SmartData

NEUROMATION

# DEEP NEURAL NETWORKS FOR OBJECT DETECTION

**Sergey Nikolenko**

Steklov Institute of Mathematics at St. Petersburg

October 21, 2017, St. Petersburg, Russia

## Outline

- Bird’s eye overview of deep learning
- Convolutional neural networks
- From CNN to object detection and segmentation
- Current state of the art
- Neuromation: synthetic data

# Neural networks: a brief history

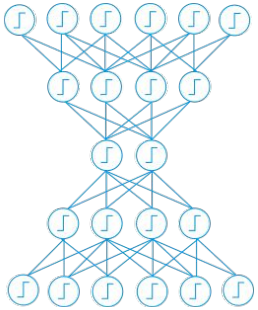
- Neural networks started as models of actual neurons
- Very old idea (McCulloch, Pitts, 1943), there were actual hardware perceptrons in the 1950s

- Several “winters” and “springs”, but the 1980s already had all basic architectures that we use today
- But they couldn’t train them fast enough and on enough data

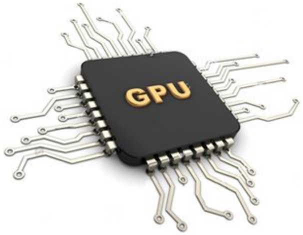
# The deep learning revolution

- 10 years ago machine learning underwent a deep learning revolution
- Since 2007-2008, we can train large and deep neural networks
- New ideas for training + GPUs + large datasets
- And now deep NNs yield state of the art results in many fields

+



+

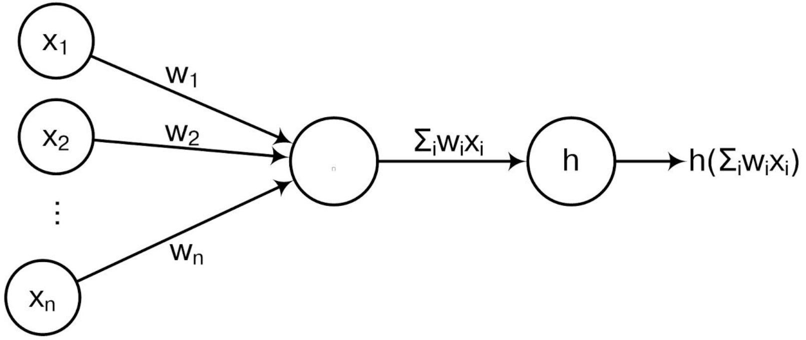


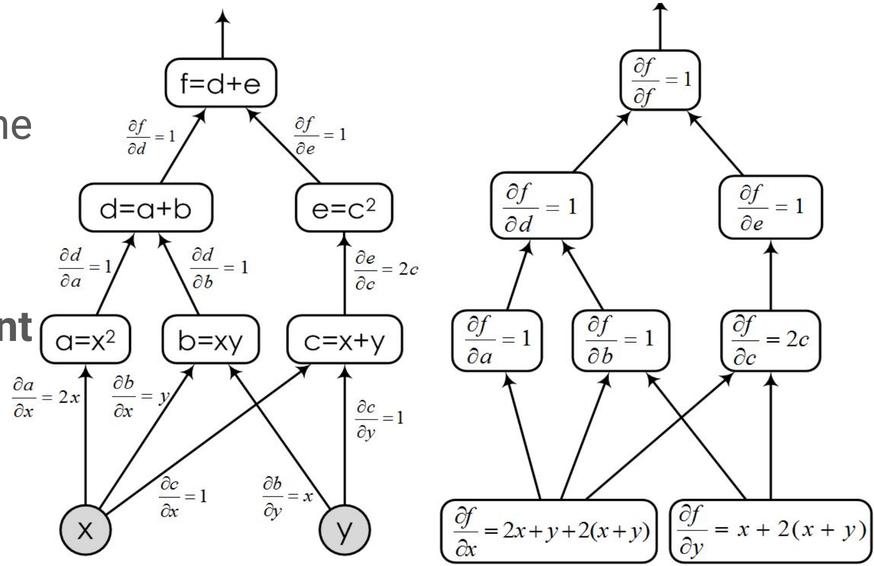

+

=

# What is a deep neural network

- A neural network is a composition of functions
- Usually linear combination + nonlinearity
- These functions comprise a **computational graph** that computes the loss function for the model
- To train the model (learn the weights), you take the gradient of the loss function w.r.t. weights with **backpropagation**
- And then you can do (stochastic) **gradient descent** and variations



# Convolutional neural networks

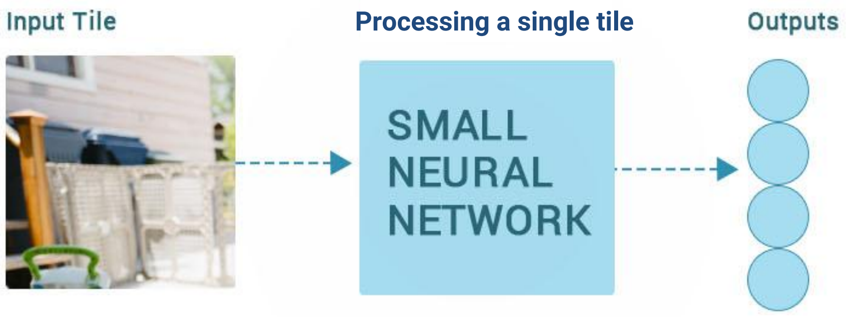
- Convolutional neural networks – specifically for image processing
- Also an old idea, LeCun’s group did it since late 1980s
- Inspired by the experiments of Hubel and Wiesel who understood (lower layers of) the visual cortex

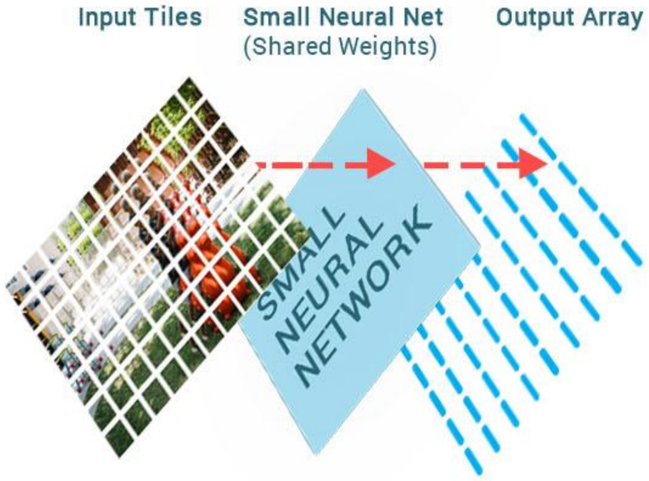
## Convolutional neural networks: idea

- Main idea: **apply the same filters to different parts of the image**.
- Break up the picture into windows:

# Convolutional neural networks: idea

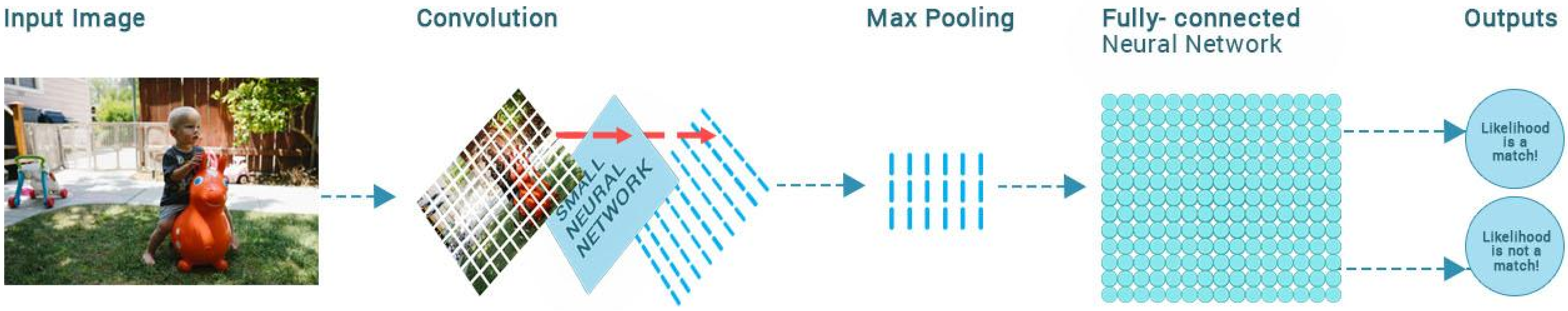
- Main idea: **apply the same filters to different parts of the image**.
- Apply a small neural network to each window:



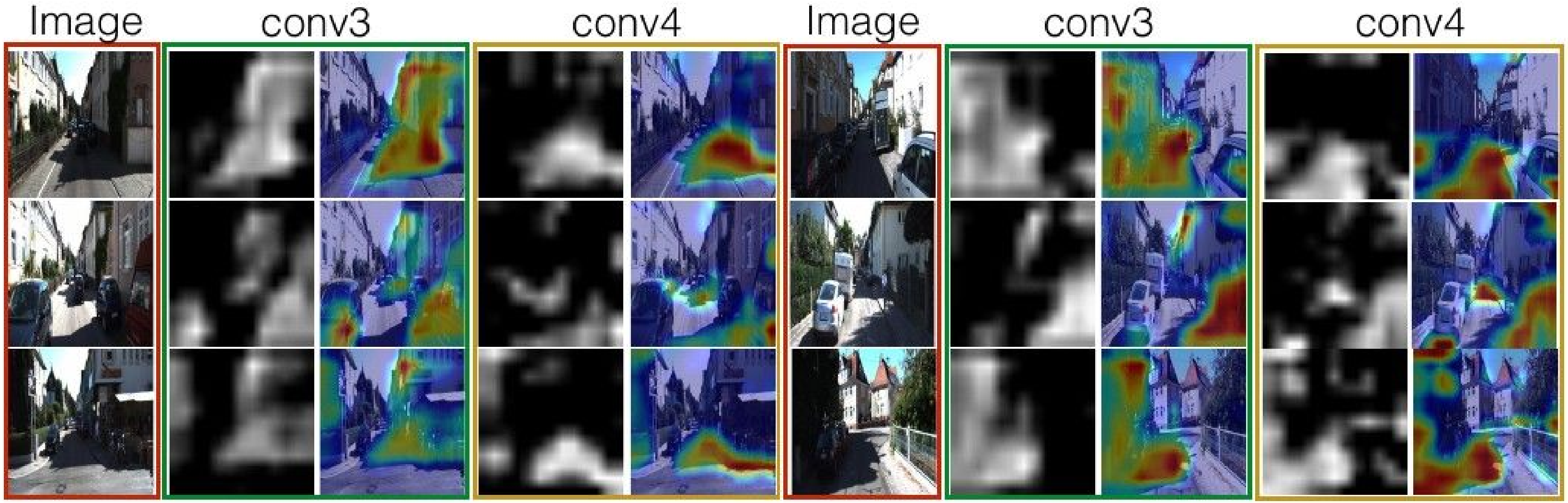
# Convolutional neural networks: idea

- Main idea: **apply the same filters to different parts of the image**.
- Compress with max-pooling
- Then use the resulting features:



# Convolutional neural networks: idea

- We can also see which parts of the image activate a specific neuron, i.e., find out what the features do for specific images:

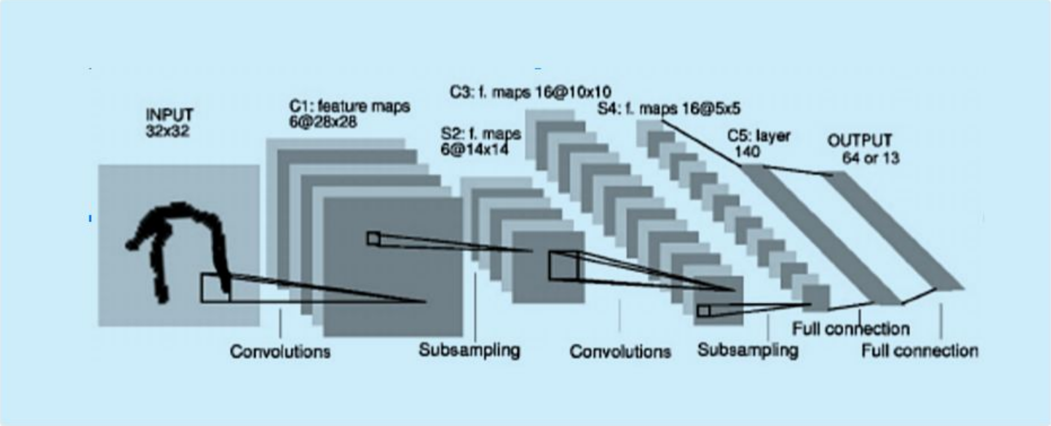


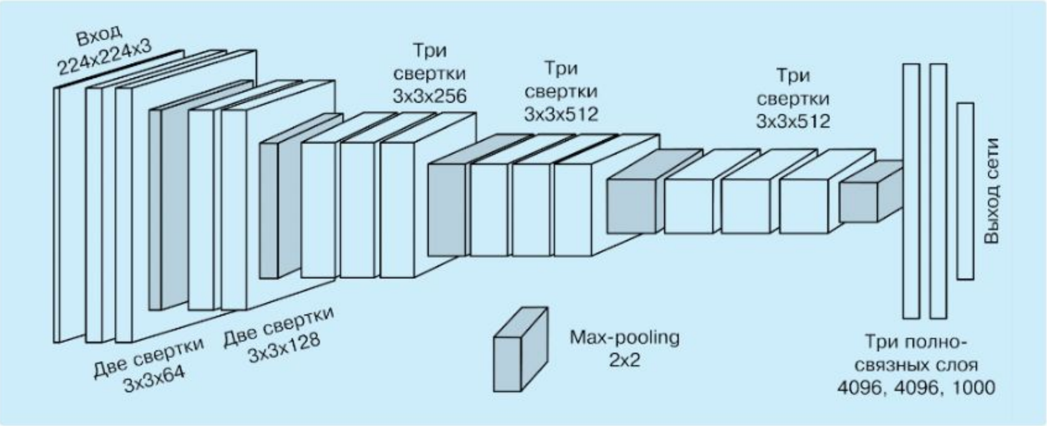
# Deep CNNs

- CNNs were deep from the start – **LeNet**, late 1980s:

- And they started to grow quickly after the deep learning revolution – **VGG**:

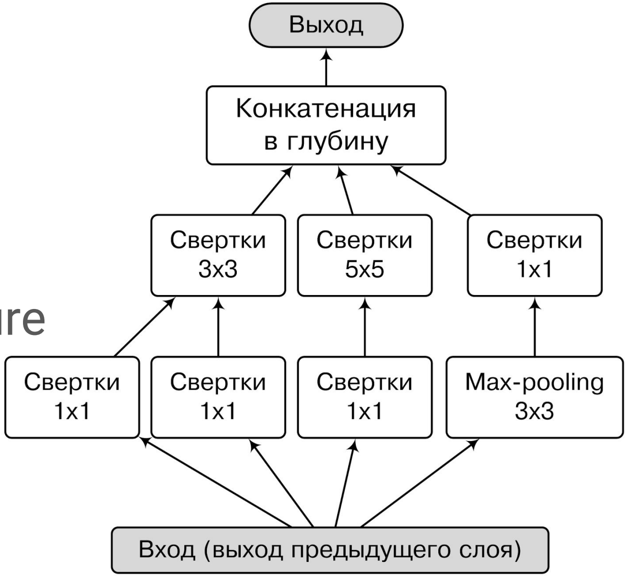
UNIVERSITY OF OXFORD

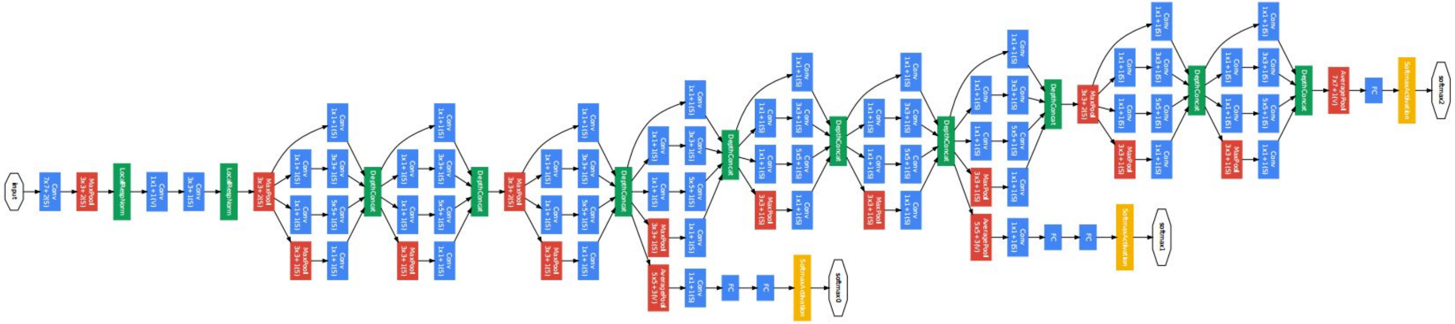


# Inception

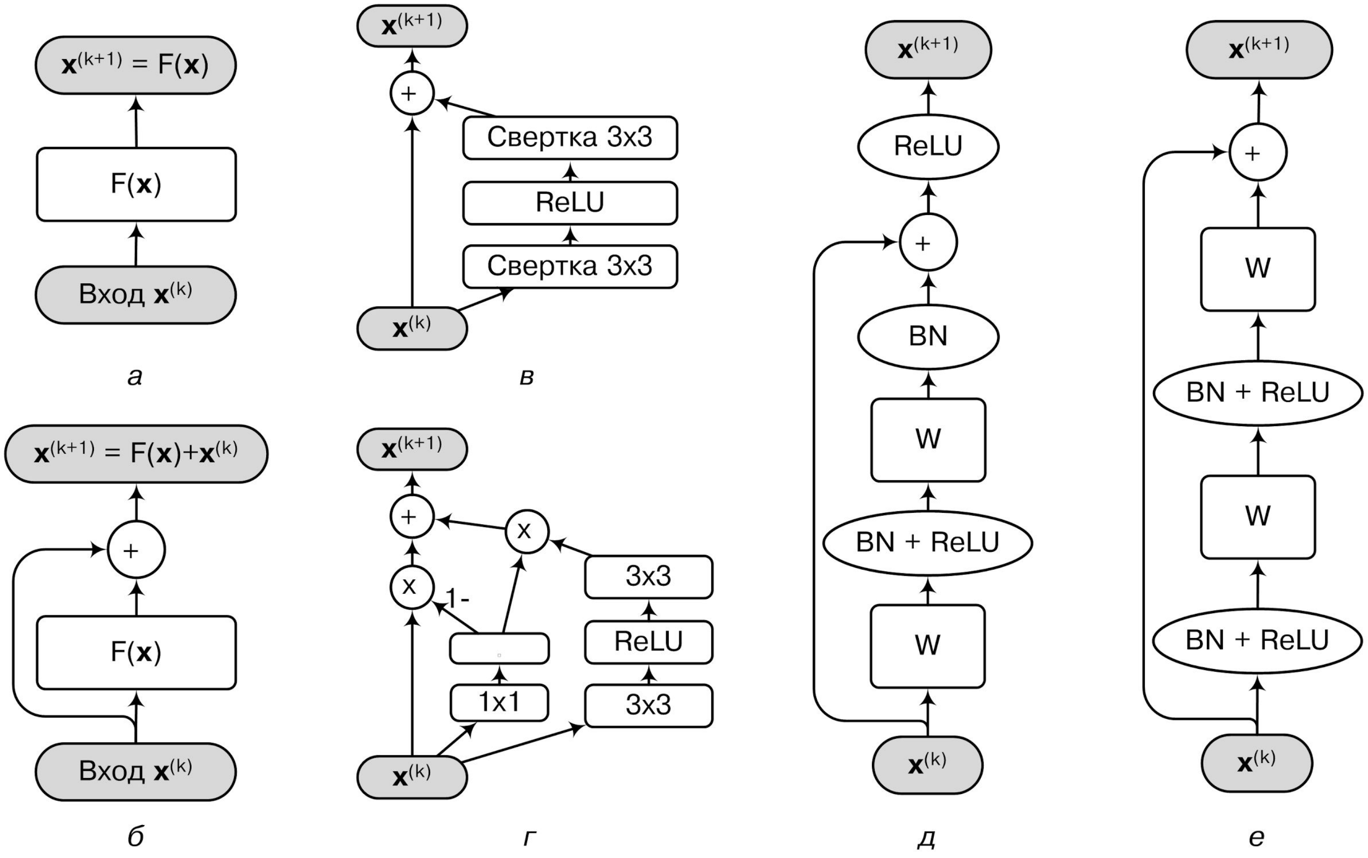
- **Network in network**: the “small network” does not have to be trivial
- **Inception**: a special network in network architecture
- **GoogLeNet**: extra outputs for the error function from “halfway” the model



# ResNet

- **Residual connections** provide the free gradient flow needed for really deep networks

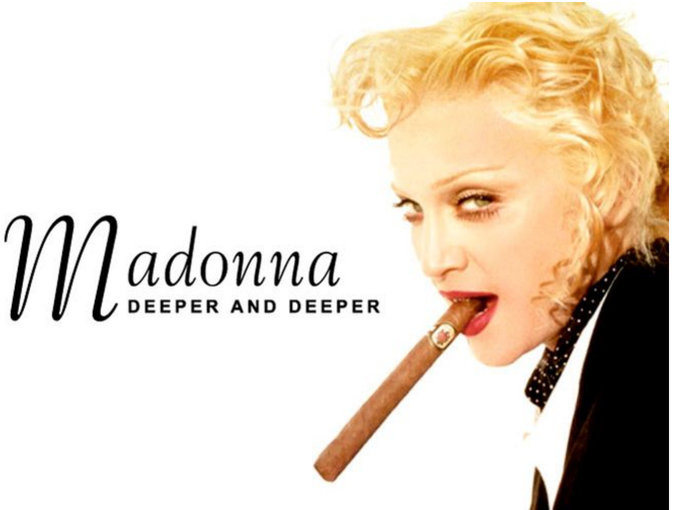


# ResNet led to the revolution of depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

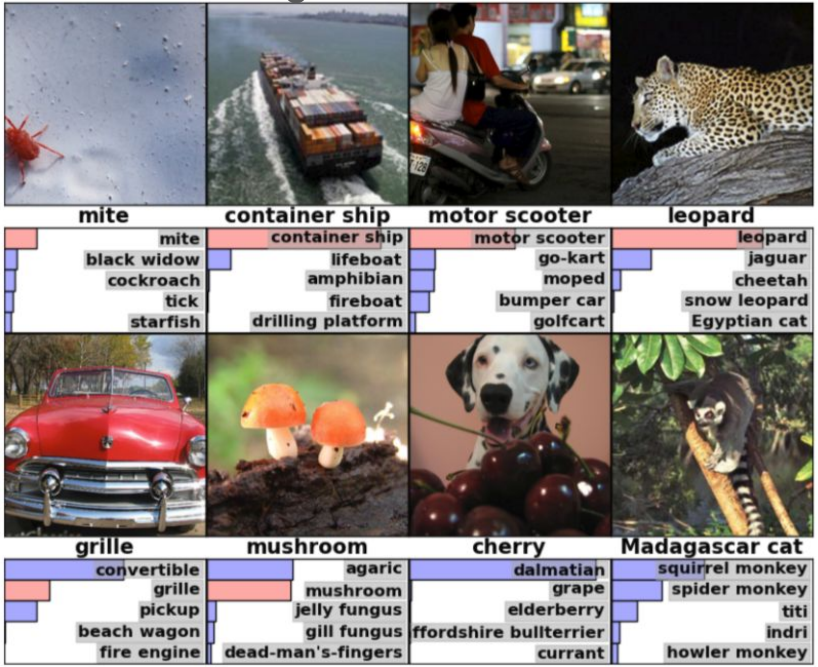


# ImageNet

- Modern CNNs have hundreds of layers
- They usually train on **ImageNet**, a huge dataset for image classification: >10M images, >1M bounding boxes, all labeled by hand

amazon mechanicalturk beta

IMAGENET



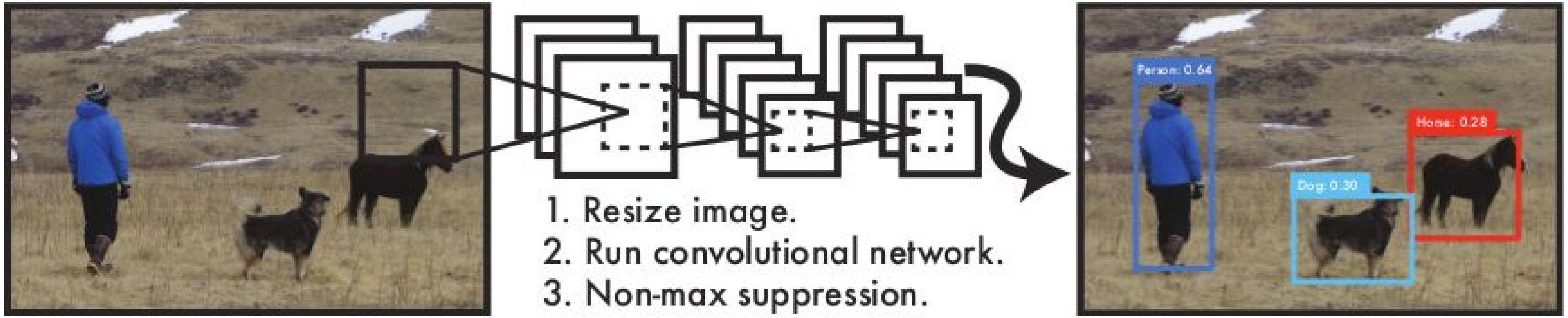
# Object detection

- In practice we also need to know **where** the objects are
- PASCAL VOC dataset for segmentation:

- Relatively small, so recognition models are first trained on ImageNet

# YOLO

- YOLO: you only look once; look for bounding boxes and objects in one pass:



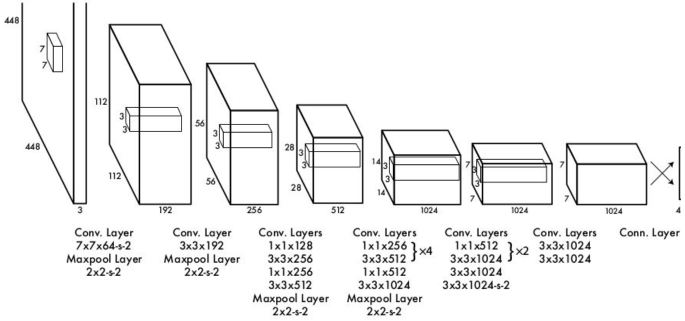

- YOLO v.2 has recently appeared and is one of the fastest and best object detectors right now

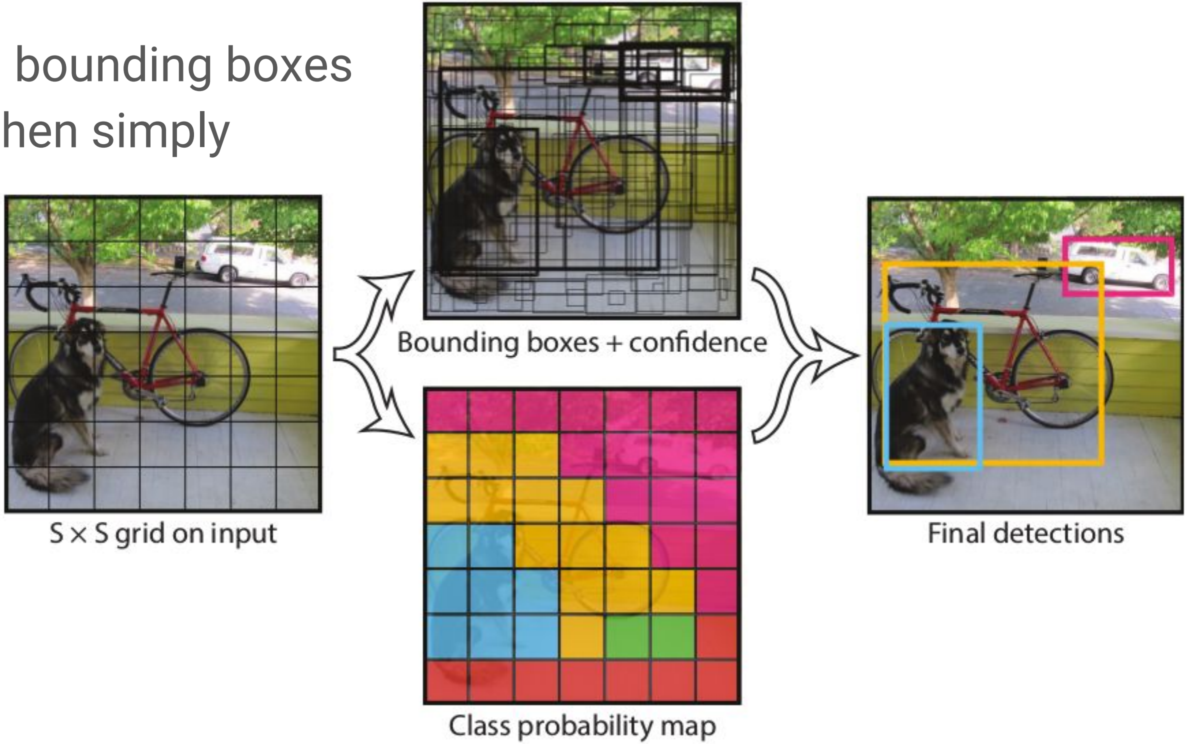
# YOLO

- Idea: split the image into an SxS grid.
- In each cell, predict both bounding boxes and class probabilities; then simply

$$p(\text{class}_i \mid \text{obj})p(\text{obj})p(\text{bbox})$$

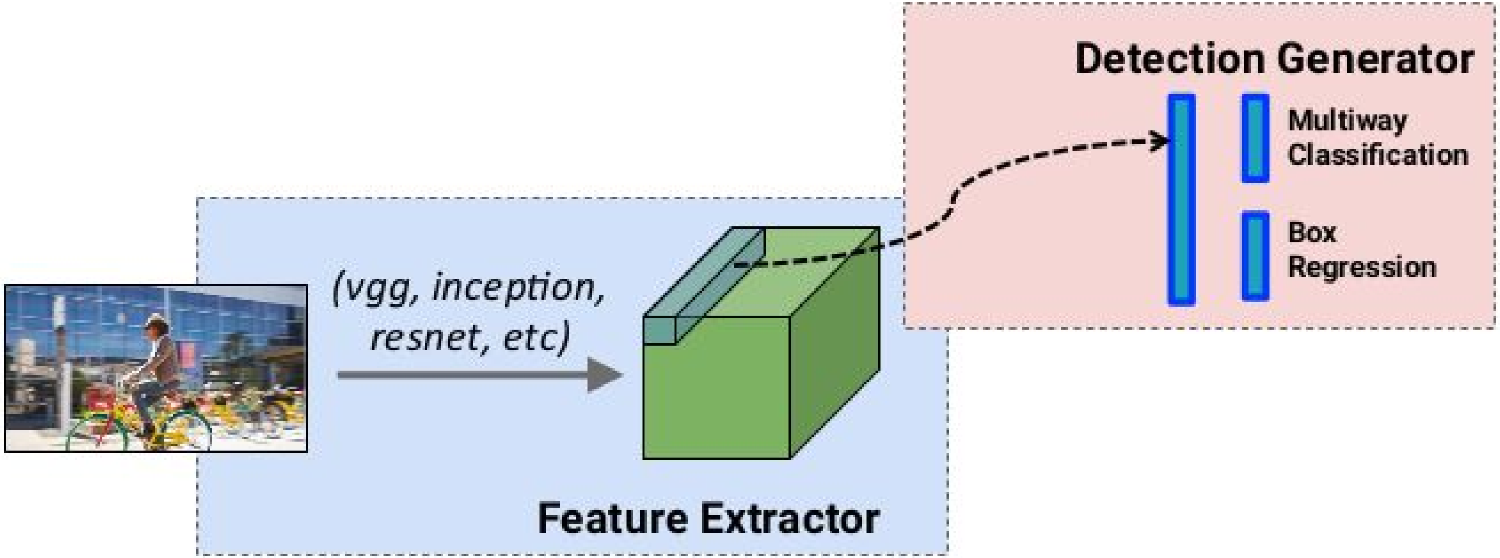
- CNN architecture in YOLO is standard:



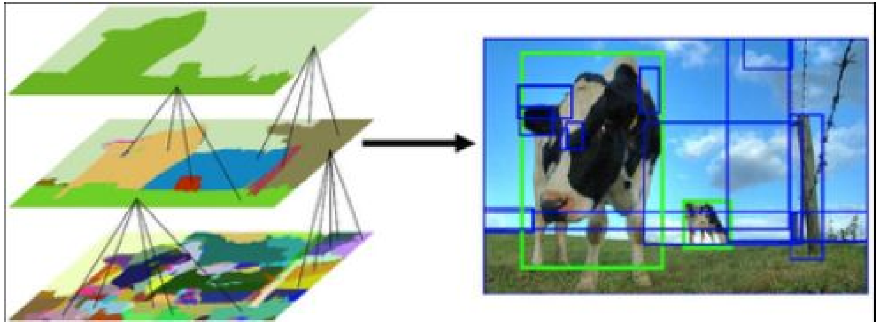



## Single Shot Detectors

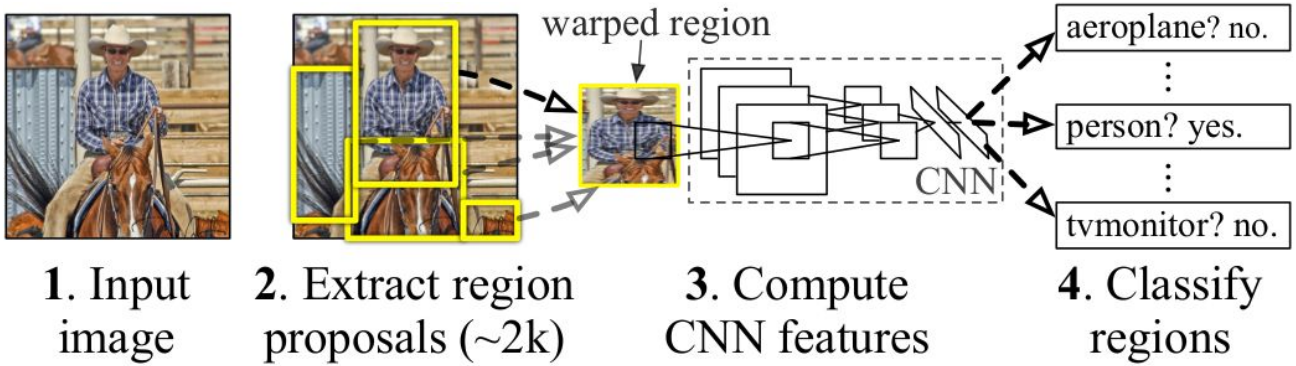
- Further development of this idea: **single-shot detectors** (SSD)
- A single network that predicts several class labels and several corresponding positions for **anchor boxes** (bounding boxes of several predefined sizes).



# R-CNN

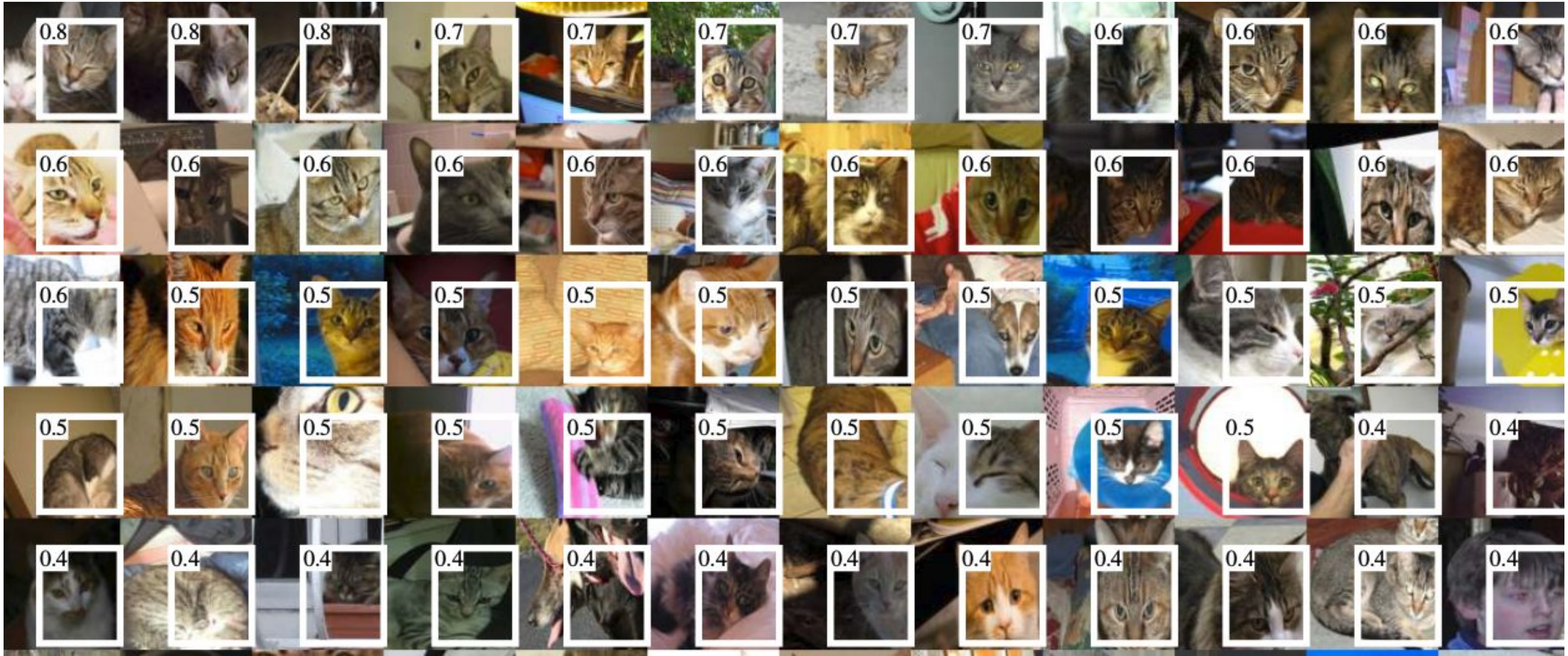


- R-CNN: **Region-based ConvNet**
- Find bounding boxes with some external algorithm (e.g., selective search)
- Then extract CNN features (from a CNN trained on ImageNet and fine-tuned on the necessary dataset) and classify



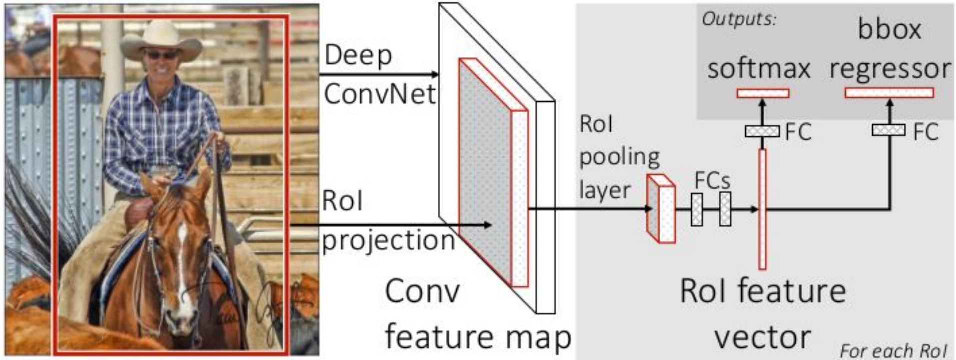
# R-CNN

- Visualizing regions of activation for a neuron from a high layer:



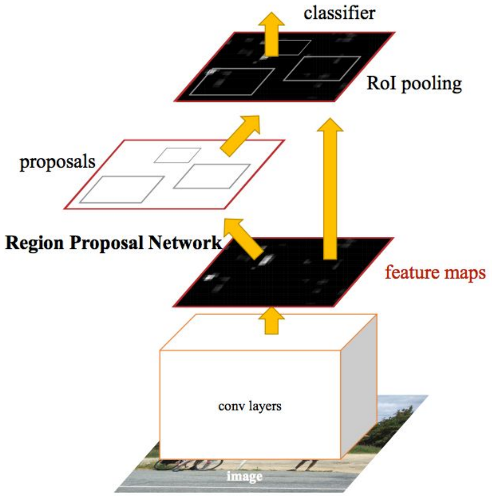
# Fast R-CNN

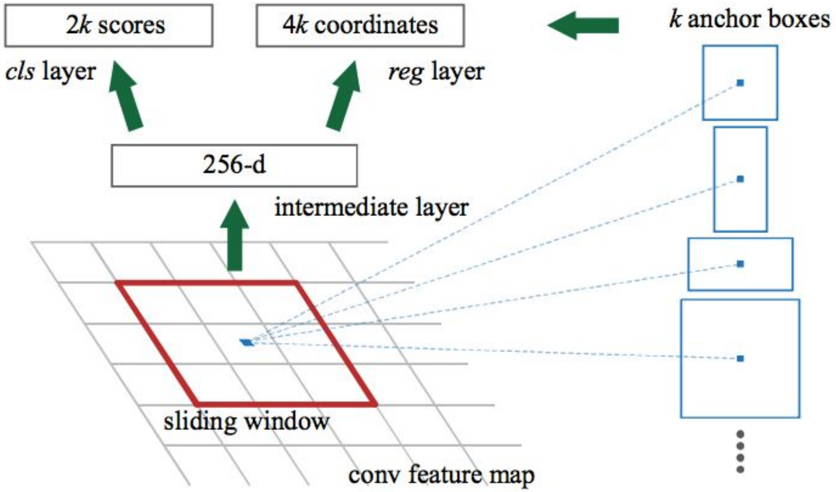
- But R-CNN has to be trained in several steps (first CNN, then SVM on CNN features, then bounding box regressors), very long, and recognition is very slow (47s per image even on a GPU!)
- The main reason is that we need to go through the CNN for every region
- Hence, **Fast R-CNN** makes RoI (region of interest) projection that collects features from a region.
- One pass of the main CNN for the whole image.
- Loss = classification error + bounding box regression error



## Faster R-CNN

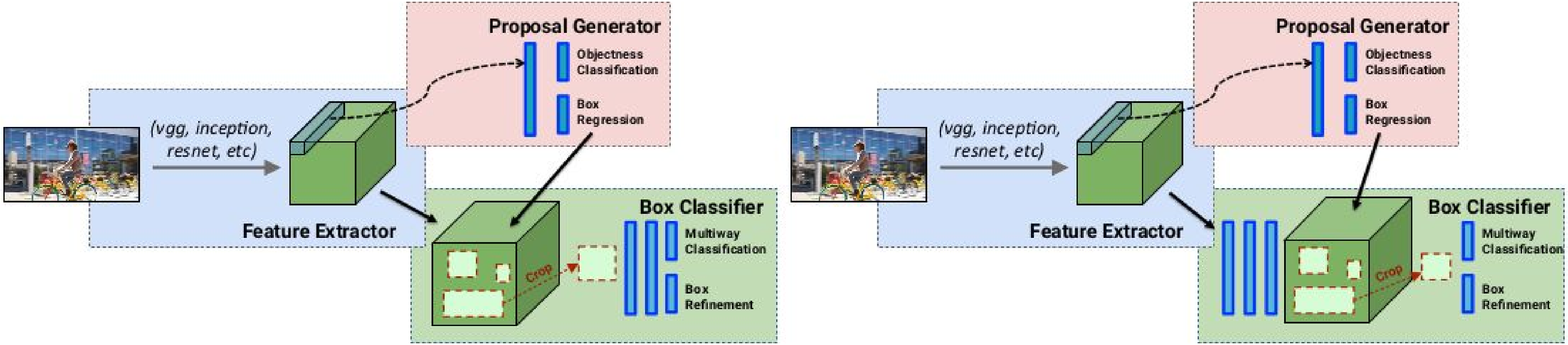
- One more bottleneck left: selective search to choose bounding boxes.
- **Faster R-CNN** embeds it into the network too with a separate **Region Proposal Network**
- Evaluates each individual possibility from a set of predefined anchor boxes



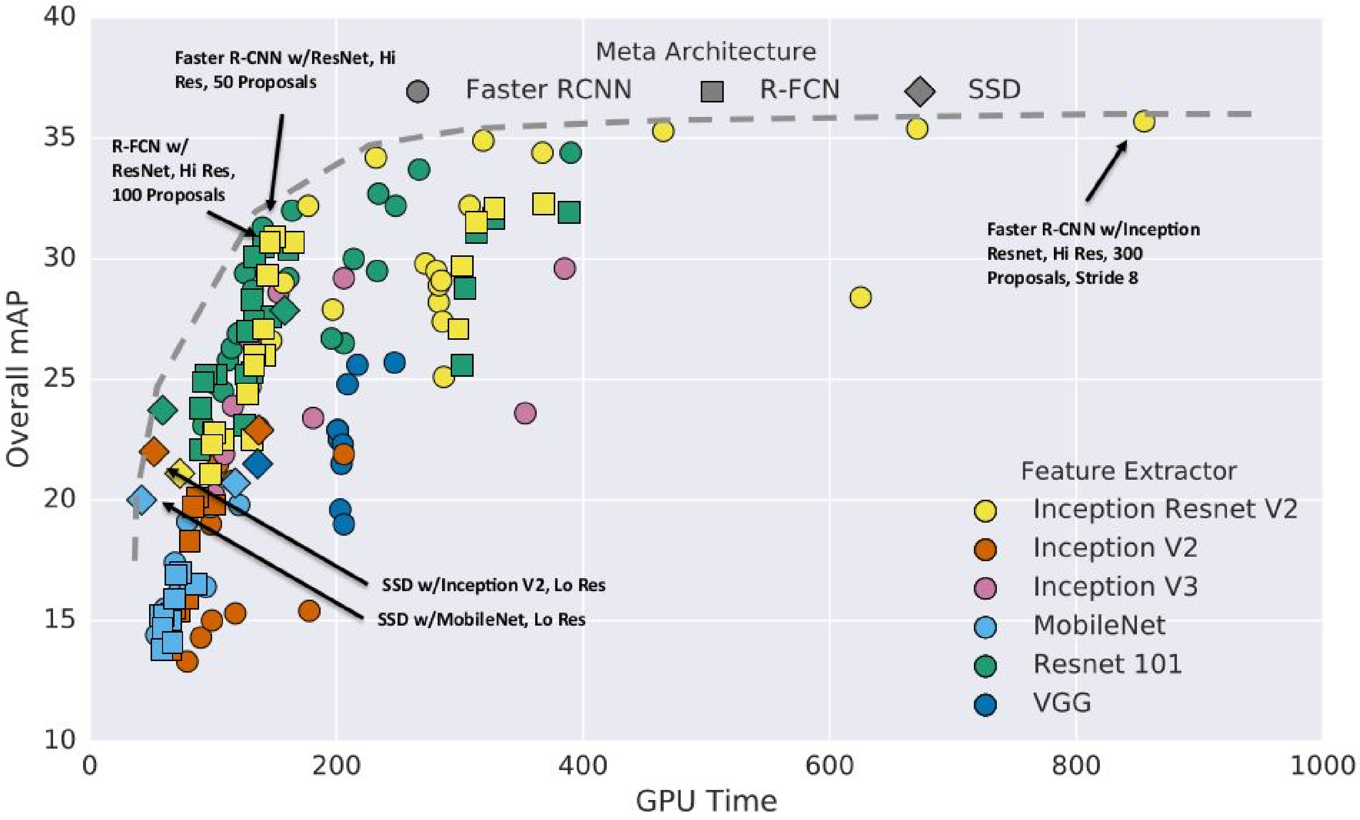
## R-FCN

- We can cut the costs even further, getting rid of complicated layers to be computed on each region.
- R-FCN (**Region-based Fully Convolutional Network**) cuts the features from the very last layer, immediately before classification

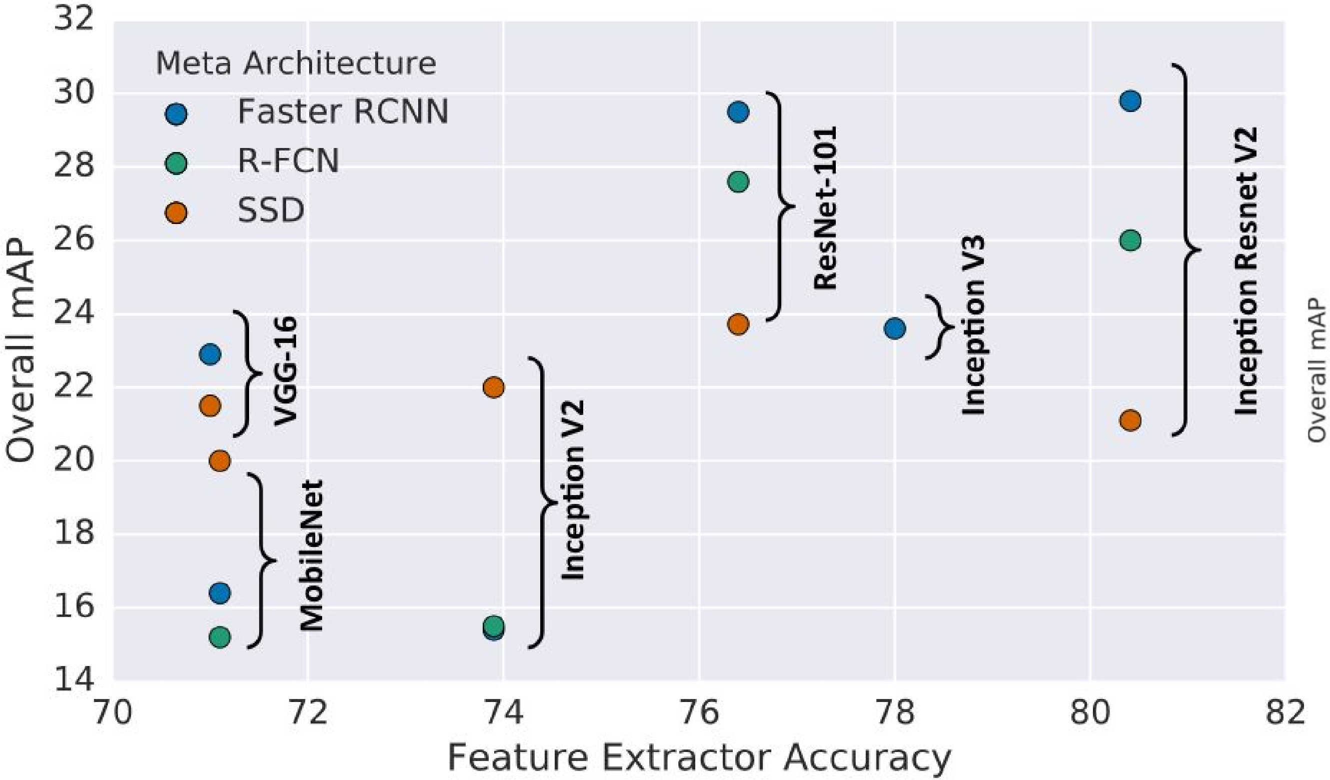


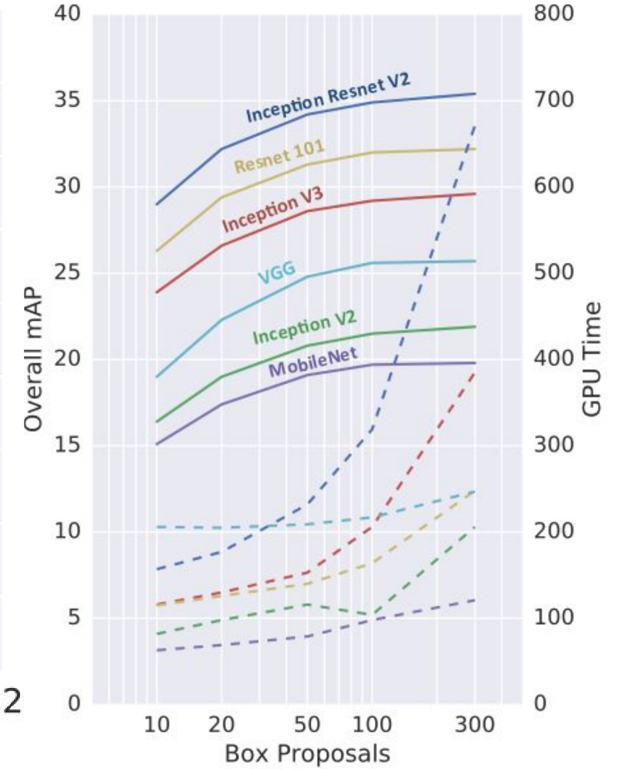

(b) Faster RCNN. (c) R-FCN.

# How they all compare



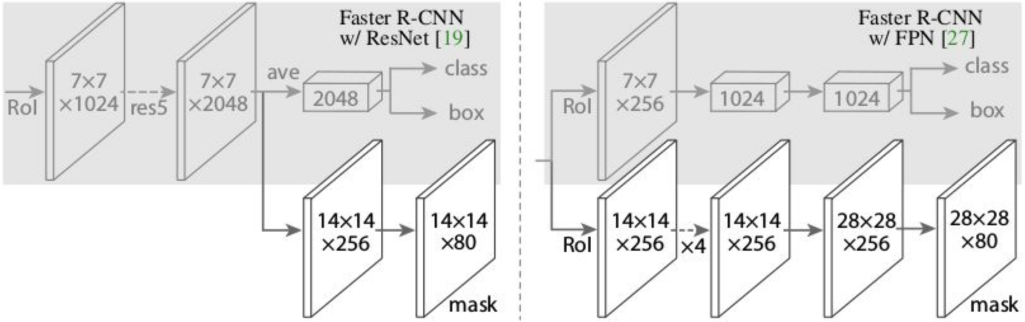
# How they all compare

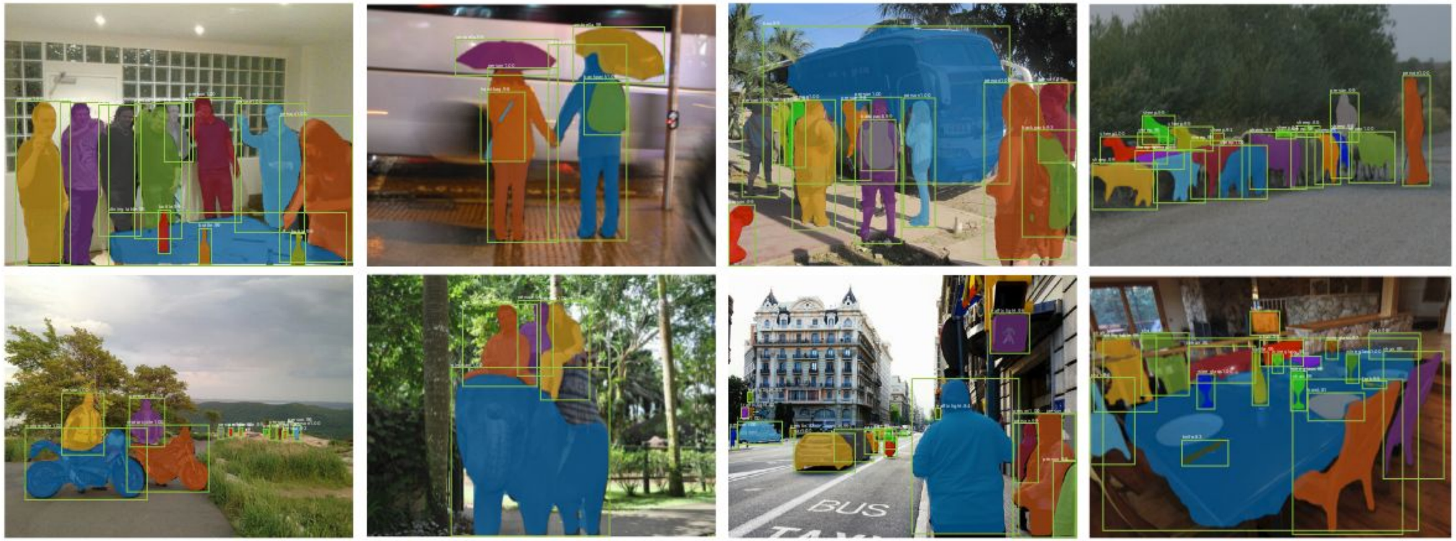


# Mask R-CNN
# for image segmentation

- To get segmentation, just add a pixel-wise output layer



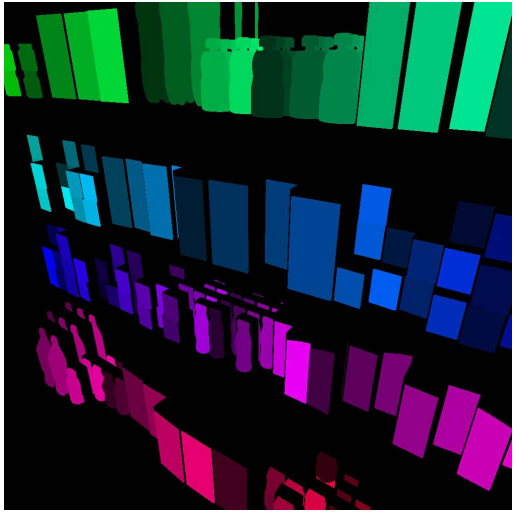



# Synthetic data

- But all of this still requires lots and lots of data
- The **Neuromation** approach: create **synthetic data** ourselves
- We create a 3D model for each object and render images to train on

# Synthetic data

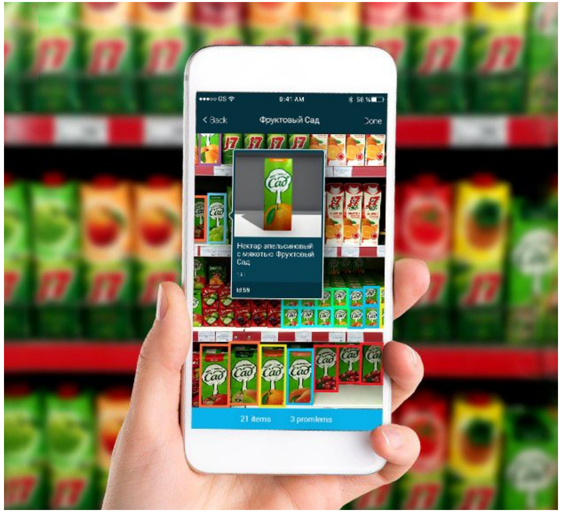
- Synthetic data can have **pixel perfect** labeling, something humans can't do
- And it is 100% correct and free



# Transfer learning

- Problem: we need to do **transfer learning** from synthetic images to real ones
- We are successfully solving this problem from both sides



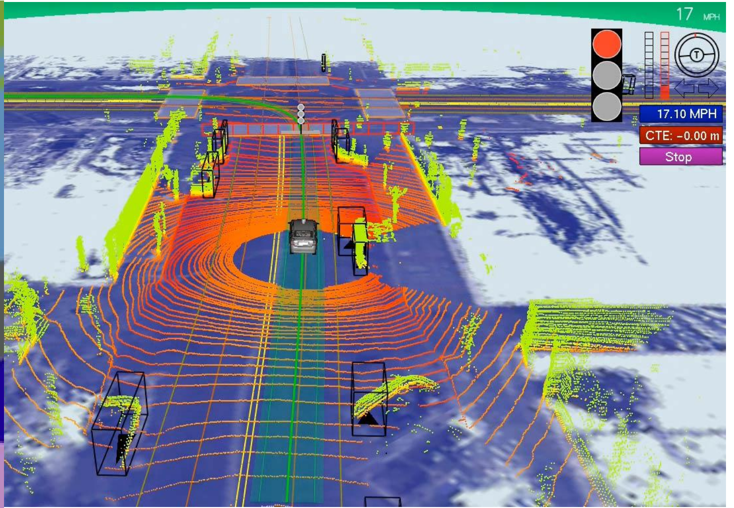


# Next step

- **Retail Automation Lab** needs to scale up synthetic data
- Challenge: **170000 SKU** in the Russian retail catalogue only

# Synthetic data for industrial automation

- Another great fit for synthetic data – industrial automation
- Self-driving cars, flying drones, industrial robots… labeled data is limited
- Synthetic environments can help

NEUROMATION

Where androids dream of electric sheep

SmartData

# OUR TEAM:

**Maxim Prasolov**
CEO

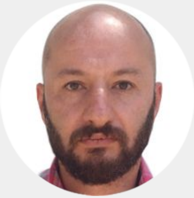
**Fedor Savchenko**
CTO

**Sergey Nikolenko**
Chief Research Officer

**Denis Popov**
Chief Information Officer

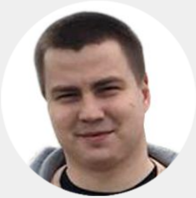
**Constantine Goltsev**
Investor / Chairman

**Andrew Rabinovich**
Adviser



**Yuri Kundin**
ICO Compliance Adviser

**Esther Katz**
VP of Communication



**Kiryl Truskovskyi**
Lead Researcher

**David Orban**
Adviser

## THANK YOU FOR YOUR ATTENTION!

NEUROMATION

SmartData

# KNOWLEDGE MINING - A NEW ERA OF DISTRIBUTED COMPUTING

**THANK YOU!**

neuromation.io