

session 5:
Deep Learning

M. Kundegorski

13th December 2019

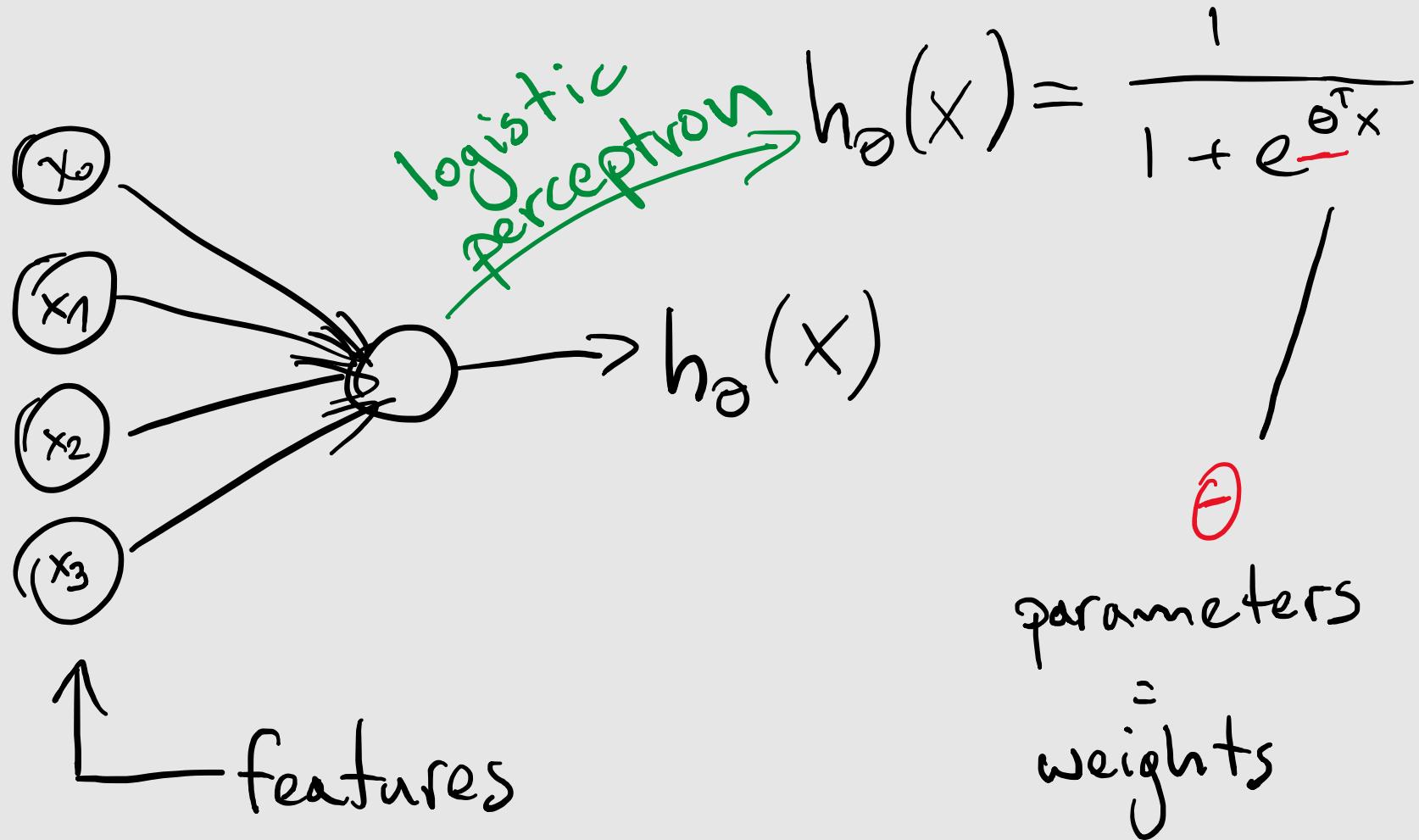
IAFIG-RMS - Bioimage Analysis With Python
Cambridge Bioinformatics Training Centre



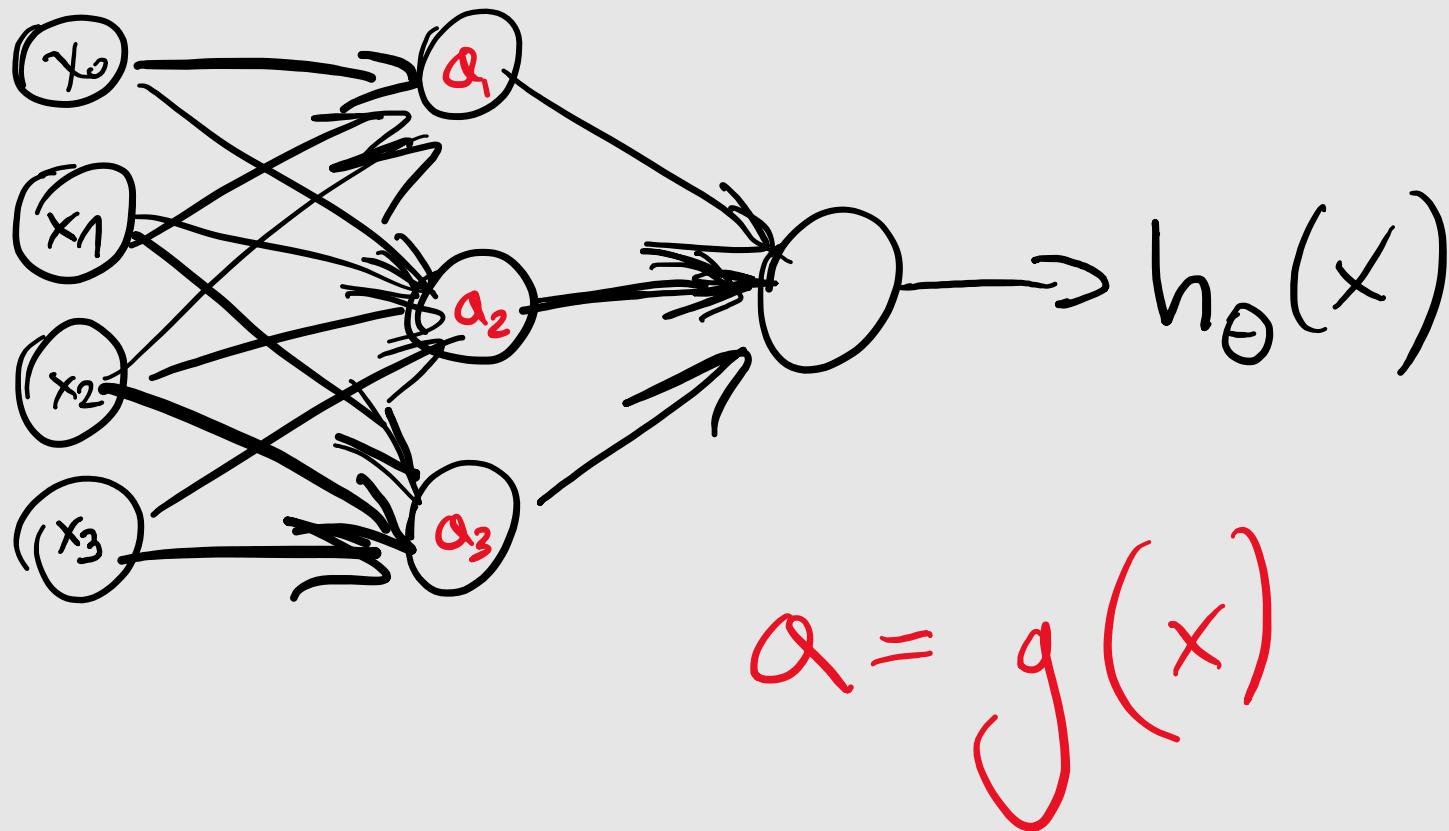
Deep Learning

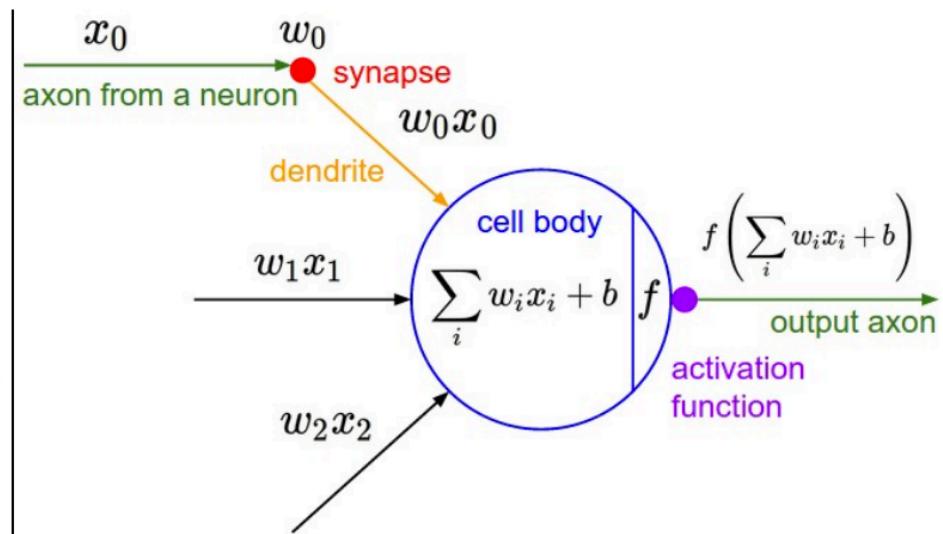
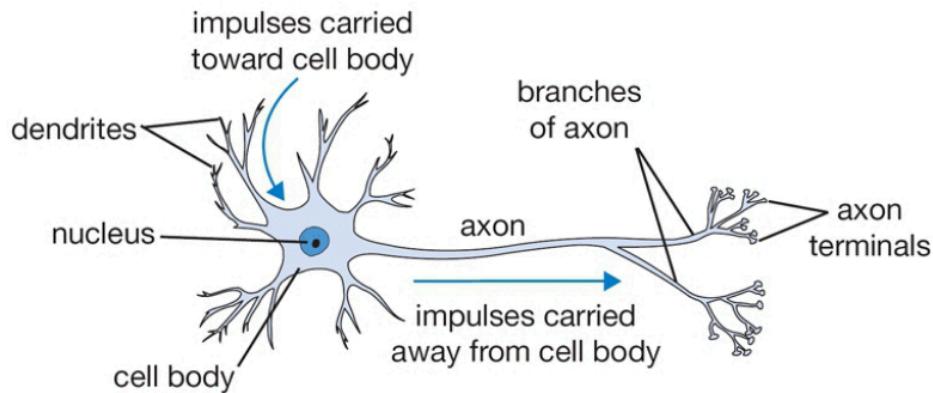
Accept that this is not magic.

From logistic regression to neural network



From logistic regression to neural network





A cartoon drawing of a biological neuron (left) and its mathematical model (right).

source:CS231n Convolutional Neural Networks for Visual Recognition
<https://cs231n.github.io>

Deep Learning Revolution

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Model	Top-1	Top-5
<i>Sparse coding [2]</i>	47.1%	28.2%
<i>SIFT + FVs [24]</i>	45.7%	25.7%
CNN	37.5%	17.0%

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

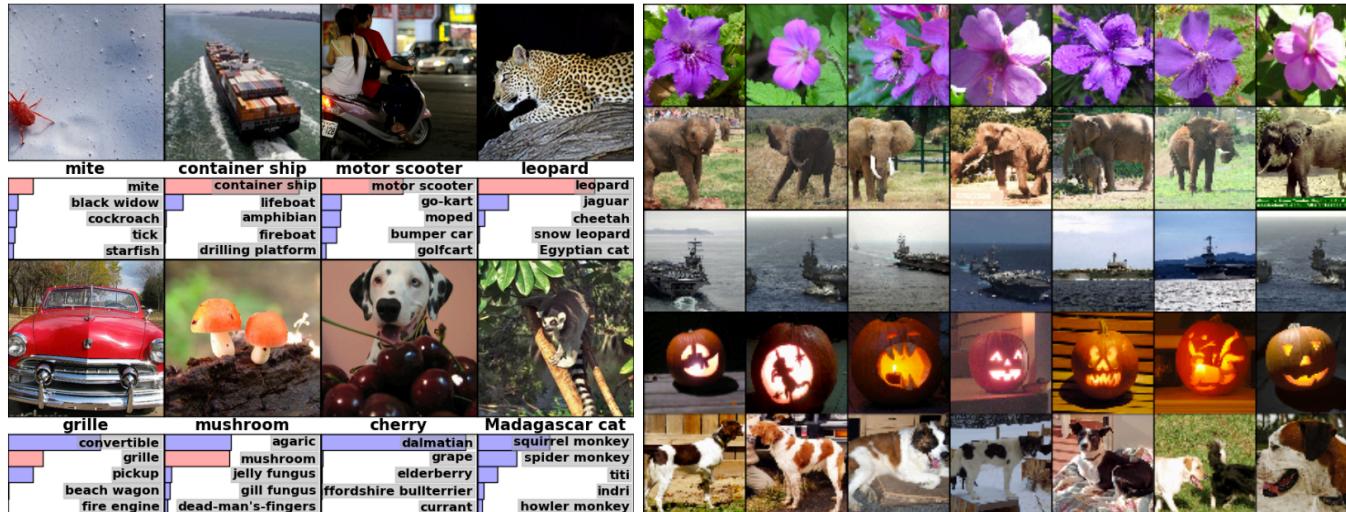
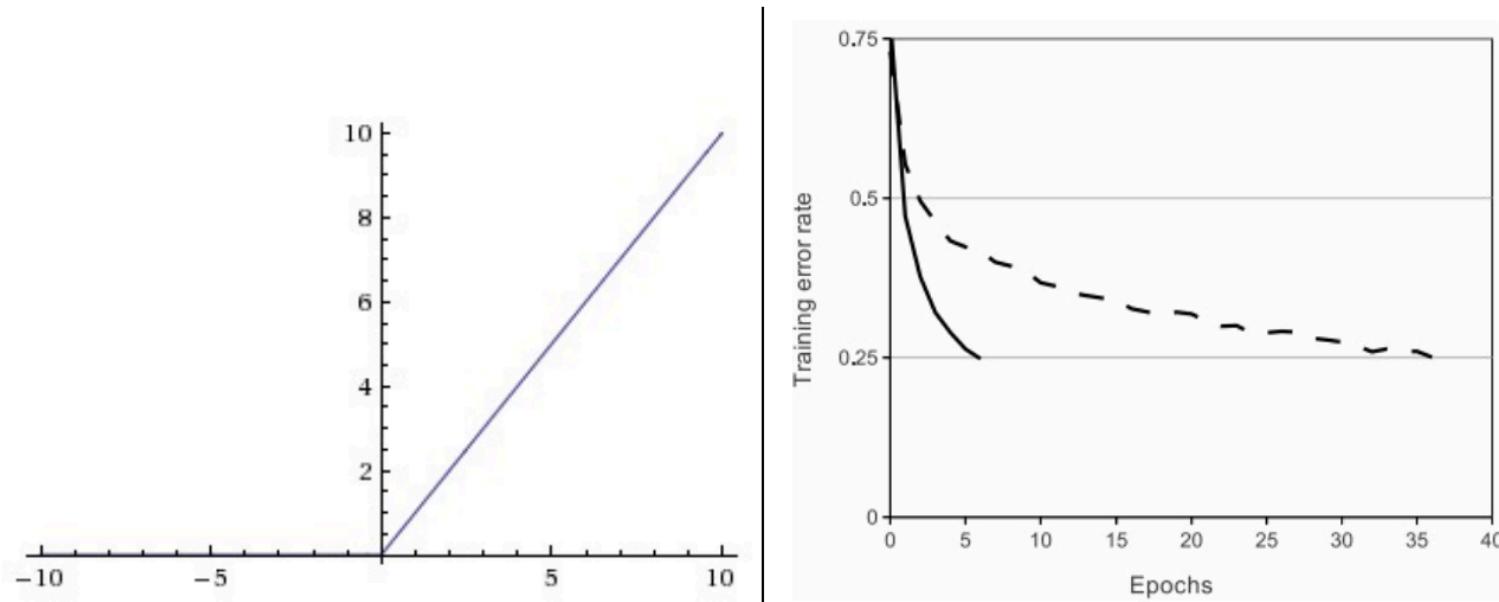


Figure 4: (Left) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (Right) Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

ReLU activation – instead of Sigmoid



Left: Rectified Linear Unit (ReLU) activation function, which is zero when $x < 0$ and then linear with slope 1 when $x > 0$. **Right:** A plot from Krizhevsky et al. (pdf) paper indicating the 6x improvement in convergence with the ReLU unit compared to the tanh unit.

Convolution

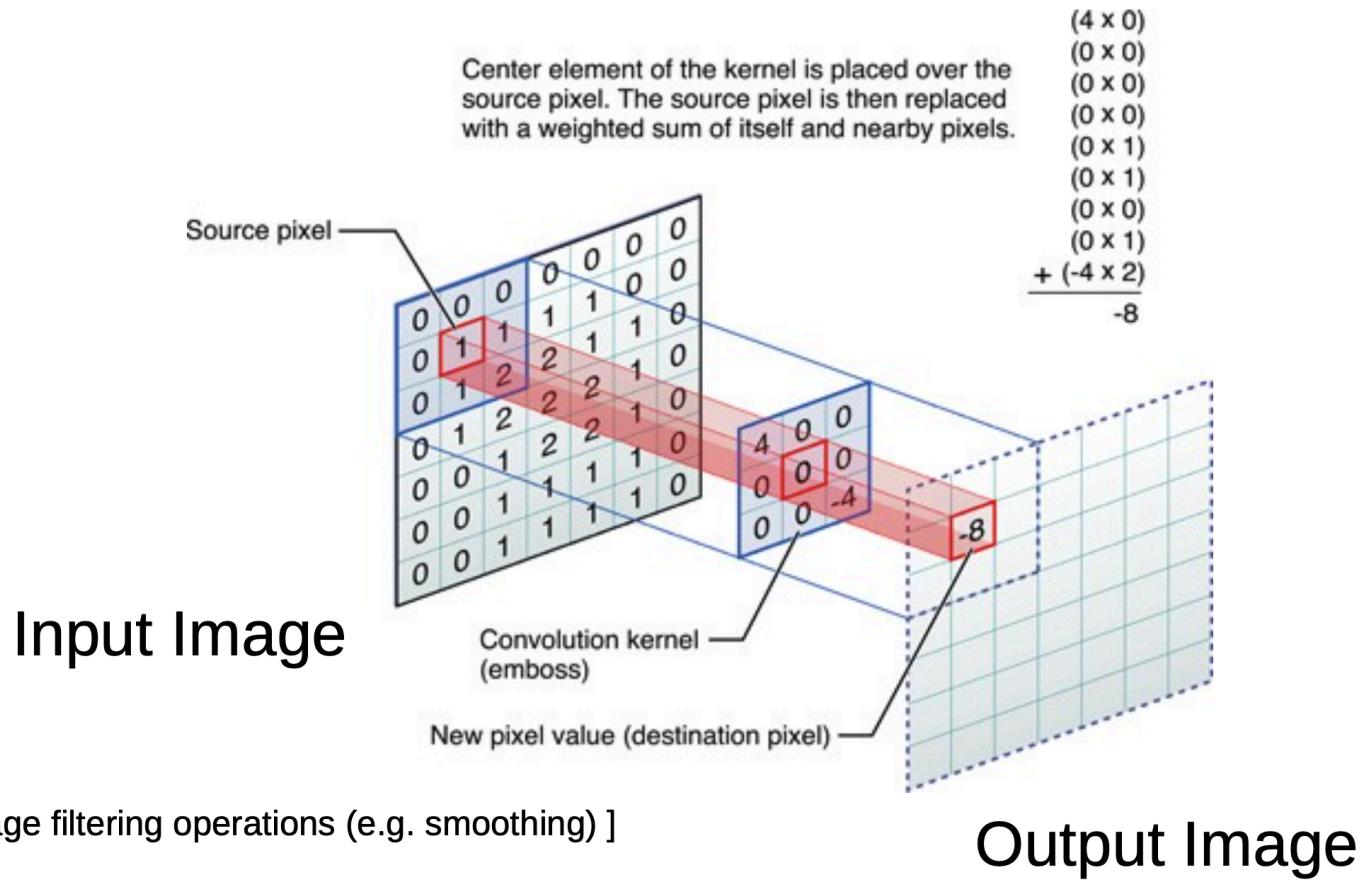


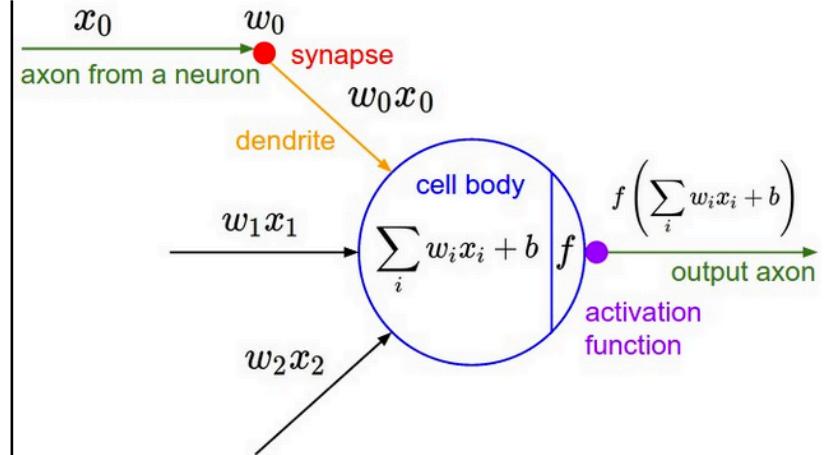
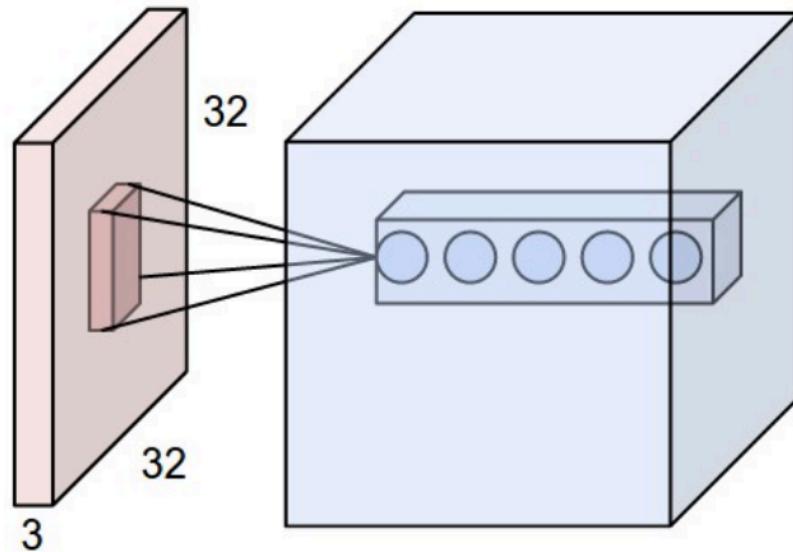
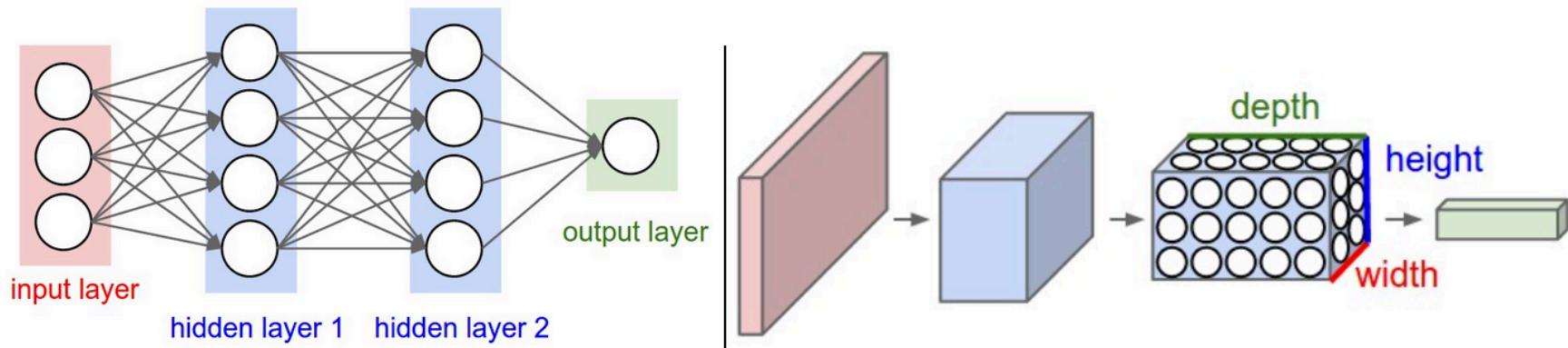
Image source: developer.apple.com

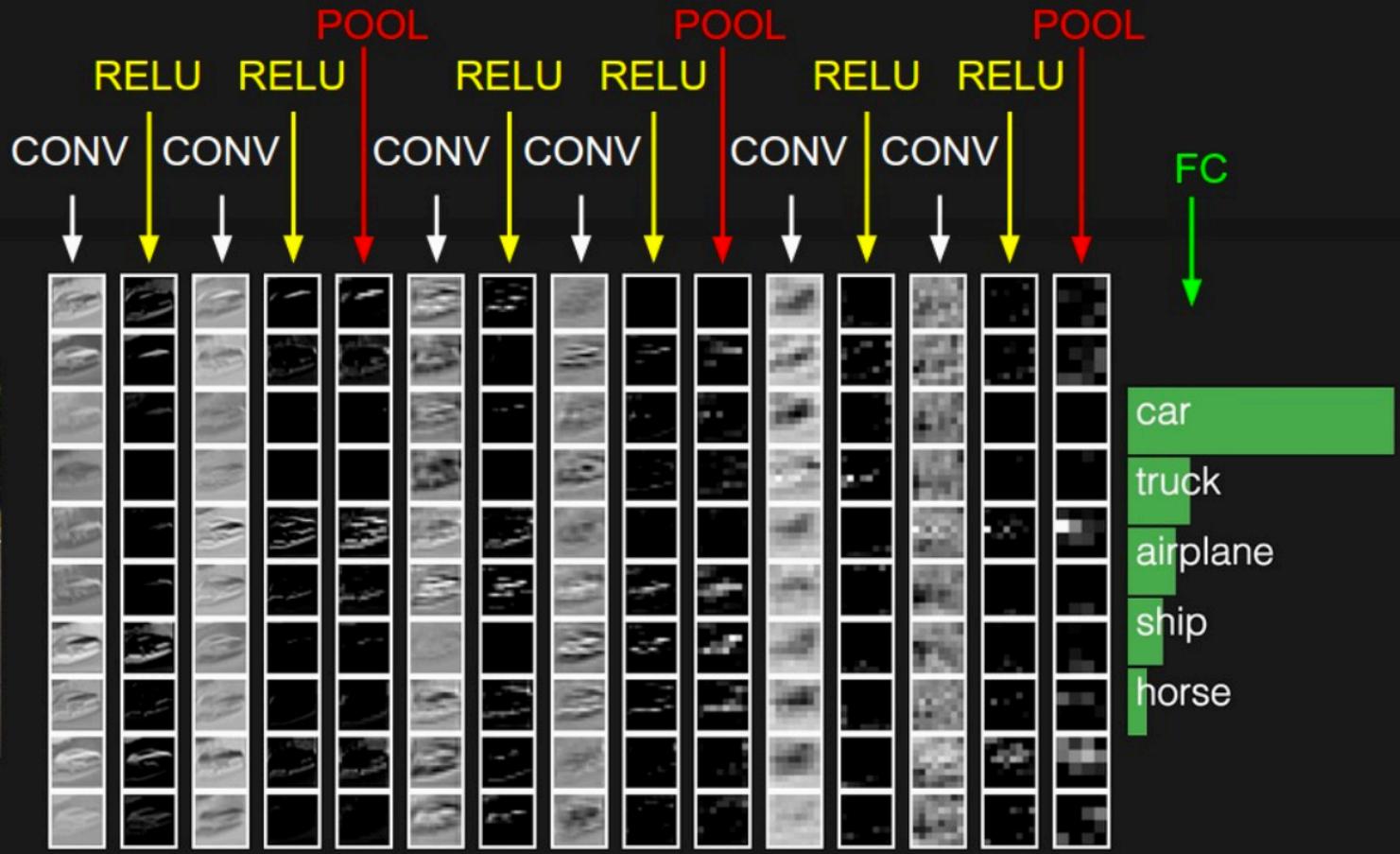
convolution

original	filter (3 x 3)	blur
	$\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array}$	
	$\begin{array}{ccc} 0 & -1 & 0 \\ 1 & 5 & 1 \\ 0 & -1 & 0 \end{array}$	
	$\begin{array}{ccc} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{array}$	

source: <http://breckon.eu/toby/teaching/mltutorial/>

Convolutional Neural Networks

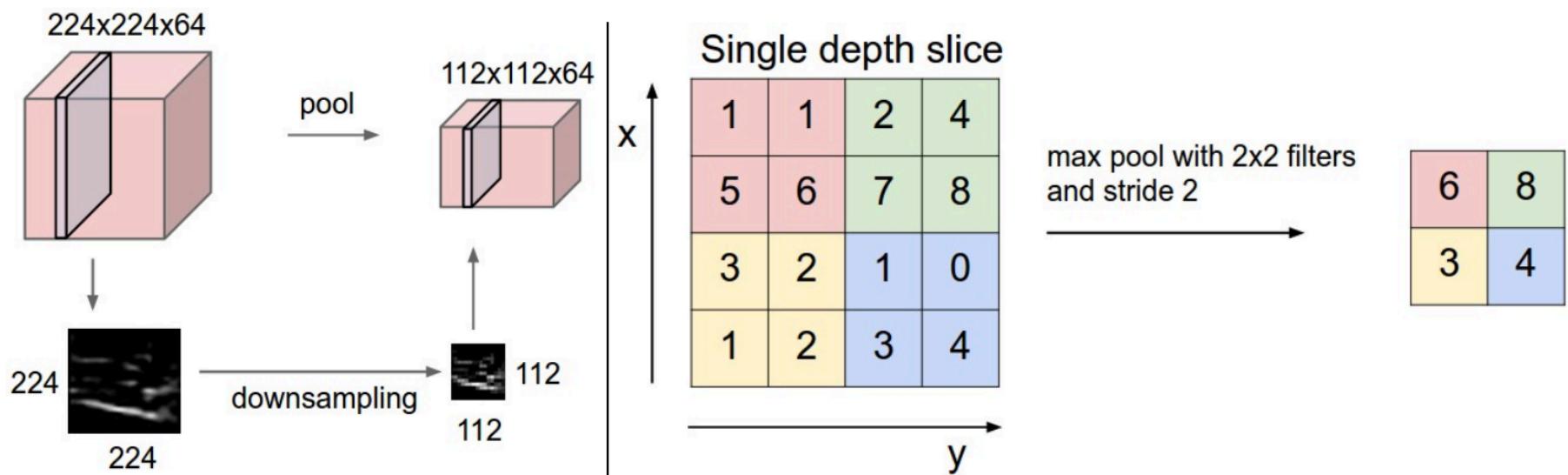




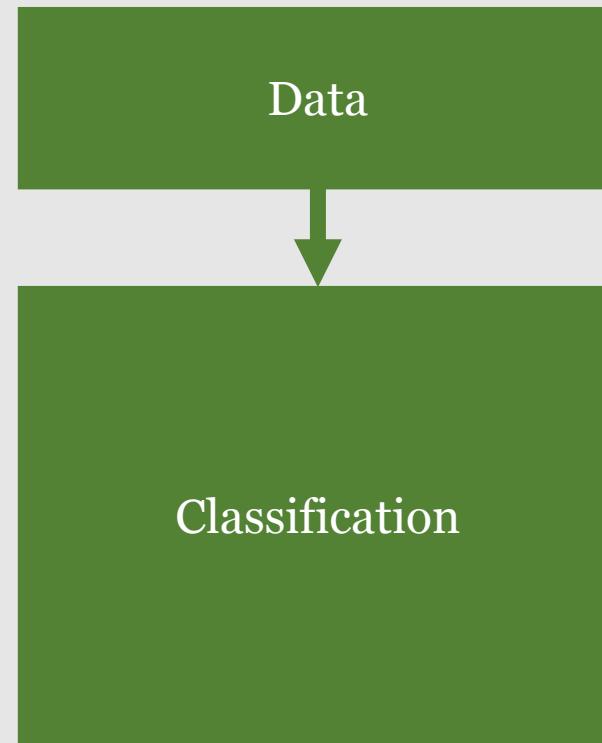
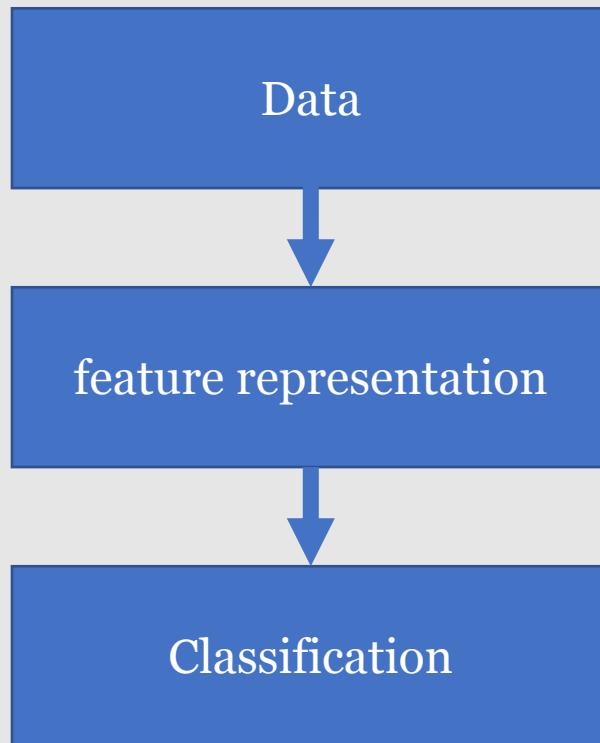
Features from Deep Learning



Max pooling



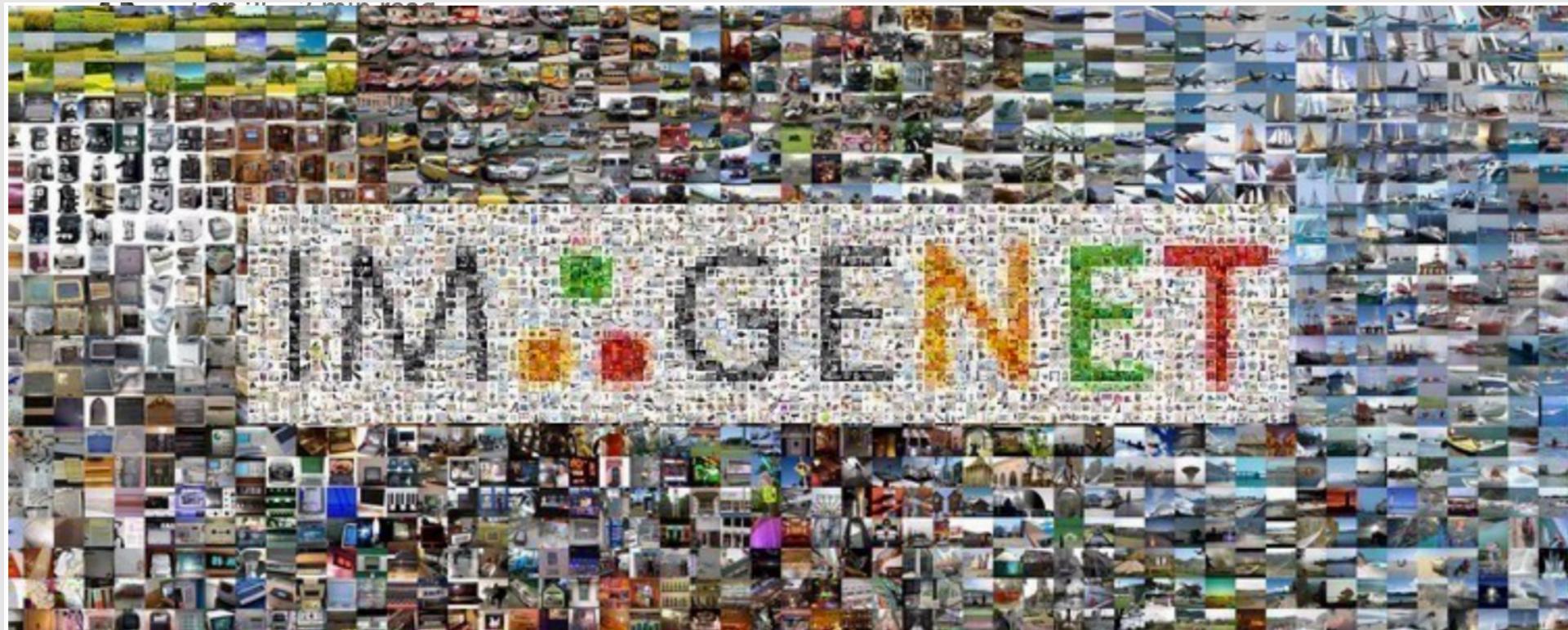
Shallow vs Deep



Why not to use deep learning

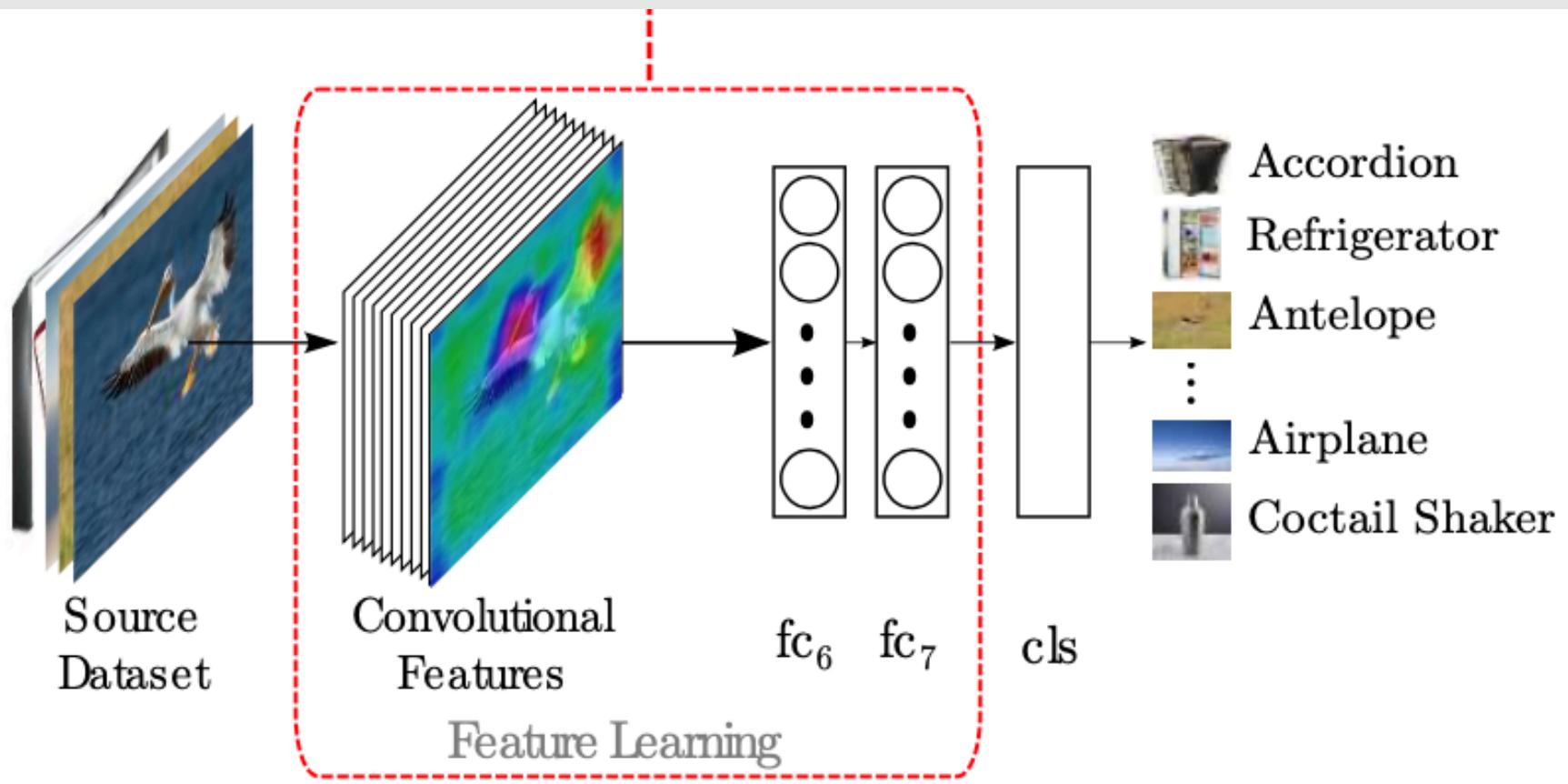
- requires big datasets
- takes long training time to learn hyperparameters
- Can a better classification be achieved using SVM on learned deep-learned features?

“Big” dataset



14 million annotated images

Transfer Learning



ImageNet Classification with Deep Convolutional Neural Networks
Krizhevsky , 2012 / Illustration Samet Ackay

Applications

Object detection

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Abstract—State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Although the original R-CNN [1] and Fast R-CNN [2] have reduced the running time of these detection networks, exposing region proposal work to the network, we introduce a *Region Proposal Network* (RPN) that shares full-image convolutional features and objectness scores at each position. An RPN is trained end-to-end to generate region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network, using the recently popular terminology of neural networks with “*feature reuse*”.

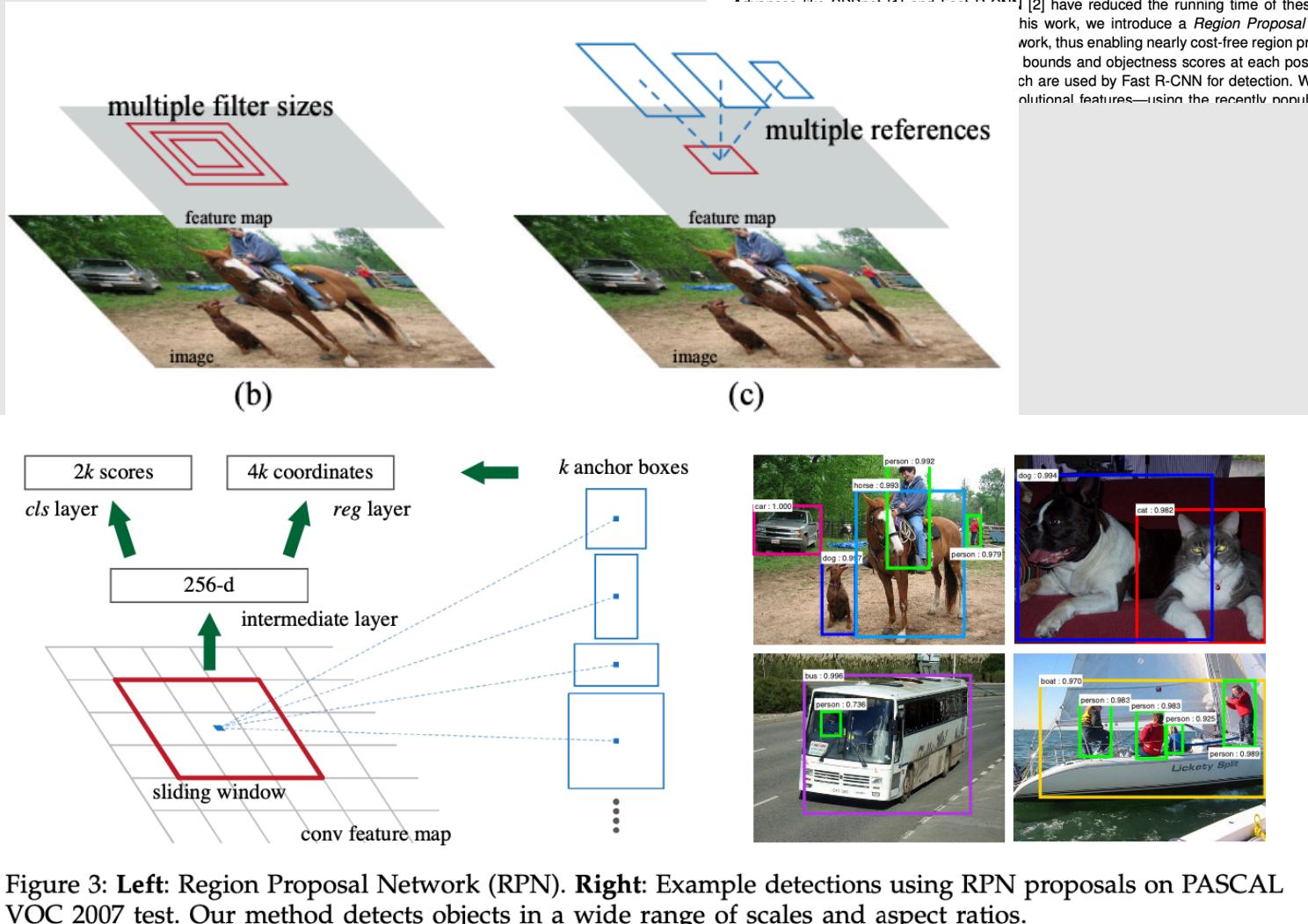


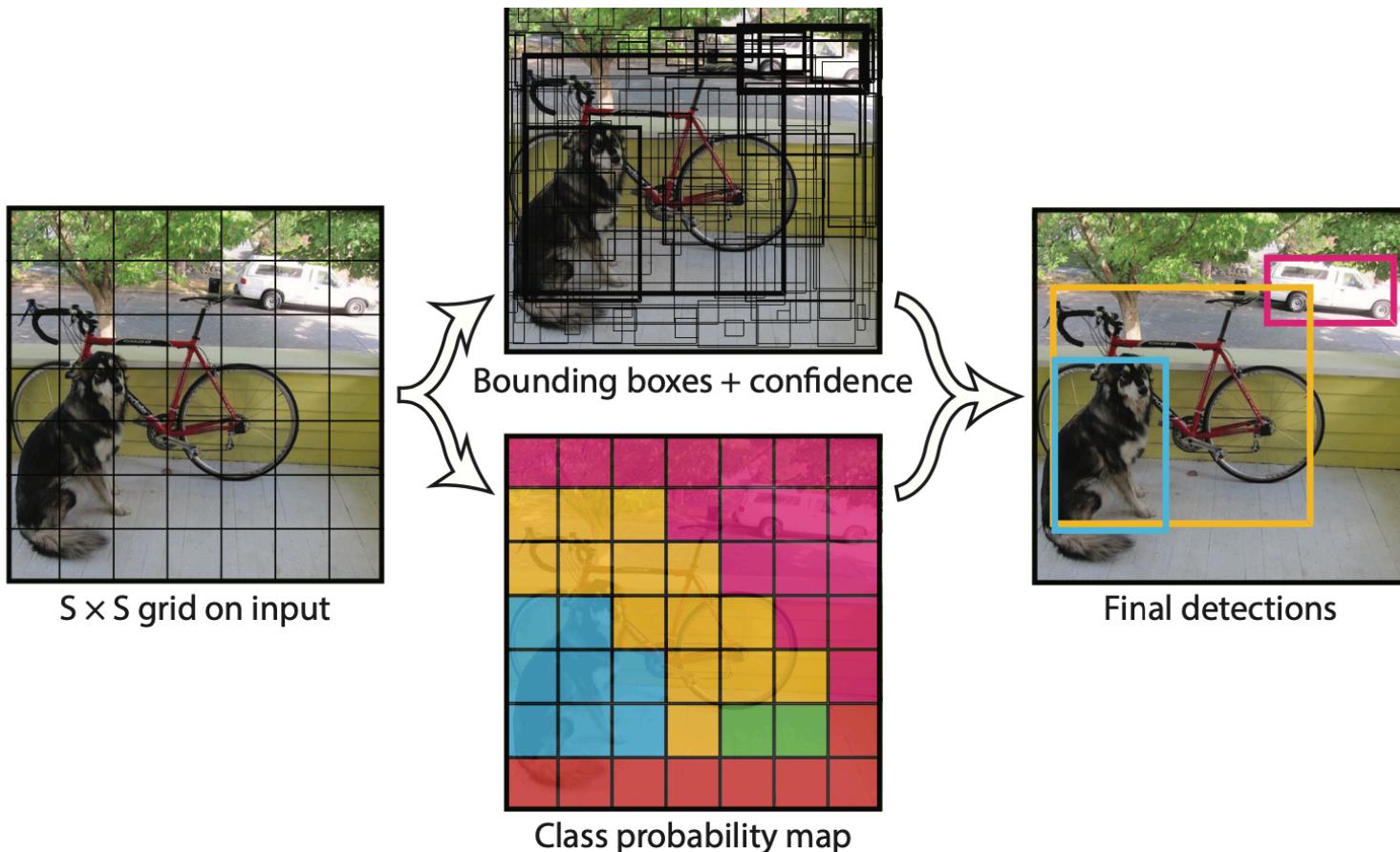
Figure 3: **Left:** Region Proposal Network (RPN). **Right:** Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*[†], Ross Girshick[¶], Ali Farhadi*[†]

University of Washington*, Allen Institute for AI[†], Facebook AI Research[¶]

<http://pjreddie.com/yolo/>



Avoiding Over-Detection: Towards Combined Object Detection and Counting

cell counting

Philip T. G. Jackson and Boguslaw Obara*

School of Engineering and Computing Sciences

Durham University

South Road, DH1 3LE, Durham, UK

{p.t.g.jackson, boguslaw.obara}@durham.ac.uk

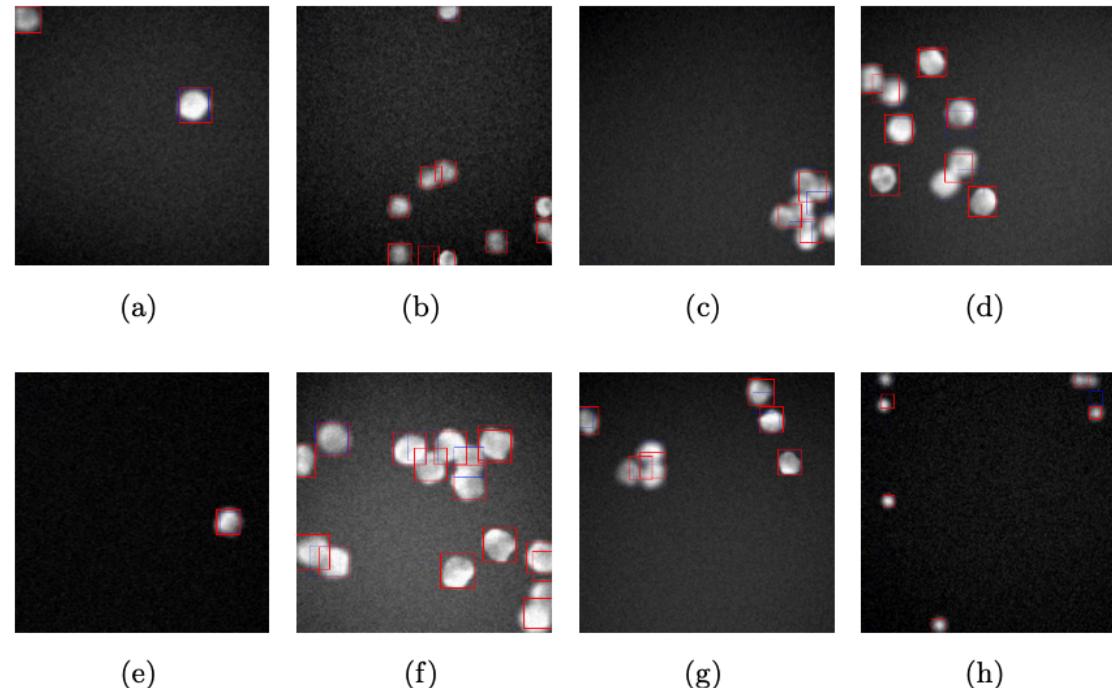


Fig. 2: A sample of detection results on SIMCEP images. Confidence is represented in the transparency of the boxes; all output boxes with confidence above 0.1 are shown. Instead of post-processing with NMS, we simply take boxes with confidence above 0.5 (shown in red) as positive detections. Boxes with confidence

The iNaturalist Species Classification and Detection Dataset

Grant Van Horn¹ Oisin Mac Aodha¹ Yang Song² Yin Cui³ Chen Sun²
Alex Shepard⁴ Hartwig Adam² Pietro Perona¹ Serge Belongie³

¹Caltech

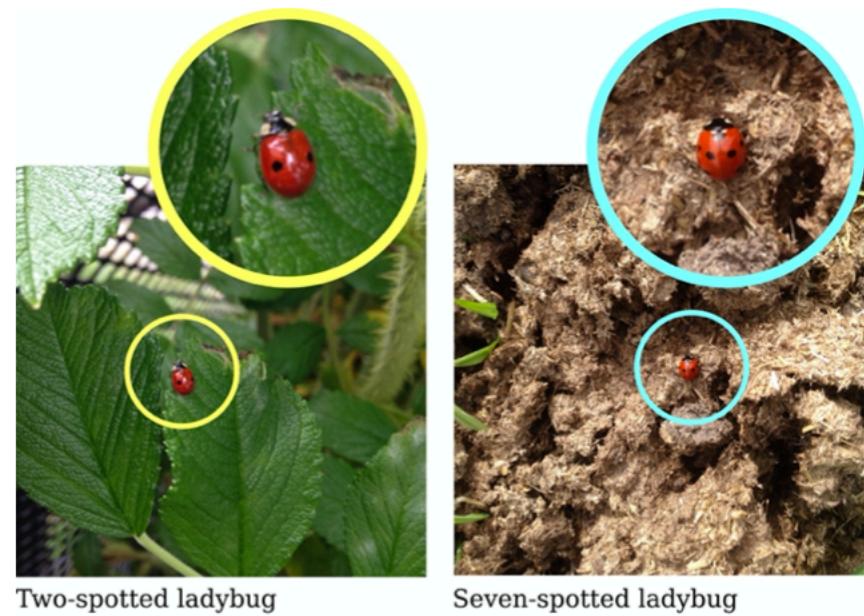
²Google

³Cornell Tech

⁴iNaturalist

	Super-Class	Class	Train	Val	BBoxes
Leaf	Plantae	2,101	158,407	38,206	-
Bee	Insecta	1,021	100,479	18,076	125,679
Bird	Aves	964	214,295	21,226	311,669
Reptile	Reptilia	289	35,201	5,680	42,351
Mammal	Mammalia	186	29,333	3,490	35,222
Mushroom	Fungi	121	5,826	1,780	-
Fish	Amphibia	115	15,318	2,385	18,281
Snail	Mollusca	93	7,536	1,841	10,821
Shark	Animalia	77	5,228	1,362	8,536
Spider	Arachnida	56	4,873	1,086	5,826
Fly	Actinopterygii	53	1,982	637	3,382
Flag	Chromista	9	398	144	-
Flower	Protozoa	4	308	73	-
	Total	5,089	579,184	95,986	561,767

Table 2. Number of images, classes, and bounding boxes in iNat2017 broken down by super-class. ‘Animalia’ is a catch-all category that contains species that do not fit in the other super-classes. Bounding boxes were collected for nine of the super-classes. In addition, the public and private test sets contain 90,427 and 92,280 images, respectively.



Two-spotted ladybug
Adalia bipunctata

Seven-spotted ladybug
Coccinella septempunctata

Figure 1. Two visually similar species from the iNat2017 dataset

GANs

Generative Adversarial Nets

Ian J. Goodfellow,* Jean Pouget-Abadie,[†] Mehdi Mirza, Bing Xu, David Warde-Farley,

Sherjil Ozair,[‡] Aaron Courville, Yoshua Bengio[§]

Département d'informatique et de recherche opérationnelle

Université de Montréal

Montréal QC H3C 3J7

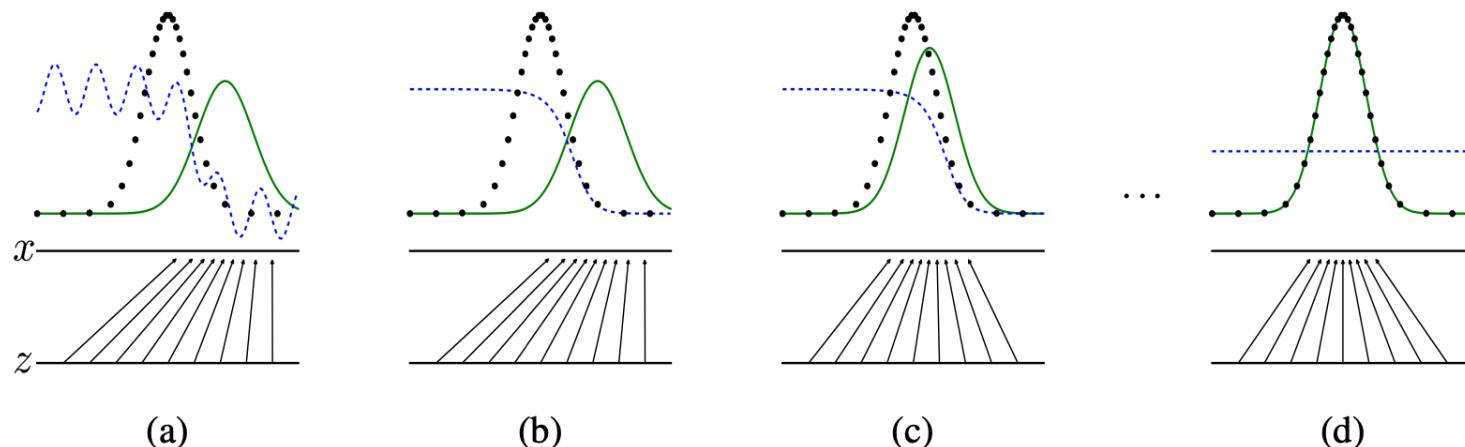


Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution (D , blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\mathbf{x}}$ from those of the generative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which \mathbf{z} is sampled, in this case uniformly. The horizontal line above is part of the domain of \mathbf{x} . The upward arrows show how the mapping $\mathbf{x} = G(\mathbf{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g . (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}$. (c) After an update to G , gradient of D has guided $G(\mathbf{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\mathbf{x}) = \frac{1}{2}$.

GANs



Fig. 3. The three rows show the synthetic bird images of TSGAN-IMG and GAN-IMG, and the real images of Real-IMG.

SYNTHESIS OF IMAGES BY TWO-STAGE GENERATIVE ADVERSARIAL NETWORKS

Qiang Huang, Philip J.B. Jackson, Mark D. Plumbley, Wenwu Wang

Centre for Vision, Speech and Signal Processing
University of Surrey, Guildford, UK

Email: {q.huang, p.jackson, m.plumbley, w.wang}@surrey.ac.uk

Detect to Track and Track to Detect

Christoph Feichtenhofer *
Graz University of Technology
feichtenhofer@tugraz.at

Axel Pinz
Graz University of Technology
axel.pinz@tugraz.at

Andrew Zisserman
University of Oxford
az@robots.ox.ac.uk

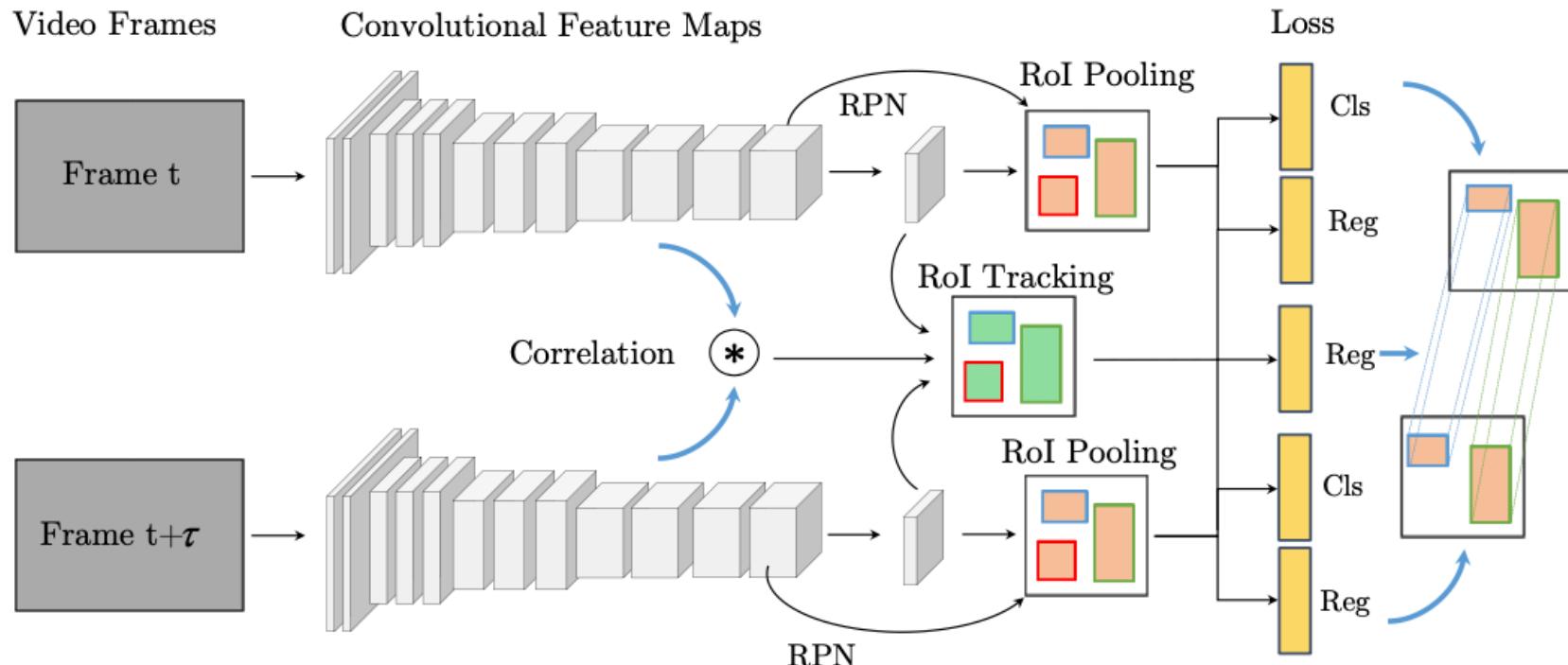
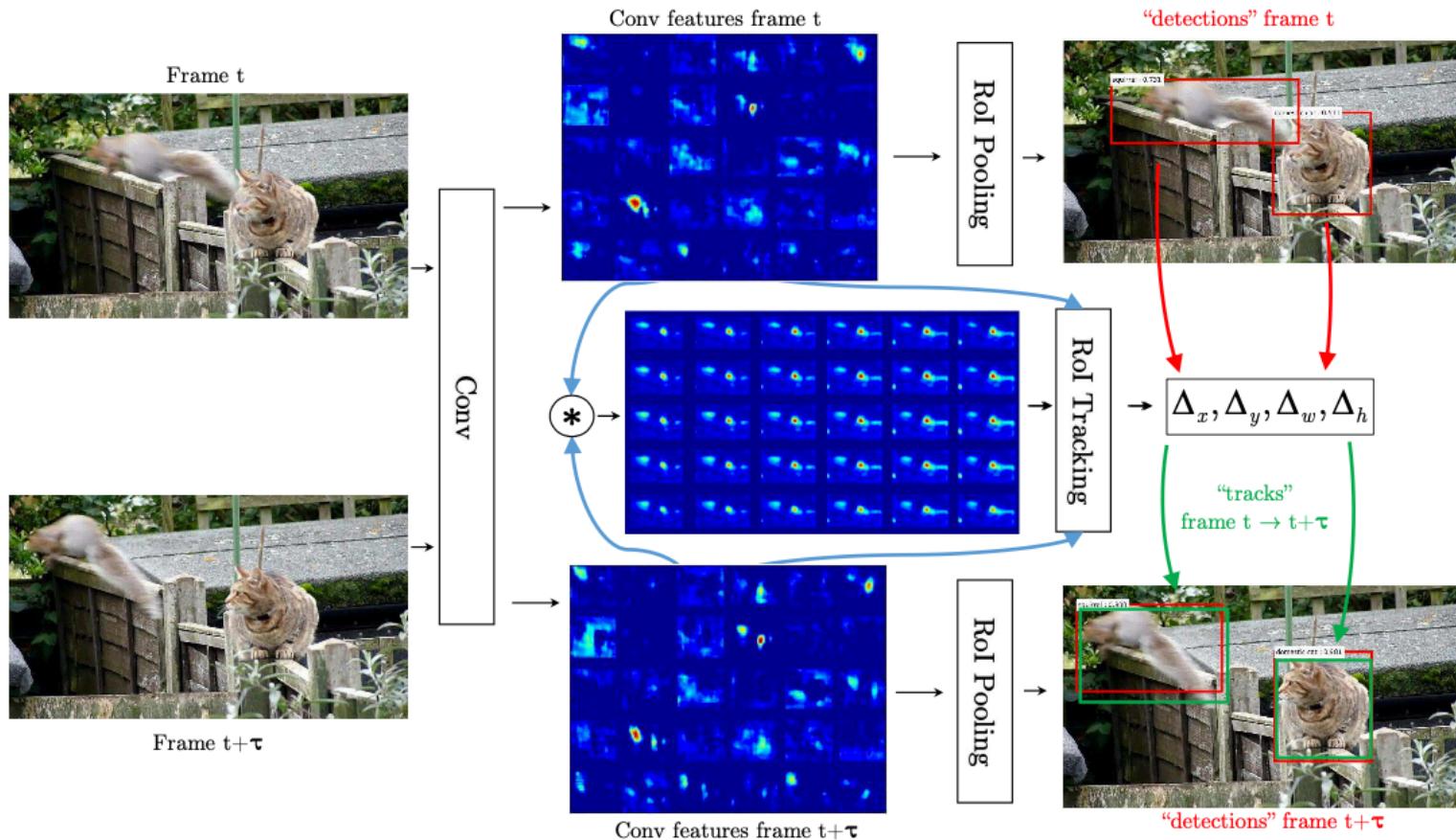


Figure 2. Architecture of our Detect and Track (D&T) approach (see Section 3 for details).

Deep Tracking





Hardware requirements

- GPU vs CPU – order of magnitude speed-up
- High memory requirements
- CUDA programming



Software

- Tensorflow
- Keras: "easy" python library
- PyTorch



An end-to-end open source machine learning platform

TensorFlow For JavaScript For Mobile & IoT For Production

The core open source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.

[Get started with TensorFlow](#)



Keras: The Python Deep Learning library



You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](#), [CNTK](#), or [Theano](#). It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that: