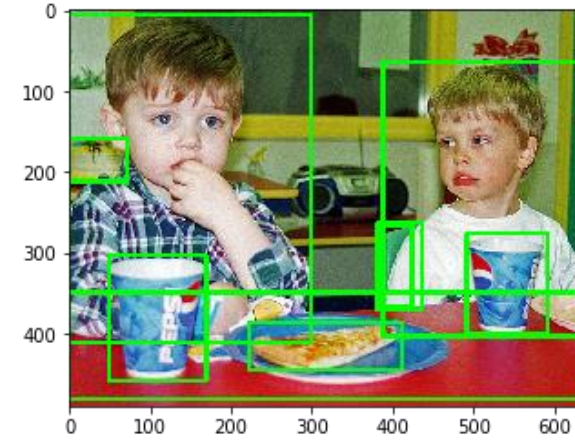
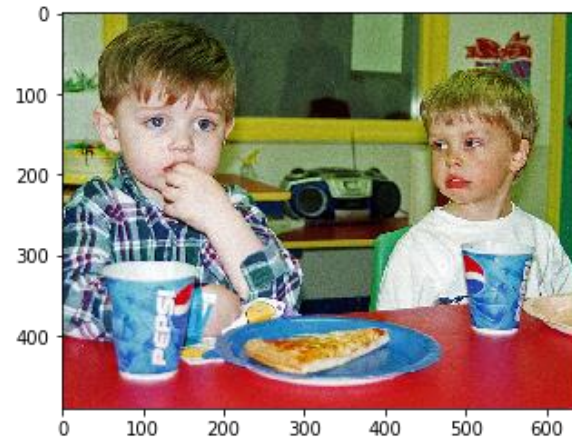


## Detection comparison - Summary

### **The object detection net trained on images of both faces and chairs:**

- Performed worse as a face detector than the object detection net trained on images of faces alone
- Did not detect any of the chairs

## Check for correct bbox position extraction



	category_ids	filename	ids	image_ids	xmax	xmin	ymax	ymin
201426	62	000000201426.jpg	377803	201426	423.93	379.89	367.44	266.14
201426	67	000000201426.jpg	414419	201426	638.65	0.00	480.09	349.06
201426	1	000000201426.jpg	440231	201426	640.00	387.60	403.01	64.97
201426	47	000000201426.jpg	677913	201426	169.19	49.44	458.80	303.89
201426	47	000000201426.jpg	679377	201426	590.16	490.05	400.04	276.60
201426	59	000000201426.jpg	1075152	201426	410.72	221.33	445.96	386.49
201426	61	000000201426.jpg	1085132	201426	72.75	0.00	212.38	159.16
201426	1	000000201426.jpg	1213103	201426	298.40	0.00	410.72	6.61
201426	62	000000201426.jpg	1930281	201426	435.83	380.05	371.88	262.28

Check if any chairs are found if we only train for chair detection



Only one chair was found in test images.

# Learning curves comparison

Horizontal Axis

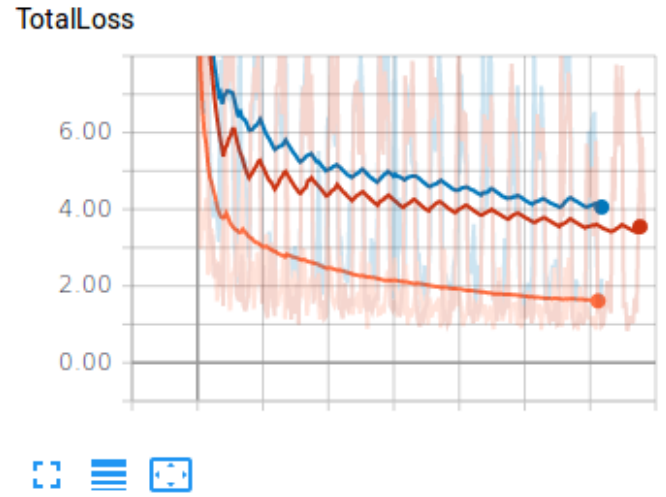
STEP RELATIVE WALL

---

Runs

Write a regex to filter runs

- face
- face + chair
- .



Horizontal Axis

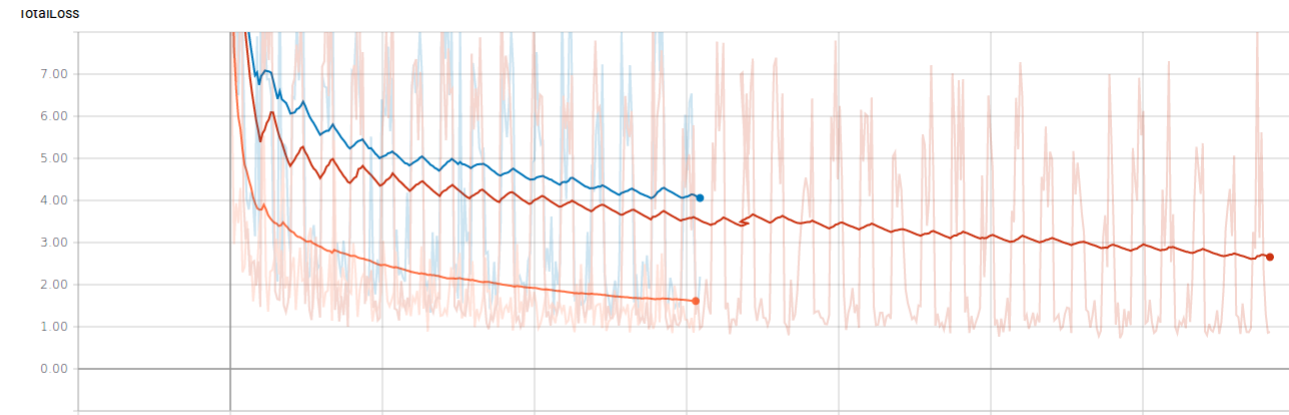
STEP RELATIVE WALL

---

Runs

Write a regex to filter runs

- face
- face + chair
- .



## Detection comparison - Summary

### **The object detection net trained on images of both faces and chairs:**

- Performed worse as a face detector than the object detection net trained on images of faces alone
- Did not detect any of the chairs

### **Conclusions & thoughts:**

- Length of the learning process
- Incorrect input data – more suitable for localization & classification problem



## Sources

- <https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4>
- Alexnet: <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>
- VGG: <https://arxiv.org/pdf/1409.1556.pdf>
- ResNet: <https://arxiv.org/pdf/1512.03385.pdf>
- R-CNN: <https://arxiv.org/abs/1311.2524>
- Fast R-CNN: <https://arxiv.org/abs/1504.08083>
- Faster R-CNN: <https://arxiv.org/abs/1506.01497>
- SDD: <https://arxiv.org/pdf/1512.02325.pdf>
- YOLO: <https://pjreddie.com/media/files/papers/yolo.pdf>