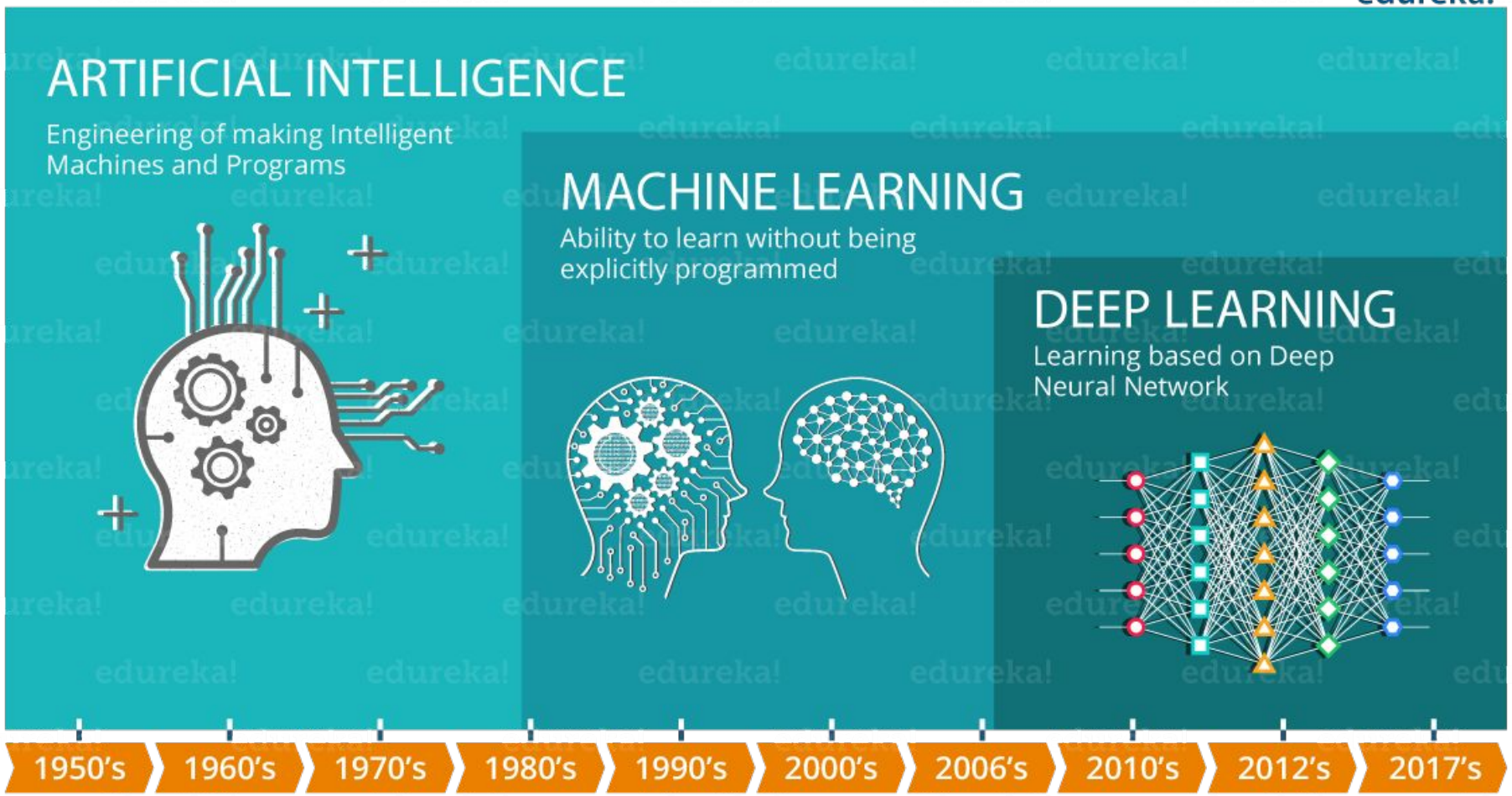
# Efficient deep network for vision-based object detection in robotic applications

keyu Lu, Xiangjing An, Jian Li, Hangen He
Neurocomputing 245 (2017) 31–45

รัชกฤช มัญยานนท์ |รหัสนักศึกษา 6014450017

# Introduction Deep learning



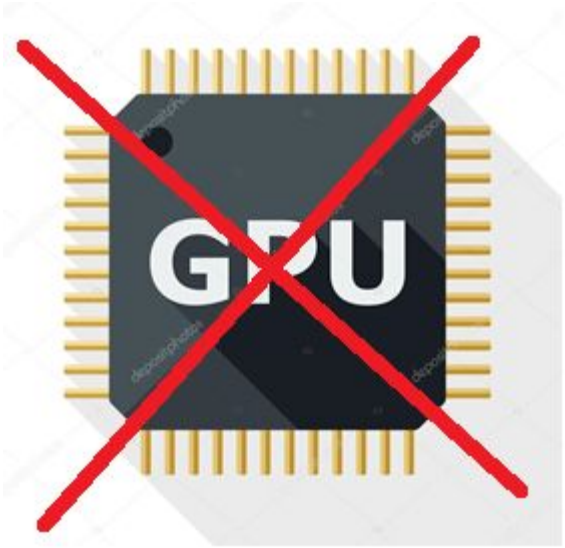
# Introduction Deep learning

# Abstract

- Vision-based object detection is essential for robotic.
- Strict constraints that apply to many robot systems in terms of run-time, power and space. To meet these special requirements of robotic applications.

# Abstract

- The motivation of this paper is to reduce the run-time of CNN based object detection system and make it possible to be utilized in robotic applications without GPU support.




# Abstract

- Introduce proposal layer to efficiently generate potential object bounding-boxes.
- Construct a robust detection layer which contains a multiple population genetic algorithm-based convolutional neural network (MPGA-based CNN) module and TLD-based multi-frame fusion procedure.

# Introduction

Vision-based object detection

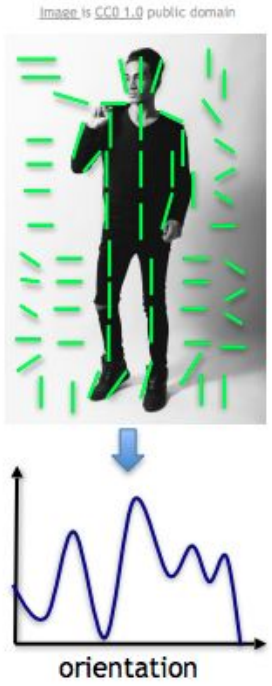
Hand-craft solutions

- HOG+SVM
- Viola and Jones
- SIFT

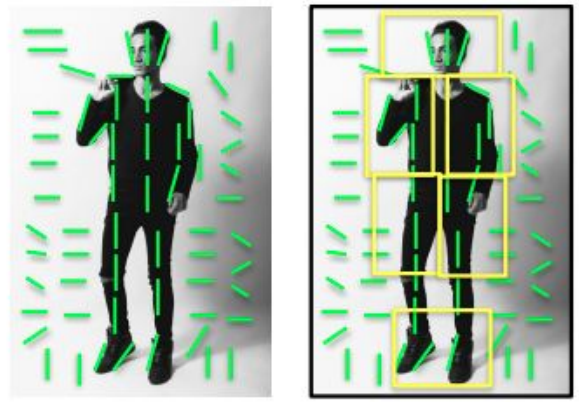
Deep learning solutions

- Data Driven
- CNN Based

# Hand Craft Solution


Histogram of Gradients (HoG)
Dalal & Triggs, 2005



Deformable Part Model
Felzenswalb, McAllester, Ramanan, 2009

IMAGENET

www.image-net.org

**22K** categories and **14M** images

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
- Food
- Materials
- Structures
- Artifact
  - Tools
  - Appliances
  - Structures
- Person
- Scenes
  - Indoor
  - Geological Formations
- Sport Activities

Deng, Dong, Socher, Li, Li, & Fei-Fei, 2009

## IMAGENET Large Scale Visual Recognition Challenge

Steel drum

The Image Classification Challenge:

1,000 object classes

1,431,167 images

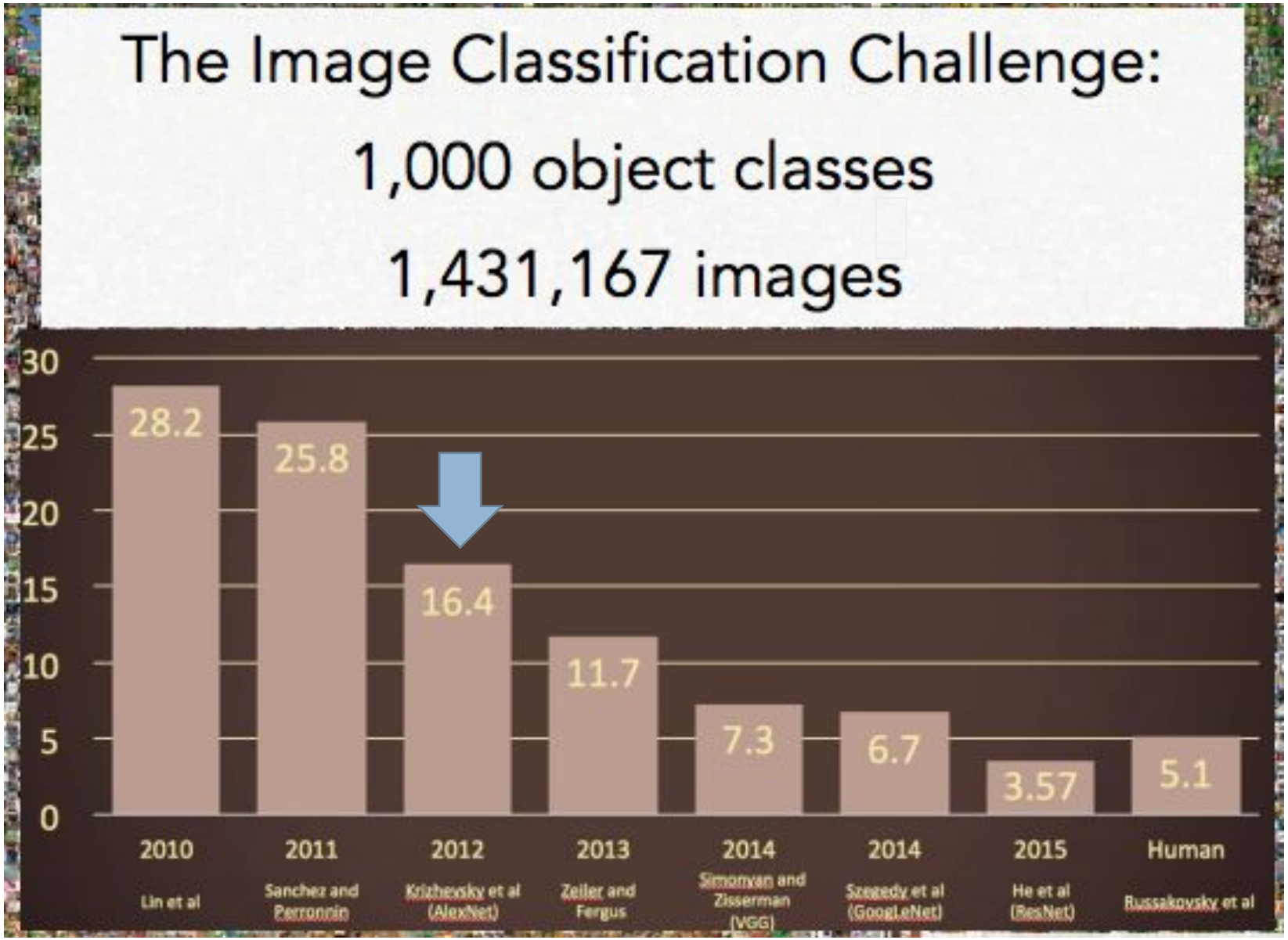
**Output:**
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

✔

**Output:**
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

Russakovsky et al. arXiv, 2014


The Image Classification Challenge:
1,000 object classes
1,431,167 images
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.57
5.1
2010
Lin et al
2011
Sanchez and Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler and Fergus
2014
Simonyan and Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
Human
Russakovsky et al

# A bit of history:

## ImageNet Classification with Deep Convolutional Neural Networks

*[Krizhevsky, Sutskever, Hinton, 2012]*

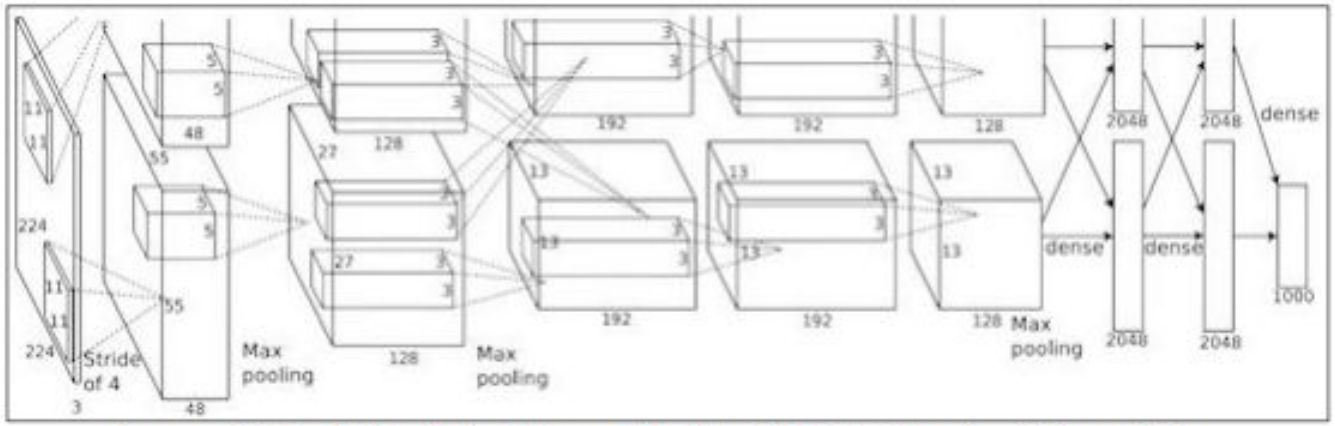
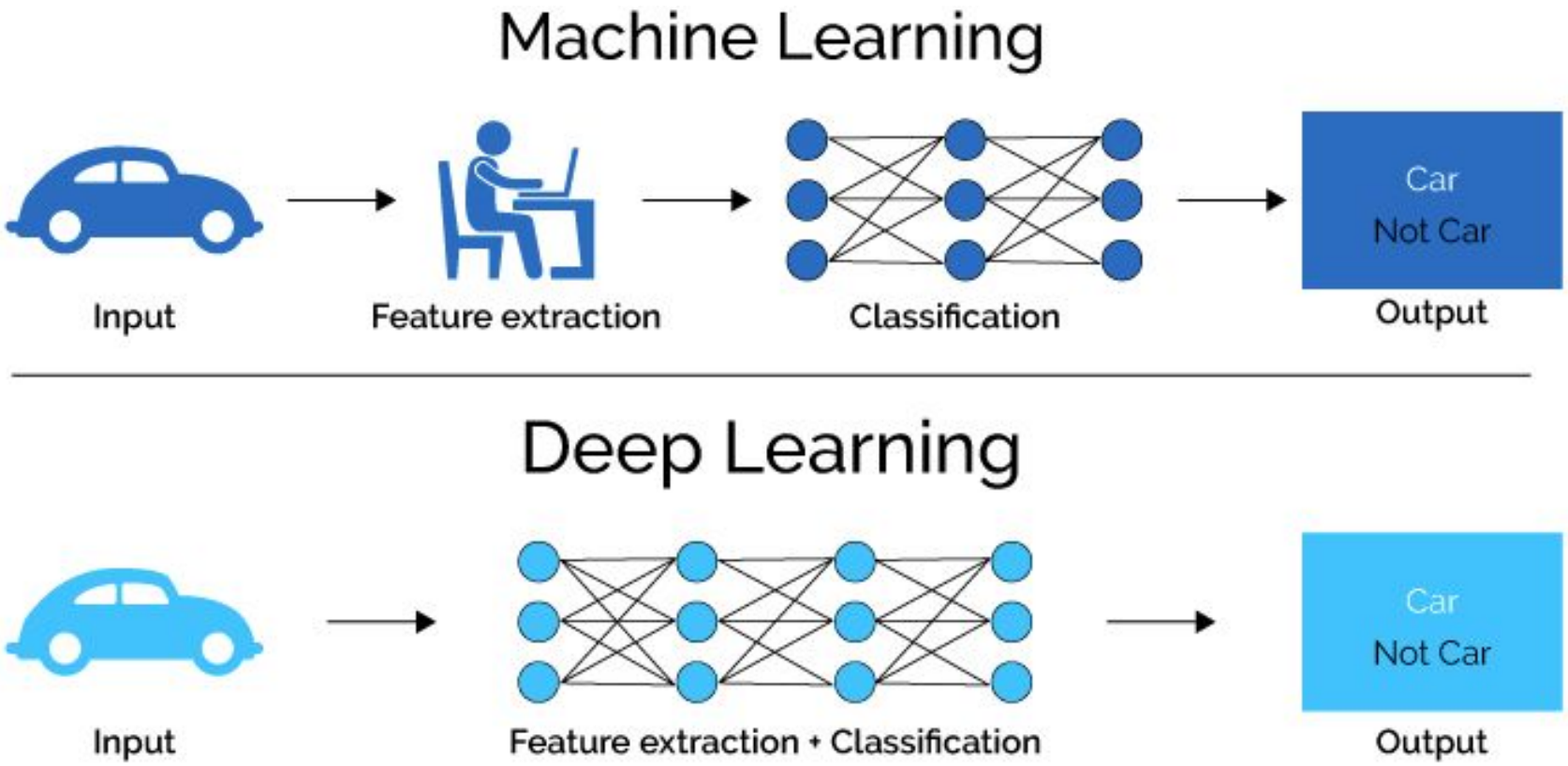



Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

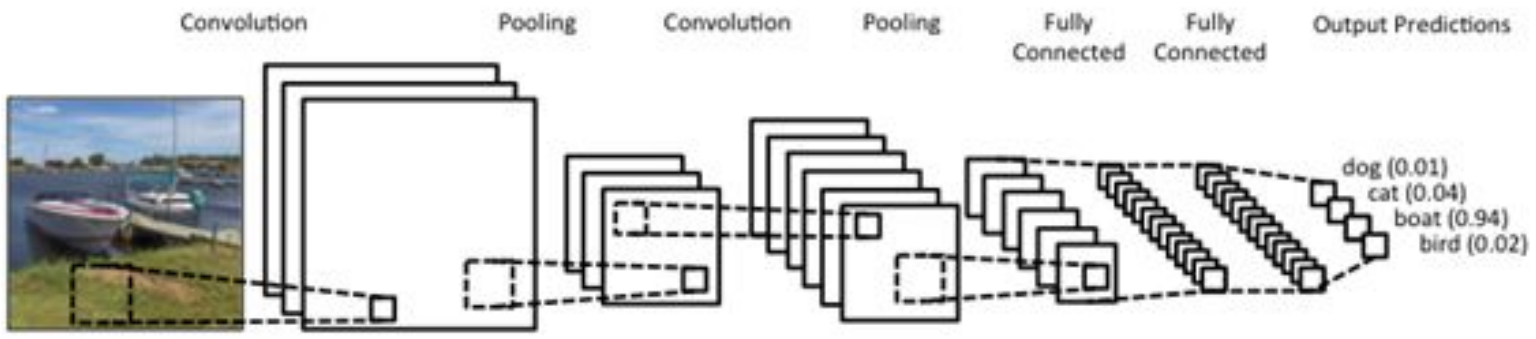
"AlexNet"

# Introduction Deep learning



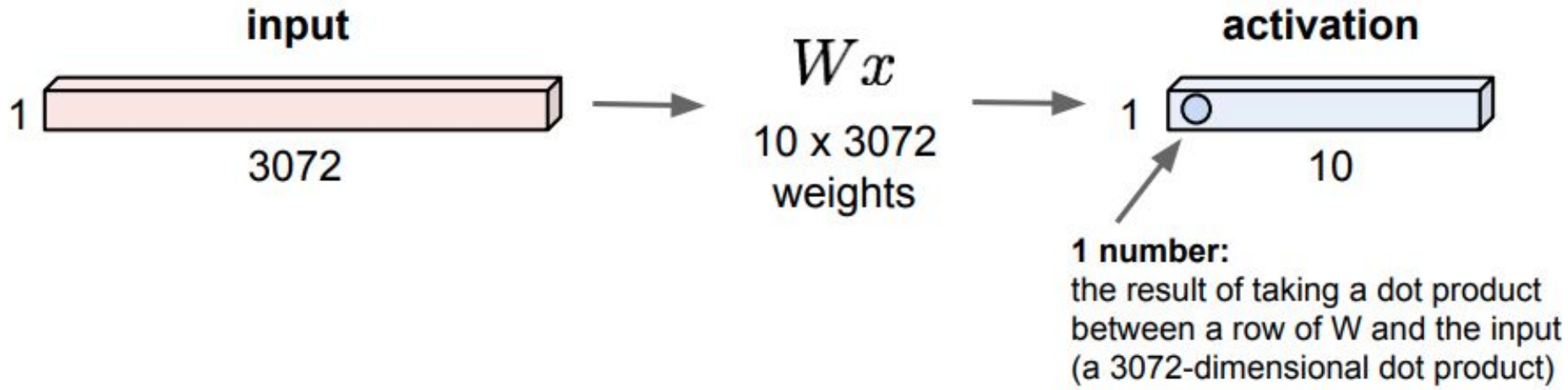
# Type of Layer on CNN Deep Learning

- Fully Connected Layer
- Convolution Layer
- Pooling layer



# Fully Connected Layer

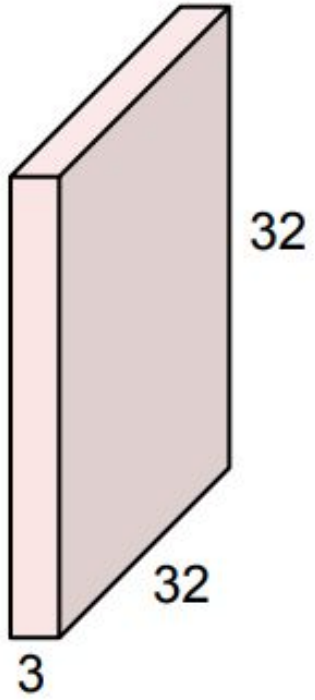
32x32x3 image -> stretch to 3072 x 1



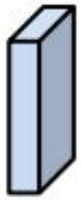
# Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

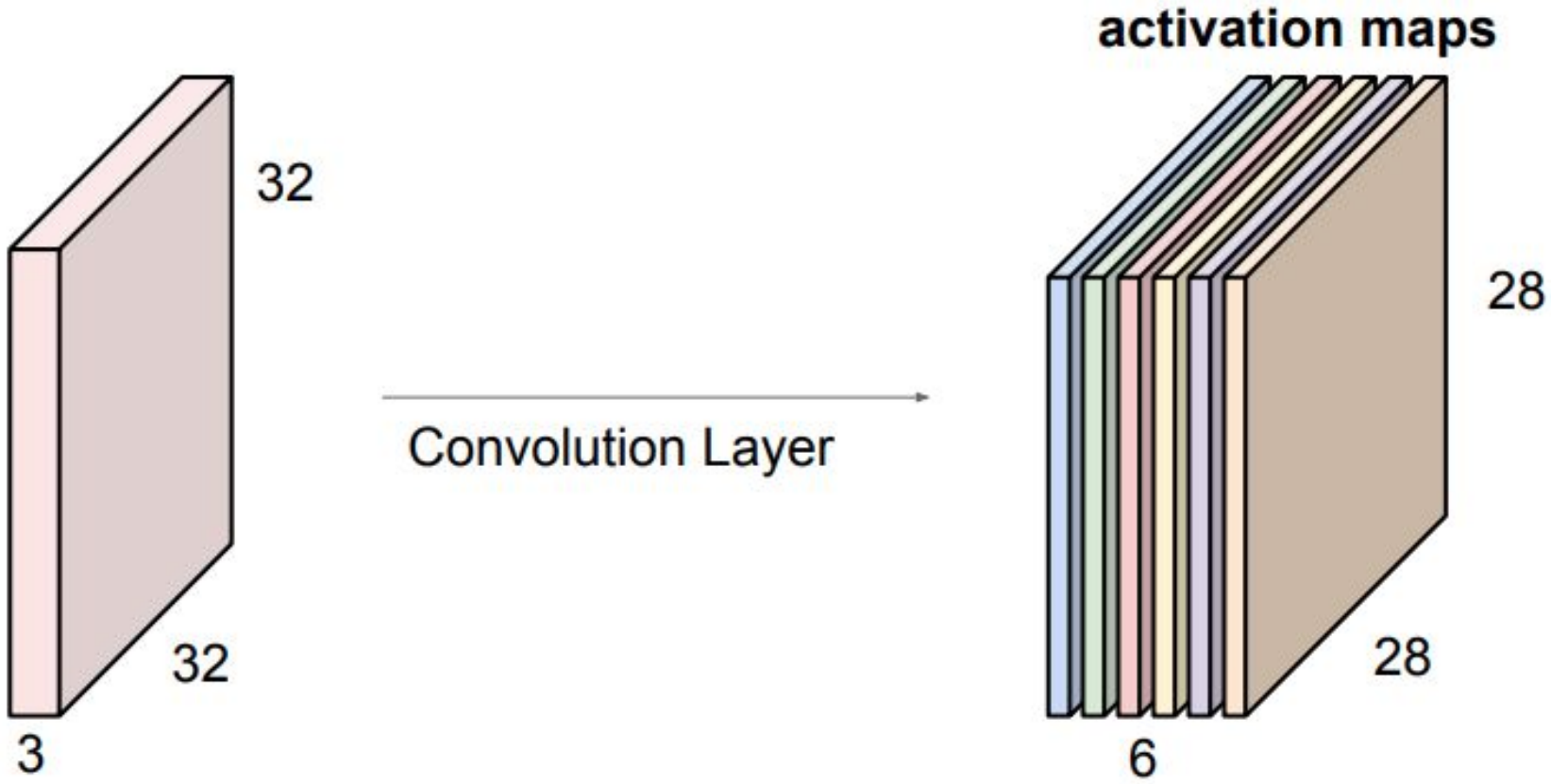



5x5x3 filter



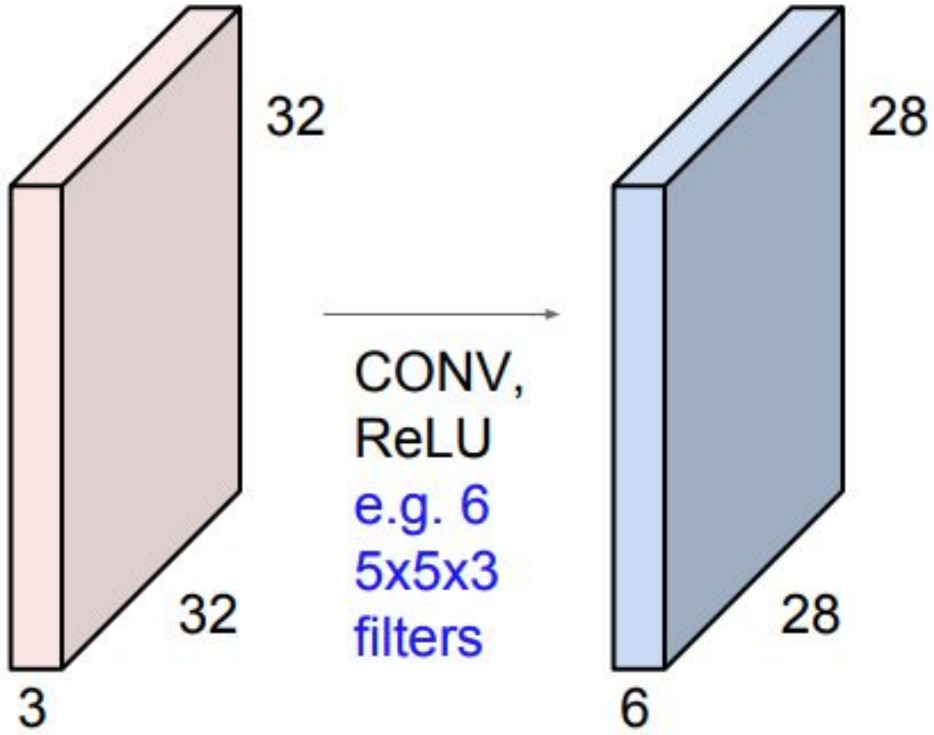
**Convolve** the filter with the image
i.e. “slide over the image spatially, computing dot products”

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



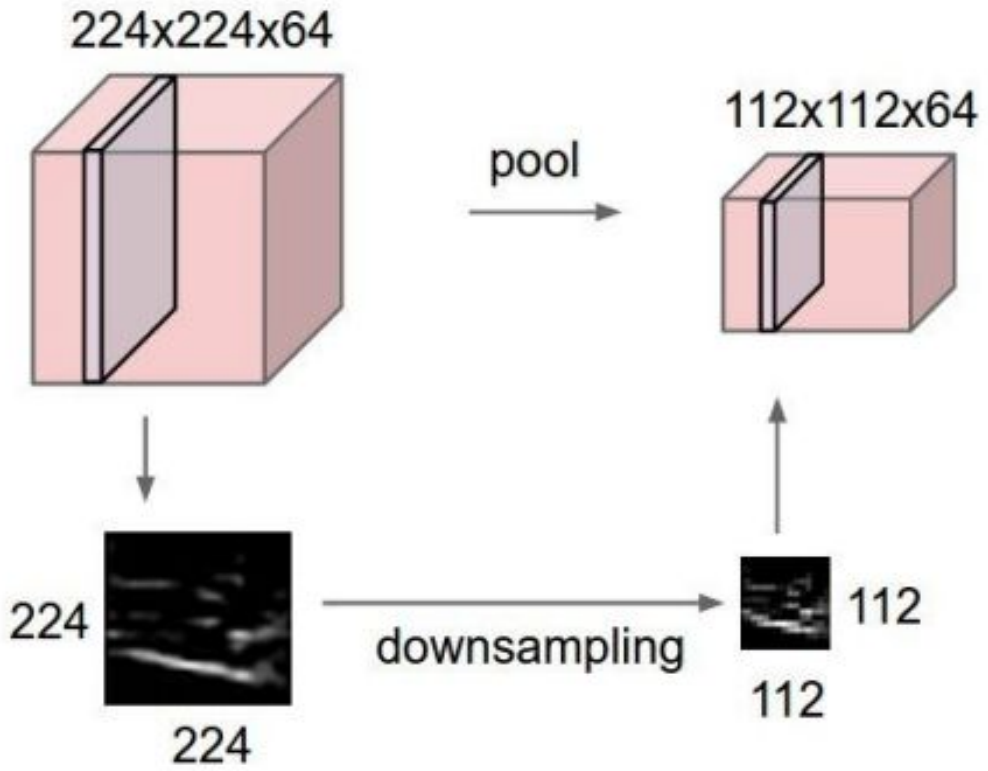

We stack these up to get a "new image" of size 28x28x6!

**Preview:** ConvNet is a sequence of Convolution Layers, interspersed with activation functions

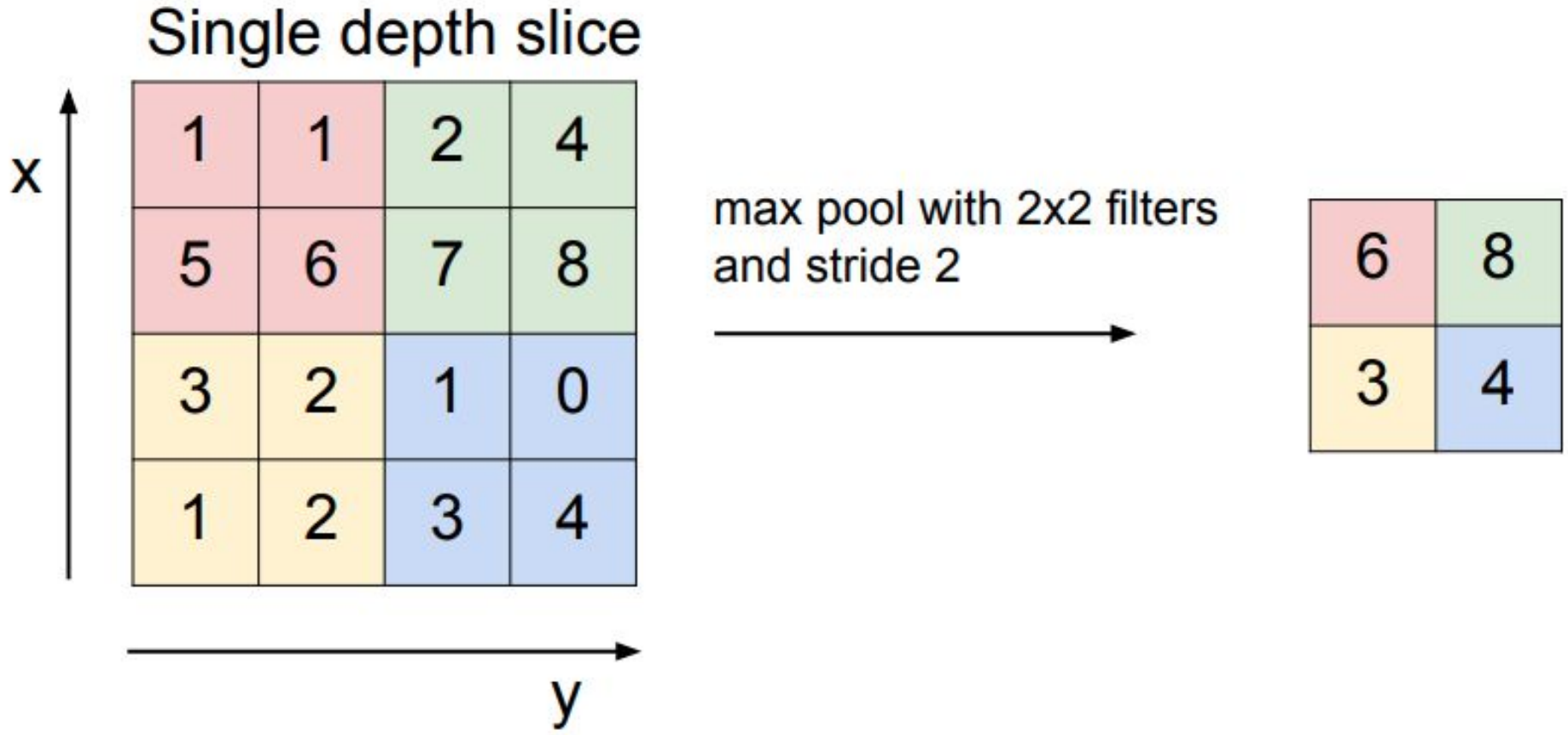


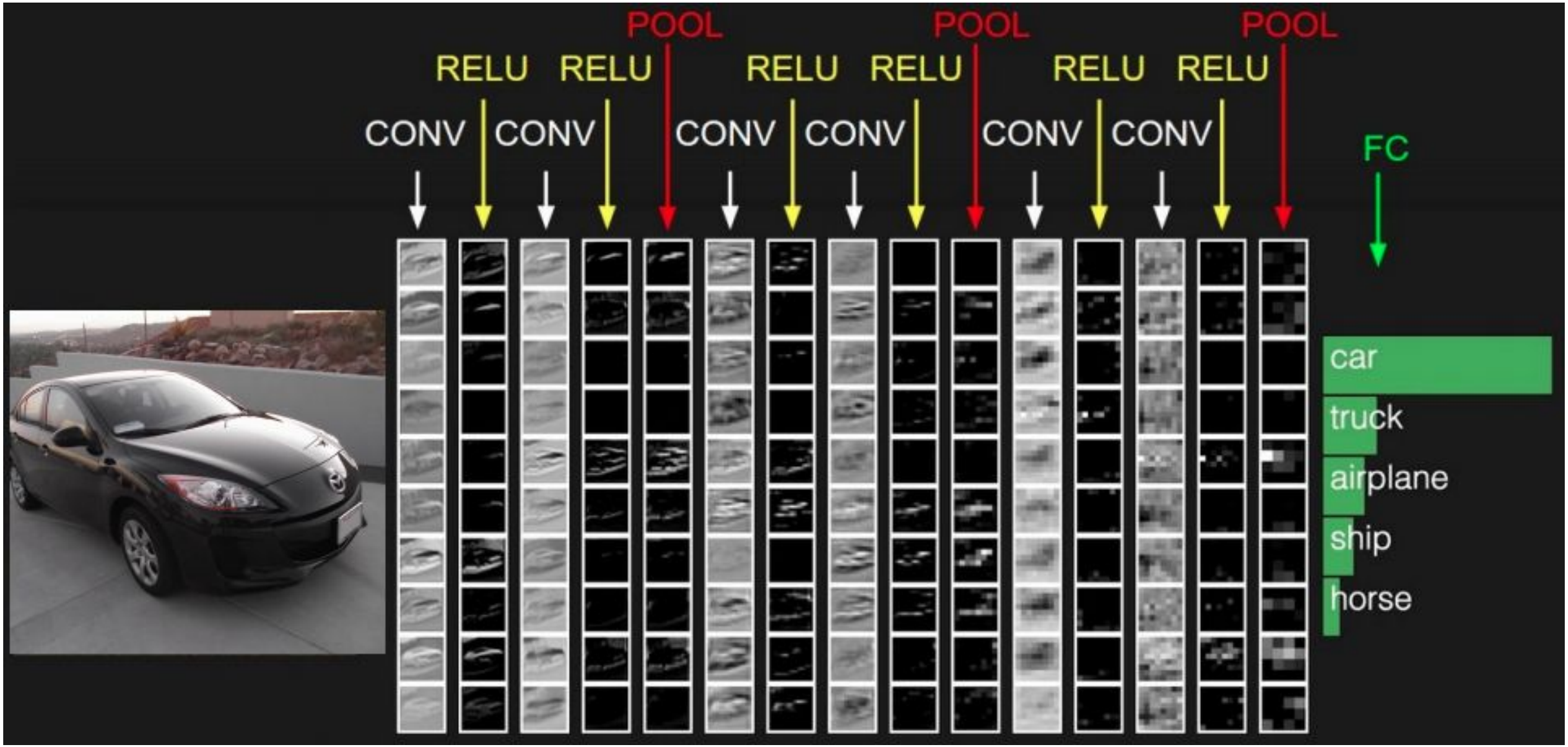
# Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



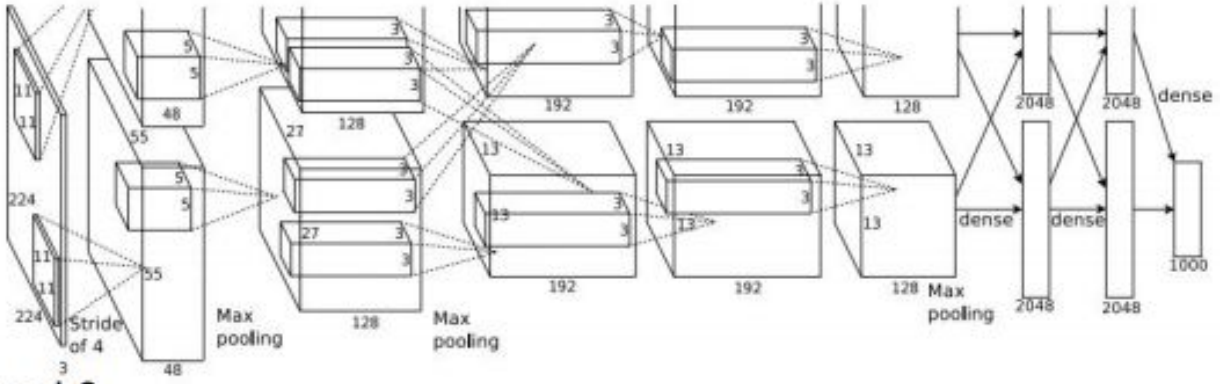
# MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

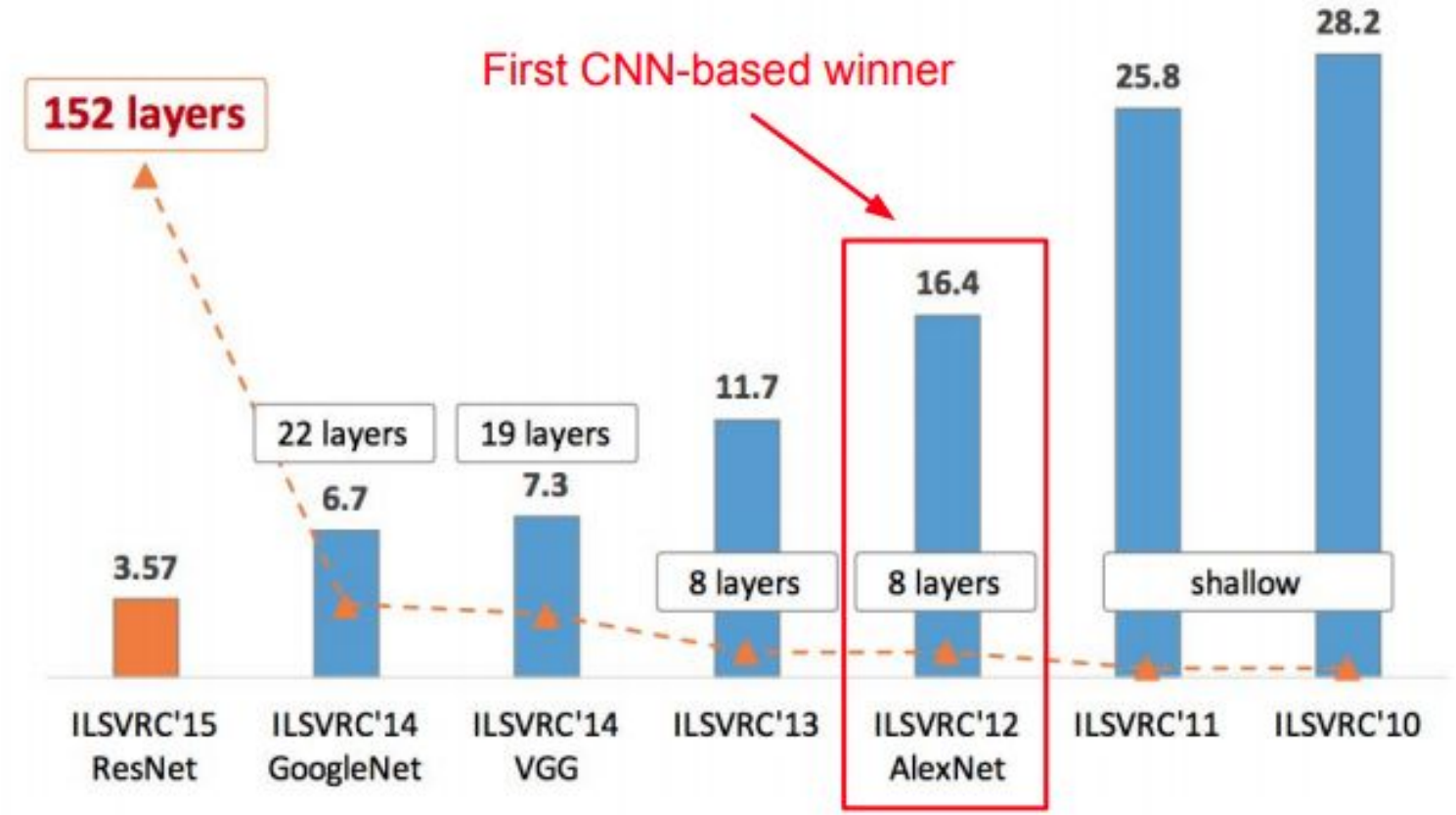
# Case Study: AlexNet

*[Krizhevsky et al. 2012]*




Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



# Case Study: VGGNet

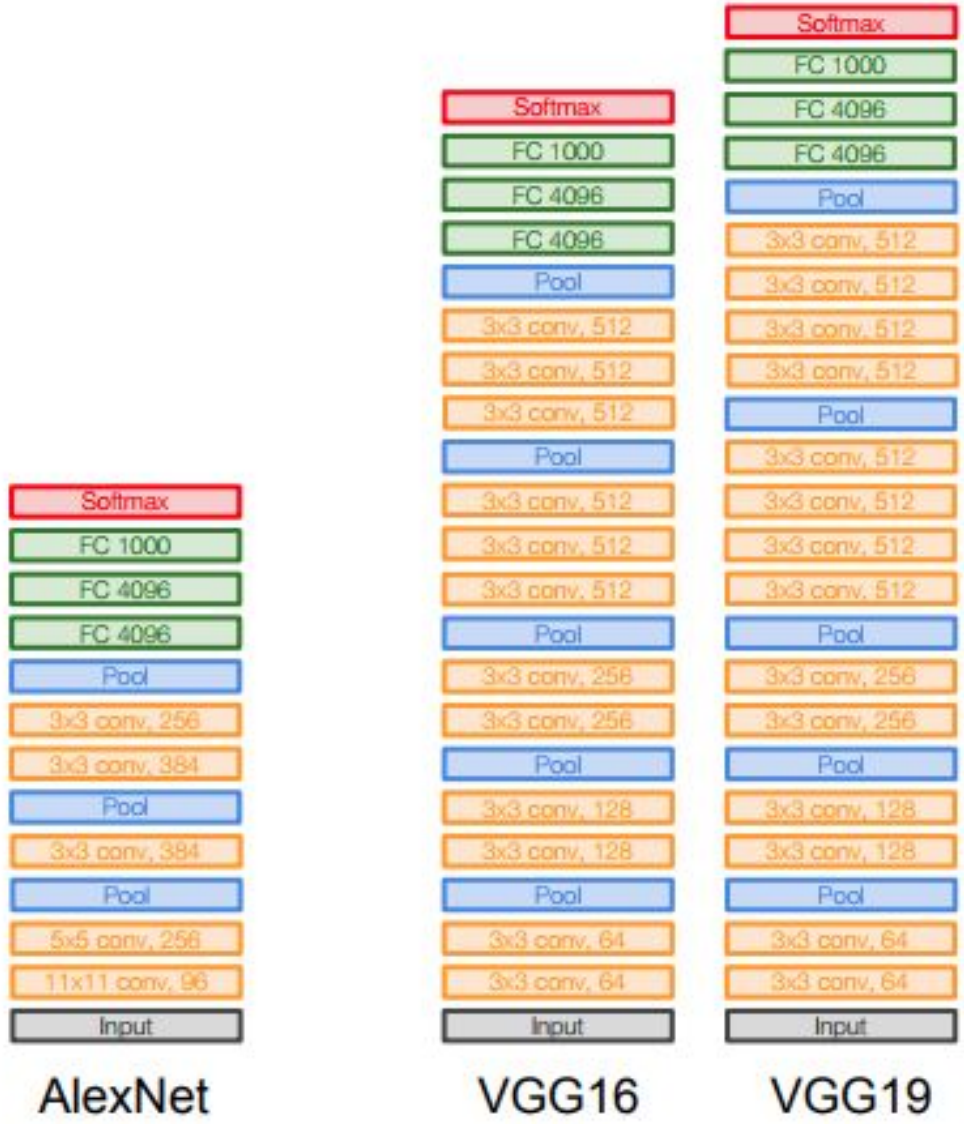
*[Simonyan and Zisserman, 2014]*

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14

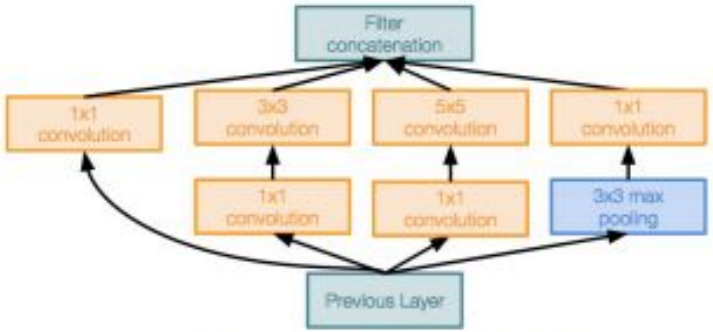


# Case Study: GoogLeNet

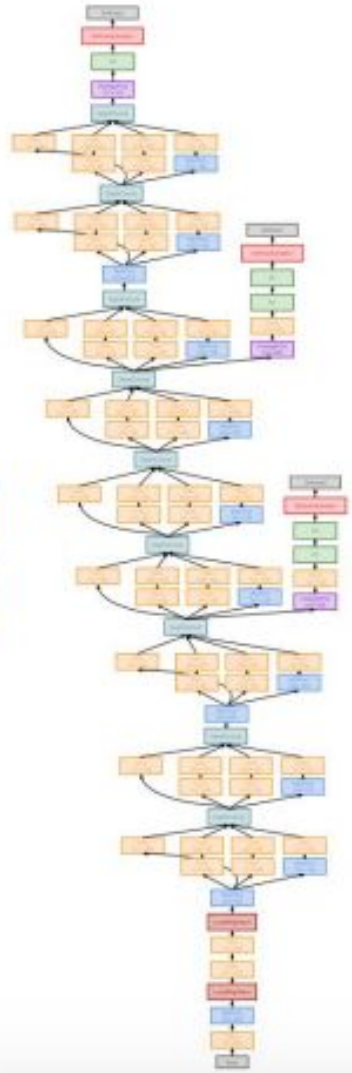
*[Szegedy et al., 2014]*

Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!
  12x less than AlexNet
- ILSVRC’14 classification winner
  (6.7% top 5 error)




Inception module

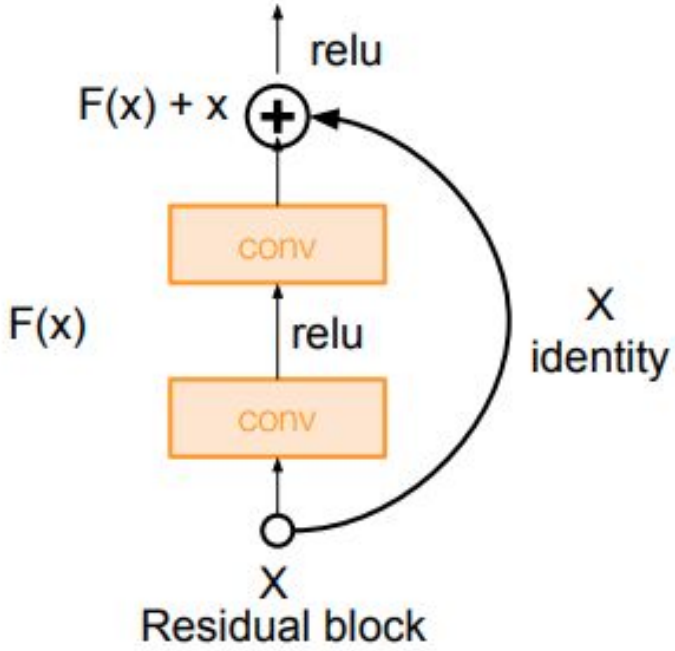


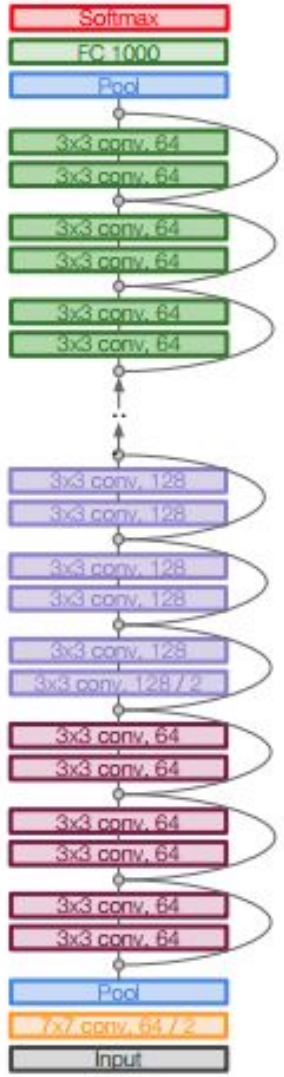
# Case Study: ResNet

*[He et al., 2015]*

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



# Summary: CNN Architectures

## Case Studies

- AlexNet
- VGG
- GoogLeNet
- ResNet

## Also....

- NiN (Network in Network)
- Wide ResNet
- ResNeXT
- Stochastic Depth
- DenseNet
- FractalNet
- SqueezeNet

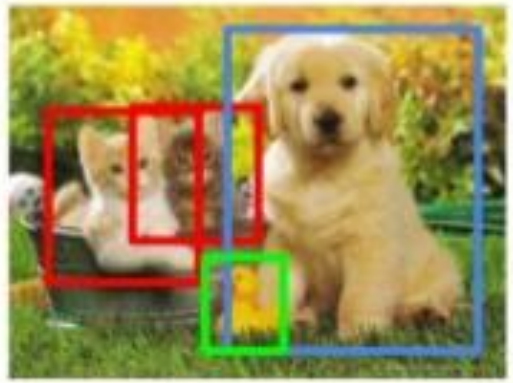
# Classification VS Object Detection

**Classification**

CAT

one image -> one label

**Object Detection**

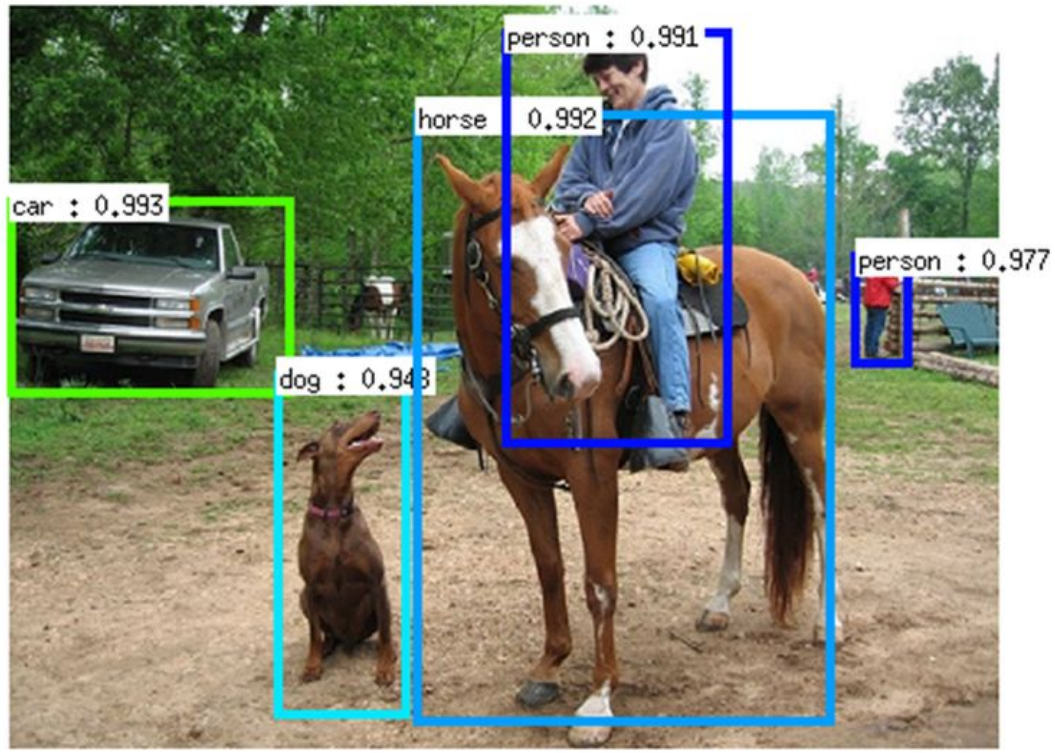


CAT, DOG, DUCK

one image -> labels + bounding boxes

# Object Detection Algorithms

- **R-CNN (2014)**
- **Fast R-CNN (2015)**
- **Faster R-CNN (2015)**
- **YOLO (2016)**
- **SSD and R-FCN  (2016)**



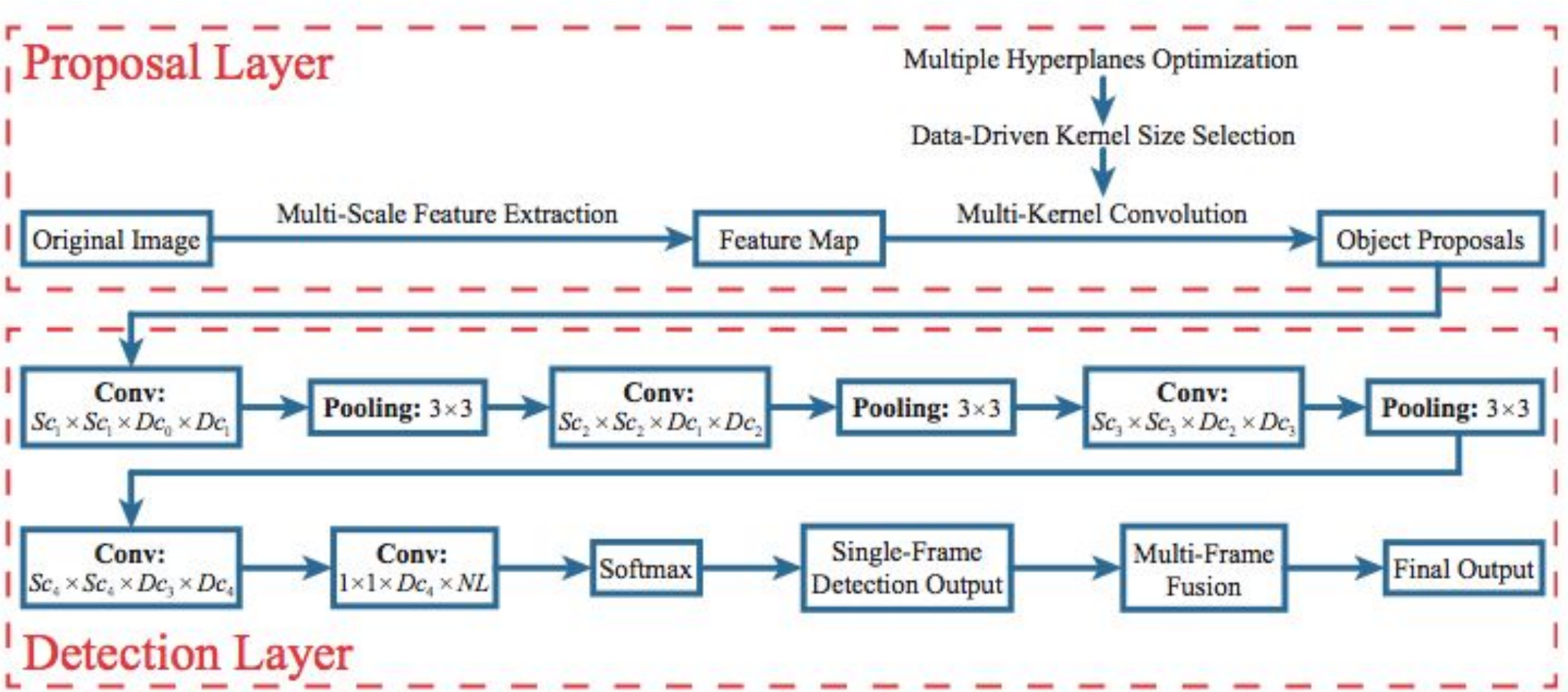
# YOLO VS SSD (Off line with GPU)


https://www.youtube.com/watch?v=8QL69cAj2kU

# Outline this Paper

- Section 1, introduction.
- Section 2,  introduce the proposal layer of our object detection network.
- Section 3, present the detection layer, including MPGA-based CNN and multi-frame fusion.
- Section 4  Experimental results.
- Section 5 Conclusions  and future works.

# Architecture



## Section 2, proposal layer (object detection)

- *2.1. Proposal feature extraction*
- *2.2. Convolution-based proposal generation*
- *2.3. Data-driven Kernel size selection*

## *2.1. Proposal feature extraction*

- Inspired by Dollár [11], initially extract a set of gradient histograms as the proposal feature of the given image, which is computationally efficient and able to represent the edge and texture of an object. Instead of generating features from grey level images [12], we compute gradient histograms over RGB color images. The respective horizontal and vertical gradients of each color channel can be obtained by applying the Sobel filter [−1, 0, 1] and [−1,0,1]T on the image.

# 2.1. Proposal feature extraction

$$\begin{cases} Gx(i,j) = \max(Grx(i,j), Ggx(i,j), Gbx(i,j)) \\ Gy(i,j) = \max(Gry(i,j), Ggy(i,j), Gby(i,j)) \end{cases}, \quad (1)$$

where $(Gr, Gg, Gb)$ denotes the gradients of the three color channels. Having defined the gradient of each pixel, the magnitude $M(i, j)$ and orientation $Ang(i, j)$ of gradient can be written as:

$$\begin{cases} M(i,j) = \sqrt{Gx(i,j)^2 + Gy(i,j)^2} \\ Ang(i,j) = \arctan\left(\frac{Gy(i,j)}{Gx(i,j)}\right) \end{cases}. \quad (2)$$

$Ang(i, j)$ of each pixel is then quantized into $k$ bins. Let $b(i, j)$ denote the bin that the pixel $(i, j)$ belongs to; it can be obtained by:

$$b(i,j) = \left\lfloor \frac{k \cdot Ang(i,j)}{2\pi} \right\rfloor, \quad (3)$$

## *2.1. Proposal feature extraction*

- To compute the gradient histogram of each location, we divide the image into small spatial regions. A larger size of the regions will result in a lower localization precision, while a smaller size of that will lead to a higher dimension of feature vector, which will increase the computational complexity or even cause the curse of dimensionality for SVM [28]. To balance the localization precision and dimension of feature vector, we use the region size of (4× 4) for calculating the gradient histogram.

# 2.1. Proposal feature extraction

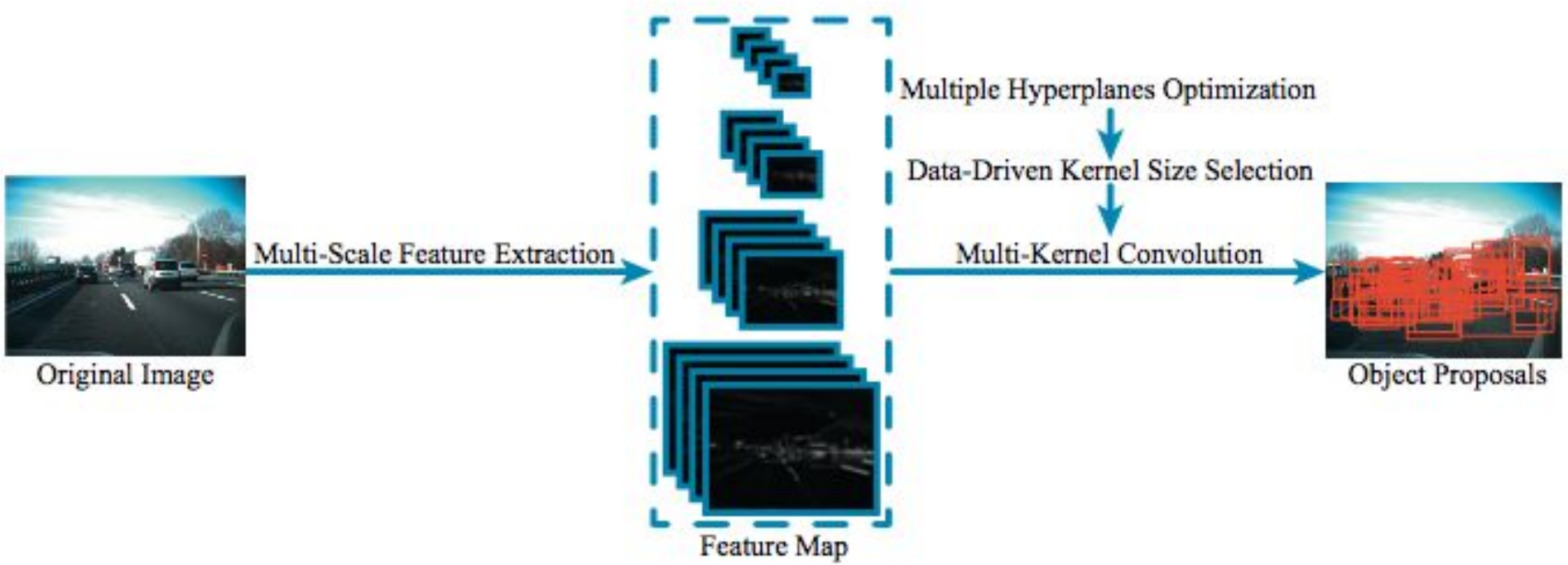



**Fig. 2.** Architecture of the proposal layer.

# 2.2. Convolution-based proposal generation

Convolution becomes a highly efficient operation in CPU-based image processing owing to the support of many proven algorithm libraries such as MKL [29], BLAS [30] and Caffe (CPU version) [31]. According to this fact, we attempt to achieve efficient proposal generation by transforming the dense sliding windows paradigm [10,14] to convolution operation.

As we know, convolution is similar to sliding windows paradigm. That is, each element in convolution result is correspond to a spatial region in input map. Let $\vec{x}_{i,j}$ denote the proposal feature vector of the bounding-box region which is correspond to the element located at the $i$th row and $j$th column of the gradient histogram feature space (center-aligned). The prediction of linear support vector machine (SVM) [13] can be formulated as the form of generic linear regression:

$$y = \vec{w} \cdot \vec{x} + b, \tag{5}$$

# Pipeline of convolution-based proposal generation

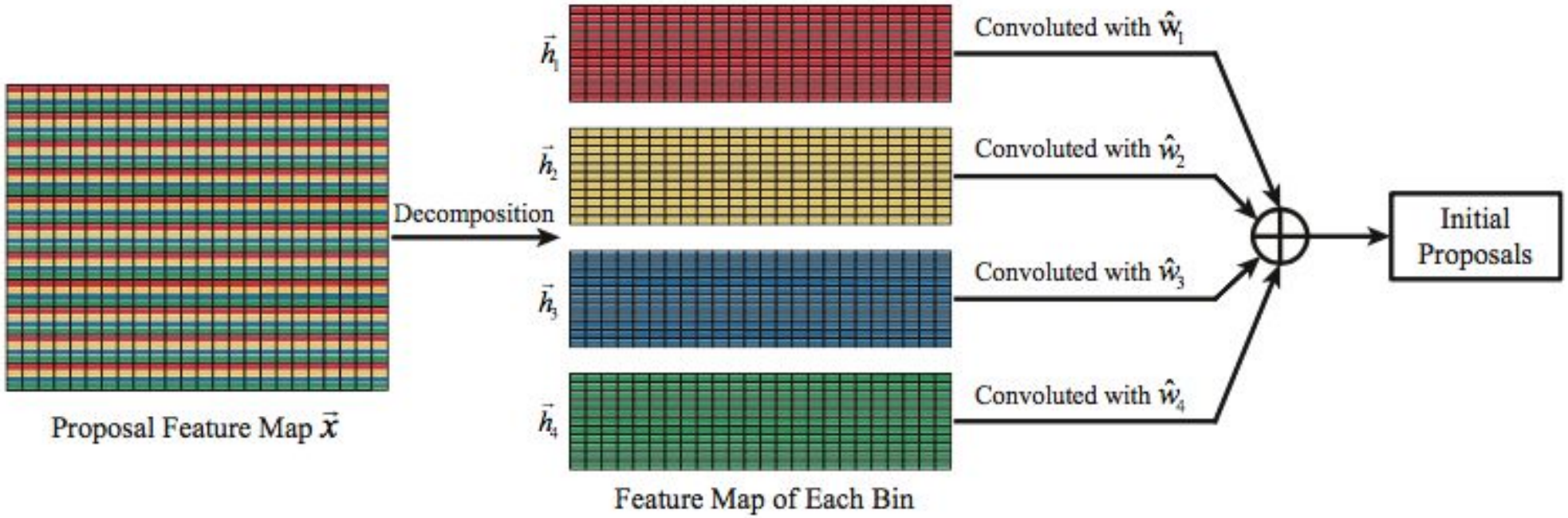



**Fig. 3.** Pipeline of convolution-based proposal generation (each color represents a histogram channel).

## *2.3. Data-driven Kernel size selection*

- propose to select the size of bounding-box using a data-driven approach, which is an off-line procedure and only needs to be performed once.
- We illustrate the bounding-box size selection problem using the TME Motorway dataset [3],



For detailed explanation see paper, paragraph IV.

# TME Motorway Dataset

The "Toyota Motor Europe (TME) Motorway Dataset" is composed by 28 clips for a total of approximately 27 minutes (30000+ frames) with vehicle annotation. Annotation was semi-automatically generated using laser-scanner data. Image sequences were selected from acquisition made in North Italian motorways in December 2011. This selection includes variable traffic situations, number of lanes, road curvature, and lighting, covering most of the conditions present in the complete acquisition.

The dataset comprises:

- Image acquisition: stereo, 20 Hz frequency , 1024x768 grayscale losslessly compressed images, 32° horizontal field of view, bayer coded color information (in OpenCV use CV_BayerGB2GRAY and CV_BayerGB2BGR color conversion codes; please note that left camera was rotated upside down, convert to color/grayscale BEFORE flipping the image). A checkboard calibration sequence is made available.
- Ego-motion estimate (confidential computing method).
- Laser-scanner generated vehicle annotation and classification (car/truck).
- A software evaluation toolkit (C++ source code).

The data provided is timestamped, and includes extrinsic calibration.
The dataset has been divided in two sub-sets depending on lighting condition, named "daylight" (although with objects casting shadows on the road) and "sunset" (facing the sun or at dusk). For each clip, 5 seconds of preceding acquisition are provided, to allow the algorithm stabilizing before starting the actual performance measurement.

The data has been acquired in cooperation with VisLab (University of Parma, Italy), using the BRAiVE test vehicle.

# 2.3. Data-driven Kernel size selection

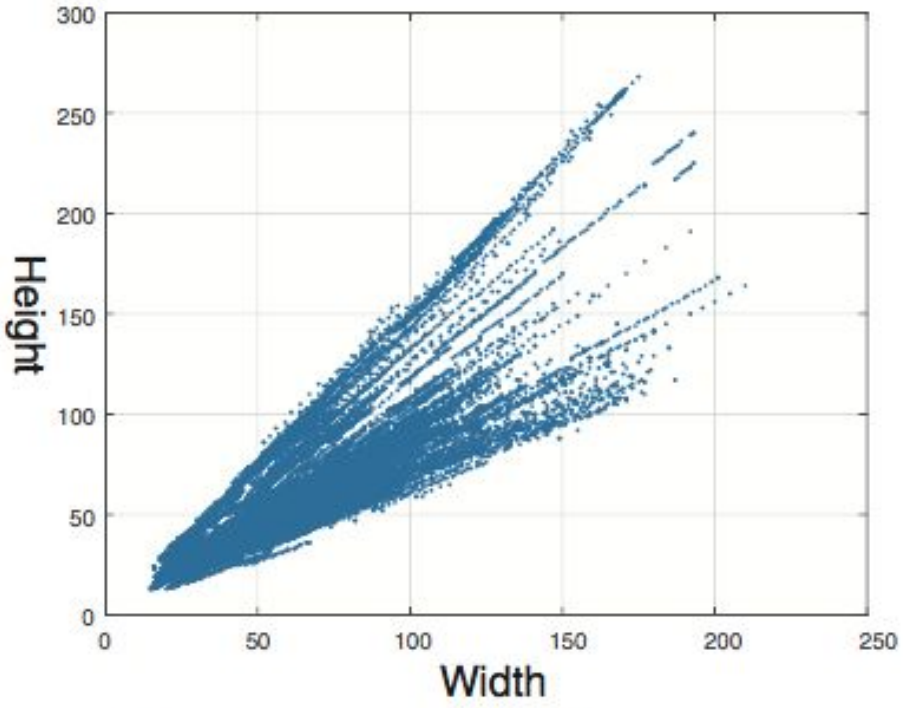



**Fig. 4.** The distribution of object size in TME Motorway dataset.

Apply

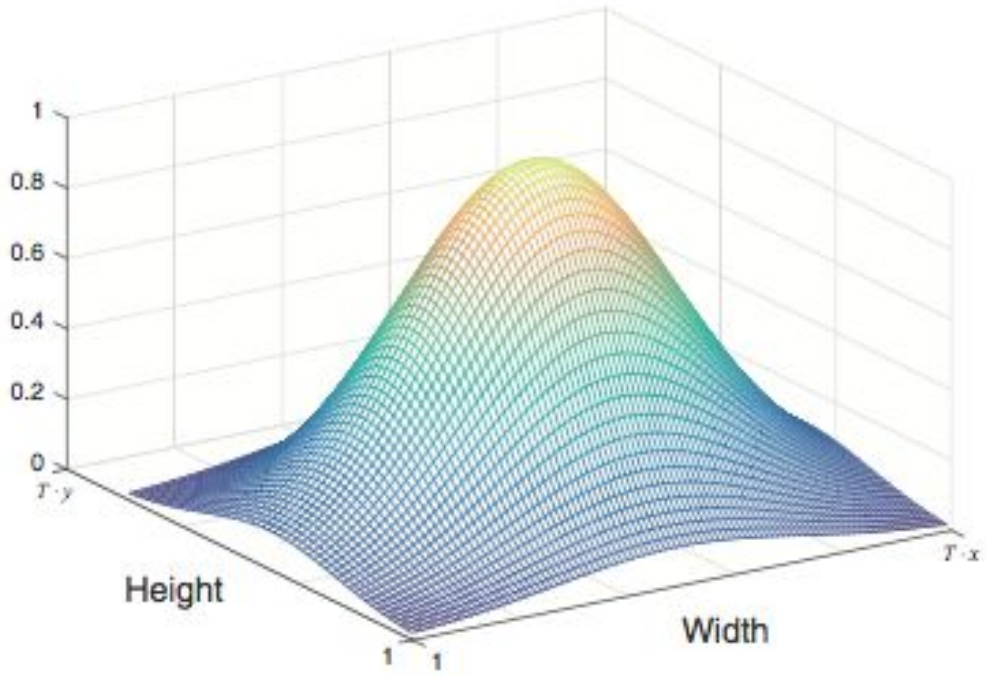



**Fig. 6.** 2D Gaussian filter $G_{x,y}(u, v)$ for location $(x, y)$ in distribution map.

# 2.3. Data-driven Kernel size selection

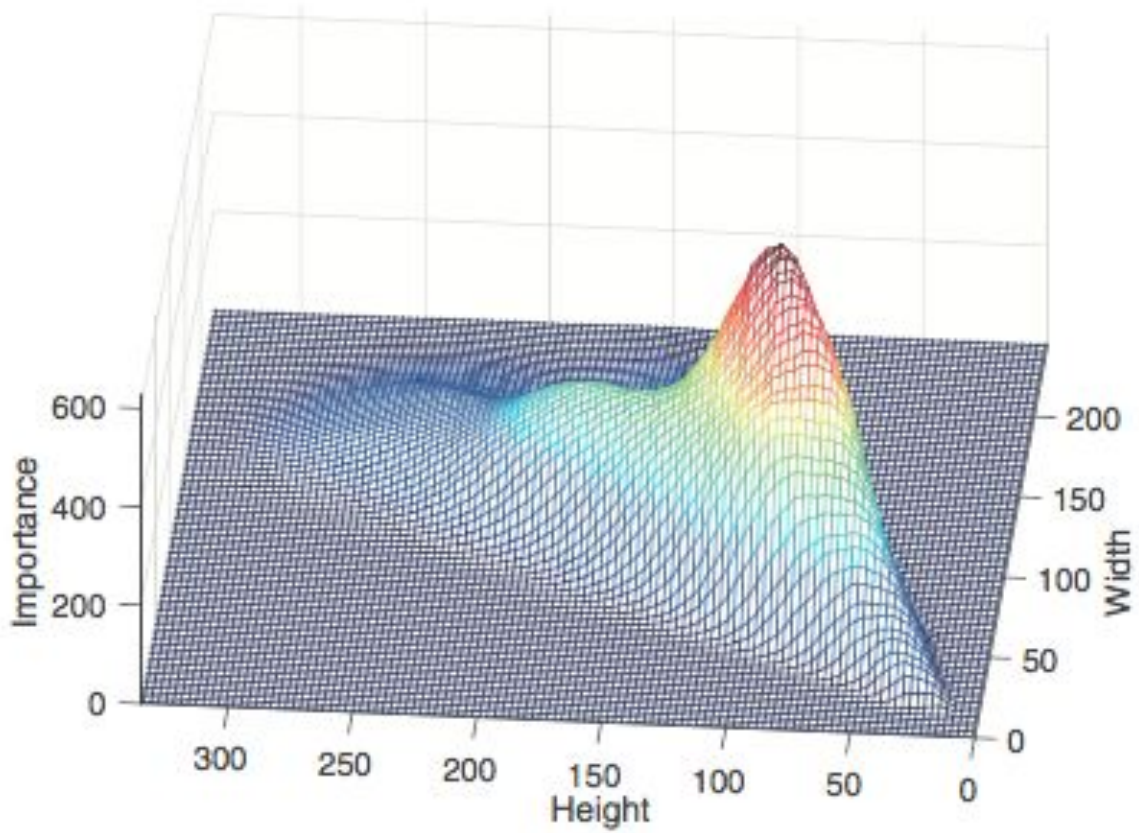



**Fig. 7.** Original importance map.

# 2.3. Data-driven Kernel size selection

**Algorithm 1** Bounding-box size selection.

**Input:** Original distribution map $Ds_0$;
The number of bounding-box types $q$;
**Output:** $Wm$ and $Hm$;
**Initialization:** $Wm = \varnothing$, $Hm = \varnothing$ and $i$=0;
**Procedure:**

1: Transform $Ds_i$ to $Es_i$ by applying $G_{x,y}$ on it;
2: Find the position $(xm_i, ym_i)$ which has the maximum value in $Es_i$;
3: Update $Wm$ and $Hm$: $Wm = Wm \bigcup \{xm_i\}$,
$Hm = Hm \bigcup \{ym_i\}$;
4: If $i < q$ then calculate $Ds_{i+1}$ by applying $Gs_{xm_i,ym_i}$ at $(xm_i, ym_i)$ of $Ds_i$;
5: Set $i \leftarrow i + 1$;
6: If $i \leq q$ then go to step 1, otherwise stop the iteration.

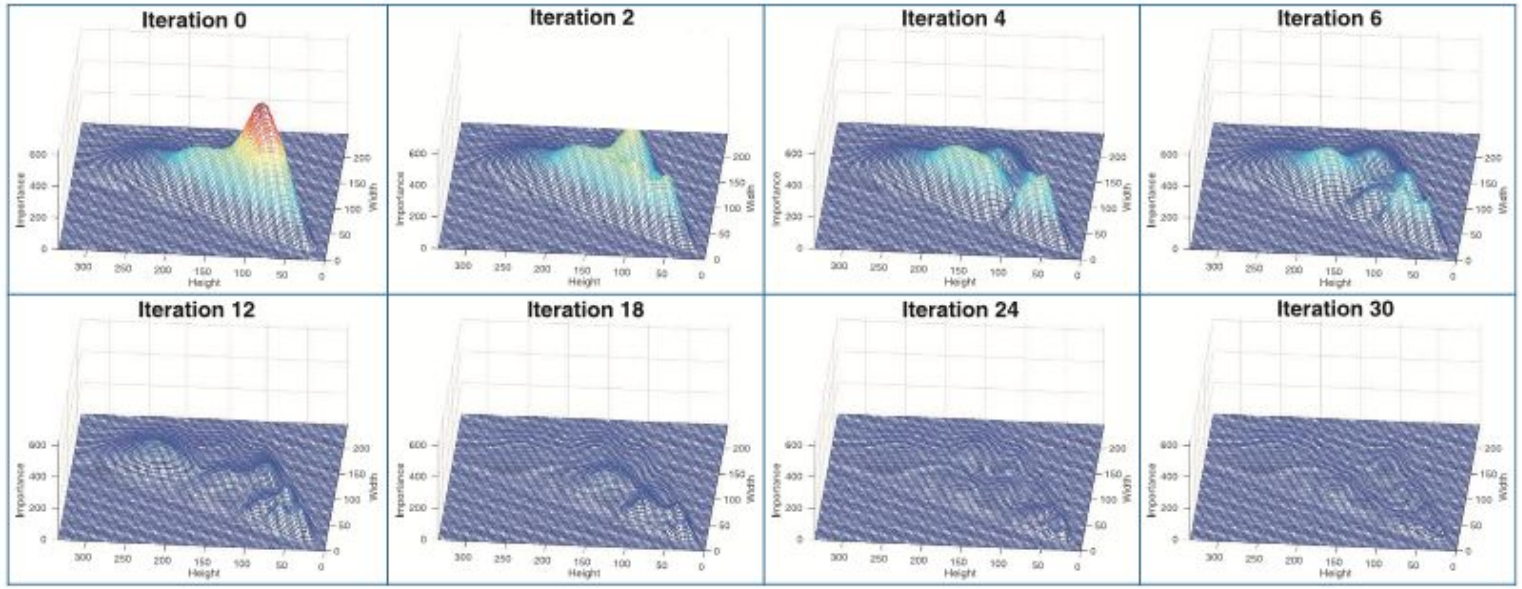



**Fig. 9.** The change of importance map with the increasing of iterations.

# 2.3. Data-driven Kernel size selection

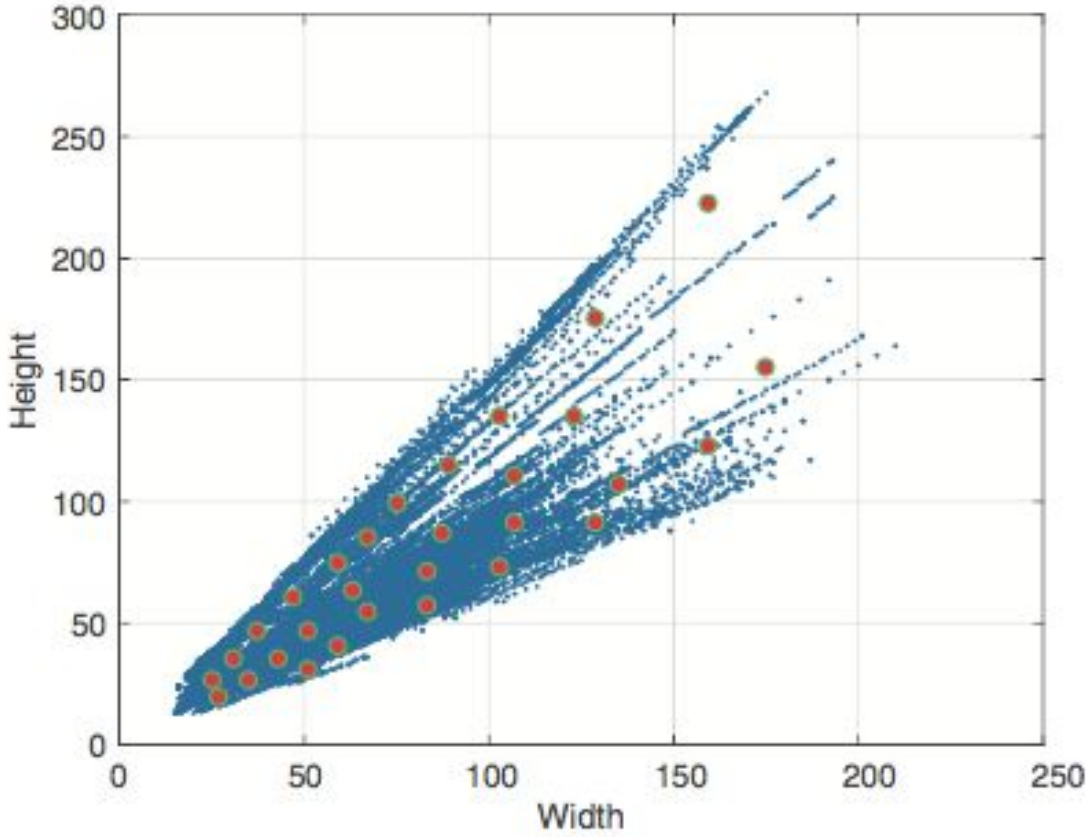



**Fig. 8.** The result of bounding-box selection. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

# 3. Detection layer

we first propose an MPGA-based convolutional neural network (CNN) module to balance the performance and computational complexity of deep convolutional neural networks. Then, we design a multi-frame fusion strategy to utilize temporal information and improve detection performance.

*3.1. Constructing convolutional neural networks using MPGA*
*3.2. Multi-frame fusion*

# 3. Detection layer

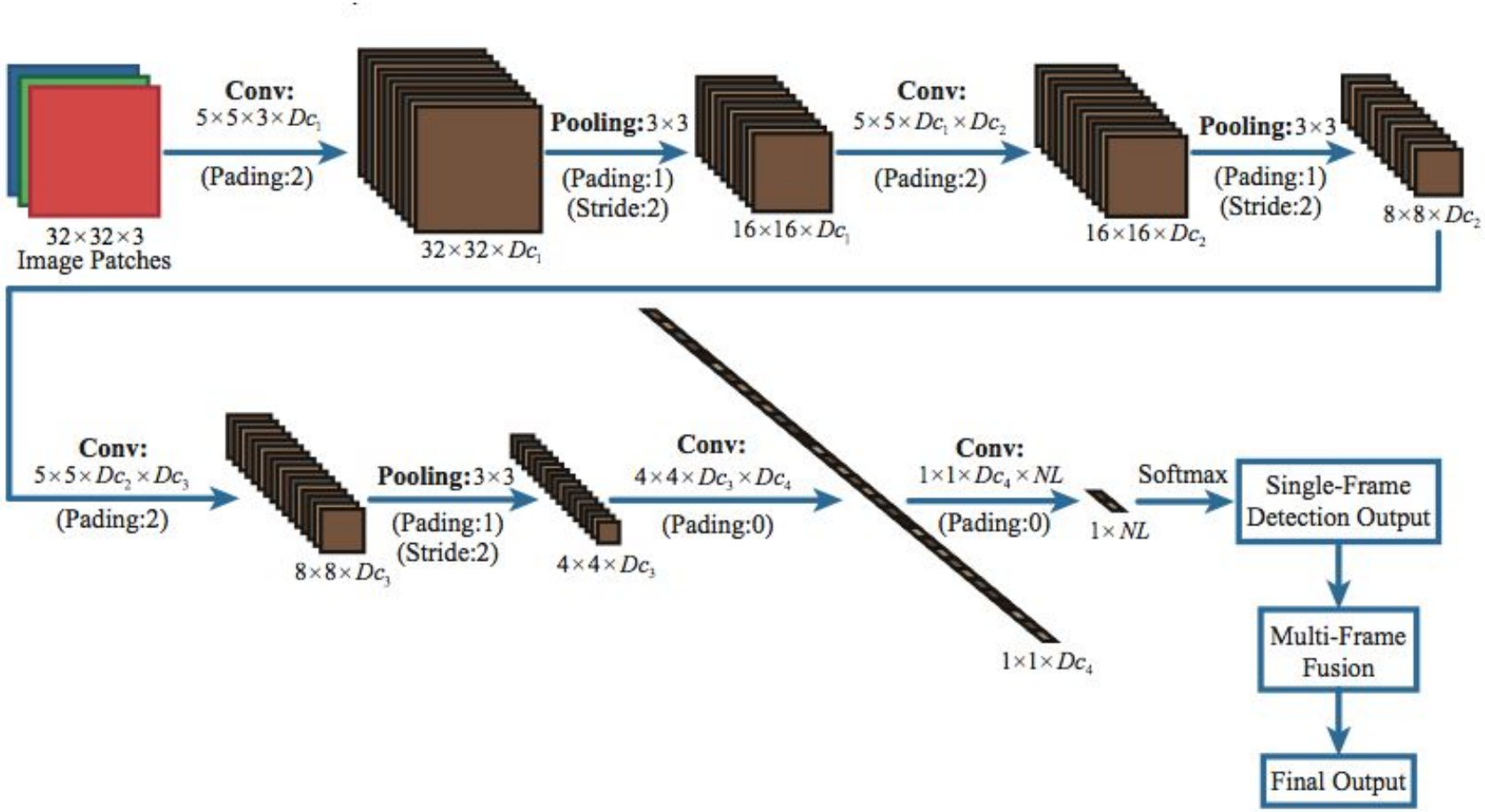



**Fig. 11.** Architecture of the detection layer.

## *3.1. Constructing CNN using MPGA*

- MPGA is an optimization algorithm inspired by biology, whose object function is not required to be continuous or differentiable. Moreover, MPGA is more likely than traditional genetic algorithms to obtain global optimal solutions [41]. In MPGA, the variables that need to be optimized are first encoded to chromosomes, and then two or more populations that contain several chromosomes are generated. The populations evolve in parallel until they converge to a stable status. In this way, the optimal solutions are obtained.

# 3.1. Constructing CNN using MPGA

- For the optimization problem in this work, since $Dc0$ and $Dc5$ are determined (see Eq. (30)), we first transform $\vec{D}c = [Dc1, Dc2, Dc3, Dc4]$ to binary codes which act as chromosomes of MPGA. Then $Np$ populations are generated (each population is made up of $Nc$ chromosomes). The fitness function $Fit(\vec{D}c, er)$ of MPGA is defined as:

$$Fit(\vec{D}c, e_r) = -\frac{Oc(\vec{D}c) + Ks \cdot (e_r - e_p)^2}{1 + Ks}, \tag{37}$$

where $e_r$ and $e_p$ are the training and expected error respectively. $Oc(\vec{D}c)$ is the structure complexity, which is formulated as:

$$Oc(\vec{D}c) = \sum_{i=1}^{4} Sc_i^2 \cdot Dc_{i-1} \cdot Dc_i, \tag{38}$$

# 3.2. Multi-frame fusion

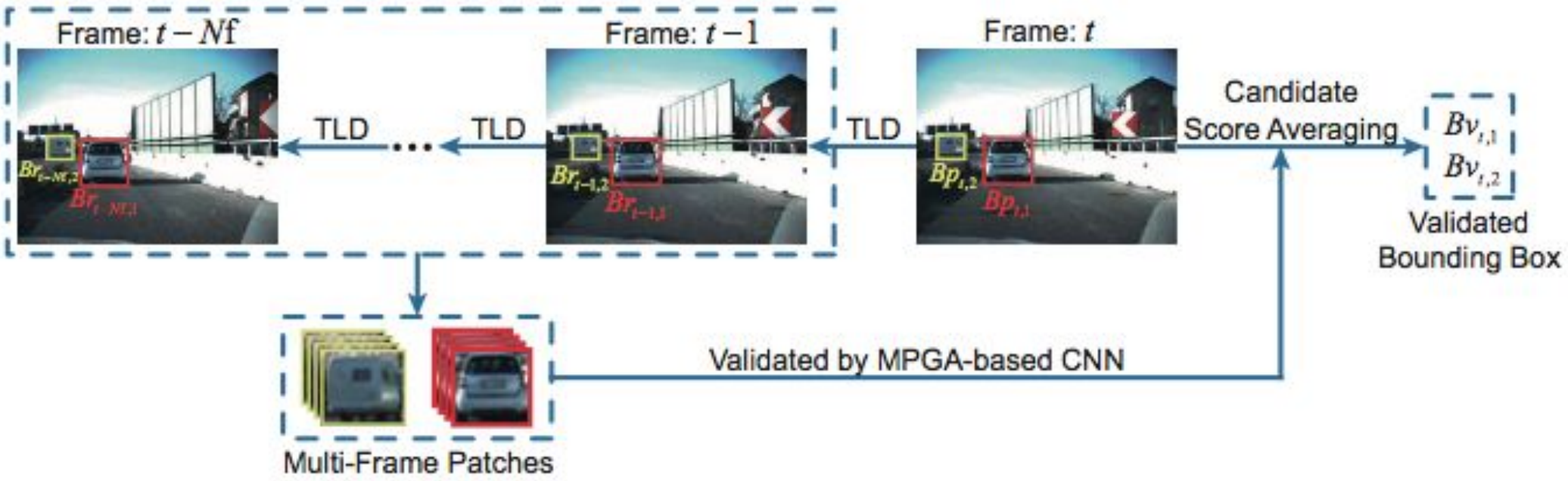


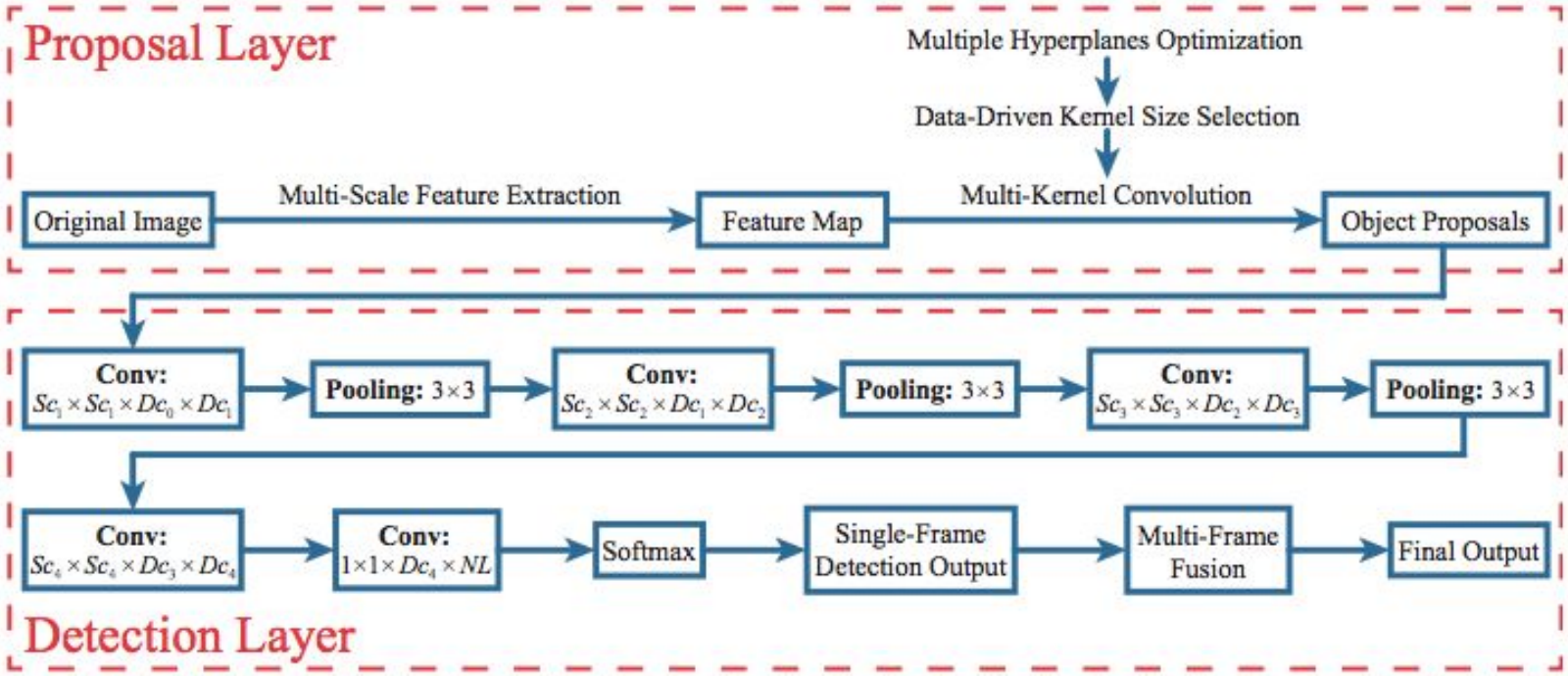

**Fig. 12.** Illustration of multi-frame validation.

## *3.2. Multi-frame fusion*

- the probability of a non- object region giving a false positive in several consecutive frames is much lower than that in a single frame.
- Image region is detected to be positive in several consecutive frames, then there is a high probability that the image region is an object re-gion.
- multi-frame fusion can effectively improve the true positive rate and yet reduce the false positive rate. More- over, detection in different frames can be smoothed by multi-frame fusion.

# 4. Experimental results

- *4.1. The proposal layer*
- *4.2. MPGA-based CNN*
- *4.3. Full system*



# 4.1. The proposal layer

**Table 1**
The evaluations of proposal layer.

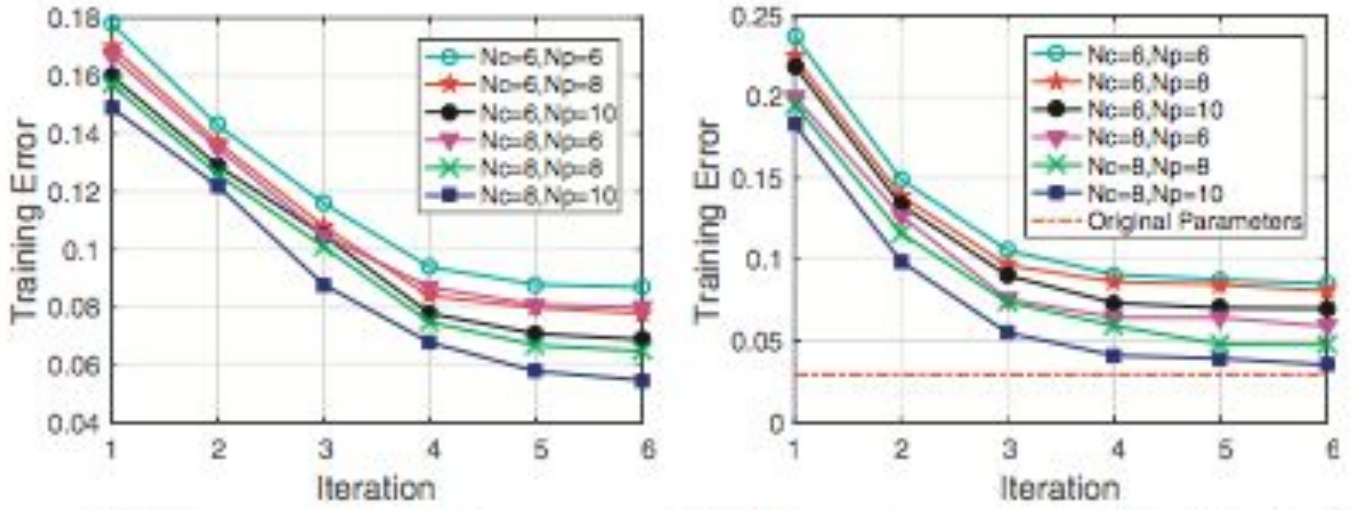
| Method | TME motorway dataset [3] | | | ETHZ pedestrian dataset [47] | | |
|---|---|---|---|---|---|---|
| | Recall@2000 | N@90% | Time | Recall@2000 | N@90% | Time |
| Emp-Ker(15) | 80.2% | 4282 | **0.1 s** | 79.7% | 3157 | **0.1 s** |
| Emp-Ker(20) | 82.1% | 3418 | **0.1 s** | 82.5% | 2439 | **0.1 s** |
| Emp-Ker(25) | 82.9% | 2773 | 0.2 s | 85.8% | 2076 | 0.2 s |
| Emp-Ker(30) | 84.7% | 2165 | 0.2 s | 87.1% | 1634 | 0.2 s |
| Driven-Ker(15) | 81.9% | 3279 | **0.1 s** | 81.6% | 2421 | **0.1 s** |
| Driven-Ker(20) | 84.6% | 2425 | **0.1 s** | 83.2% | 1863 | **0.1 s** |
| Driven-Ker(25) | 87.7% | 1973 | 0.2 s | 86.9% | 1647 | 0.2 s |
| Driven-Ker(30) | 90.8% | 1695 | 0.2 s | 89.3% | 1265 | 0.2 s |
| Driven-Ker(30) + MHO(0.1) | 93.2% | **1217** | 0.2 s | 92.7% | **1076** | 0.2 s |
| Driven-Ker(30) + MHO(0.4) | 93.8% | 1934 | 0.2 s | 93.3% | 1856 | 0.2 s |
| Driven-Ker(30) + MHO(0.7) | 94.3% | 2776 | 0.2 s | 93.8% | 2749 | 0.2 s |
| Driven-Ker(30) + MHO(1.0) | **95.1%** | 3853 | 0.3 s | **94.2%** | 3558 | 0.3 s |
| Edge Boxes [21] | 92.5% | 1754 | 0.2 s | 91.6% | 1584 | 0.2 s |
| Selective Search [20] | 92.7% | 2276 | 7.3 s | 92.0% | 2362 | 8.2 s |

laptop which contains an Intel i7-4700MQ processor running at 2.4 GHz. Except for off-line training, all the detection processes in the experiments run without GPU support.

# 4.2. MPGA-based CNN

**Table 2**
The training epochs of MPGA-based CNN.

| $Nt$ \ $Np$ / $Nc$ | 2 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|
| 4 | 40 | 80 | 120 | 128 | 200 |
| 6 | 48 | 144 | 108 | 288 | 240 |
| 8 | 80 | 192 | 144 | 256 | 480 |




(a) The structure of ours (b) The structure of AlexNet [15]

**Fig. 13.** The training error of MPGA-based CNN.

# 4.3. Full system

**Table 3**
Run-time analysis on TME motorway dataset.

| Method | Average run-time |
|---|---|
| Caraffi [3] | 0.1 s |
| Castangia [48] | 0.05 s |
| Fast R-CNN [22] | 2.7 s |
| Ours (without proposal) | 43 s |
| Ours (without MF) | 0.23 s |
| Ours (full system) | 0.25 s |

**Table 4**
Run-time analysis of each on-line step on TME motorway dataset.

| Step | Average run-time |
|---|---|
| Feature extraction | 0.12 s |
| Multi-kernel convolution | 0.08 s |
| CNN (path forward) | 0.03 s |
| Multi-frame fusion | 0.02 s |
| **Total** | 0.25 s |

# 4.3. Full system

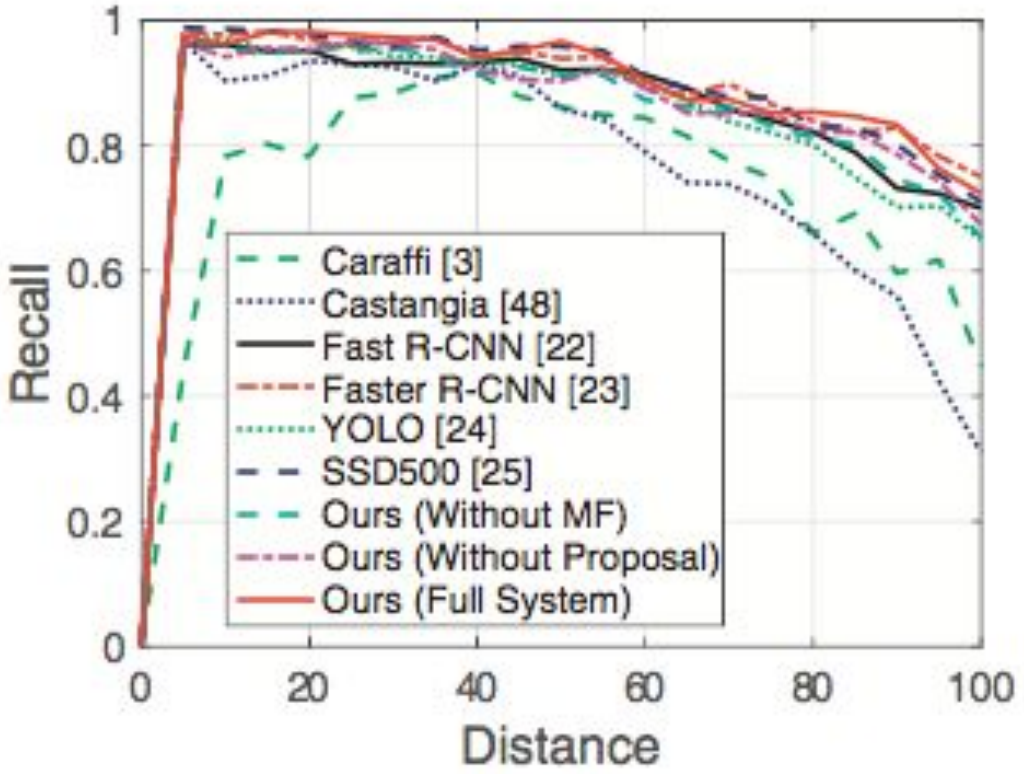



**Fig. 14.** Recall in function of vehicle distance on TME motorway dataset.

# 4.3. Full system

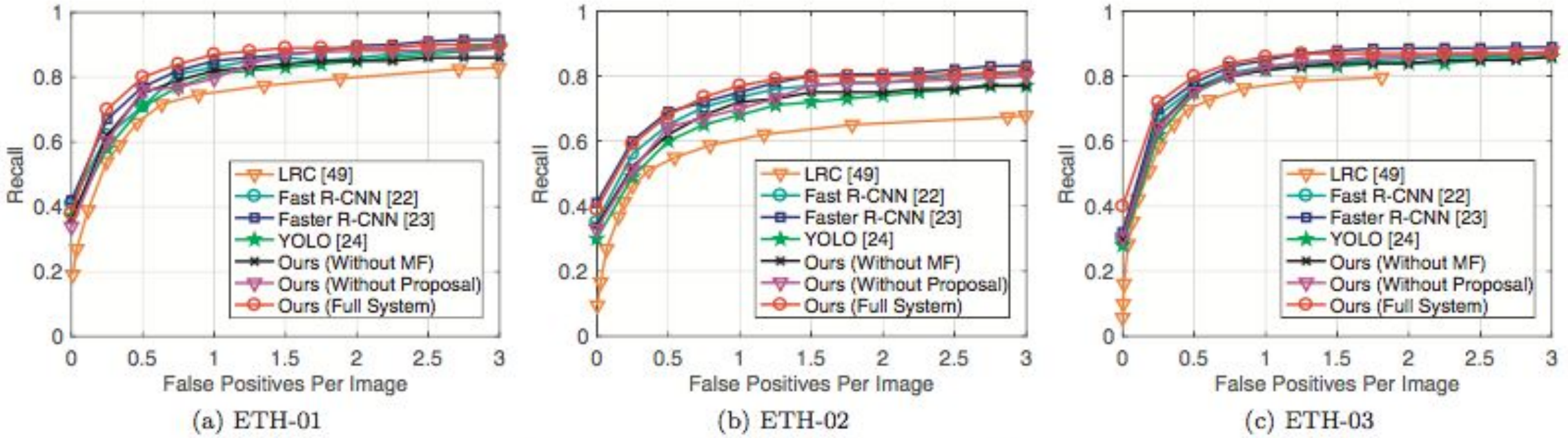
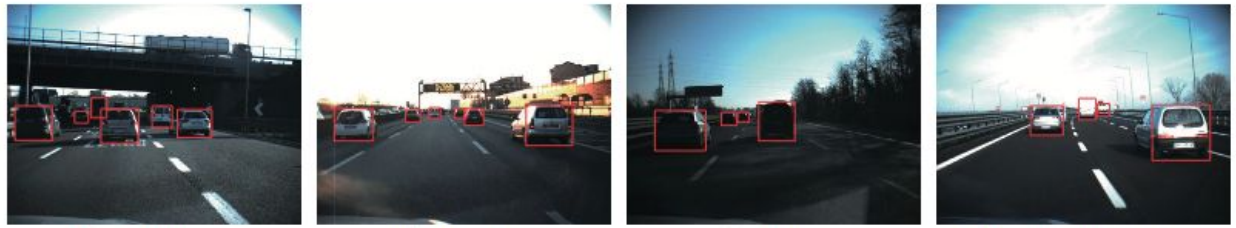



**Fig. 15.** Performance evaluation on ETHZ pedestrian dataset.

# 5. Conclusions and future works

- In the experiments, we validate each component of our pro- posed object detection system and compare the system with some recently published state-of-the-art object detection algorithms on widely used datasets. Even under challenging scenarios with shadows, backlighting, different viewpoints, overlapping objects or ambiguous boundaries, the proposed system is proved to be both efficient and effective in object detection.



(a) Common scene (b) Backlighting (c) Shadows (d) Ambiguous boundary

(e) Common scene (f) Backlighting (g) Sideview (h) Overlapping objects

**Fig. 16.** Qualitative analysis of our proposed method.

# 5. Conclusions and future works

- the proposal layer is able to work efficiently and reliably. Then, we construct a robust detection layer which works on the proposal layer. The detection layer involves an MPGA-based convolutional neural network (CNN) and a TLD-based multi-frame fusion procedure

# Reference


- Keyu Lu, Xiangjing An, Jian Li, Hangen He, Efficient deep network for vision-based object detection in robotic applications (2017)
- Slide CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.stanford.edu/syllabus.html


Thank You