Università degli Studi di Padova

Machine Learning
2024/2025

AMCO
Artificial Intelligence, Machine Learning and Control Research Group

Lecture #26
Convolutional Neural Networks

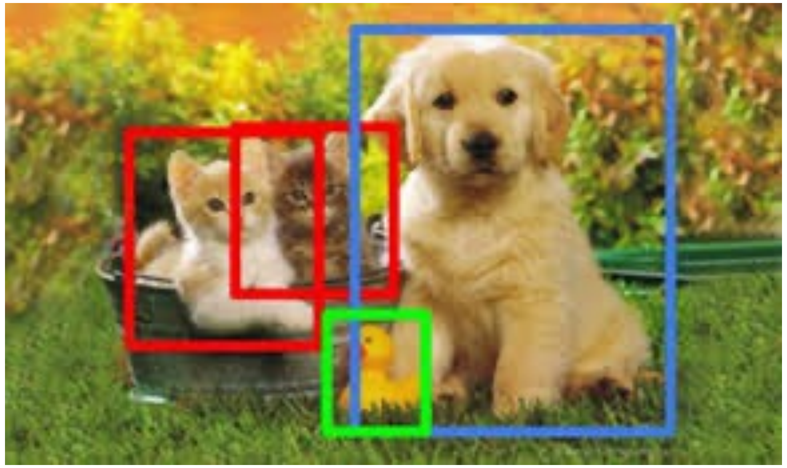
Gian Antonio Susto

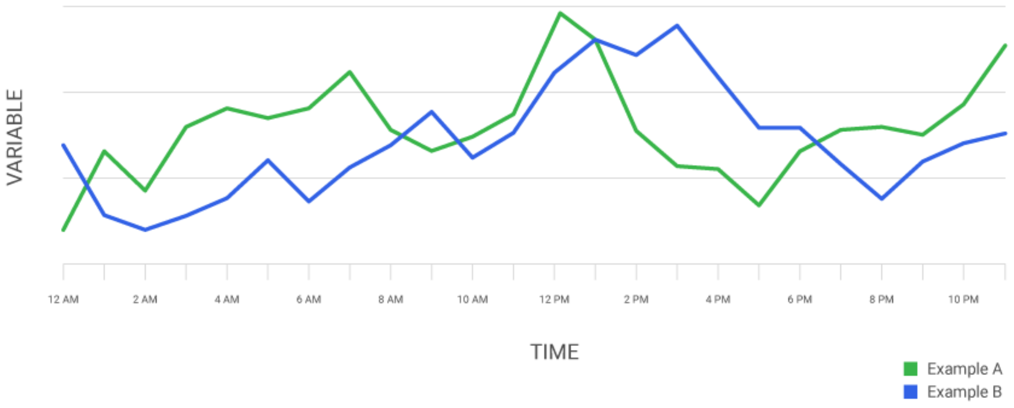
Reference Material (used for this presentation):

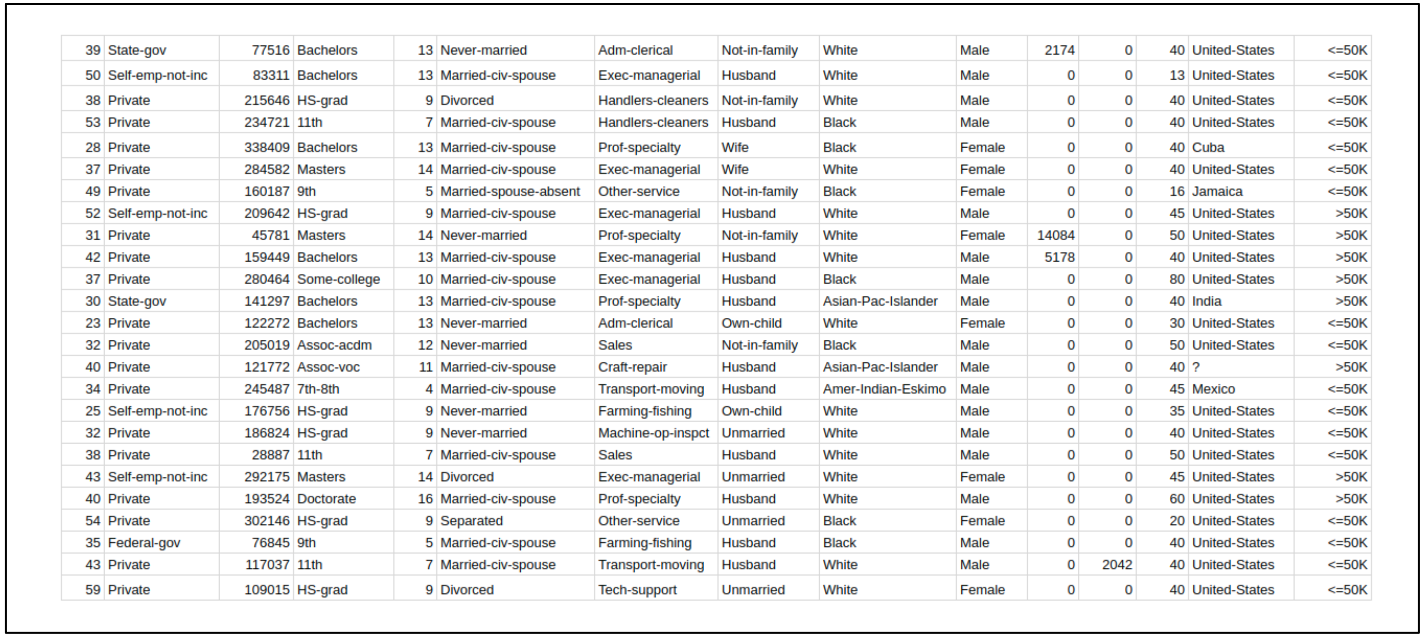
- MIT *Introduction to Deep Learning* http://introtodeeplearning.com
- L. Fridman MIT Deep Learning https://deeplearning.mit.edu/
- I. Goodfellow, Y. Bengio, A. Courville. *Deep learning*. MIT press, 2016.
- ImageNET http://www.image-net.org/
- C. Szegedy et al. *Going Deeper with Convolutions* CVPR2015
- F. Li, J. Johnson, S. Yeung. *Stanford CS231n Convolutional Neural Networks for Visual Recognition* *http://cs231n.stanford.edu/ + http://cs231n.github.io/convolutional-networks/*
- Y. LeCun http://yann.lecun.com/exdb/lenet/
- https://www.kaggle.com/zalando-research/fashionmnist
- MIT https://groups.csail.mit.edu/vision/TinyImages/
- ImageNET http://www.image-net.org/
- https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. *Imagenet classification with deep convolutional neural networks.* Advances in neural information processing systems. 2012.
- C. Szegedy et al. *Going Deeper with Convolutions* CVPR2015
- Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." Proceedings of the 26th annual international conference on machine learning. ACM, 2009
- Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014
- Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." Proceedings of the 26th annual international conference on machine learning. ACM, 2009
- Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014
- The complete history of Lenna http://www.ee.cityu.edu.hk/~lmpo/lenna/Lenna97.html
- Style transfer https://deepart.io/
- https://paperswithcode.com/sota/image-classification-on-imagenet

From lecture #02:

Different tasks (objectives), different models, different data type... and different stages of development!

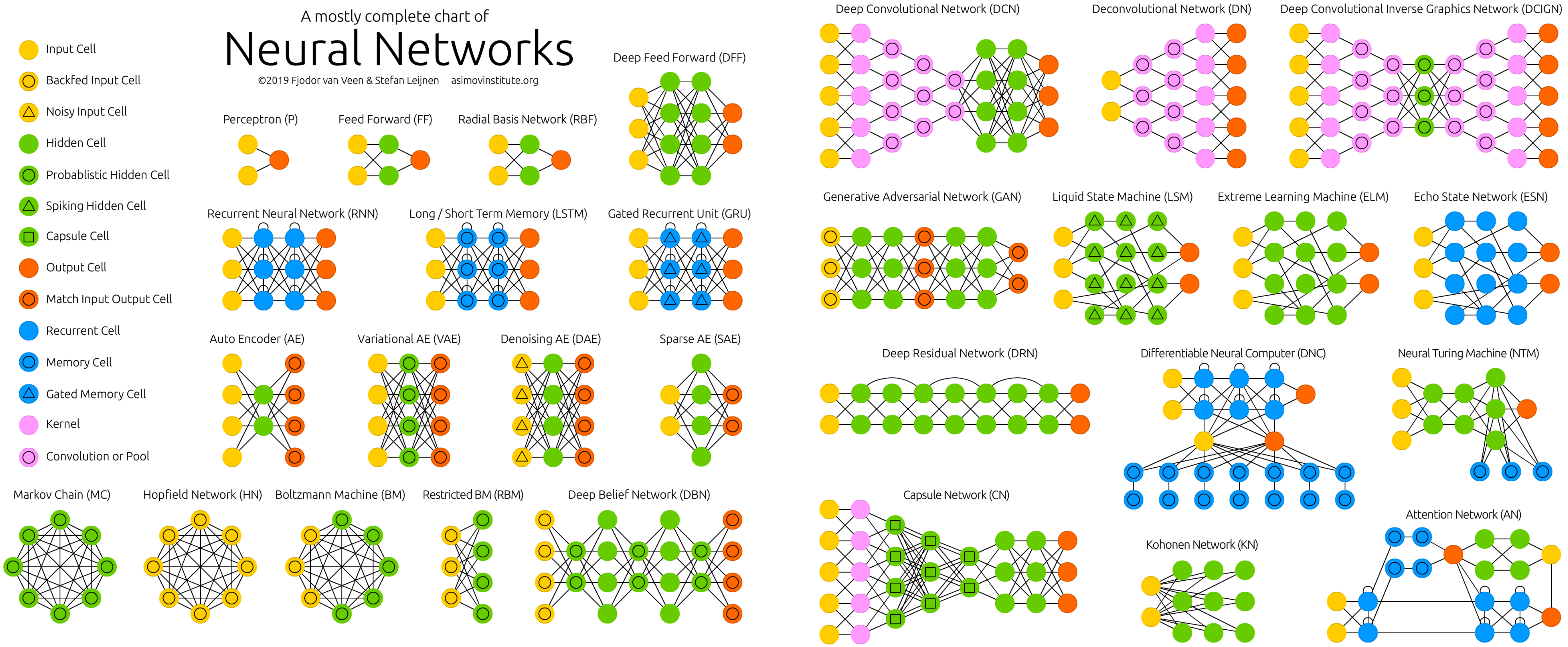






39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K
37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0	0	80	United-States	>50K
30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	40	India	>50K
23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0	30	United-States	<=50K
32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0	50	United-States	<=50K
40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	0	0	40	?	>50K
34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-moving	Husband	Amer-Indian-Eskimo	Male	0	0	45	Mexico	<=50K
25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fishing	Own-child	White	Male	0	0	35	United-States	<=50K
32	Private	186824	HS-grad	9	Never-married	Machine-op-inspct	Unmarried	White	Male	0	0	40	United-States	<=50K
38	Private	28887	11th	7	Married-civ-spouse	Sales	Husband	White	Male	0	0	50	United-States	<=50K
43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-managerial	Unmarried	White	Female	0	0	45	United-States	>50K
40	Private	193524	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	60	United-States	>50K
54	Private	302146	HS-grad	9	Separated	Other-service	Unmarried	Black	Female	0	0	20	United-States	<=50K
35	Federal-gov	76845	9th	5	Married-civ-spouse	Farming-fishing	Husband	Black	Male	0	0	40	United-States	<=50K
43	Private	117037	11th	7	Married-civ-spouse	Transport-moving	Husband	White	Male	0	2042	40	United-States	<=50K
59	Private	109015	HS-grad	9	Divorced	Tech-support	Unmarried	White	Female	0	0	40	United-States	<=50K

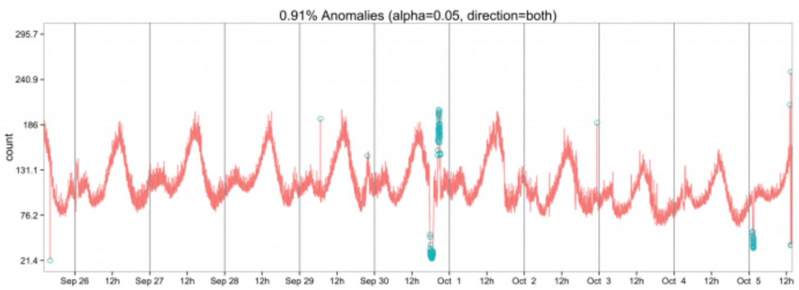
With DL architectures we are able to handle all sort of data


A mostly complete chart of
Neural Networks
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org
Input Cell
Backfed Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Capsule Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Gated Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Differentiable Neural Computer (DNC)
Neural Turing Machine (NTM)
Capsule Network (CN)
Kohonen Network (KN)
Attention Network (AN)

Today: Convolutional Neural Networks (CNNs) to deal with images/videos

Today: Convolutional Neural Networks (CNNs) to deal with images/videos

We will not see in this course Recurrent Neural Networks (RNNs) to deal with sequence learning tasks



Outline:

- CNN building blocks
- Tasks in Computer Vision
- Datasets for Computer Vision & Historical CNNs
- What a CNN sees?

Human Vision

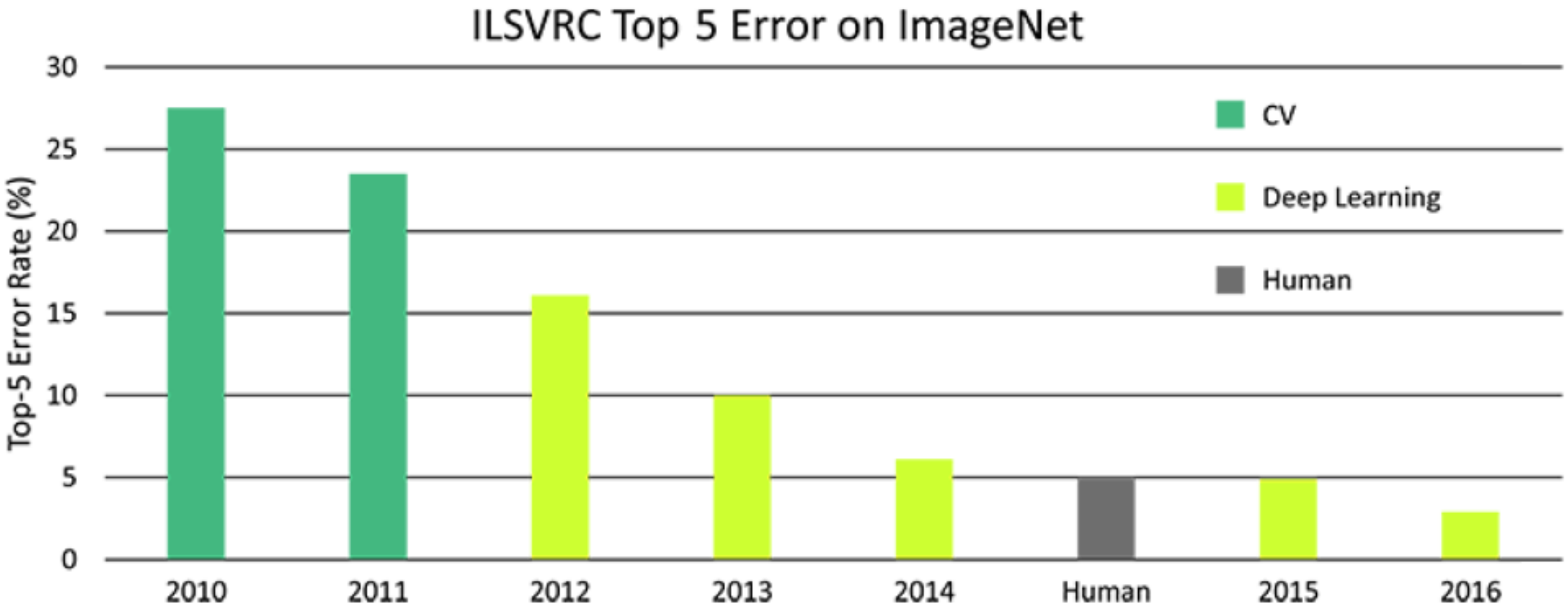
A system trained on 540 million years of data

Computer Vision

Started in the 60s aiming at automatically doing what human vision can do

CV is undoubtly THE field that mostly benefit from Deep Learning

DL has been disruptive in computer vision!

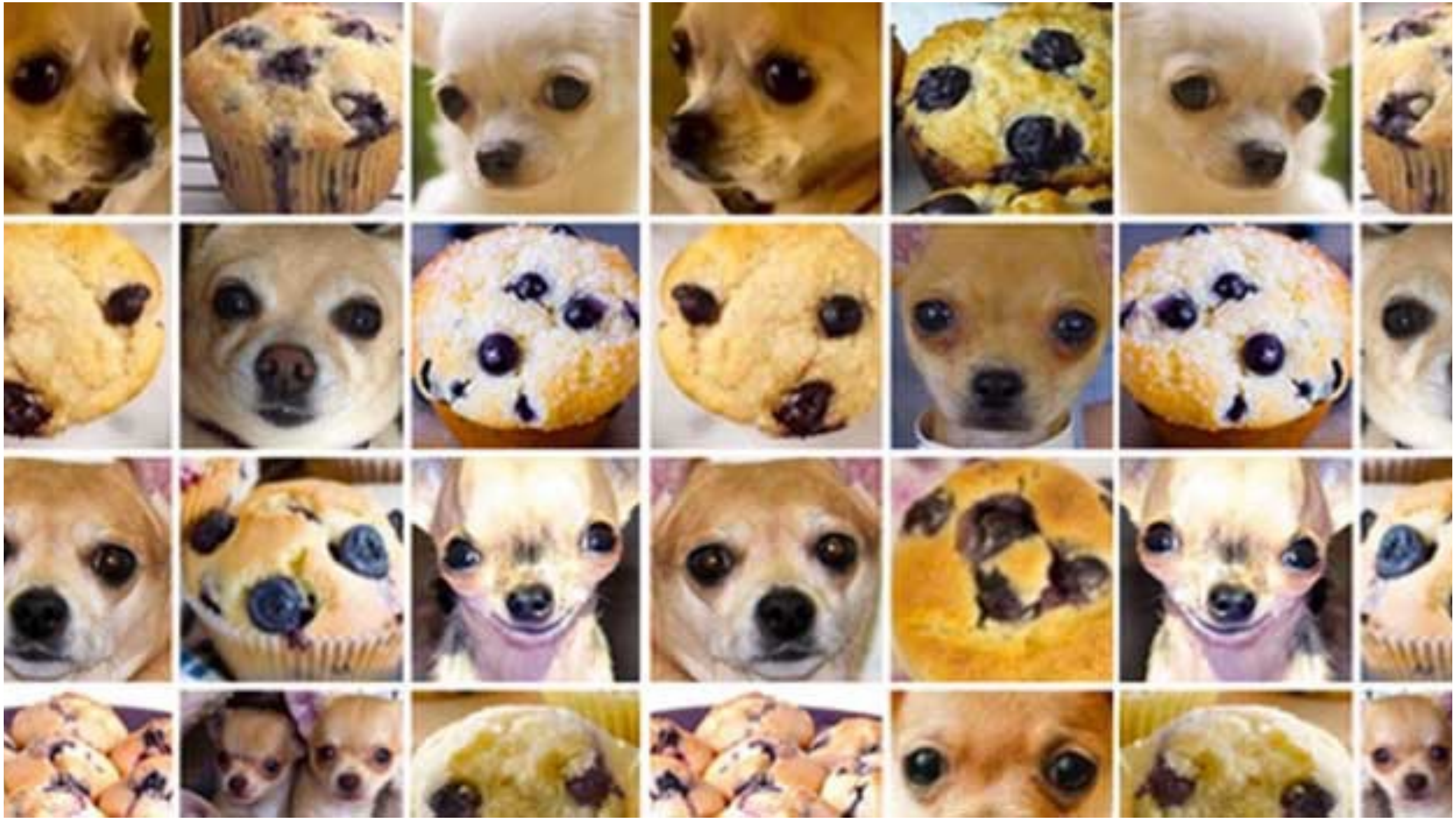


IMAGENET

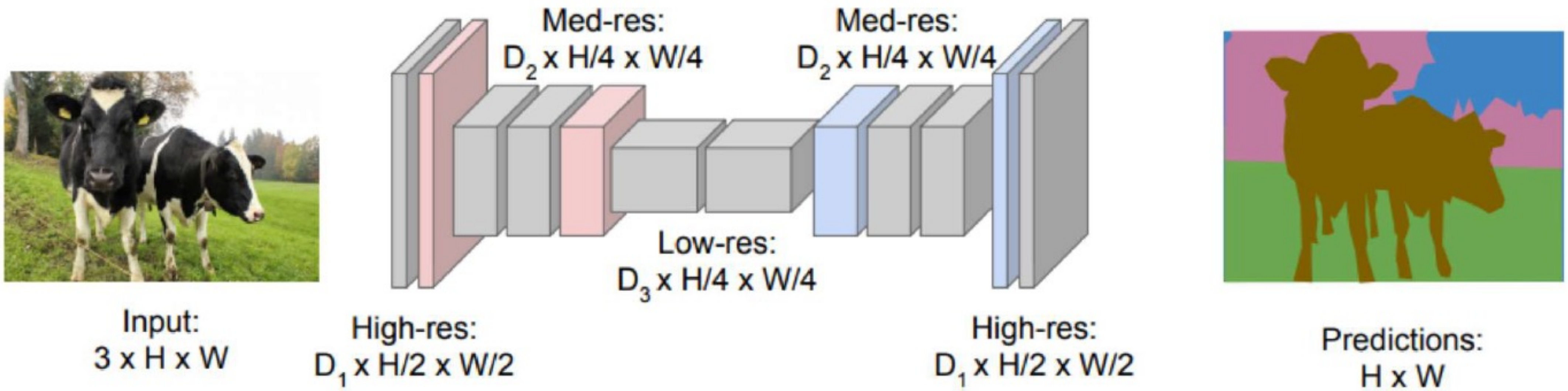
A dataset with more than 14 Million annotated images of 20K categories

G. Prando. *Seminars on Deep Learning*. UniPD, 2017

Example Task #1: Classification Chihuahua vs Muffin



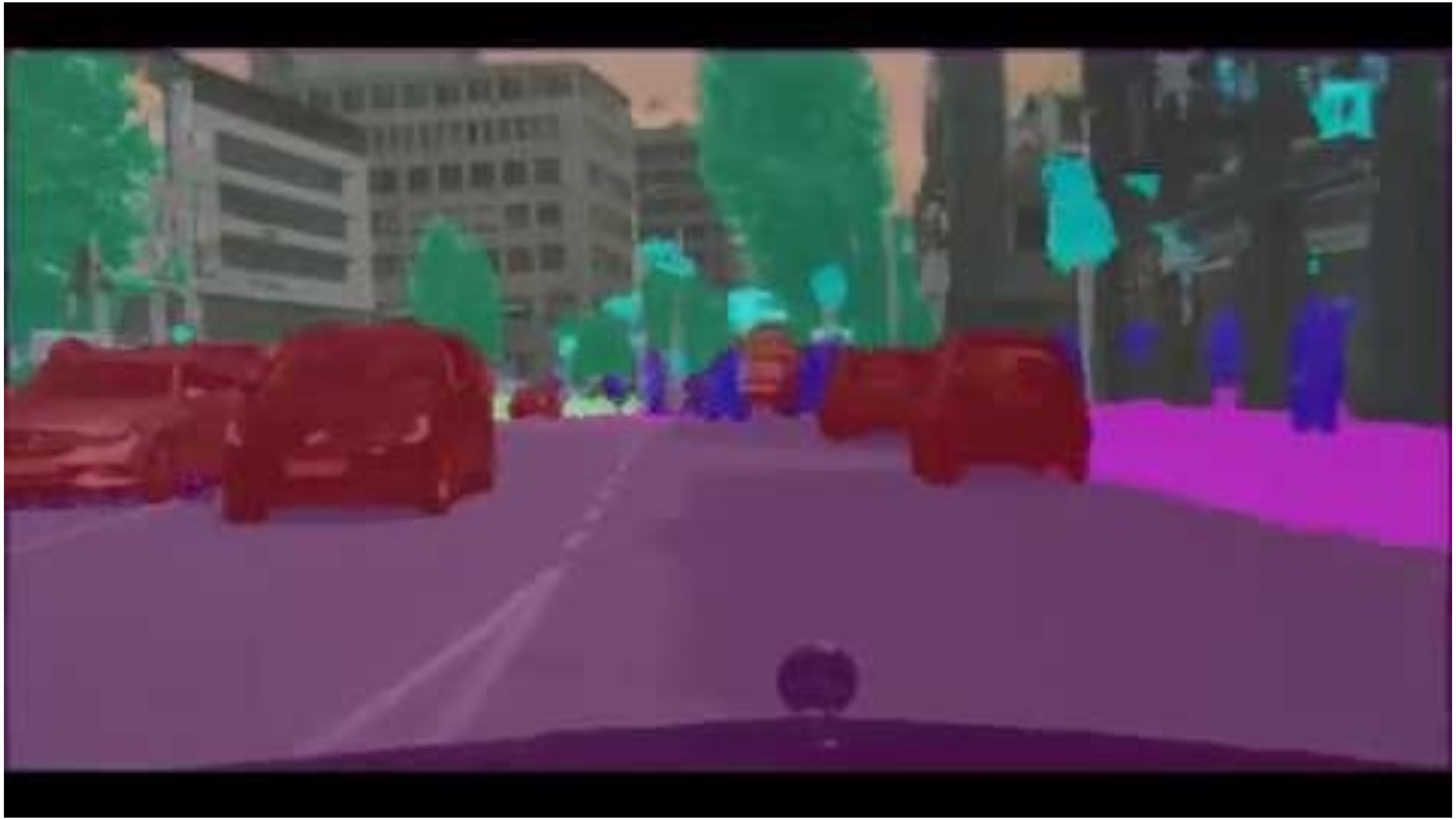
Example Task #2: Semantic Segmentation



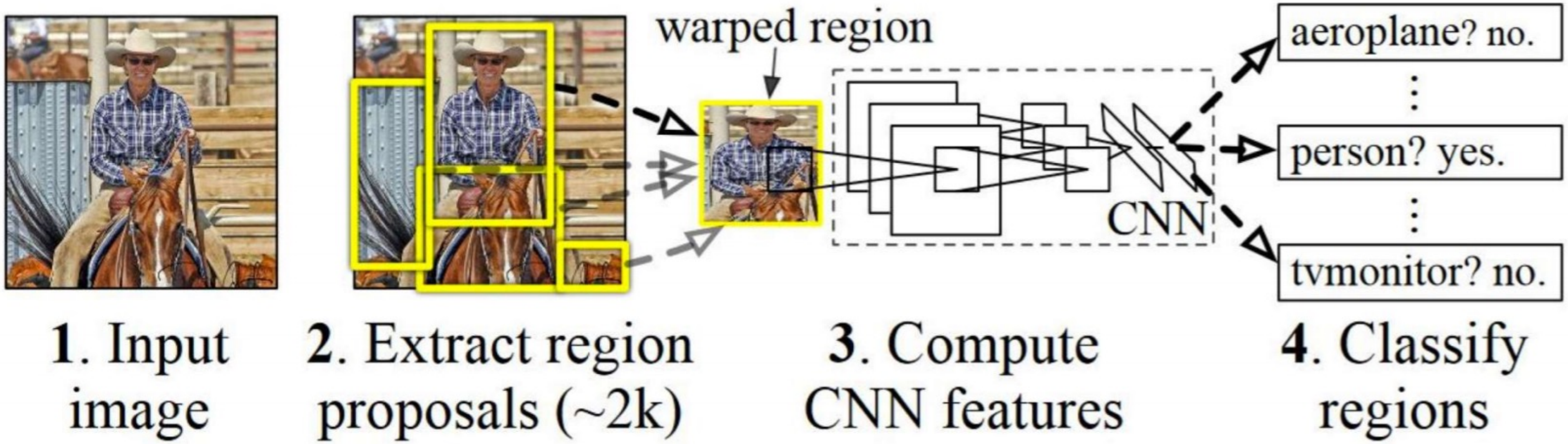
All convolutional layers with downsampling and upsampling operations

-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

Example Task #2: Semantic Segmentation



Example Task #3: Object Detection

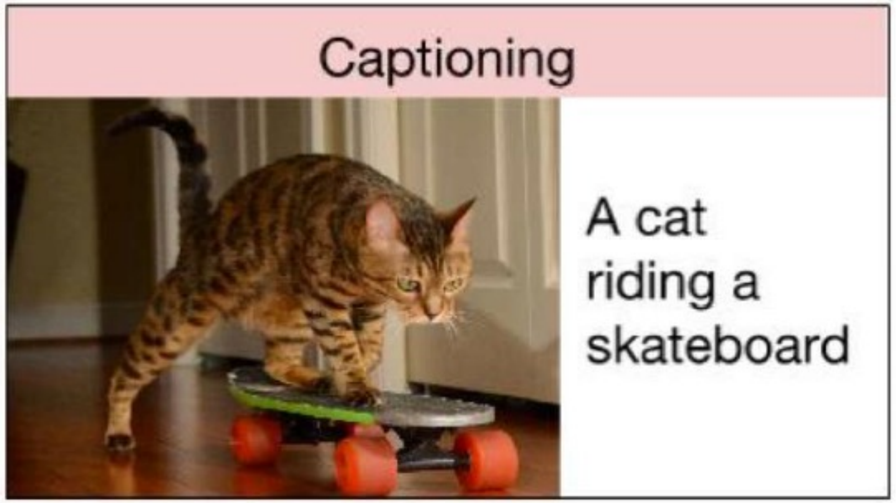


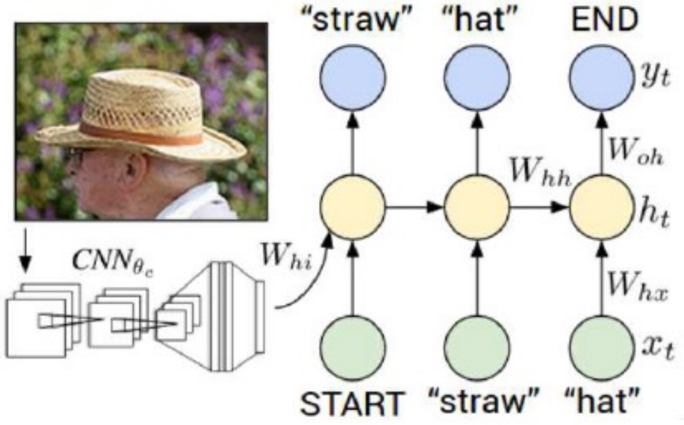
MIT *Introduction to Deep Learning* http://introtodeeplearning.com

1. Find regions that we think have objects
2. Classify such objects

Example Task #4: Image Captioning

Combination of CNN and sequence learning

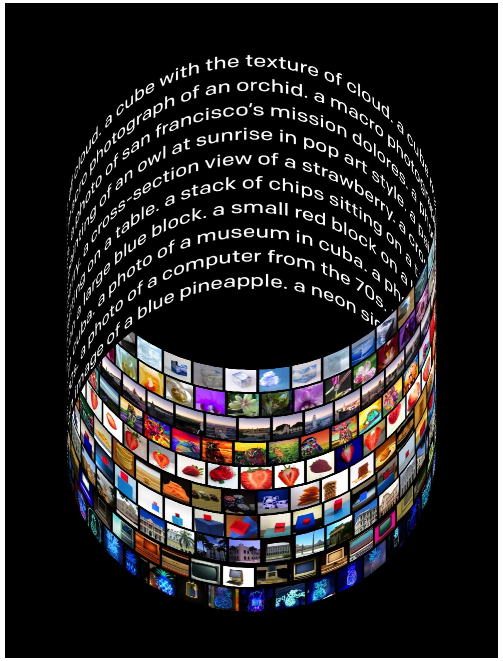


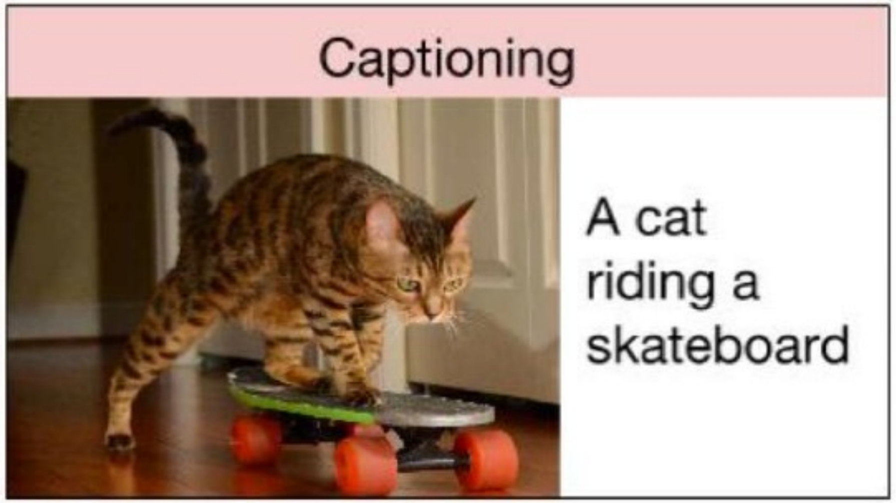


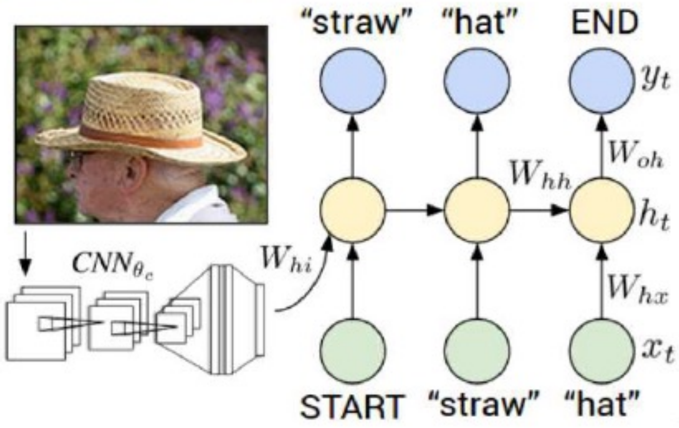
-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

Example Task #4: Image Captioning

Combination of CNN and sequence learning

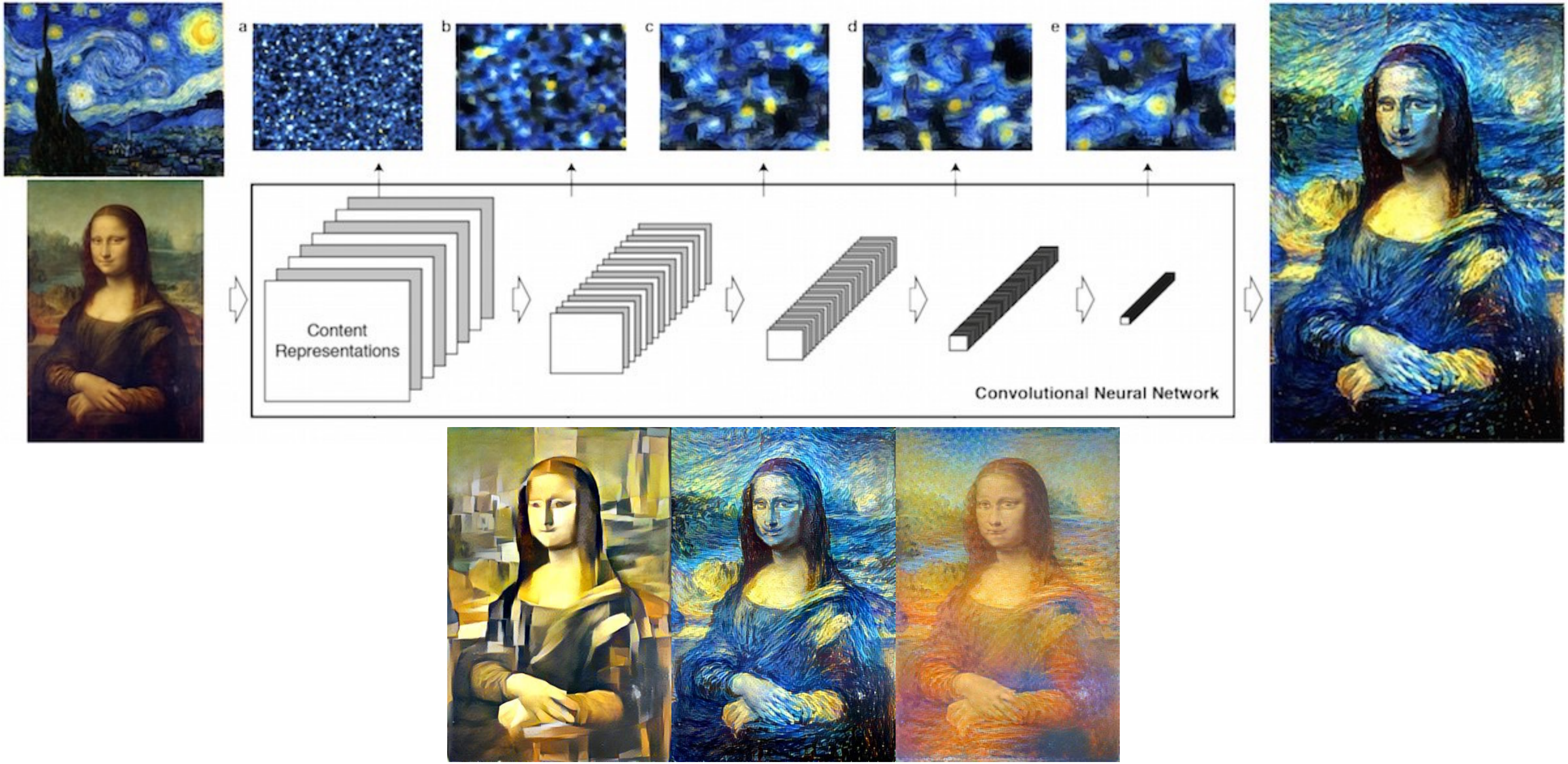

a cube with the texture of cloud. a cube
a photograph of an orchid. a macro photo
photo of san francisco's mission dolores
of an owl at sunrise in pop art style
a cross-section view of a strawberry
a table. a stack of chips sitting on a
blue block. a small red block on a
a photo of a museum in cuba. a ph
a photo of a computer from the 70s.
a photo of a blue pineapple. a neon si





-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

Example Task #5: Style Transfer



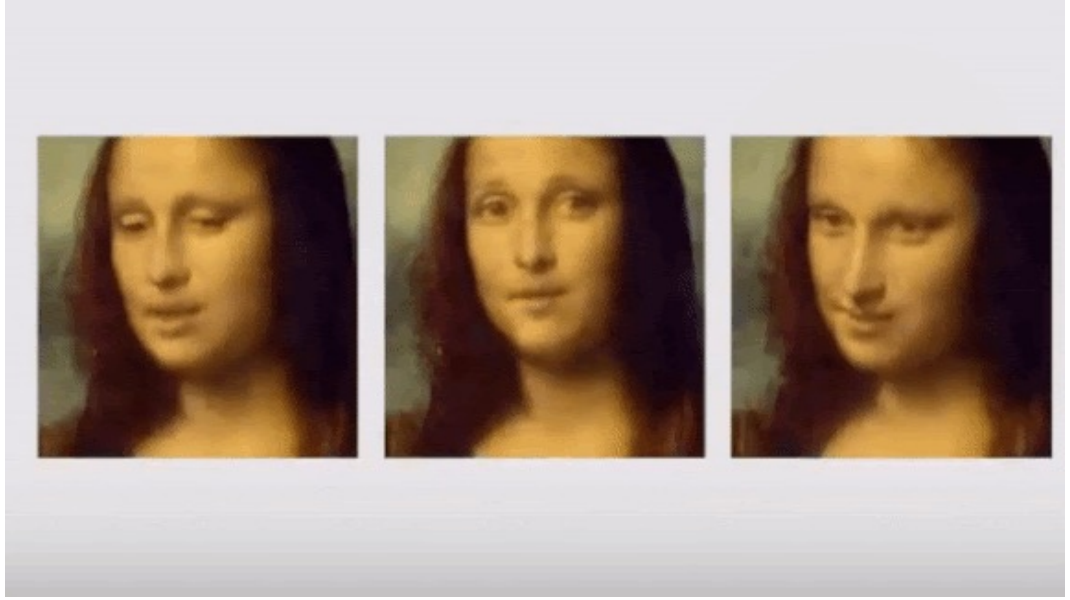
Example Task #5: Style Transfer

-https://deepart.io/ (not working anymore)

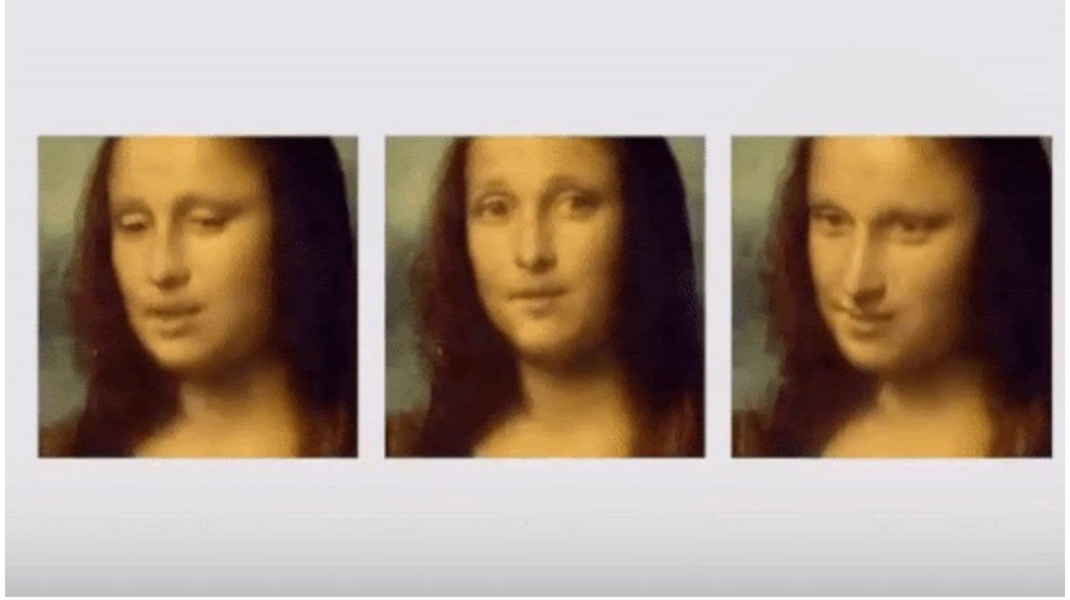
Example Task #5: Style Transfer

-https://deepart.io/ (not working anymore)

Example Task #5: Style Transfer

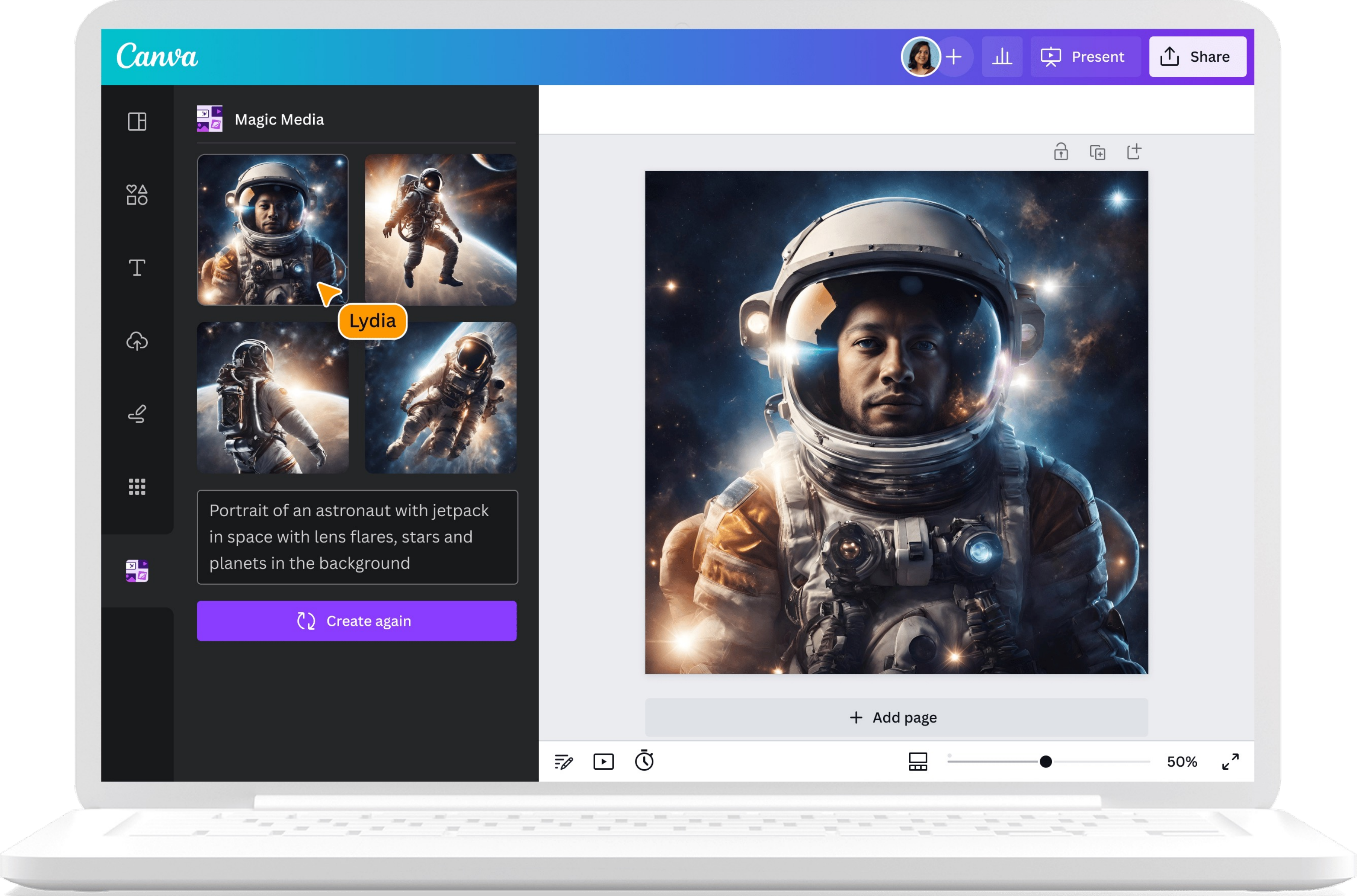


Example Task #5: Style Transfer



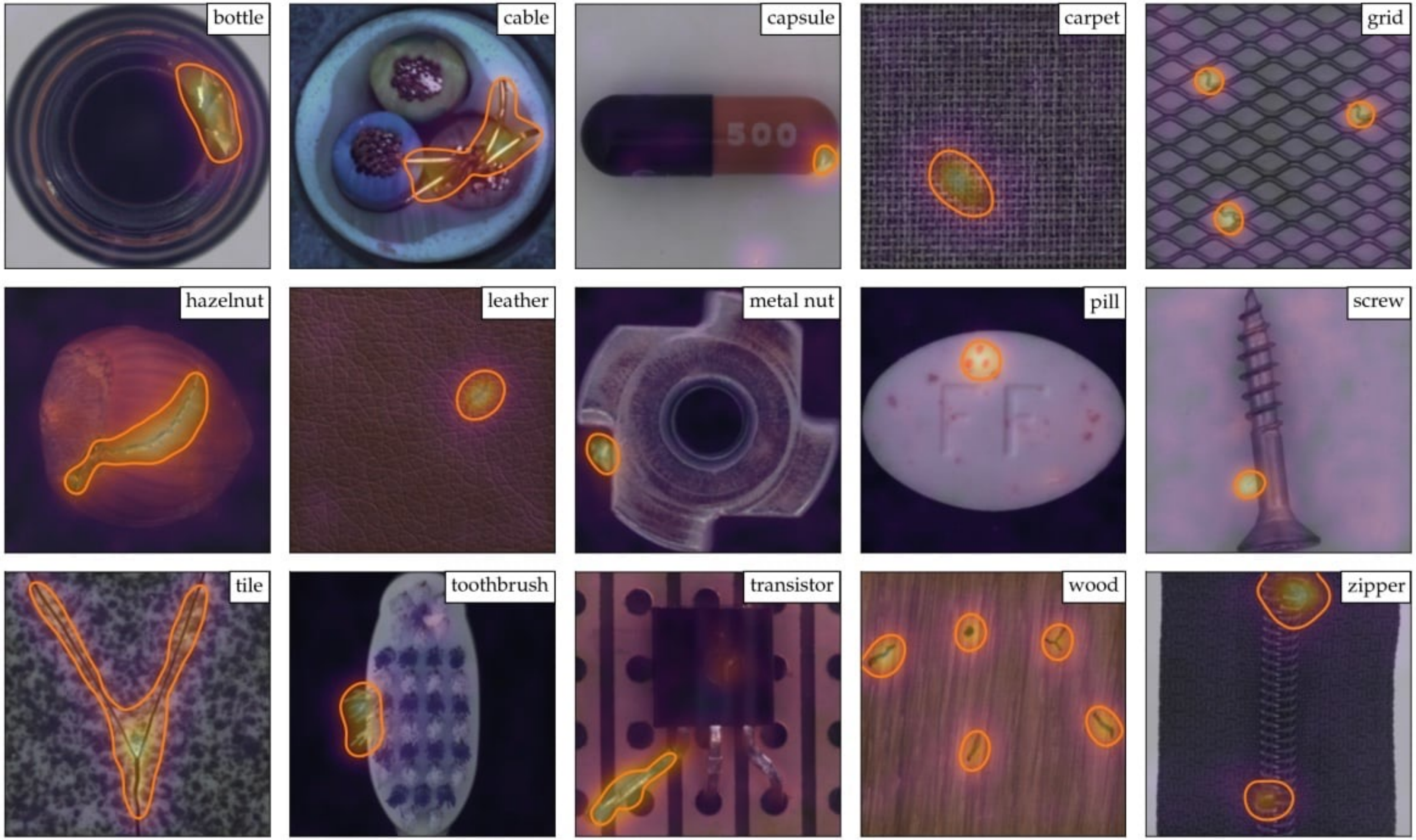
Some style transfer works
- Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image style transfer using convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2414-2423).
- Johnson, J., Alahi, A., & Fei-Fei, L. (2016, October). Perceptual losses for real-time style transfer and super-resolution. In European conference on computer vision (pp. 694-711). Springer, Cham.
- Luan, F., Paris, S., Shechtman, E., & Bala, K. (2017). Deep photo style transfer. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4990-4998).
- i, X., Liu, S., Kautz, J., & Yang, M. H. (2019). Learning linear transformations for fast image and video style transfer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 3809-3817)

Example Task #6: Image Generation

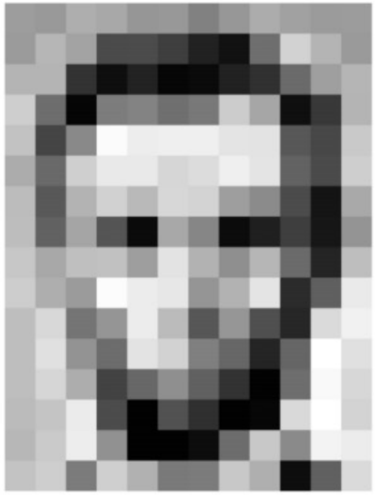


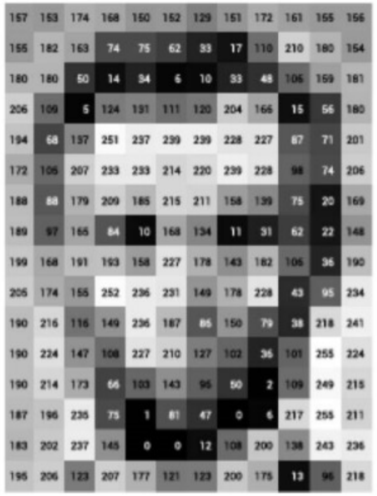
Portrait of an astronaut with jetpack in space with lens flares, stars and planets in the background

Example Task #7: Visual Anomaly Detection

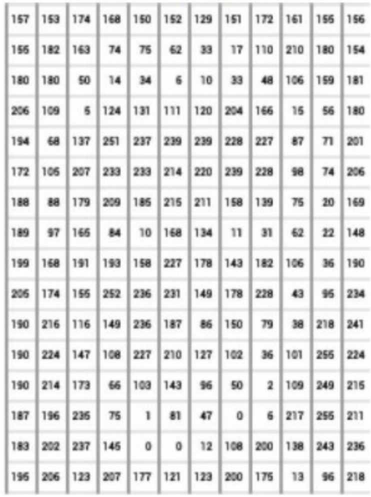


Images are numbers



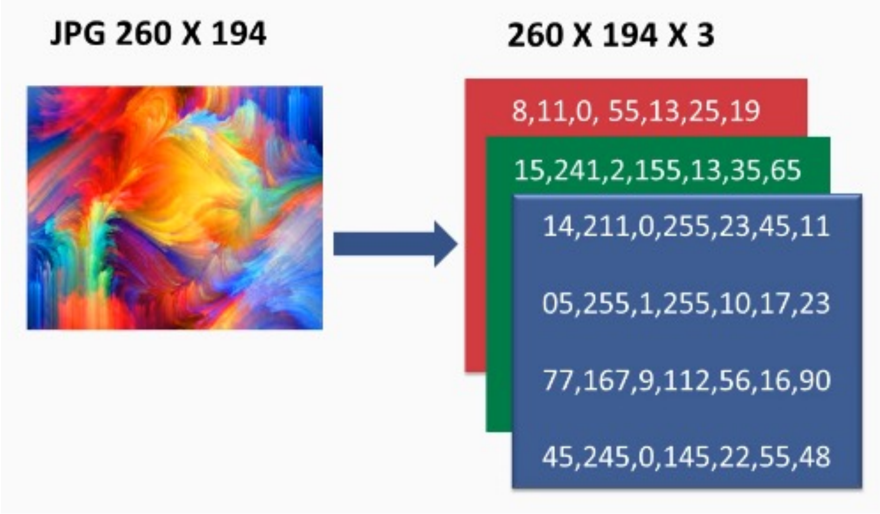


What the computer sees

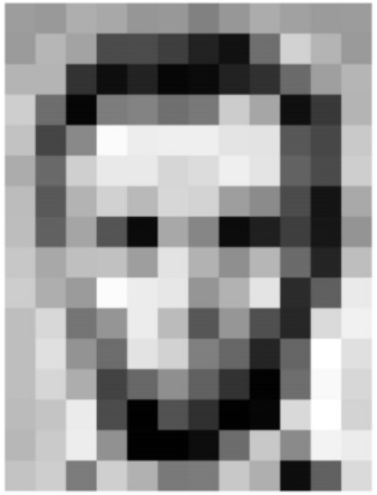


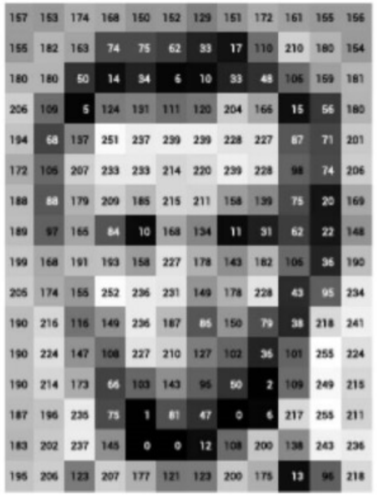
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

L. Fridman MIT Deep Learning https://deeplearning.mit.edu/

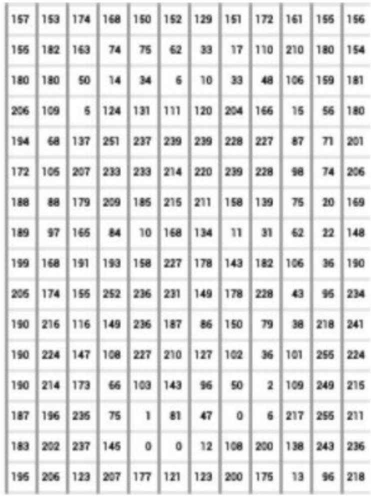


Images are numbers





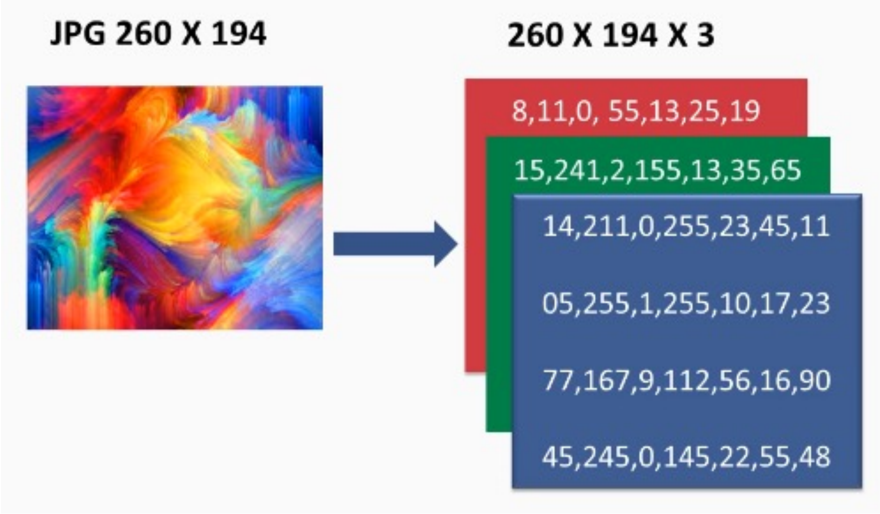
What the computer sees



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

L. Fridman MIT Deep Learning https://deeplearning.mit.edu/

Any (simple) ideas on how to feed this into a model?



8,11,0, 55,13,25,19

15,241,2,155,13,35,65

14,211,0,255,23,45,11

05,255,1,255,10,17,23

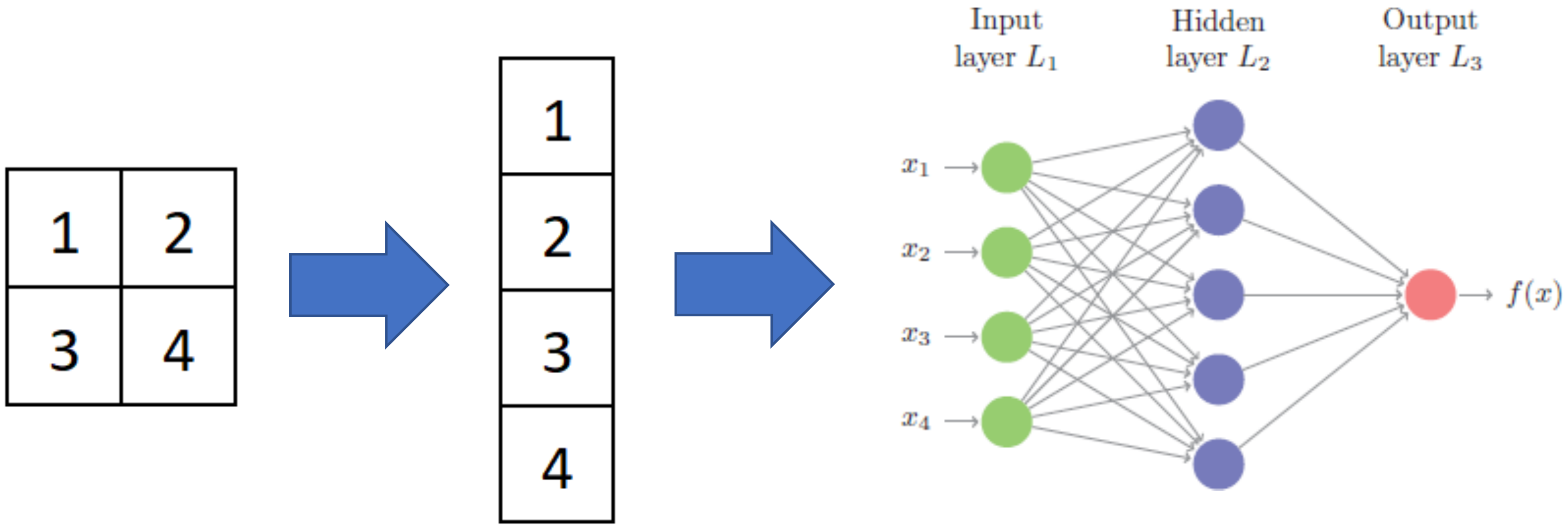
77,167,9,112,56,16,90

45,245,0,145,22,55,48

Using FFNN is not a good idea...

- 2D images converted to arrays: **spatial information is lost!**

- Each hidden unit is connected to all units in the previous layers: **lots of parameters!**



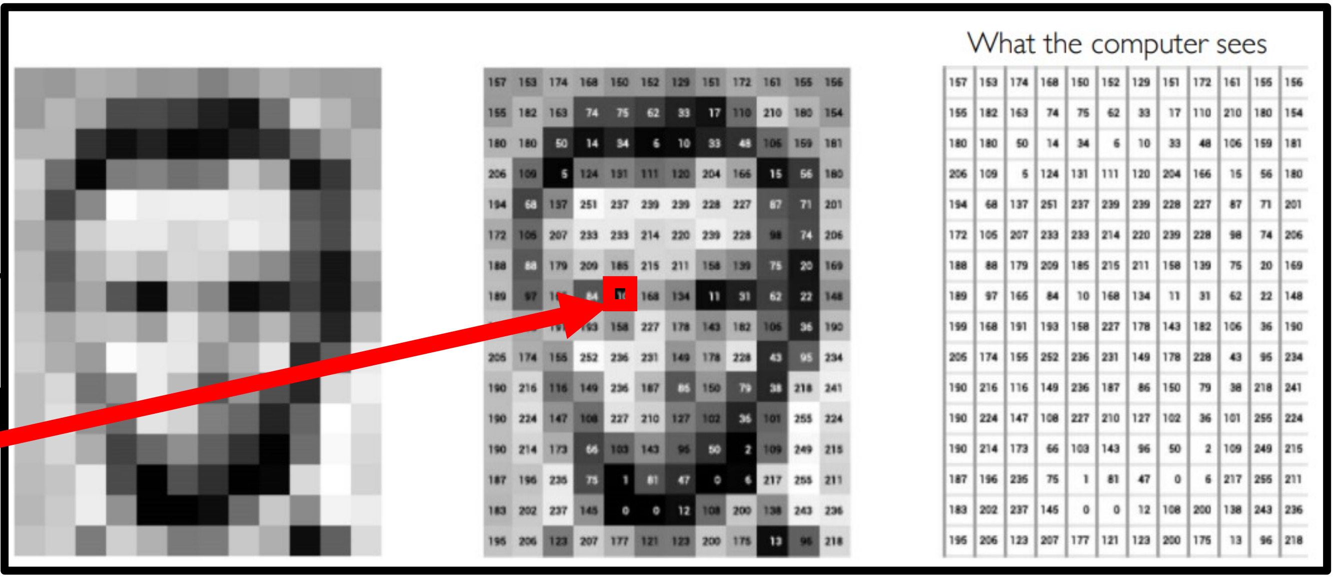
-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

Using FFNN is not a good

- 2D images converted to array

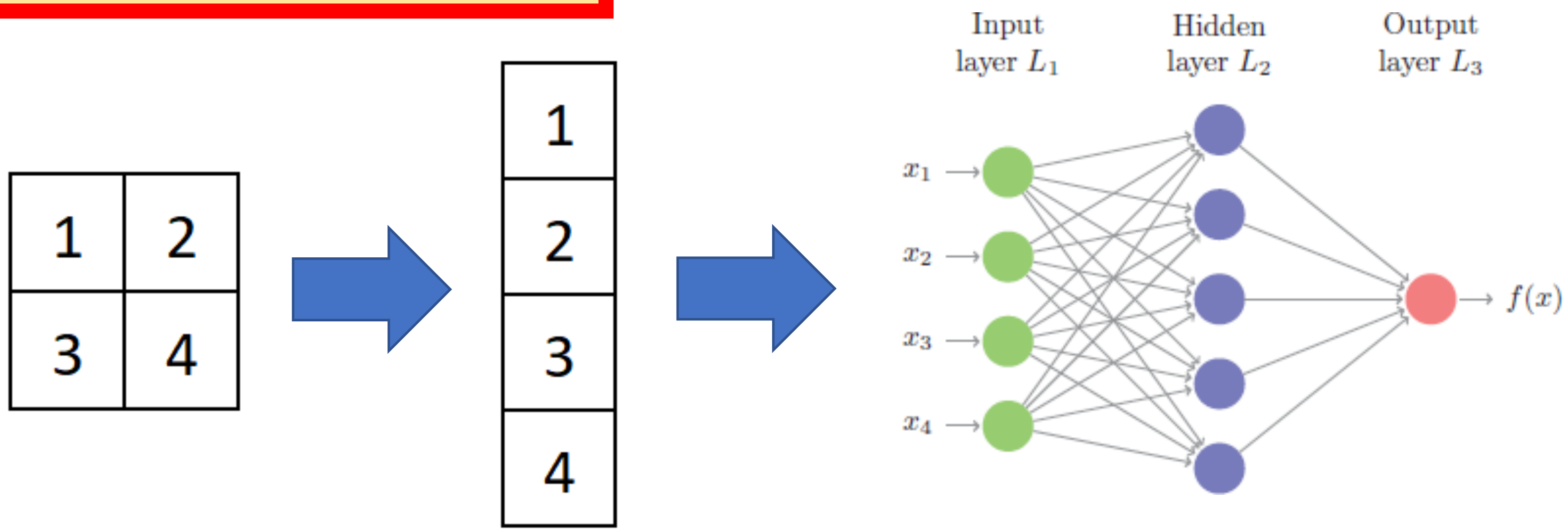
ed

Does the value in this specific pixel really matters by itself?



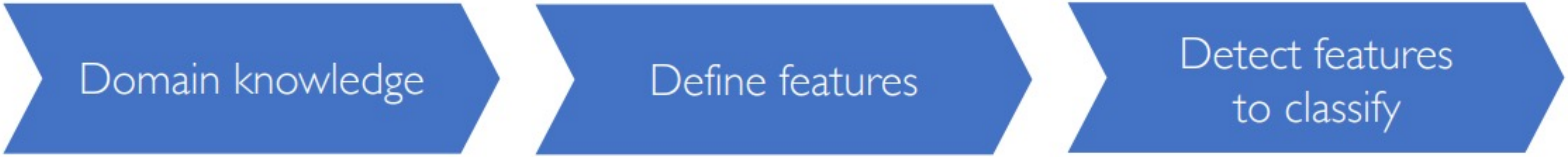
What the computer sees

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218



-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

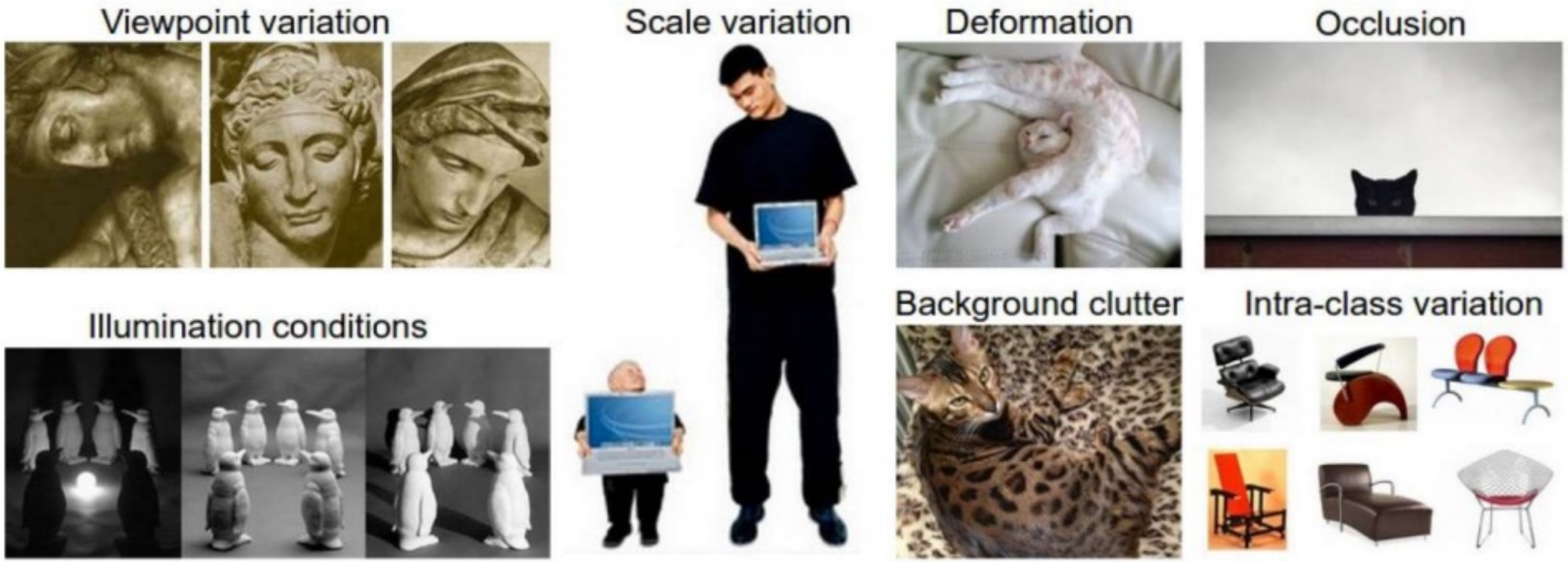
In Computer Vision, feature engineering is hard



Domain knowledge

Define features

Detect features to classify



Viewpoint variation

Scale variation

Deformation

Occlusion

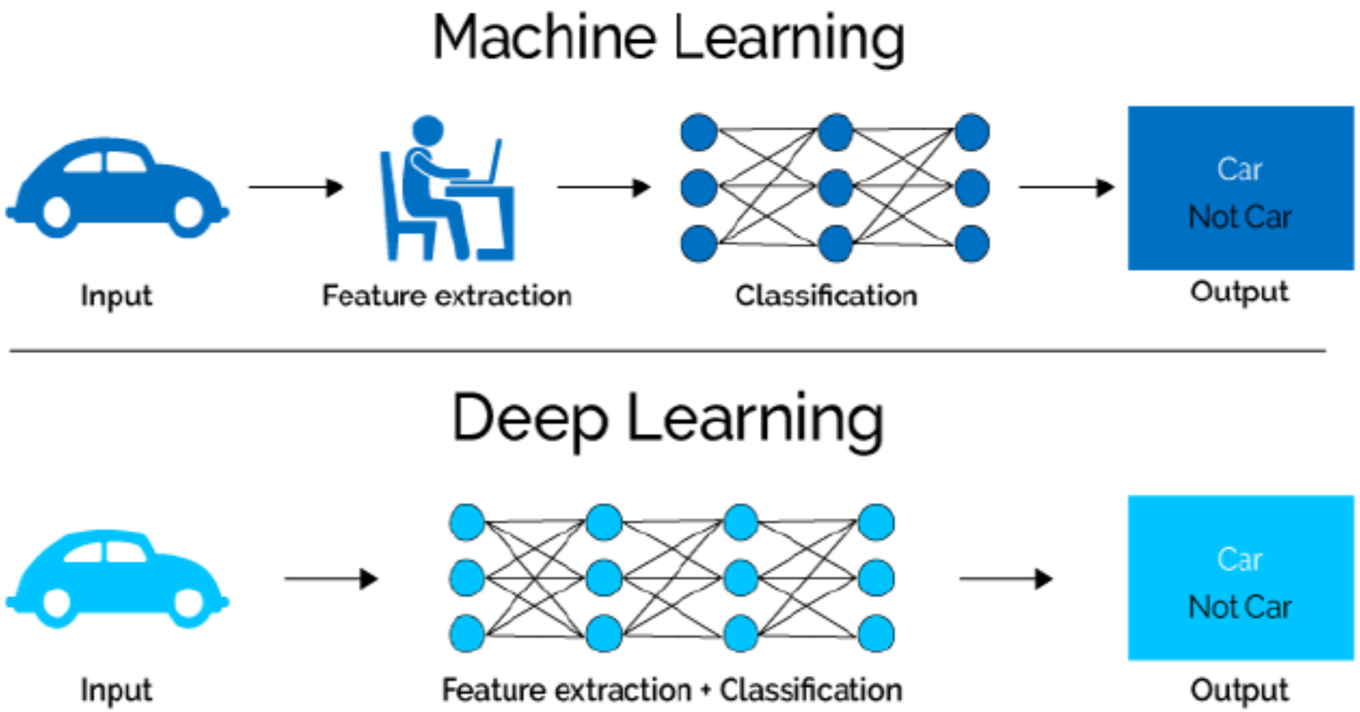
Illumination conditions

Background clutter

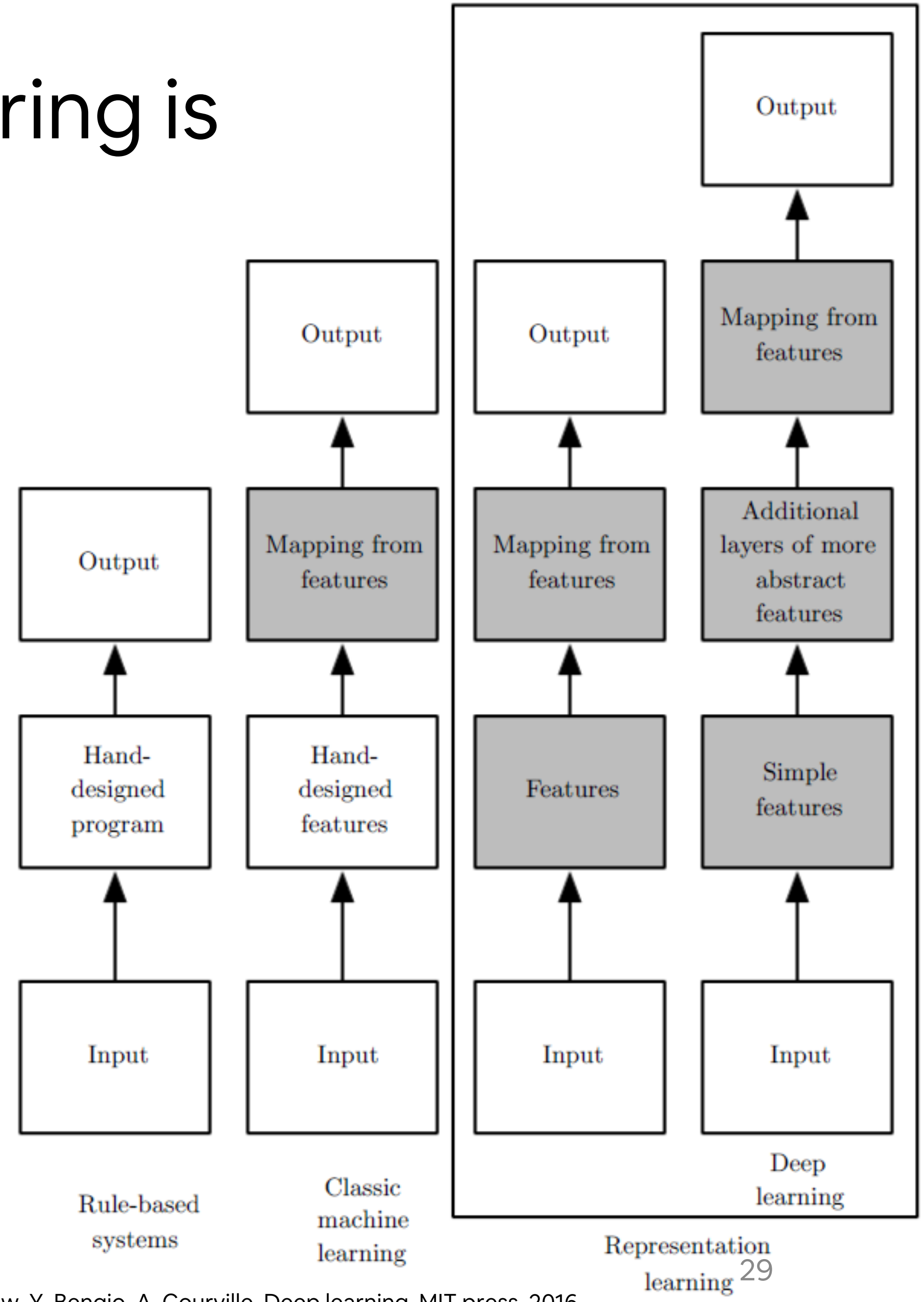
Intra-class variation

-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

As seen before: no feature engineering is needed with DL



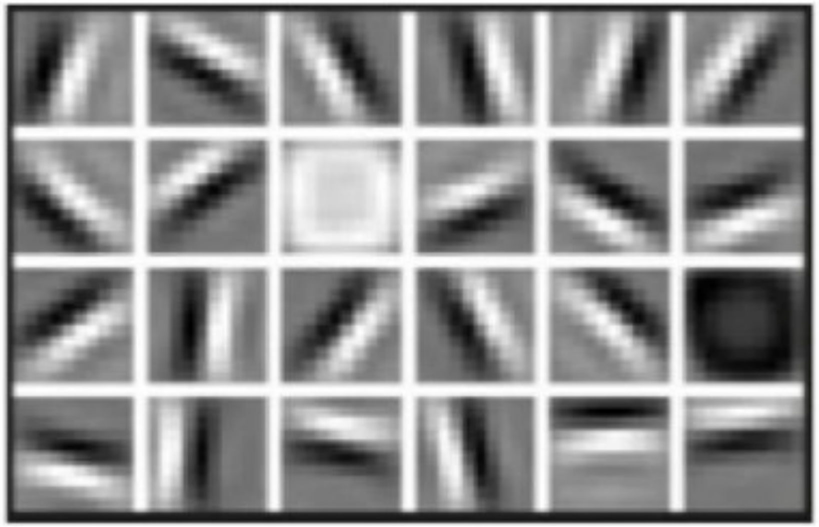
L. Fridman MIT Deep Learning https://deeplearning.mit.edu/



I. Goodfellow, Y. Bengio, A. Courville. Deep learning. MIT press, 2016.

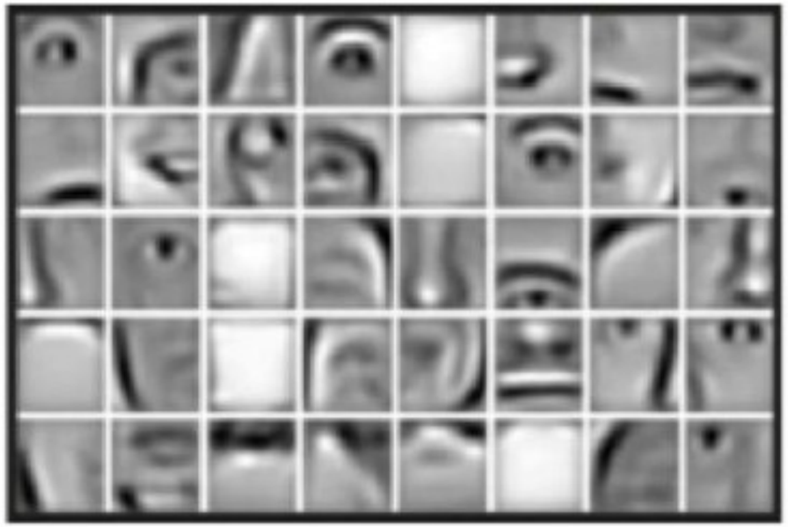
Features extracted by Deep Neural Networks

Low Level Features



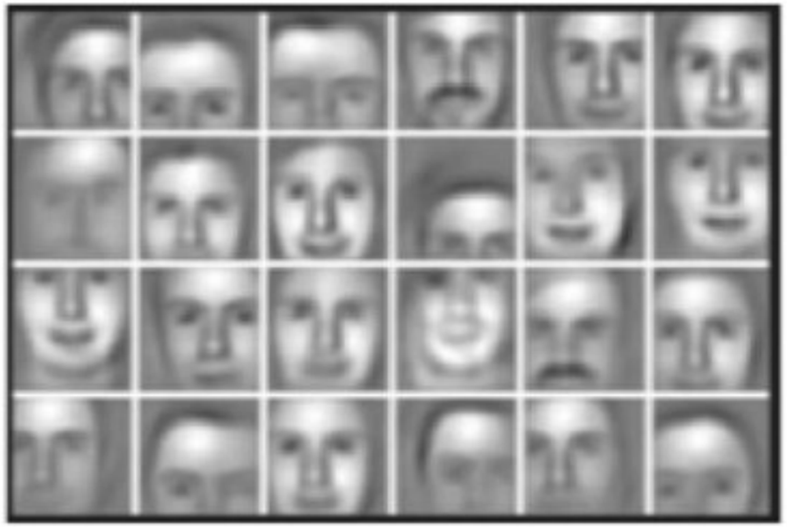
Lines & Edges

Mid Level Features



Eyes & Nose & Ears

High Level Features

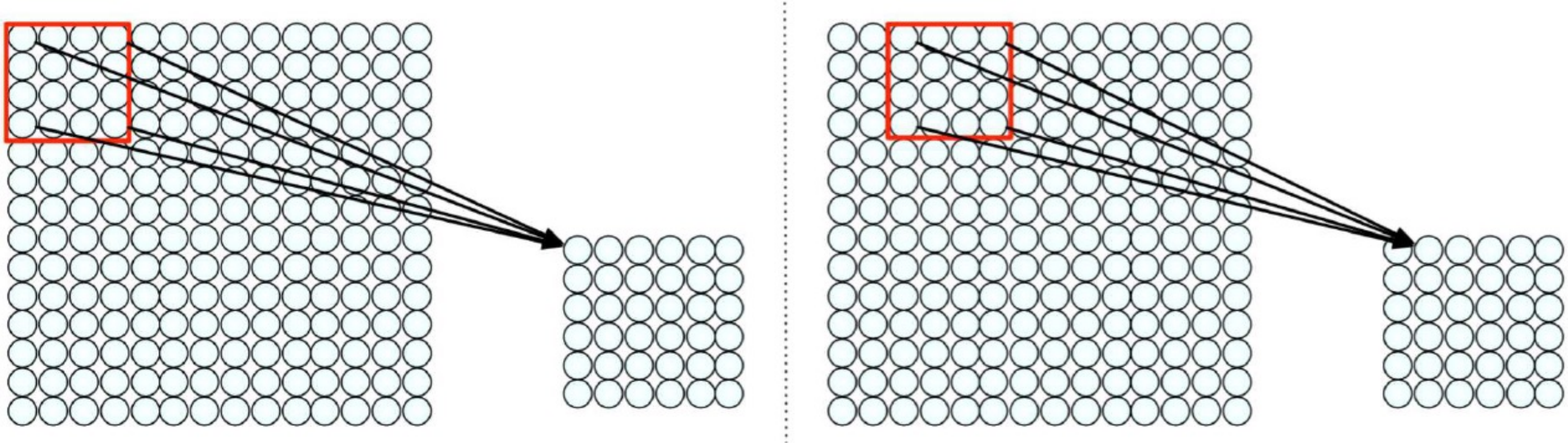


Facial Structure

These features were obtained in an automatic fashion!

Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009.

Idea: using the spatial structure (patches)

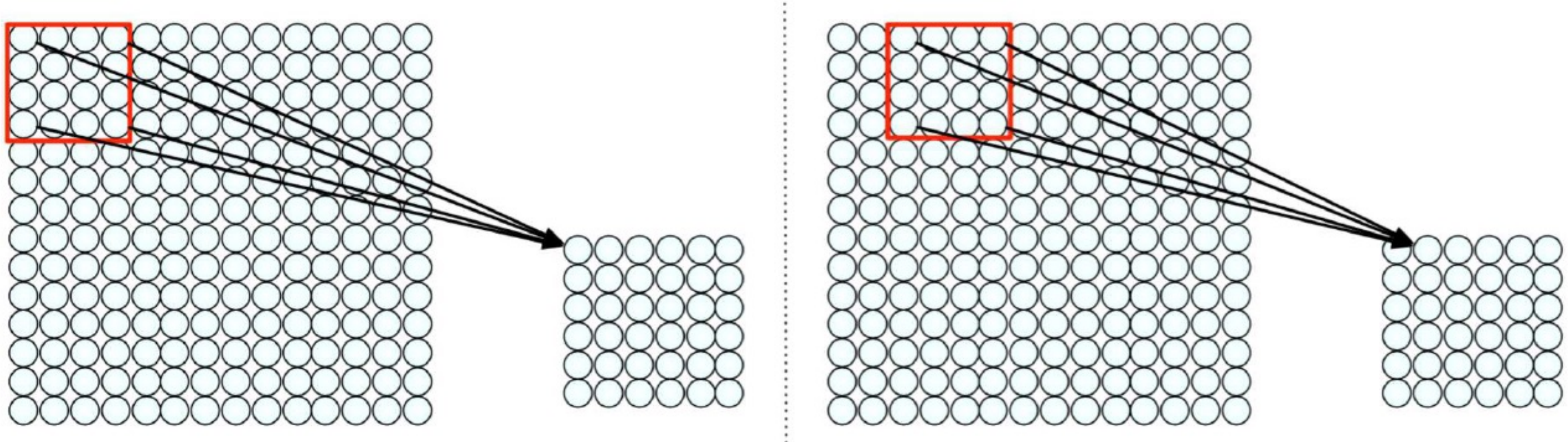


-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

- Connect patch in input layer to a single neuron in subsequent layer.

- Use a sliding window to define connections.

Idea: 2D convolutions



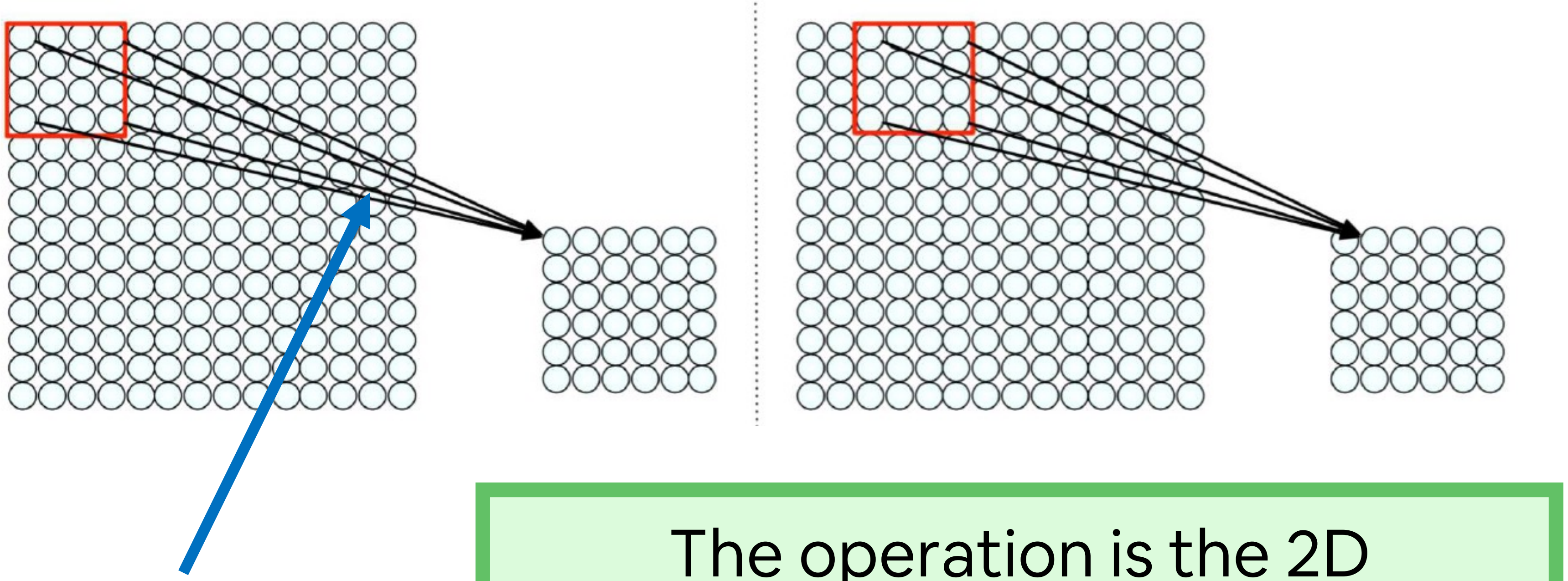
1	2	3
4	5	6
7	8	9

We would like to extract something useful out of this patch! Why not doing a ‘linear combination’ as we did with tabular data?

From https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionO

From http://cs231n.github.io/convolutional-networks/

Idea: 2D convolutions



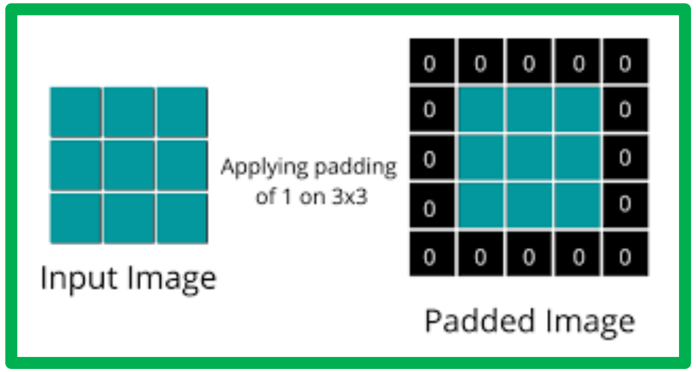
1	2	3
4	5	6
7	8	9

1	2	1
0	0	0
-1	-2	-1

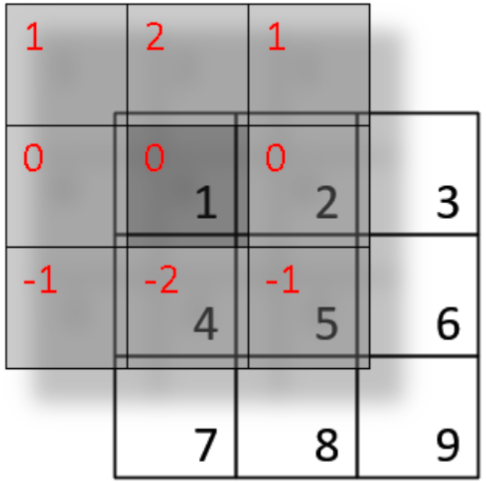
The operation is the 2D convolution: we multiply element-wise 2 matrices (the input and the kernel – weights), then we sum and derive some 'features'

From https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionO

From http://cs231n.github.io/convolutional-networks/



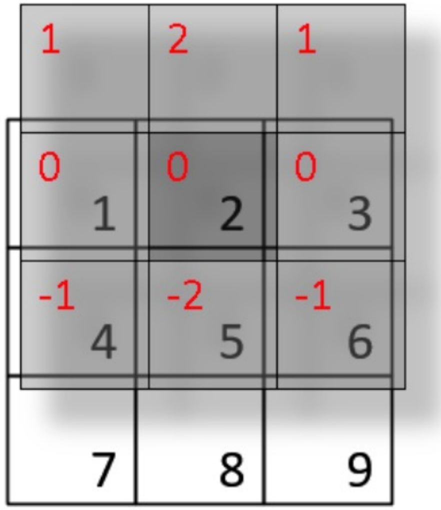
[Optional procedure] We include 'padding' the image (equal to 1): we add a frame around the image of pixels with value = 0. When performing the convolution operation, padding allow border pixels to have similar importance to inner pixels



$$
\begin{aligned}
y[0,0] &= \sum_j \sum_i x[i,j] \cdot h[0-i, 0-j] \\
&= \quad x[-1,-1] \cdot h[1,1] + x[0,-1] \cdot h[0,1] + x[1,-1] \cdot h[-1,1] \\
&\quad + x[-1,0] \cdot h[1,0] \quad + x[0,0] \cdot h[0,0] \quad + x[1,0] \cdot h[-1,0] \\
&\quad + x[-1,1] \cdot h[1,-1] + x[0,1] \cdot h[0,-1] + x[1,1] \cdot h[-1,-1] \\
&= \quad 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 \\
&\quad + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 \\
&\quad + 0 \cdot (-1) + 4 \cdot (-2) + 5 \cdot (-1) \\
&= -13
\end{aligned}
$$

From https://www.songho.ca/dsp/convolution/convolution2d_example.html

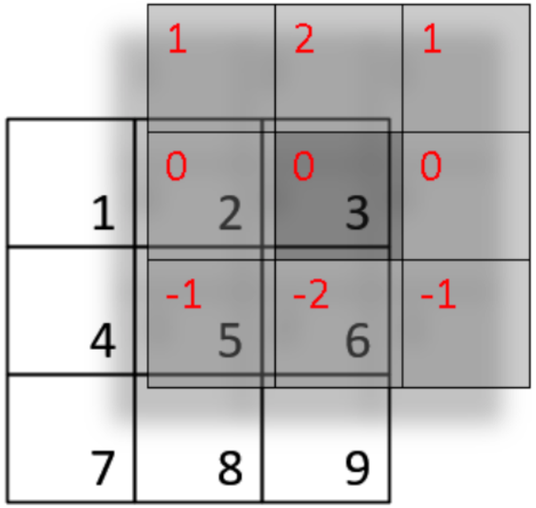
We then ‘stride’ (equal to 1): we move the kernel (ie. filter, ie. weights) horizontally to a different patch



$$
\begin{aligned}
y[1,0] &= \sum_j \sum_i x[i,j] \cdot h[1-i, 0-j] \\
&= \quad x[0,-1] \cdot h[1,1] + x[1,-1] \cdot h[0,1] + x[2,-1] \cdot h[-1,1] \\
&\quad + x[0,0] \cdot h[1,0] \quad + x[1,0] \cdot h[0,0] \quad + x[2,0] \cdot h[-1,0] \\
&\quad + x[0,1] \cdot h[1,-1] + x[1,1] \cdot h[0,-1] + x[2,1] \cdot h[-1,-1] \\
&= \quad 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 \\
&\quad + 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 \\
&\quad + 4 \cdot (-1) + 5 \cdot (-2) + 6 \cdot (-1) \\
&= -20
\end{aligned}
$$

From https://www.songho.ca/dsp/convolution/convolution2d_example.html

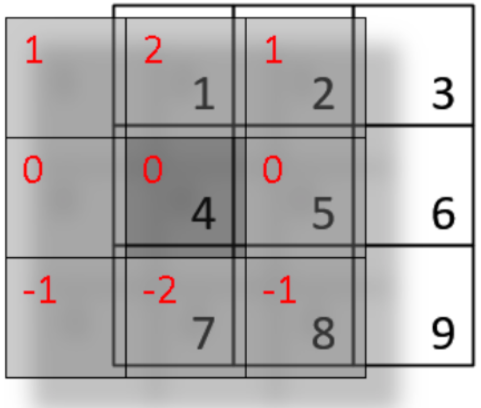
We then 'stride' (equal to 1): we move the kernel (ie. filter, ie. weights) horizontally to a different patch... until we arrive to the end of the row!



$$
\begin{aligned}
y[2,0] &= \sum_j \sum_i x[i,j] \cdot h[2-i, 0-j] \\
&= \quad x[1,-1] \cdot h[1,1] + x[2,-1] \cdot h[0,1] + x[3,-1] \cdot h[-1,1] \\
&\quad + x[1,0] \cdot h[1,0] \quad + x[2,0] \cdot h[0,0] \quad + x[3,0] \cdot h[-1,0] \\
&\quad + x[1,1] \cdot h[1,-1] + x[2,1] \cdot h[0,-1] + x[3,1] \cdot h[-1,-1] \\
&= \quad 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 \\
&\quad + 2 \cdot 0 + 3 \cdot 0 + 0 \cdot 0 \\
&\quad + 5 \cdot (-1) + 6 \cdot (-2) + 0 \cdot (-1) \\
&= -17
\end{aligned}
$$

From https://www.songho.ca/dsp/convolution/convolution2d_example.html

We then ’stride’ vertically!



$$
\begin{aligned}
y[0,1] &= \sum_j \sum_i x[i,j] \cdot h[0-i, 1-j] \\
&= \quad x[-1,0] \cdot h[1,1] \quad + x[0,0] \cdot h[0,1] \quad + x[1,0] \cdot h[-1,1] \\
&\quad + x[-1,1] \cdot h[1,0] \quad + x[0,1] \cdot h[0,0] \quad + x[1,1] \cdot h[-1,0] \\
&\quad + x[-1,2] \cdot h[1,-1] + x[0,2] \cdot h[0,-1] + x[1,2] \cdot h[-1,-1] \\
&= \quad 0 \cdot 1 + 1 \cdot 2 + 2 \cdot 1 \\
&\quad + 0 \cdot 0 + 4 \cdot 0 + 5 \cdot 0 \\
&\quad + 0 \cdot (-1) + 7 \cdot (-2) + 8 \cdot (-1) \\
&= -18
\end{aligned}
$$

From https://www.songho.ca/dsp/convolution/convolution2d_example.html

After completing the stride:

1	2	3
4	5	6
7	8	9

Input

n m	-1	0	1
-1	-1	-2	-1
0	0	0	0
1	1	2	1

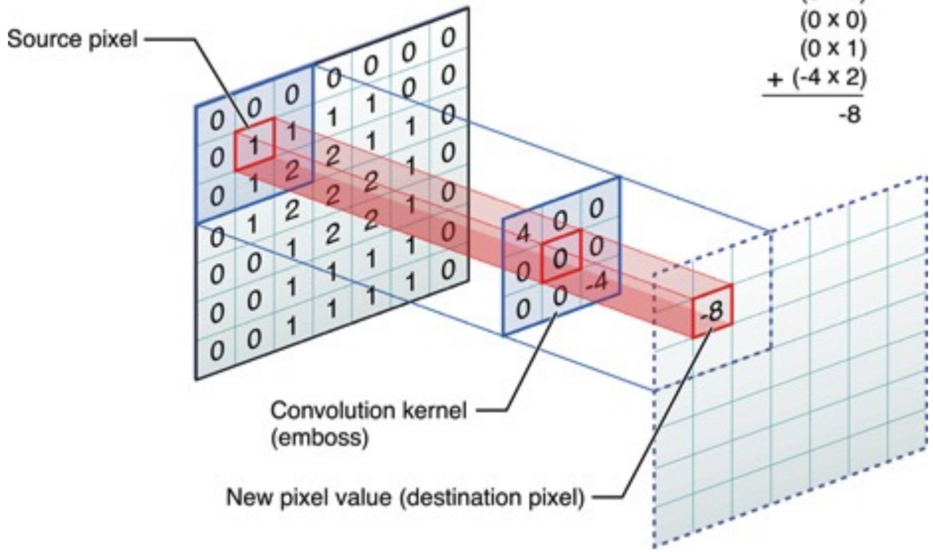
Kernel

-13	-20	-17
-18	-24	-18
13	20	17

Output

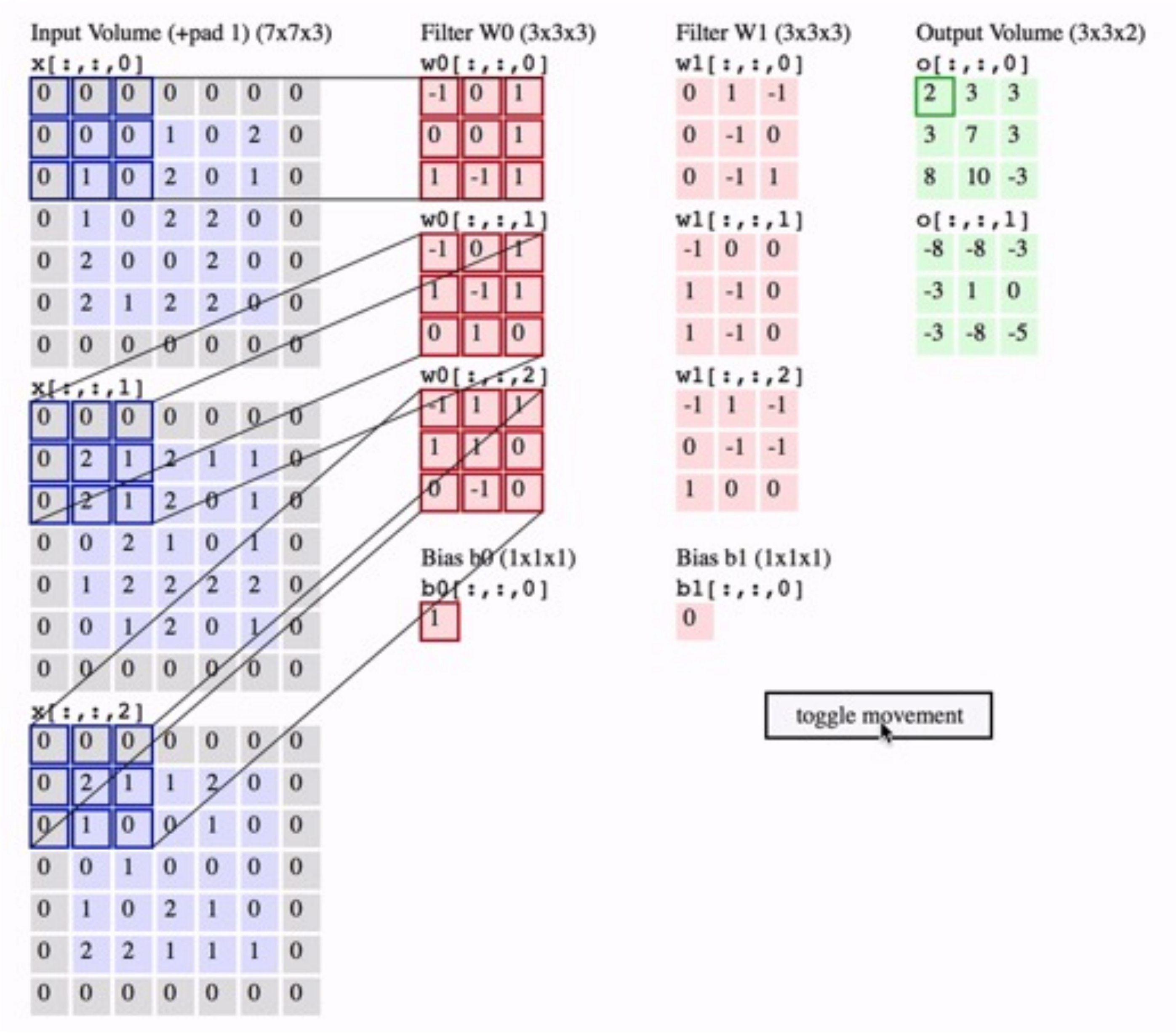
From https://www.songho.ca/dsp/convolution/convolution2d_example.html

Idea: 2D convolutions



Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

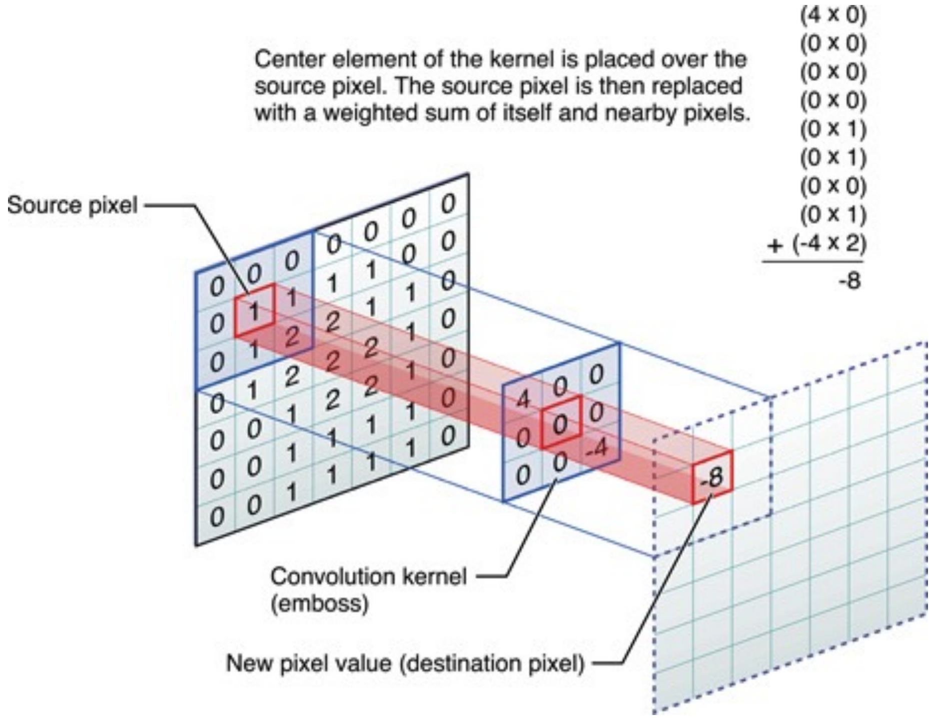
o[:,:,1]

-8	-8	-3
-3	1	0
-3	-8	-5

From https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionO

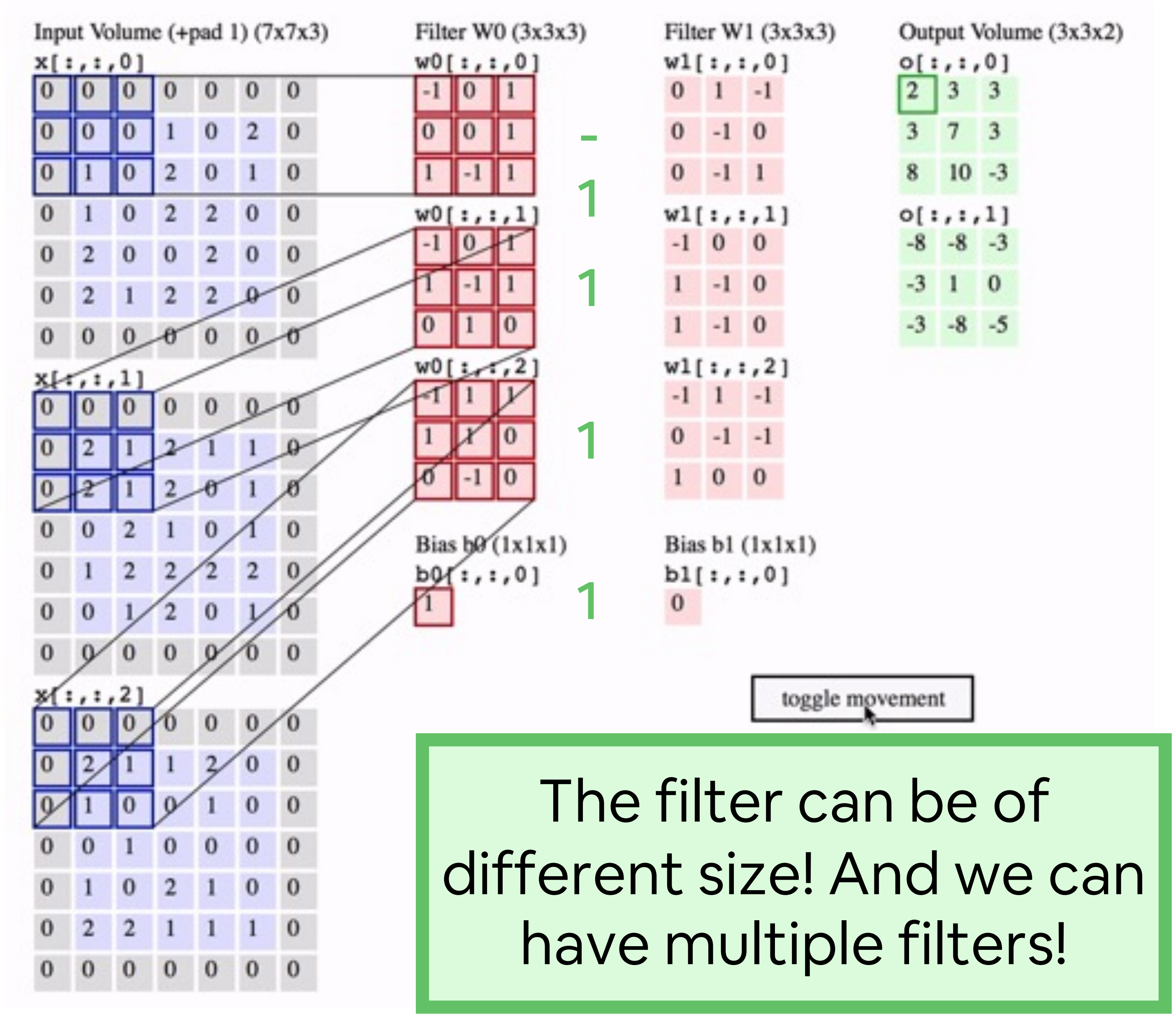
From http://cs231n.github.io/convolutional-networks/

Idea: 2D convolutions


Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8

Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)


Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

o[:,:,1]

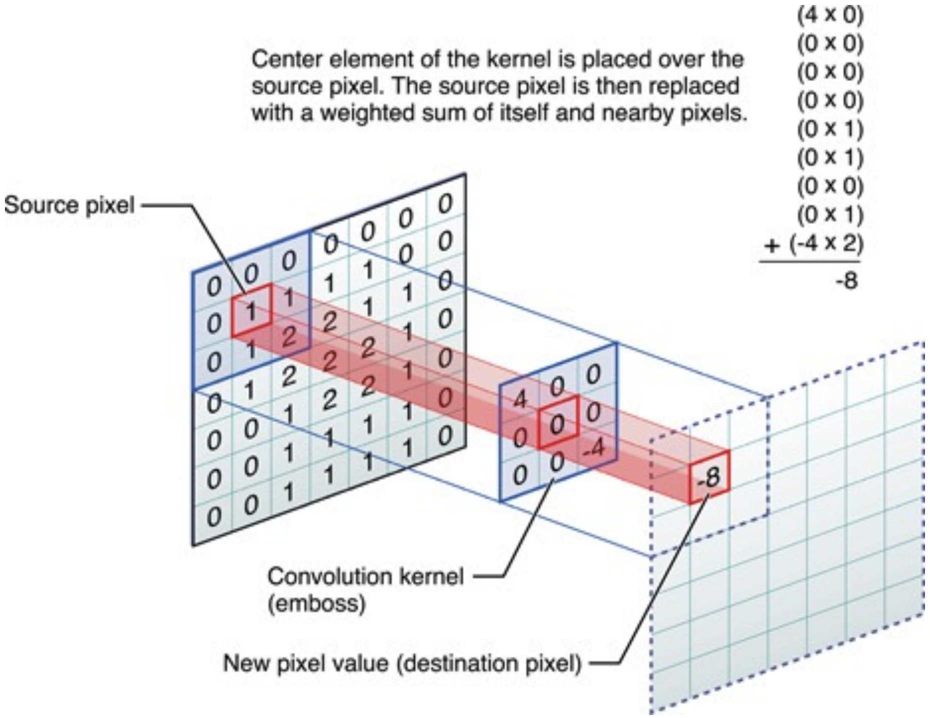
-8	-8	-3
-3	1	0
-3	-8	-5

The filter can be of different size! And we can have multiple filters!

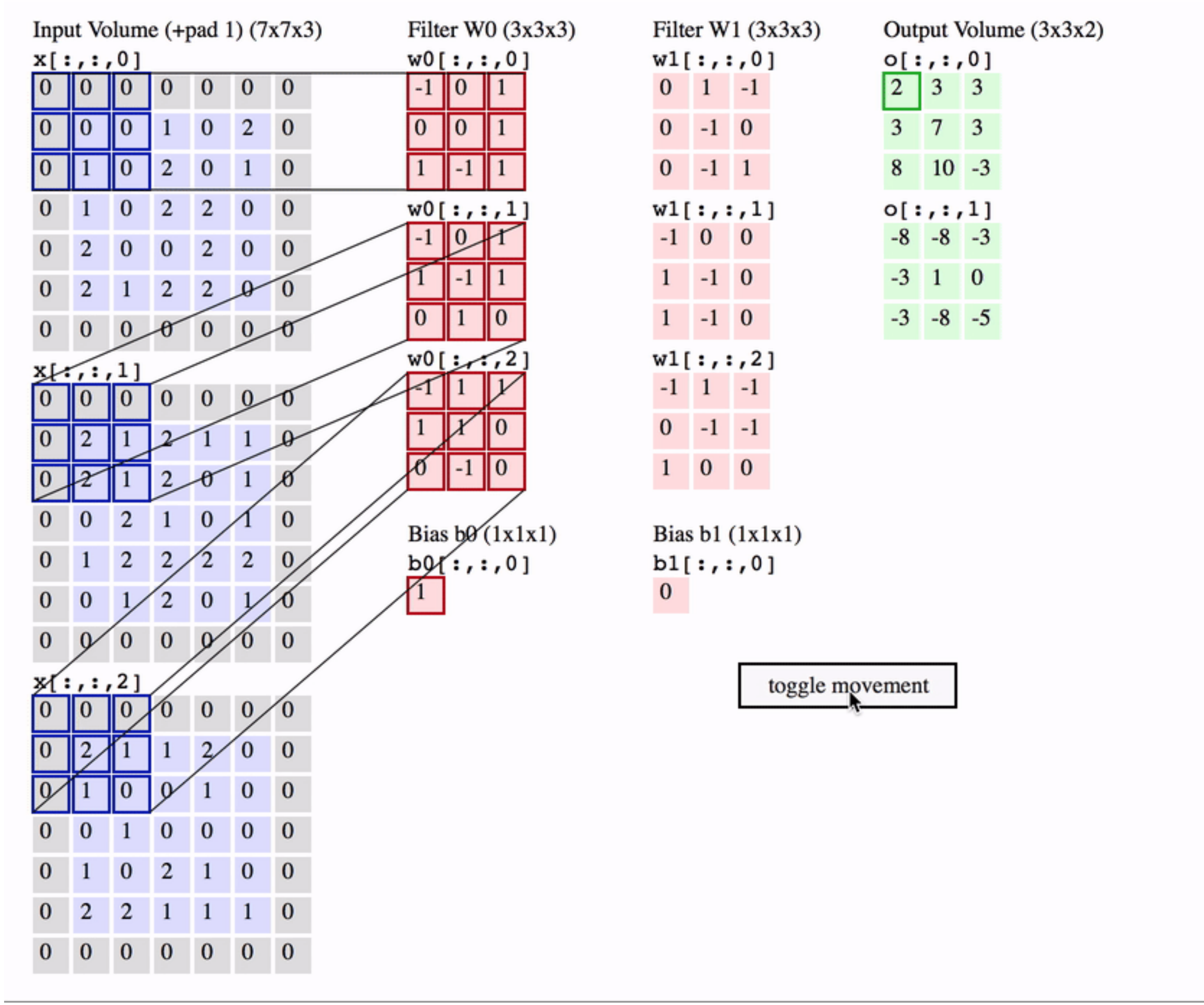
From https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionO

From http://cs231n.github.io/convolutional-networks/

Idea: 2D convolutions



Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

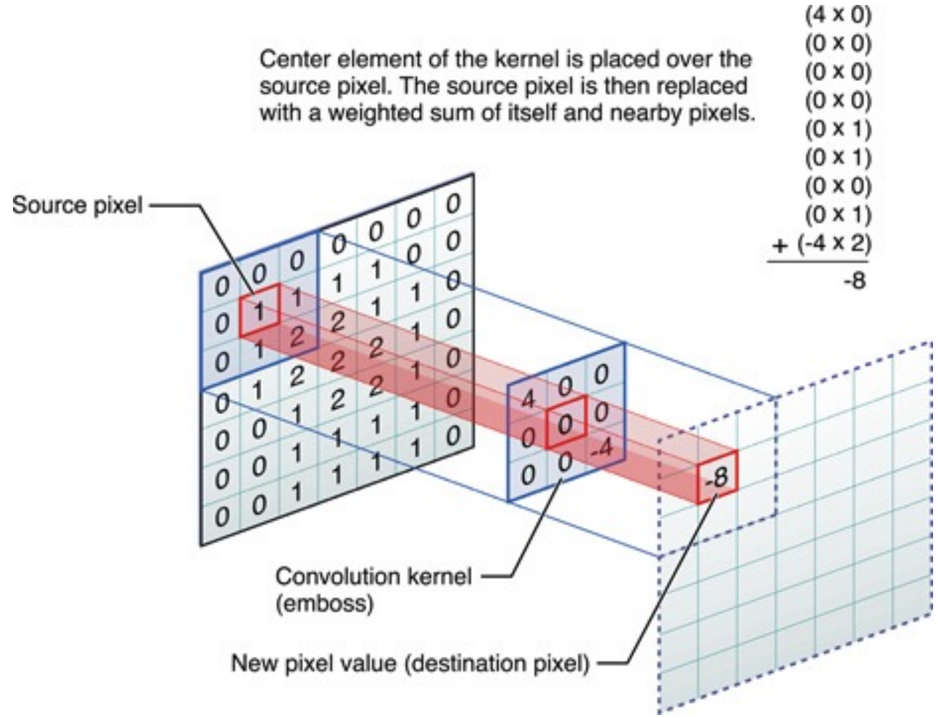
o[:,:,1]

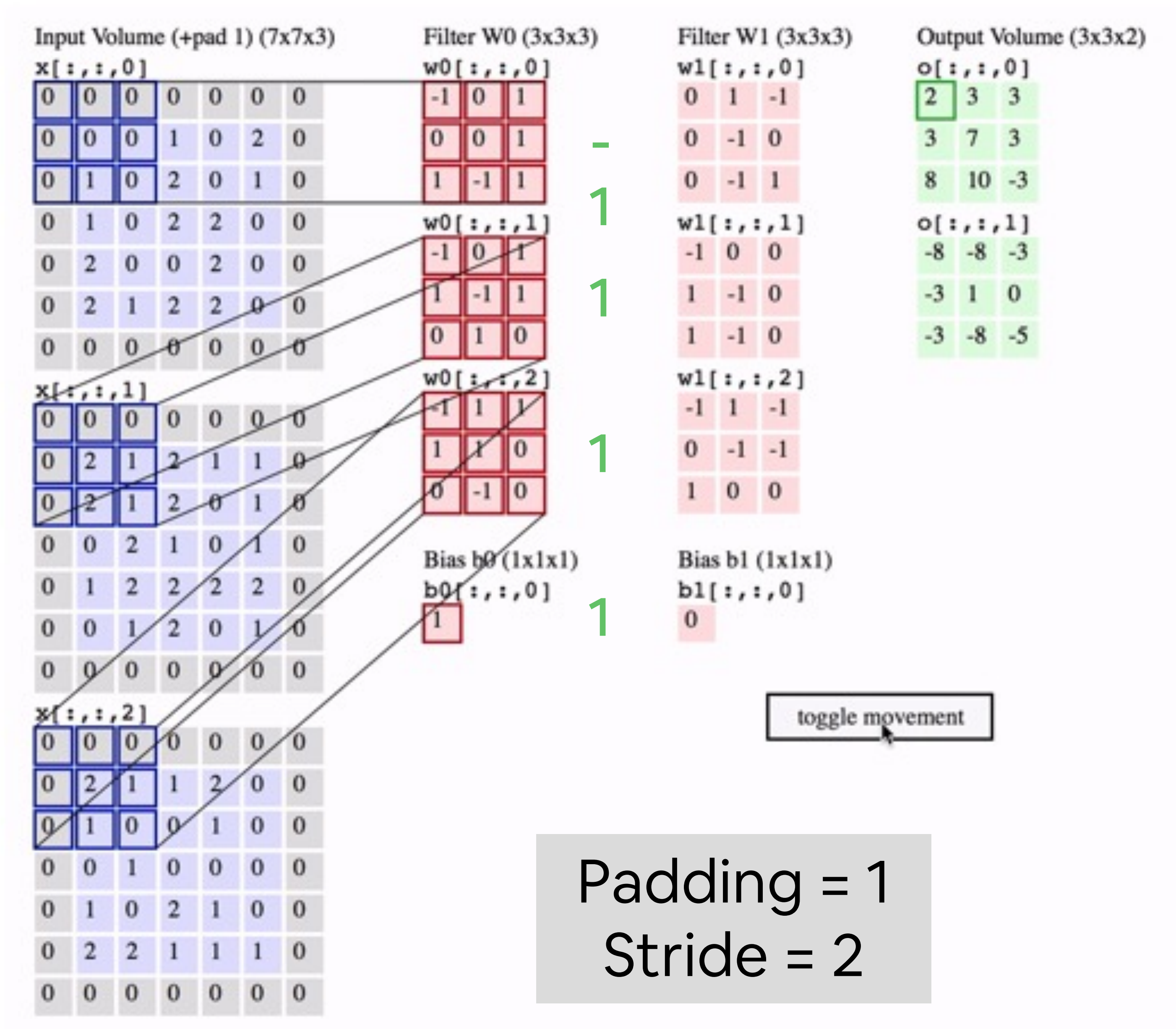
-8	-8	-3
-3	1	0
-3	-8	-5

From https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionO

From http://cs231n.github.io/convolutional-networks/

Idea: 2D convolutions


Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

o[:,:,1]

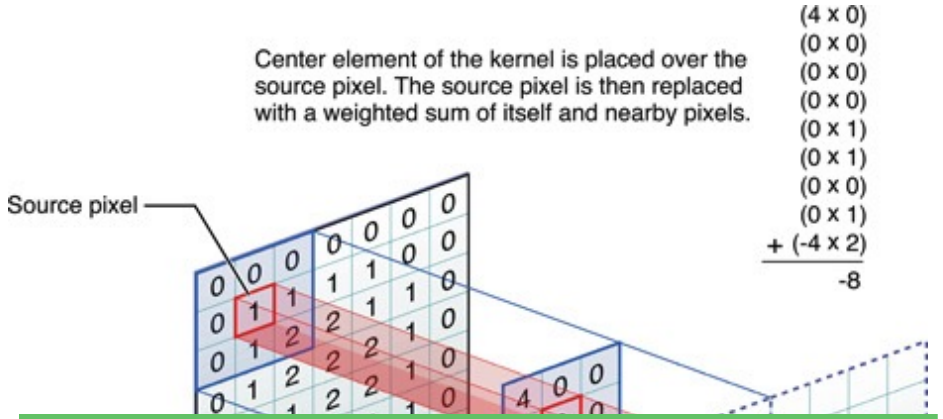
-8	-8	-3
-3	1	0
-3	-8	-5

Padding = 1
Stride = 2

From https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionO

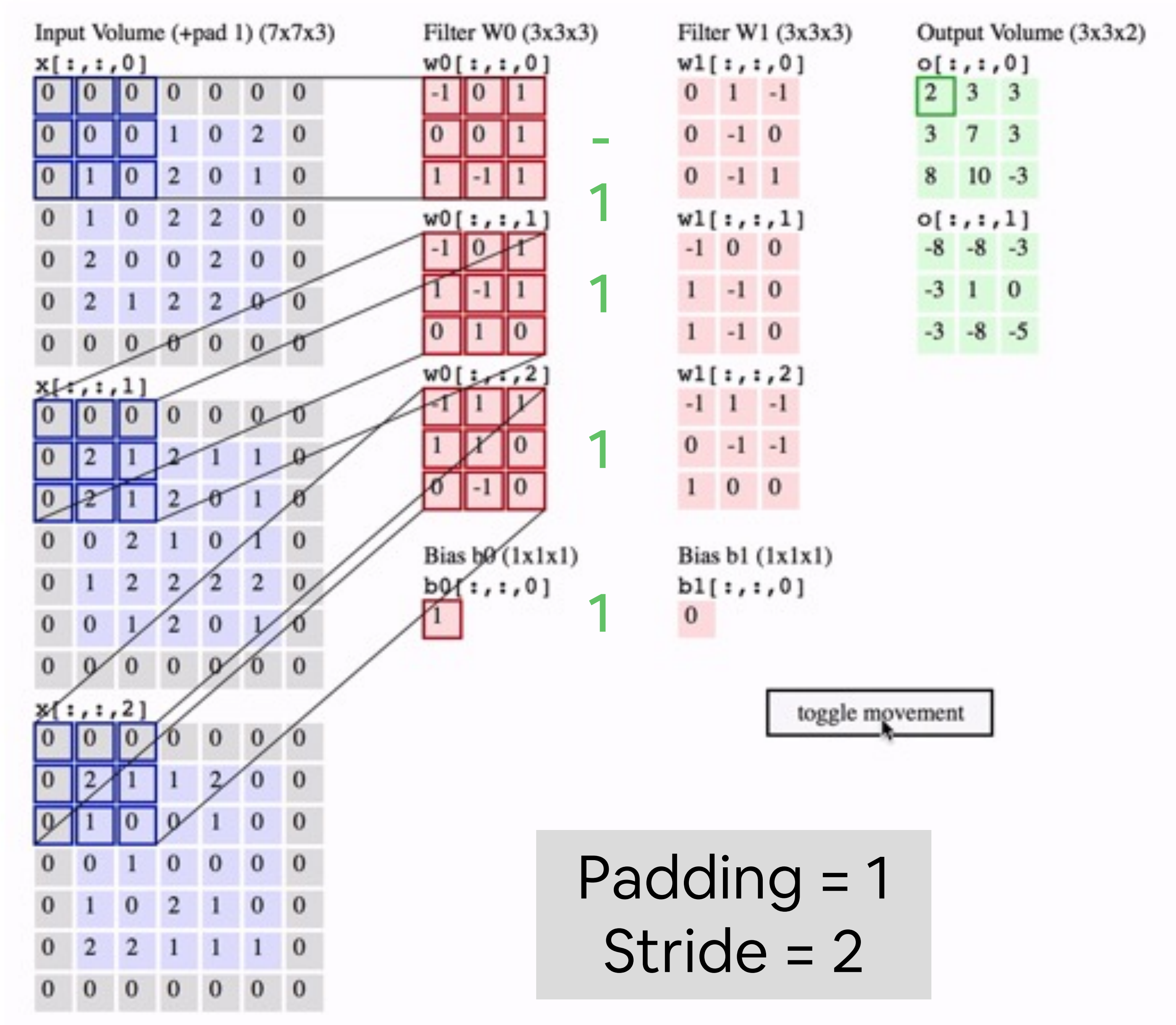
From http://cs231n.github.io/convolutional-networks/

Idea: 2D convolutions



Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

With padding we give 'right' importance to pixels in the border, while with stride we can tune the inner dimensions of the network



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

o[:,:,1]

-8	-8	-3
-3	1	0
-3	-8	-5

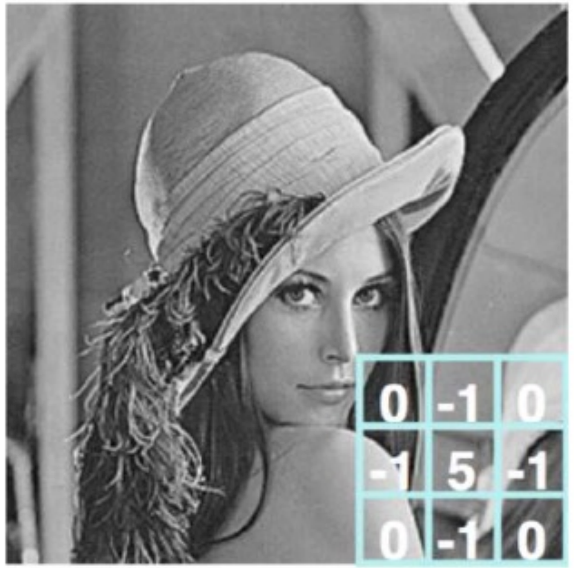
Padding = 1
Stride = 2

From https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionO

From http://cs231n.github.io/convolutional-networks/

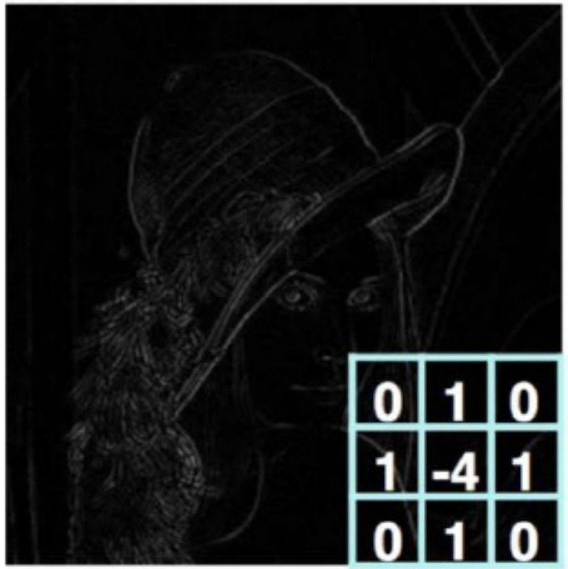
Feature maps induced by convolutions: some examples

Original



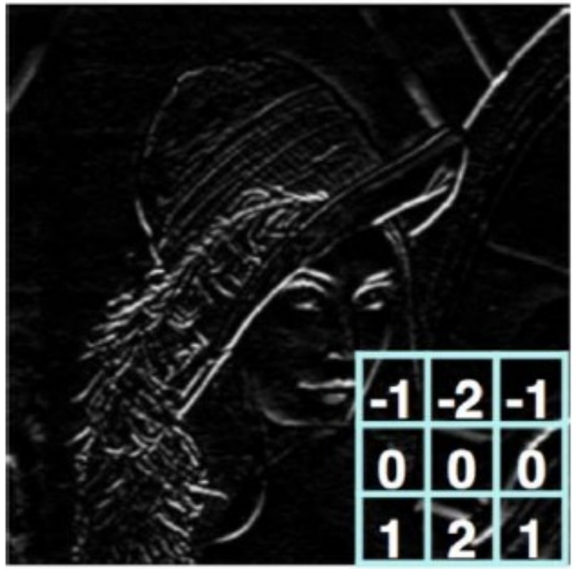
$$\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix}$$

Sharpen



$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Edge Detect



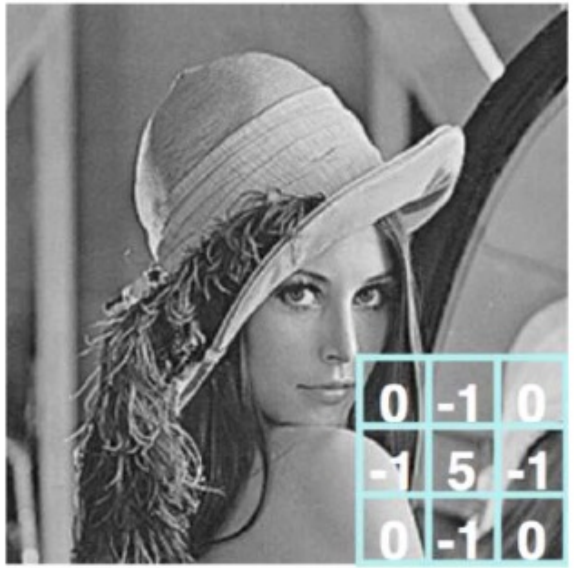
$$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

"Strong" Edge Detect

-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

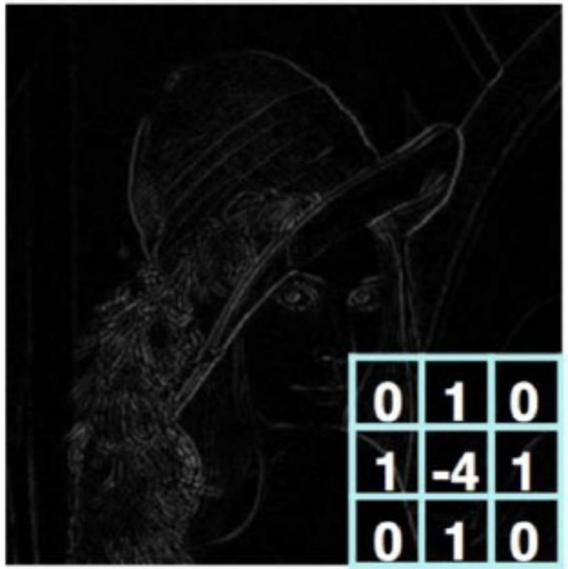
Feature maps induced by convolutions: some examples

Original



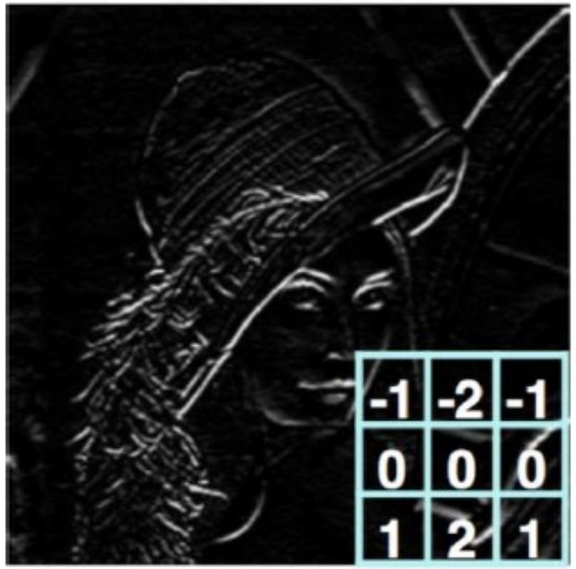
0	-1	0
-1	5	-1
0	-1	0

Sharpen



0	1	0
1	-4	1
0	1	0

Edge Detect



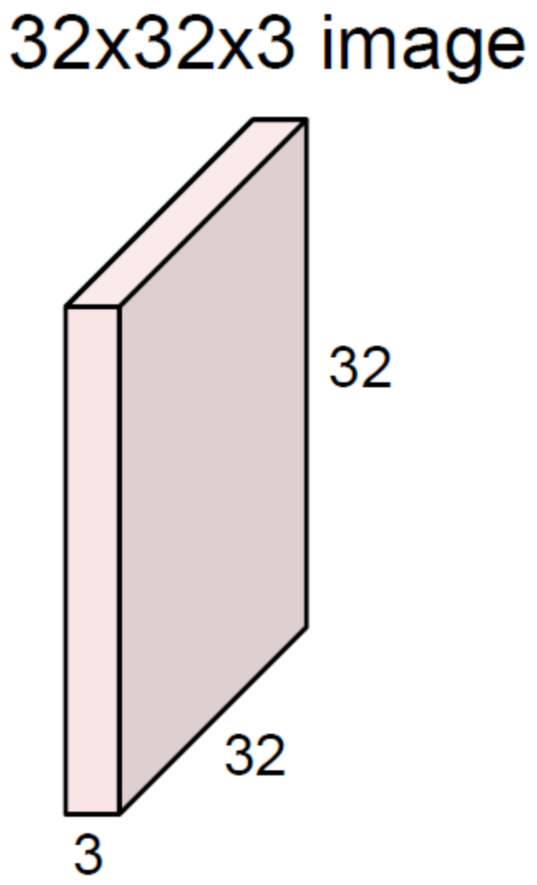
-1	-2	-1
0	0	0
1	2	1

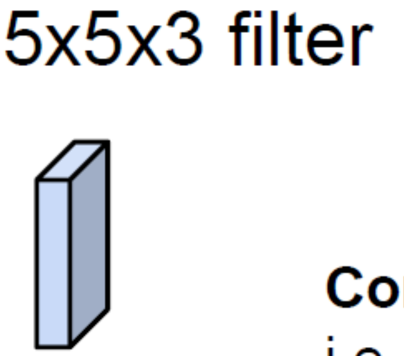
"Strong" Edge Detect

-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

?	?	?
?	?	?
?	?	?

Recap

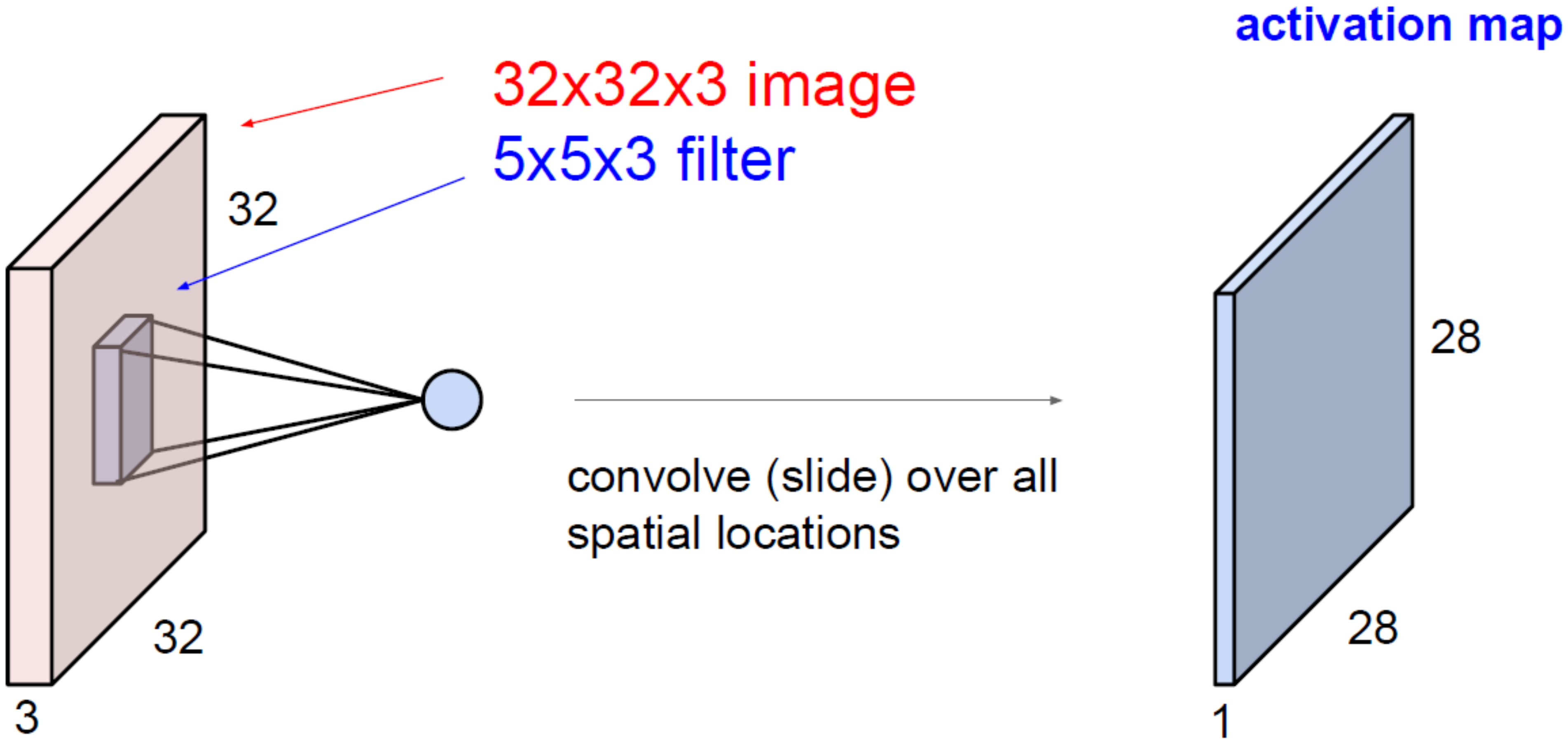




Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

F. Li, J. Johnson, S. Yeung. *Stanford CS231n Convolutional Neural Networks for Visual Recognition* http://cs231n.stanford.edu/

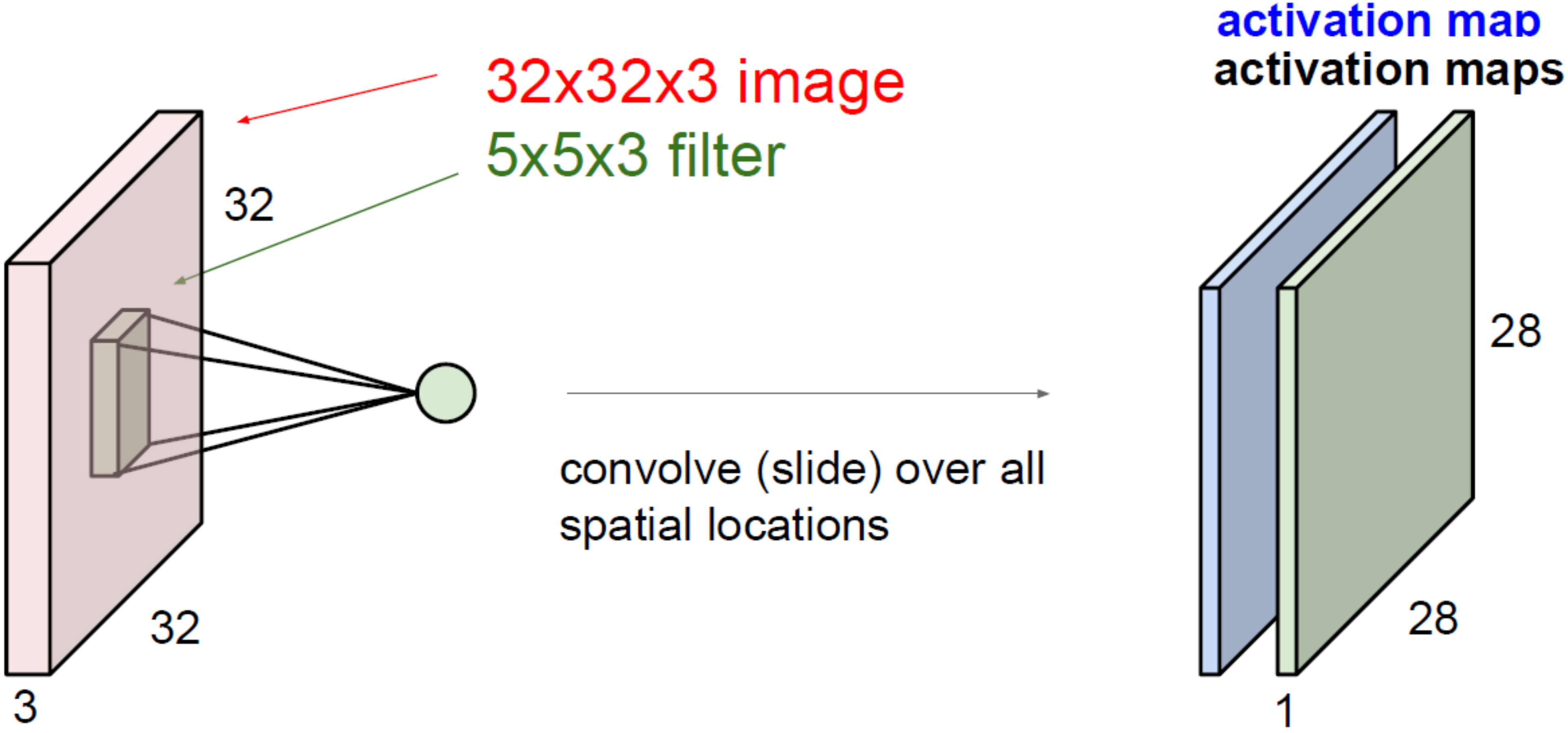
Recap



stride = 1,
no padding

F. Li, J. Johnson, S. Yeung. *Stanford CS231n Convolutional Neural Networks for Visual Recognition* http://cs231n.stanford.edu/

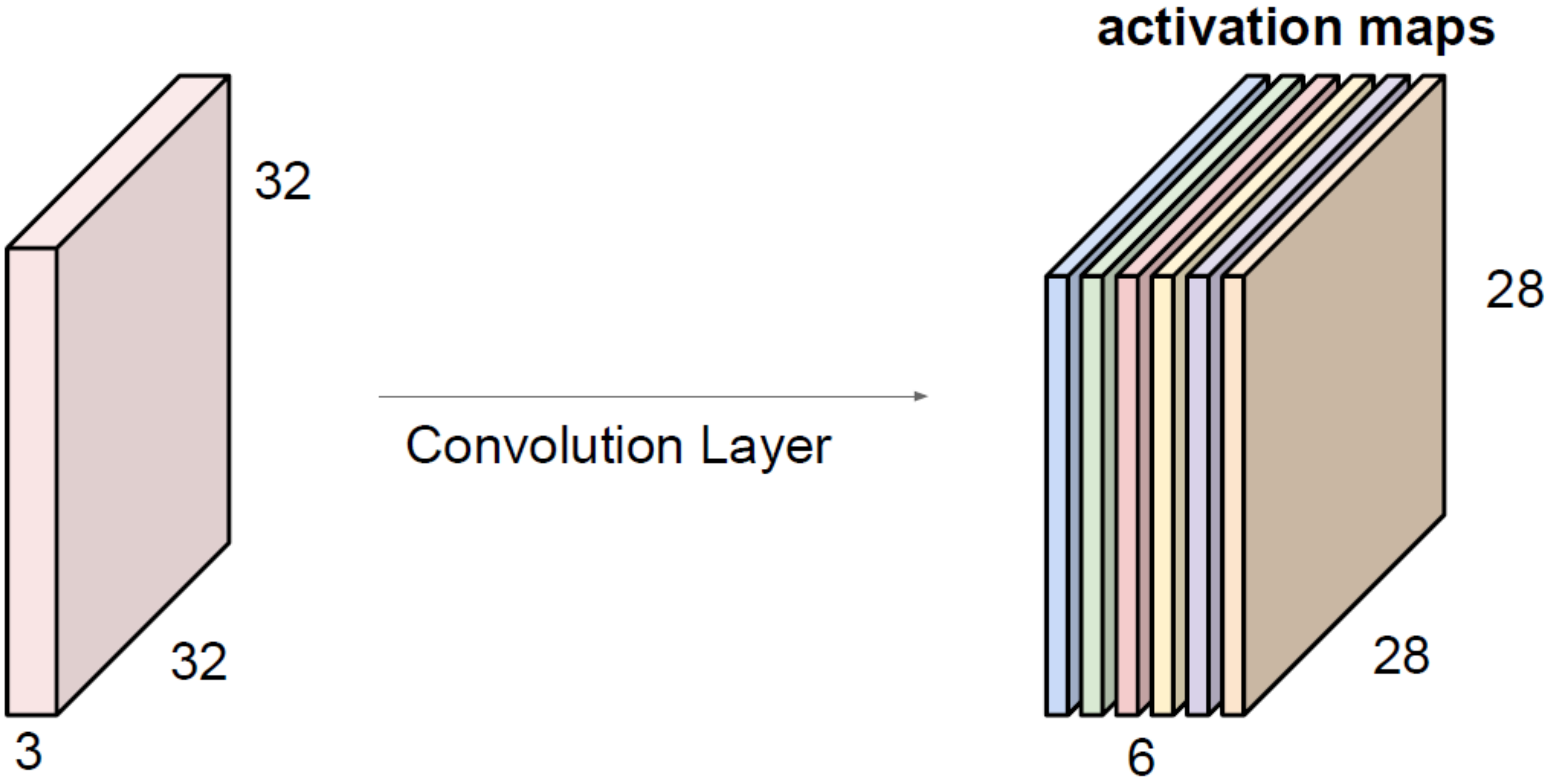
Recap



stride = 1,
no padding

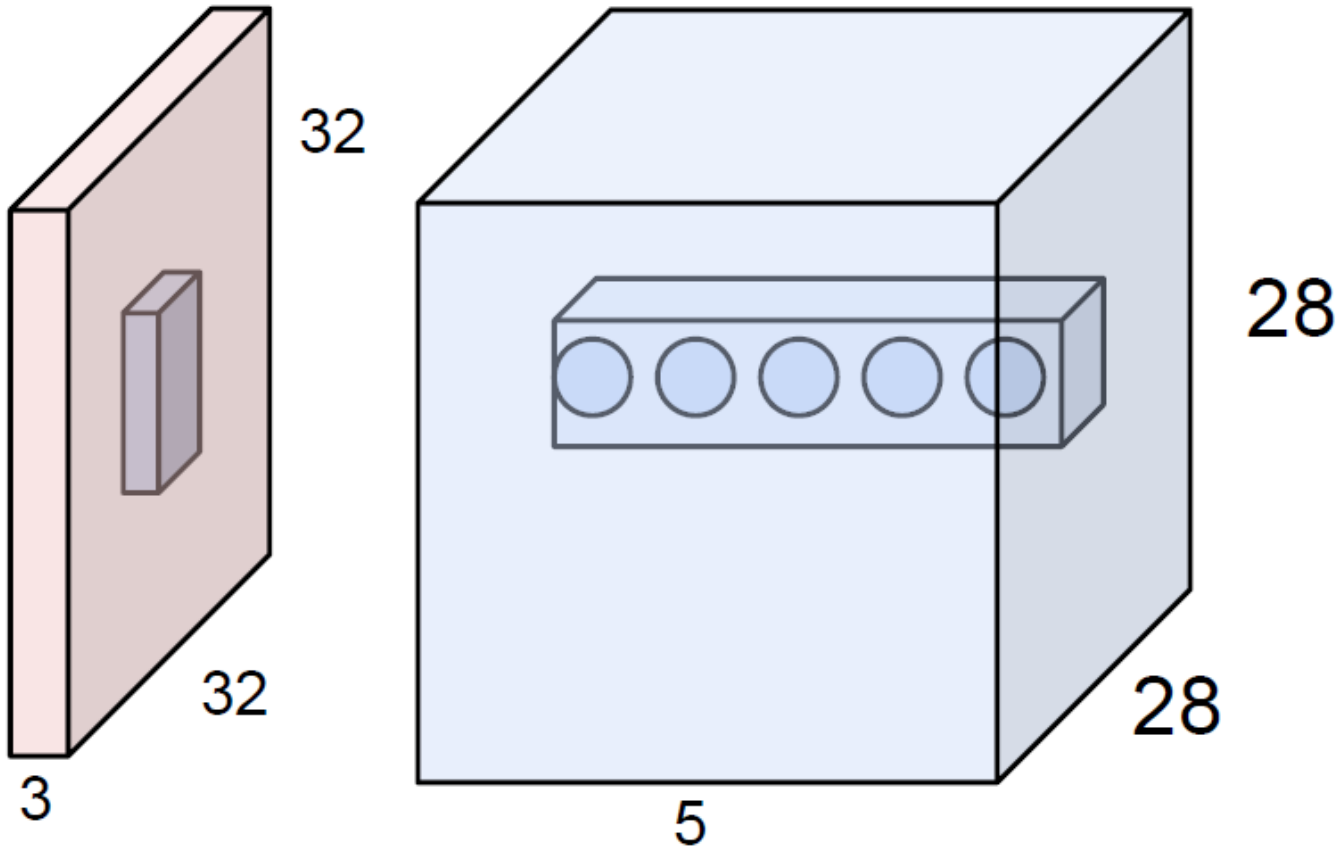
F. Li, J. Johnson, S. Yeung. *Stanford CS231n Convolutional Neural Networks for Visual Recognition* http://cs231n.stanford.edu/

Recap



F. Li, J. Johnson, S. Yeung. *Stanford CS231n Convolutional Neural Networks for Visual Recognition* http://cs231n.stanford.edu/

Receptive field



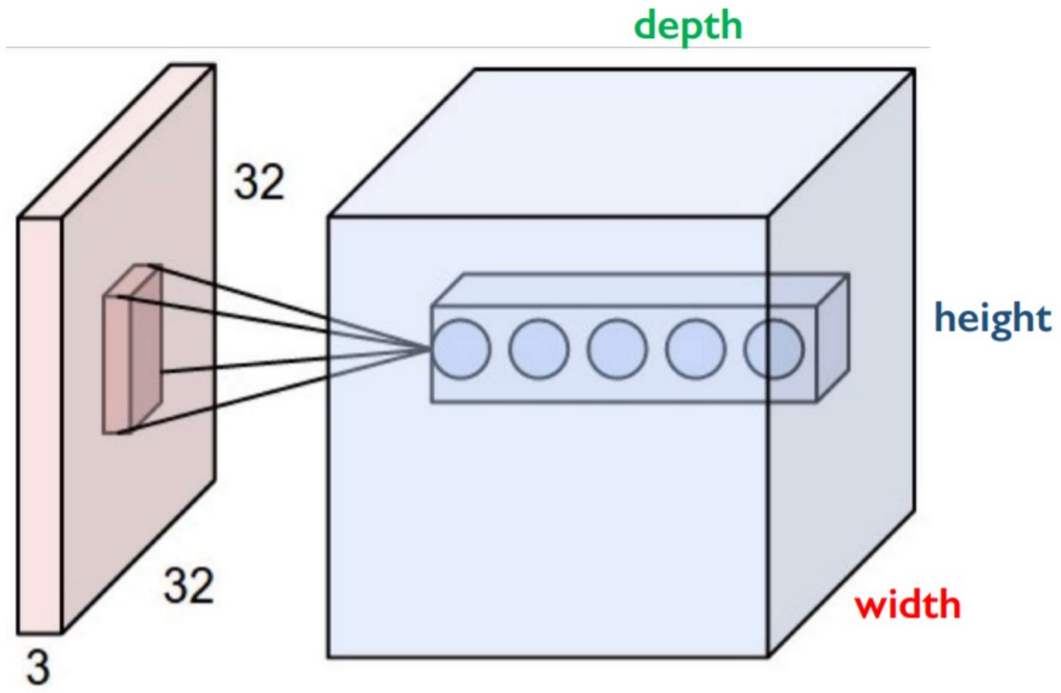
E.g. with 5 filters,
CONV layer consists of
neurons arranged in a 3D grid
(28x28x5)

There will be 5 different
neurons all looking at the same
region in the input volume

F. Li, J. Johnson, S. Yeung. *Stanford CS231n Convolutional Neural Networks for Visual Recognition* http://cs231n.stanford.edu/

Pros of 2D convolutions

- Use convolution filters (weights) to extract **local features**
- Use multiple filters to extract **different features**
- Filter parameters are **shared spatially** across the whole input



Layer Dimensions:

$h\ x\ w\ x\ d$

where h and w are spatial dimensions
d (depth) = number of filters

Stride:

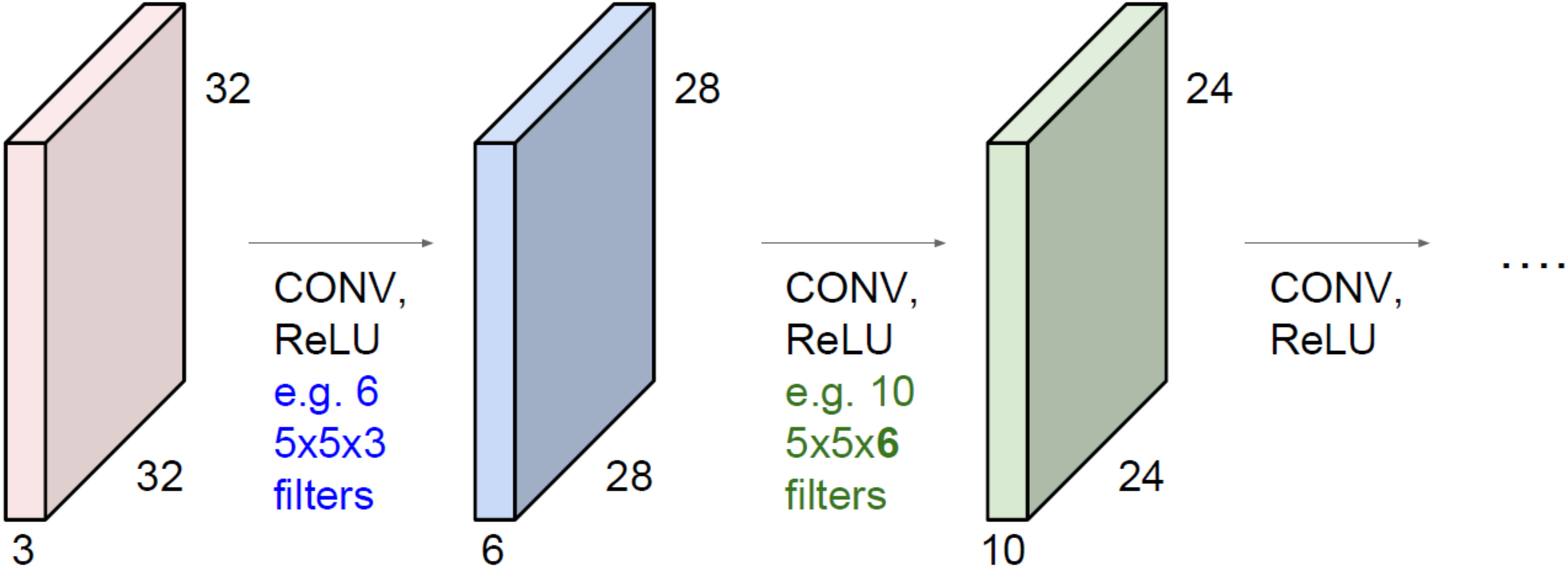
Filter step size

Receptive Field:

Locations in input image that
a node is path connected to

-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

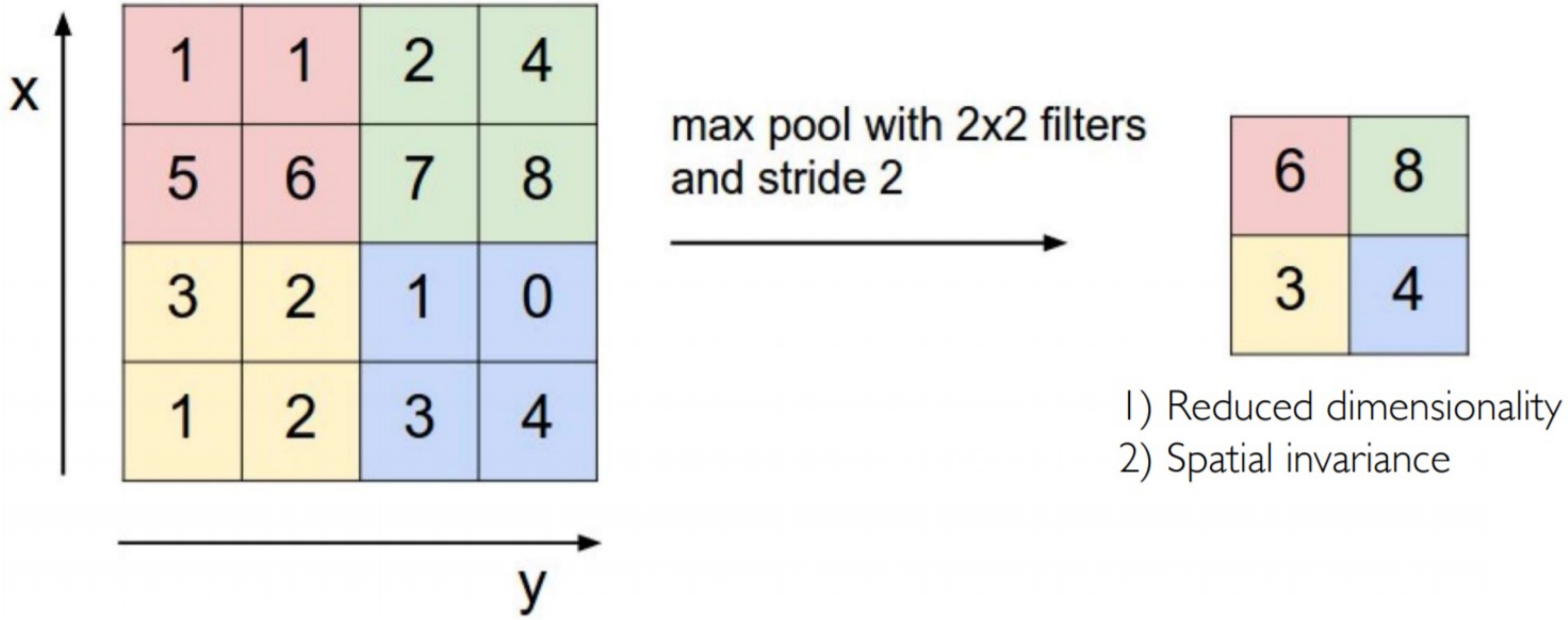
Deep Learning: we stack different layers!



F. Li, J. Johnson, S. Yeung. *Stanford CS231n Convolutional Neural Networks for Visual Recognition* http://cs231n.stanford.edu/

Make extracted features rich, robust and low-dimensional

- Introduce non-linearity: apply non-linear activation to convolution output
- Downsample and preserve spatial invariance: pooling



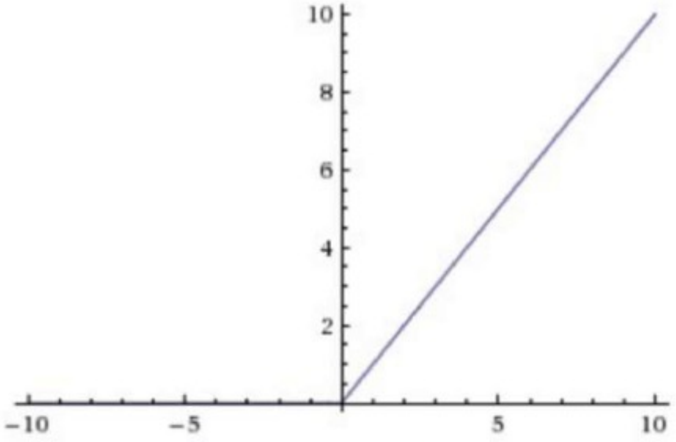
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2 filters and stride 2

6	8
3	4

1) Reduced dimensionality
2) Spatial invariance

Rectified Linear Unit (ReLU)



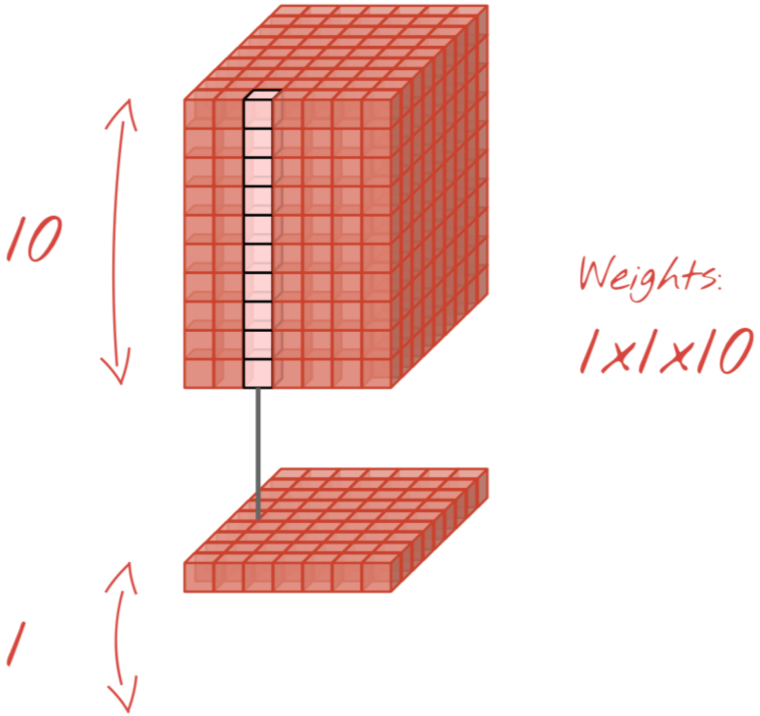
$g(z) = \max(0, z)$

-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

Dimensionality reduction (optional)

- While to reduce width and height we can exploit pooling (or bigger strides), to reduce depth we can use 1d convolutions

1d convolutions



Why 1x1 convolutions?

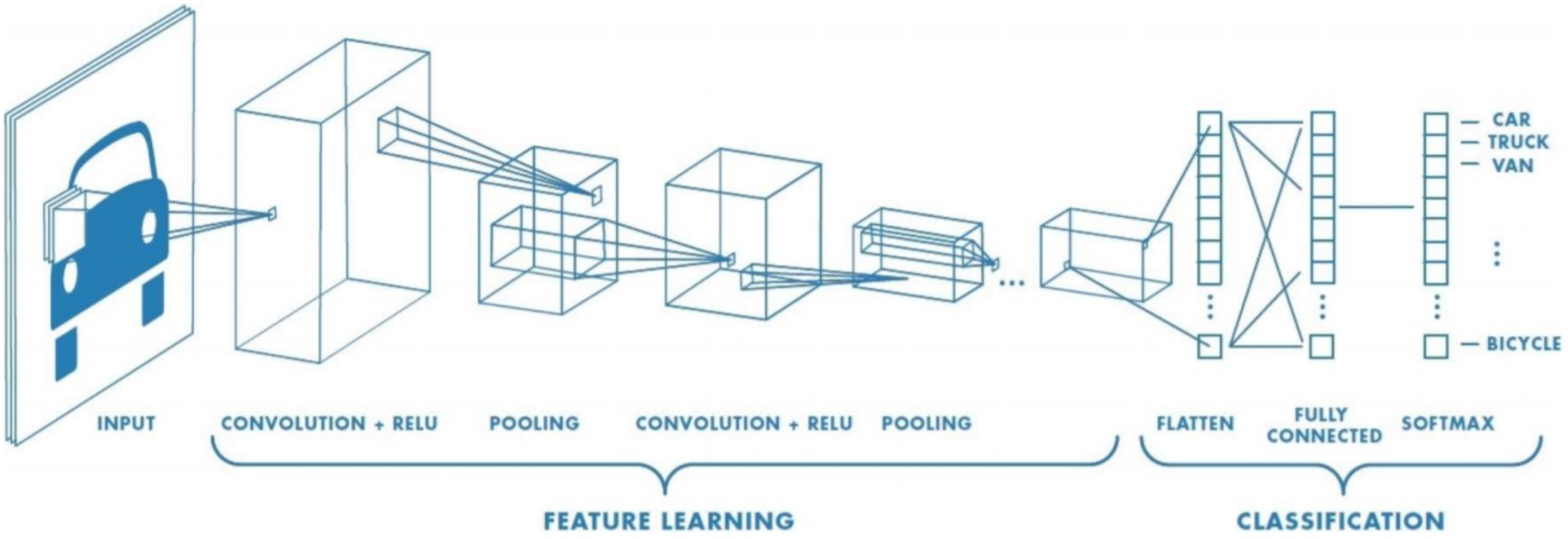
- ***Efficiency***: reduces the depth (number of channels). Width and height are unchanged. To reduce the horizontal dimensions, you would use pooling (or increase the stride of the conv).
- The 1x1 conv computes a weighted sum of input channels (or features). This allows it to "***select***" certain combinations of features that are useful downstream.

-https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd

Now put it all together to perform classification

- We apply back-propagation to train model (weights for convolution and dense layers)
- Cross entropy loss

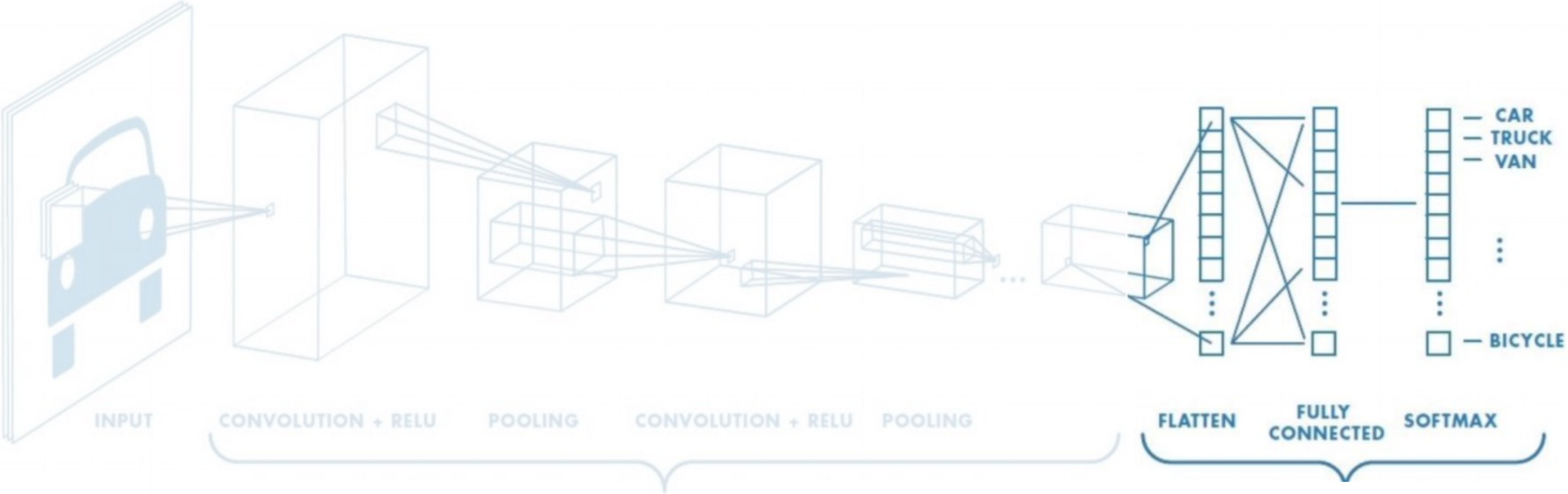
$$J(\boldsymbol{\theta}) = \sum_i y^{(i)} \log(\hat{\boldsymbol{y}}^{(i)})$$



-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

Now put it all together to perform classification

- Extract high-level features from last conv and pool layer
- Add fully connected layers to learn complex transformation of the features
- Output expressed as probability distribution over classes



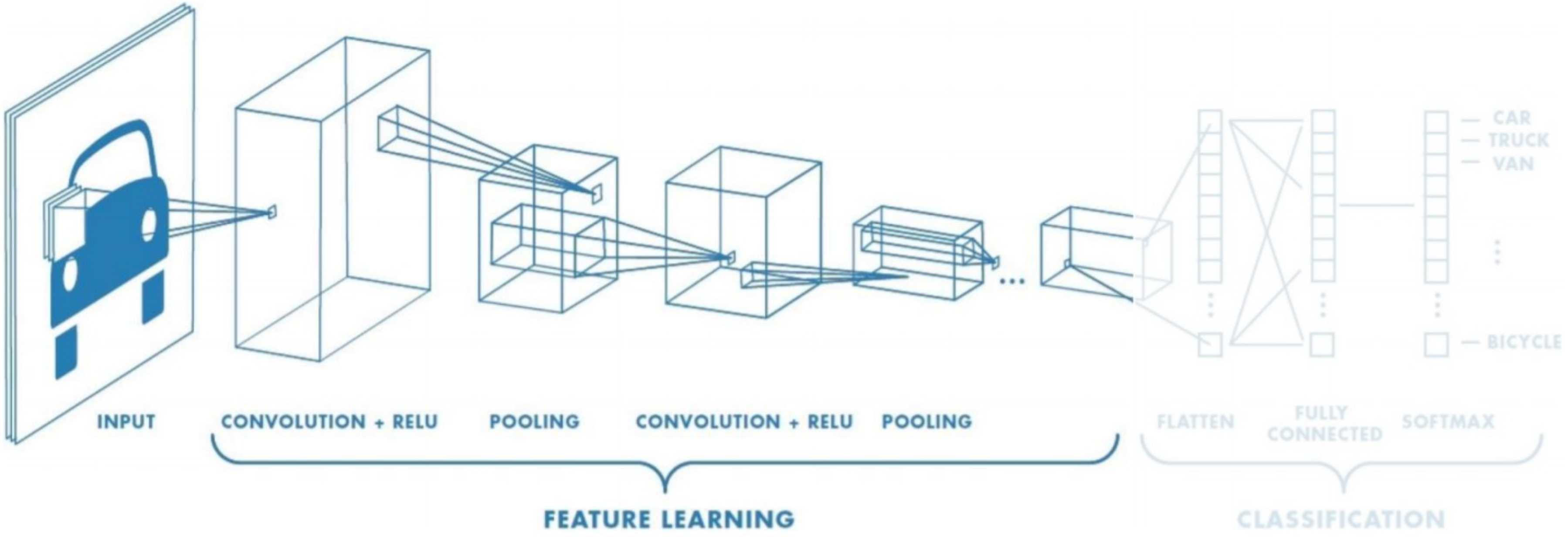
We'll see softmax on next lecture

$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

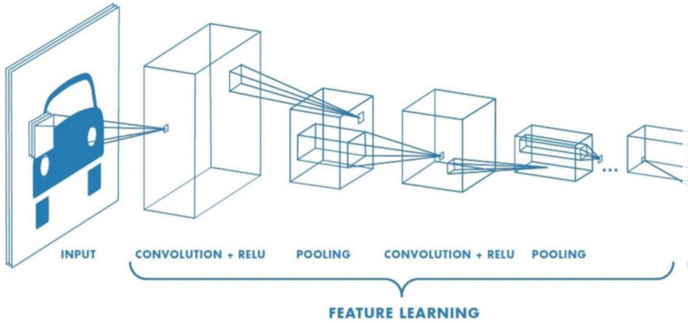
Now put it all together to perform classification

- Convolution to learn local features from input image
- Introduce non-linear activation
- Reduce dimensionality and preserve local translation invariance with pooling



-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

One feature extraction tool, many tasks!



You can use feature learning part to perform a variety of tasks!

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
LESIONS LEARNT
Artificial intelligence powers detection of skin cancer from images
person
person
person
skateboard
dog

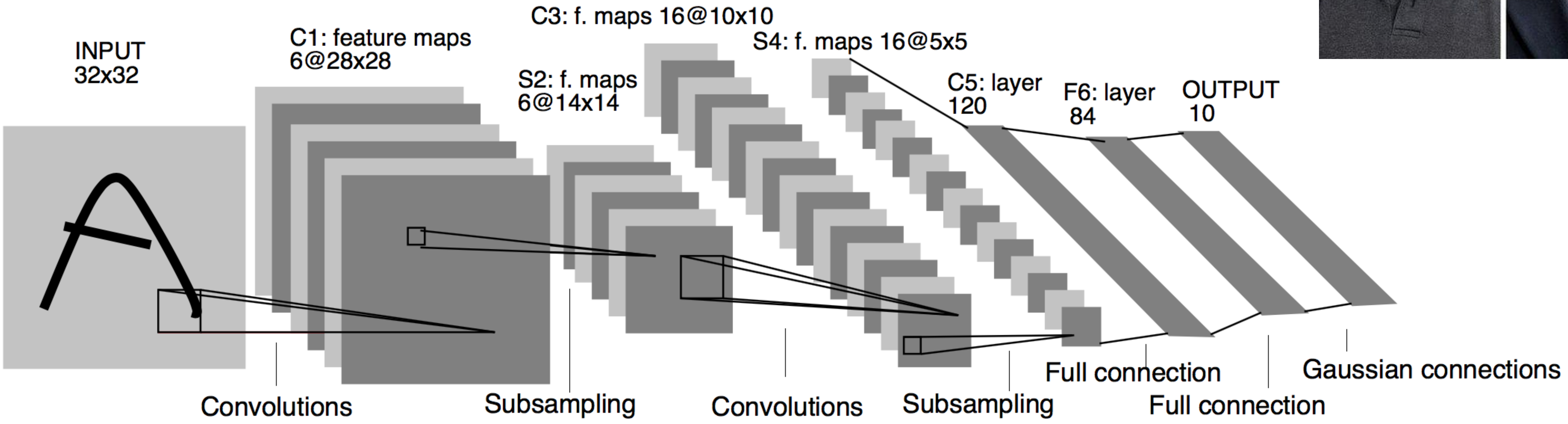
-MIT *Introduction to Deep Learning* http://introtodeeplearning.com

An overview of famous architectures and datasets

LeNet (LeNet-5)

Once upon a time... LeNet [LeCun et al., 1998]

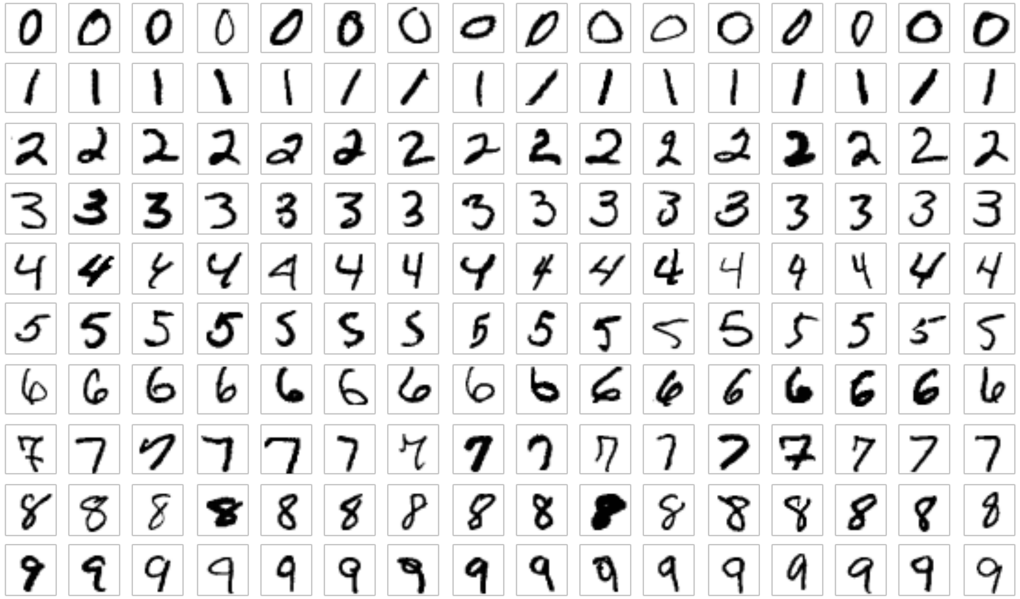
- 5x5 conv filters with stride 1
- 2x2 pooling with stride 2

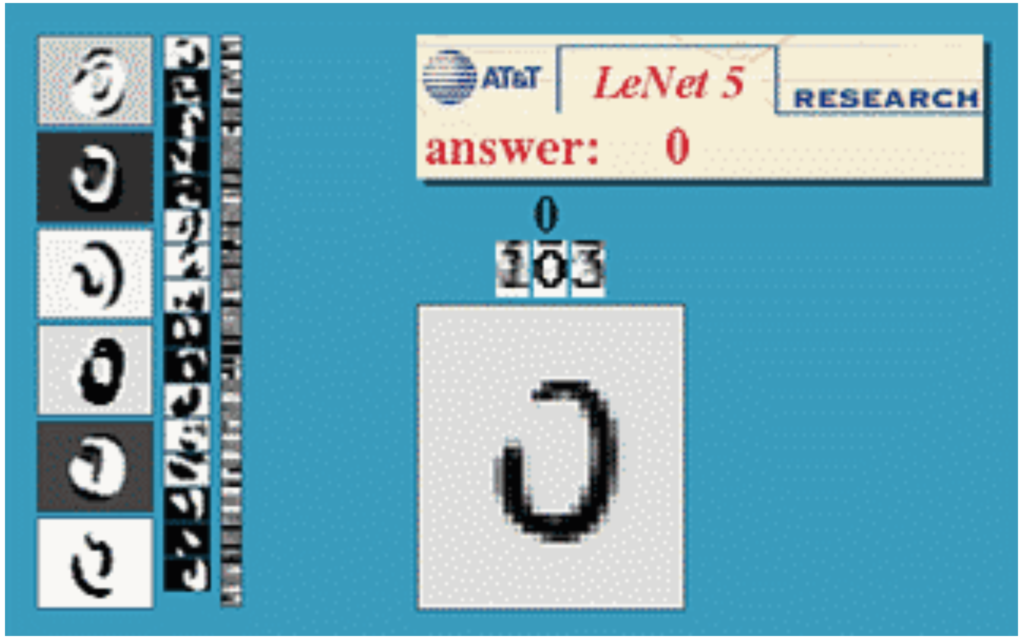


CONV-POOL-CONV-POOL-FC-FC architecture

MNIST (1998)

- Handwritten digits recognition
- 10 classes (of course!)
- 60k training images (28x28), 10k test images
- Grayscale images
- MNIST: Modified National Institute of Standards and Technology database
- Original NIST: training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students
- Available on Tensorflow
- LeNet on MNIST was the first real NN application!
- MNIST is not much of a challenge anymore...

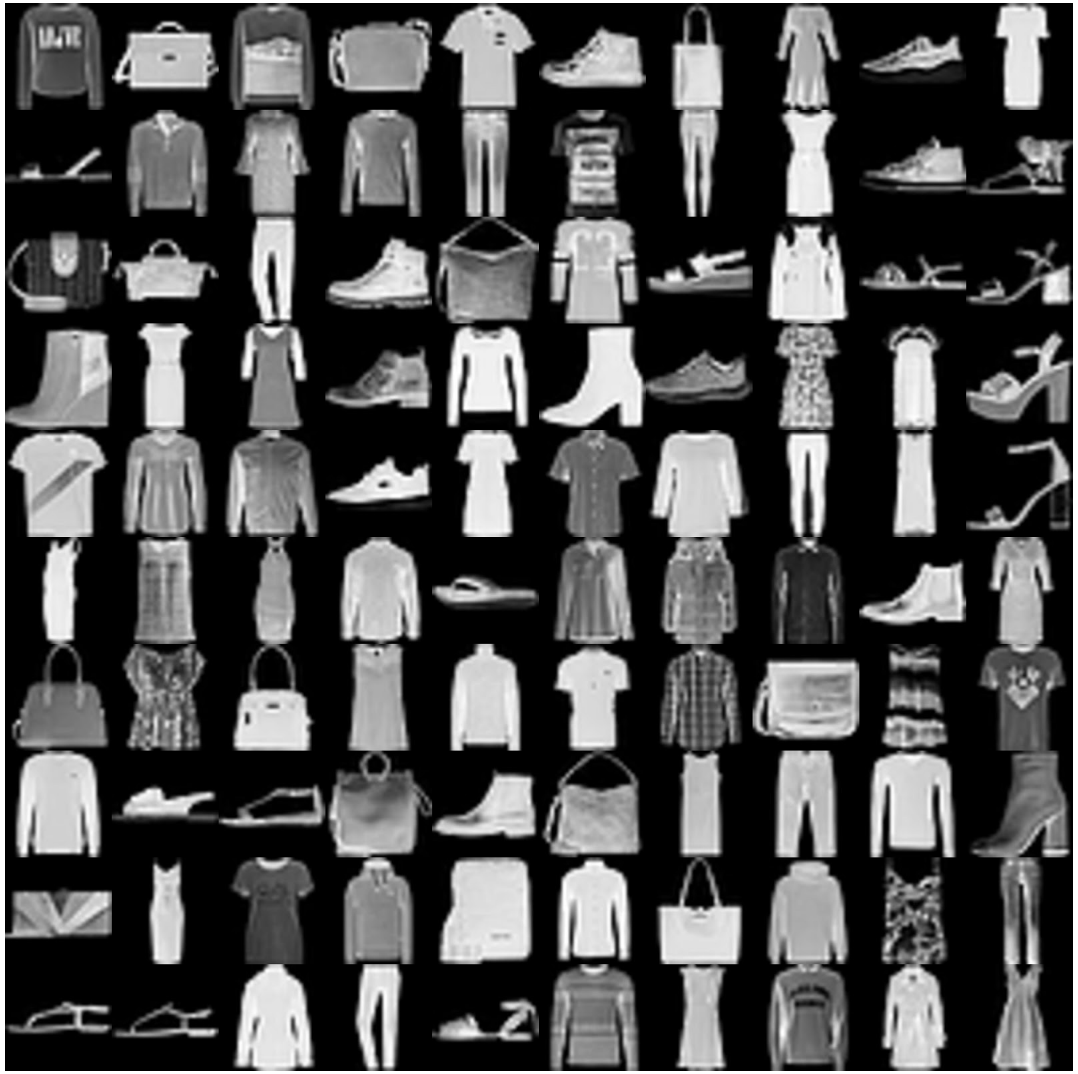




Fashion MNIST (2017)

- Clothing classification
- 10 classes
- 60k training images (28x28), 10k test images
- Grayscale images
- Available on Tensorflow
- More challenging then MNIST

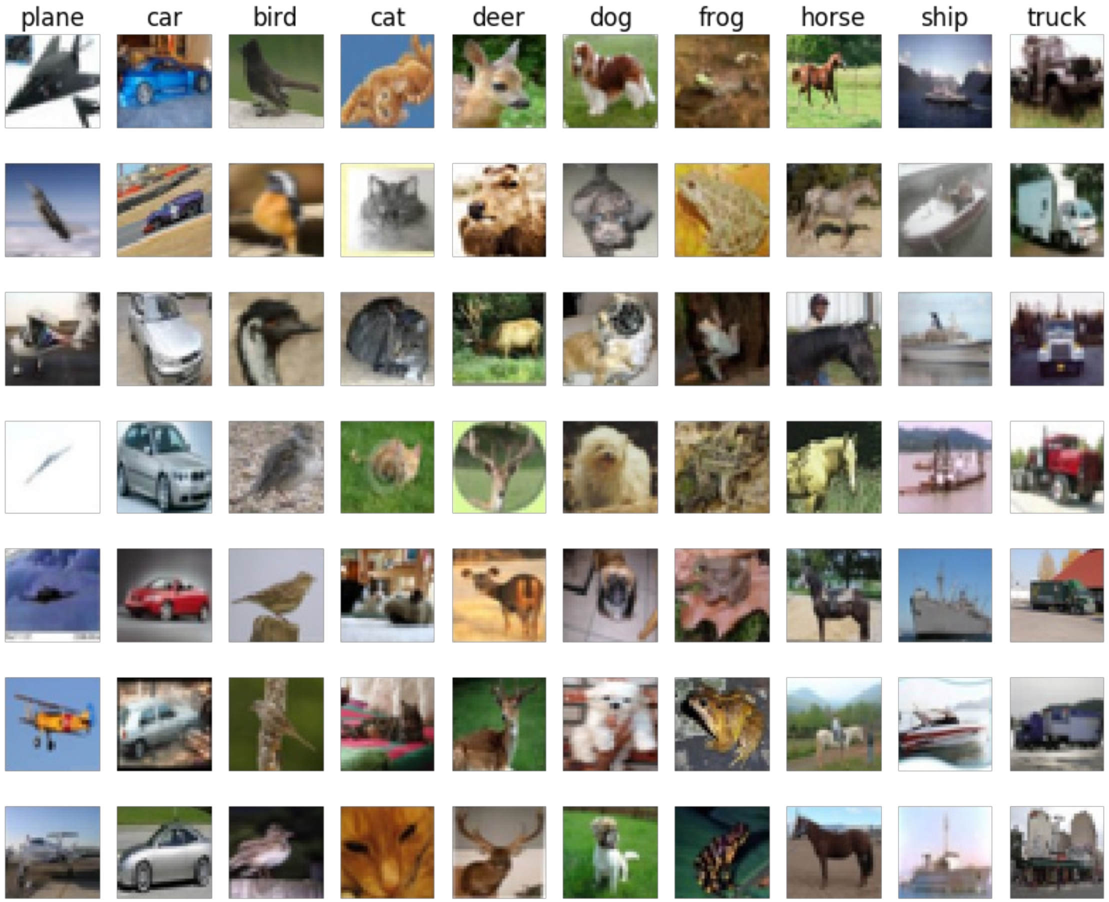
zalando research



Cifar10 (2010)

- 10 classes
- 60k images (32x32)
- Balanced dataset: 6000 images each
- Colour images
- It is a labeled subset of the 80 million tiny image dataset
- Available on Tensorflow
- Widely used for research nowadays

CIFAR CANADIAN INSTITUTE FOR ADVANCED RESEARCH

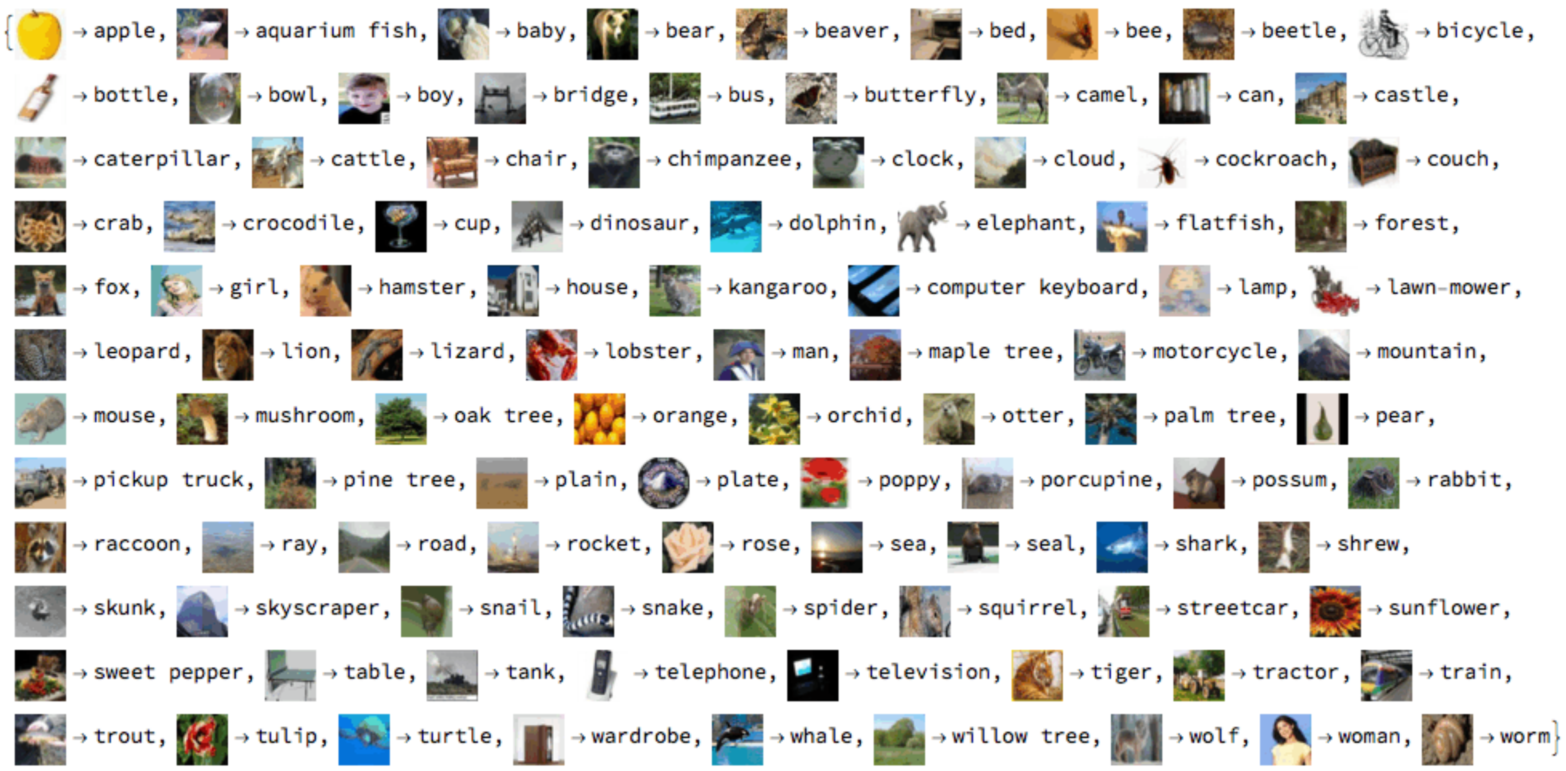


MIT https://groups.csail.mit.edu/vision/TinyImages/

Cifar100 (2010)

CIFAR
CANADIAN
INSTITUTE
FOR
ADVANCED
RESEARCH

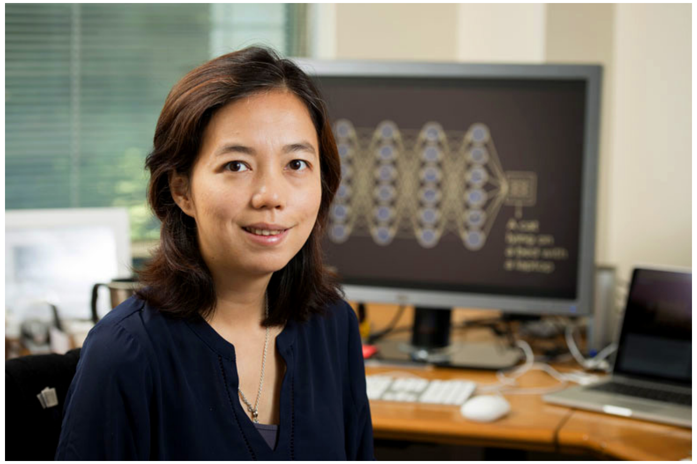
- 100 classes
- 60k images (32x32)
- Balanced dataset: 600 images each
- Colour images
- It is a labeled subset of the 80 million tiny image dataset
- Available on Tensorflow
- Widely used for research nowadays (less than Cifar10)


{→ apple, → aquarium fish, → baby, → bear, → beaver, → bed, → bee, → beetle, → bicycle,
→ bottle, → bowl, → boy, → bridge, → bus, → butterfly, → camel, → can, → castle,
→ caterpillar, → cattle, → chair, → chimpanzee, → clock, → cloud, → cockroach, → couch,
→ crab, → crocodile, → cup, → dinosaur, → dolphin, → elephant, → flatfish, → forest,
→ fox, → girl, → hamster, → house, → kangaroo, → computer keyboard, → lamp, → lawn-mower,
→ leopard, → lion, → lizard, → lobster, → man, → maple tree, → motorcycle, → mountain,
→ mouse, → mushroom, → oak tree, → orange, → orchid, → otter, → palm tree, → pear,
→ pickup truck, → pine tree, → plain, → plate, → poppy, → porcupine, → possum, → rabbit,
→ raccoon, → ray, → road, → rocket, → rose, → sea, → seal, → shark, → shrew,
→ skunk, → skyscraper, → snail, → snake, → spider, → squirrel, → streetcar, → sunflower,
→ sweet pepper, → table, → tank, → telephone, → television, → tiger, → tractor, → train,
→ trout, → tulip, → turtle, → wardrobe, → whale, → willow tree, → wolf, → woman, → worm}

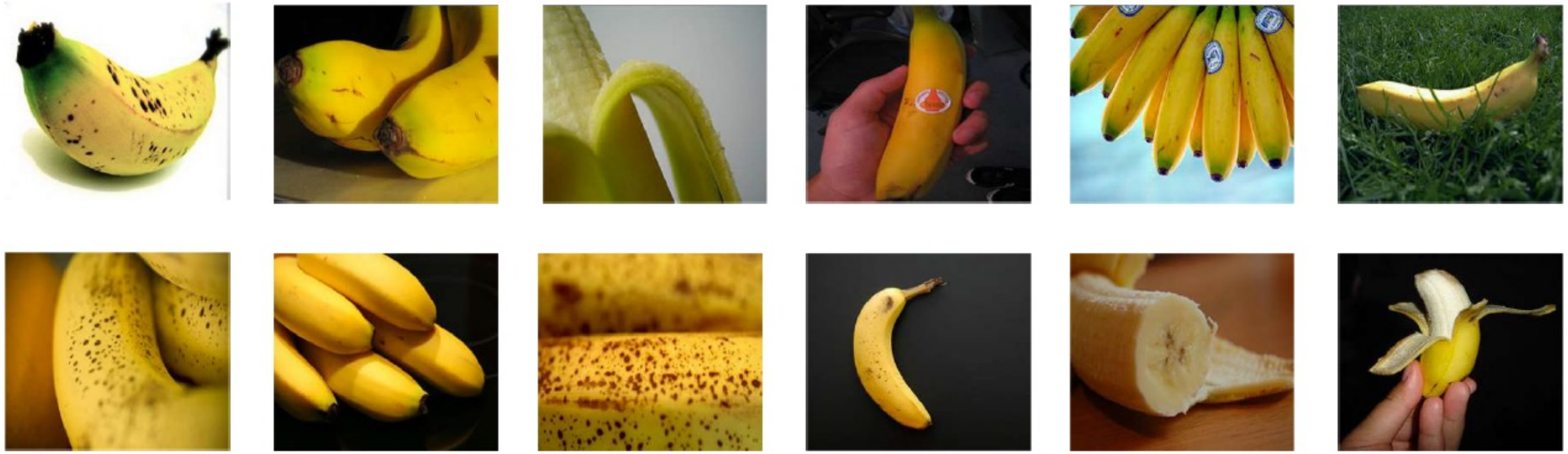
MIT https://groups.csail.mit.edu/vision/TinyImages/

Imagenet (2009)

- 21841 classes
- 14M images with different dimensions and resolutions (many apply resize to 256x256)
- Unbalanced dataset
- Colour images
- Lead developer Fei-Fei Li



"Elongated crescent-shaped yellow fruit with soft sweet flesh"



1409 pictures of bananas.

MIT https://groups.csail.mit.edu/vision/TinyImages/

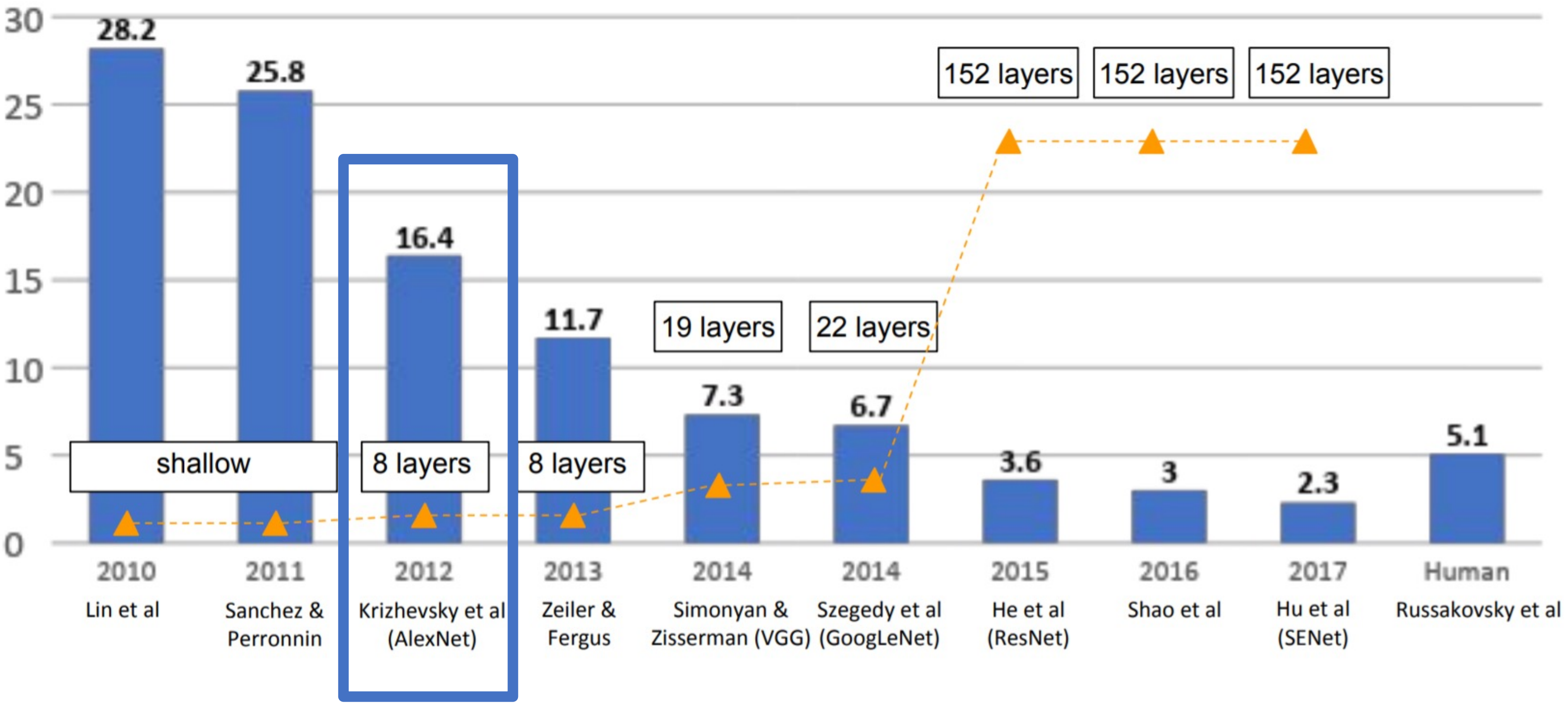
Classification task: produce a list of object categories present in image. 1000 categories.

“Top 5 error”: rate at which the model does not output correct label in top 5 predictions

An overview on the most famous architectures

Imagenet – visual recognition challenge with 1000 classes.

Winners:



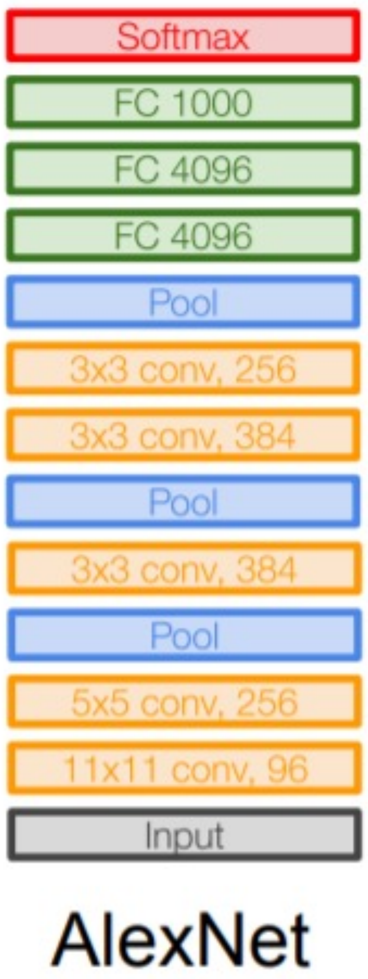
First CNN-based winner

AlexNet

AlexNet (2012) renewed interest in CNN.

It uses:

- RELU (first use)
- Layer normalization
- Dropout
- MaxPooling
- Momentum
- Data augmentation in training



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)

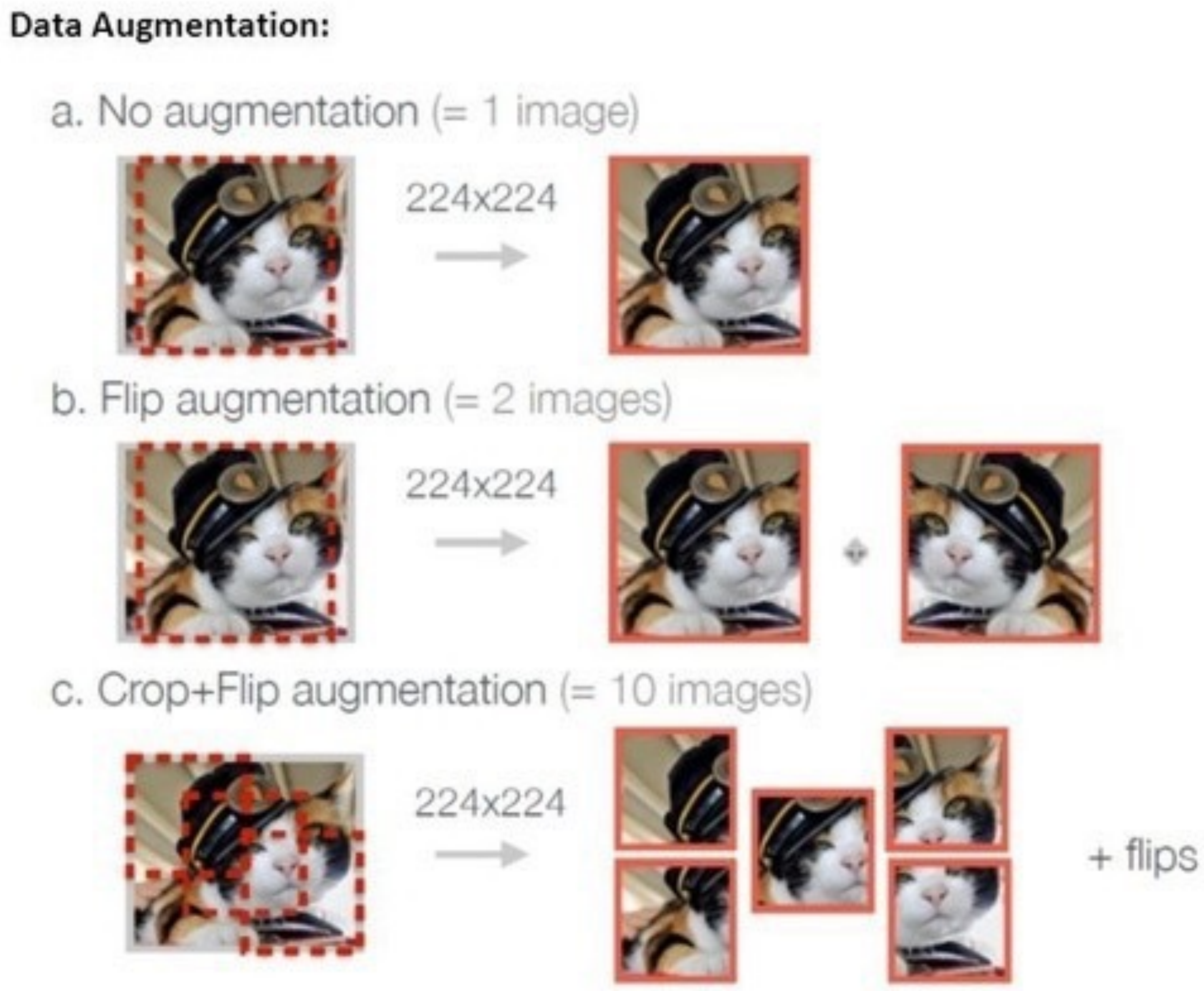
From: http://cs231n.stanford.edu - Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

AlexNet

AlexNet (2012) renewed interest in CNN.

It uses:

- RELU (first use)
- Layer normalization
- Dropout
- MaxPooling
- Momentum
- **Data augmentation in training**


Data Augmentation:
a. No augmentation (= 1 image)
224x224
b. Flip augmentation (= 2 images)
224x224
+
c. Crop+Flip augmentation (= 10 images)
224x224
+ flips

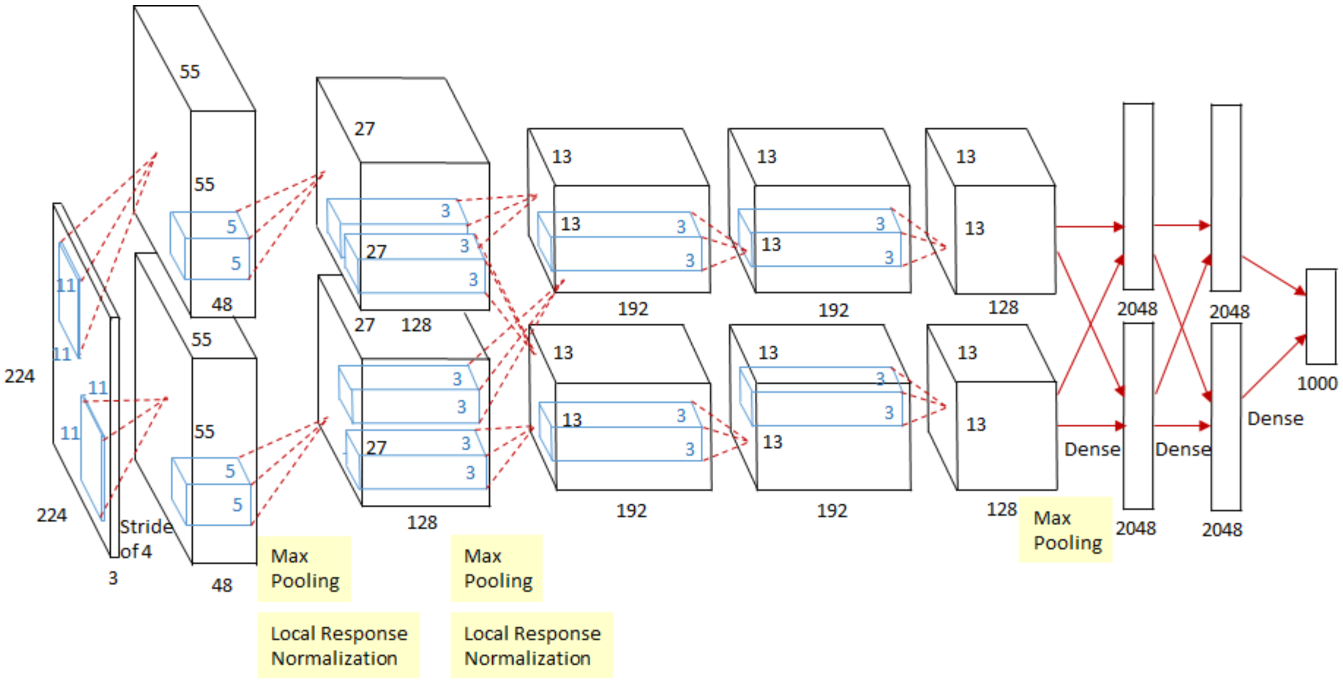
From: http://cs231n.stanford.edu - Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

AlexNet

AlexNet (2012) renewed interest in CNN.

It uses:

- RELU (first use)
- Layer normalization
- Dropout
- MaxPooling
- Momentum
- **Data augmentation in training**



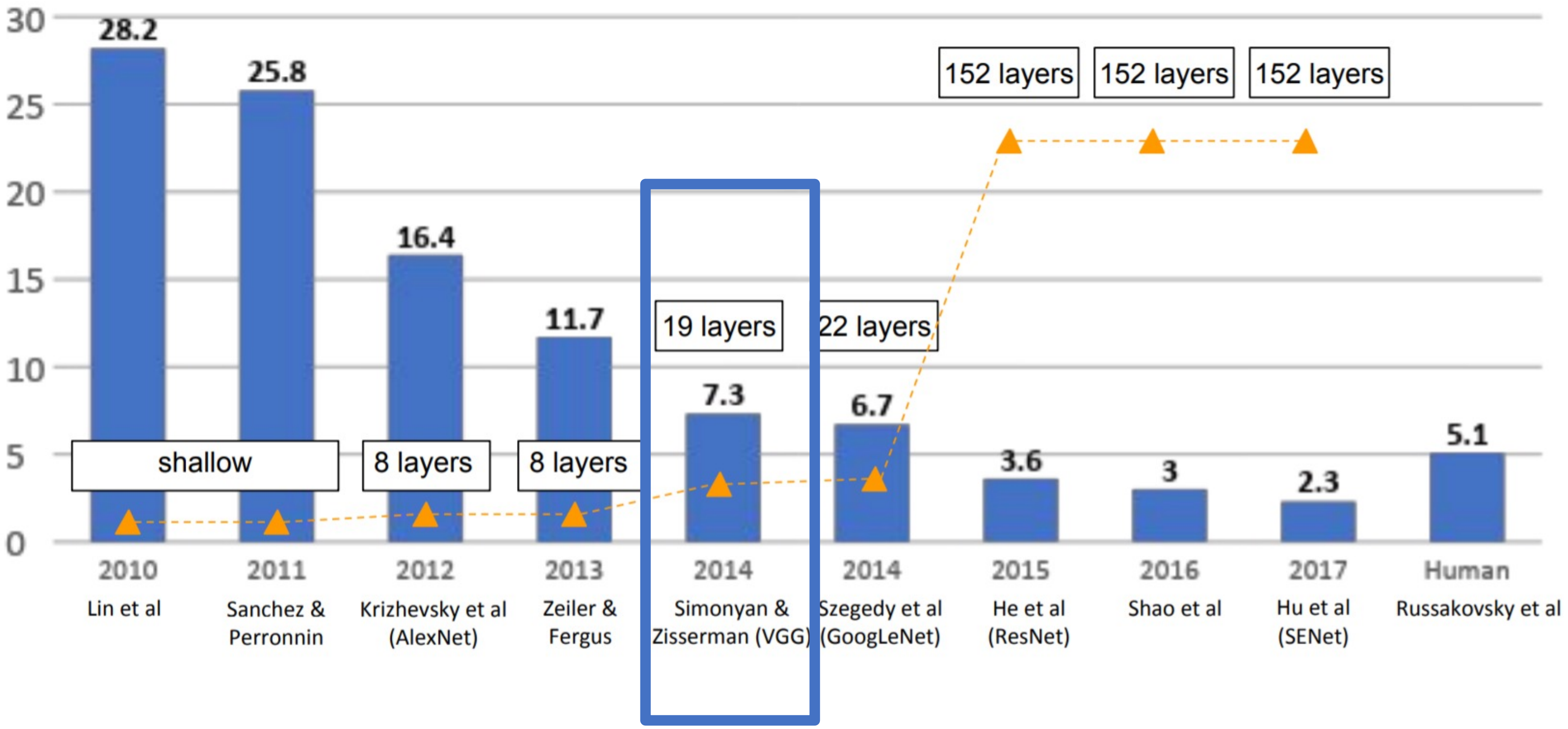
62.3 million parameters: the original paper showed that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of (GPUs) during training.

From: http://cs231n.stanford.edu - Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

An overview on the most famous architectures

Imagenet – visual recognition challenge with 1000 classes.

Winners:



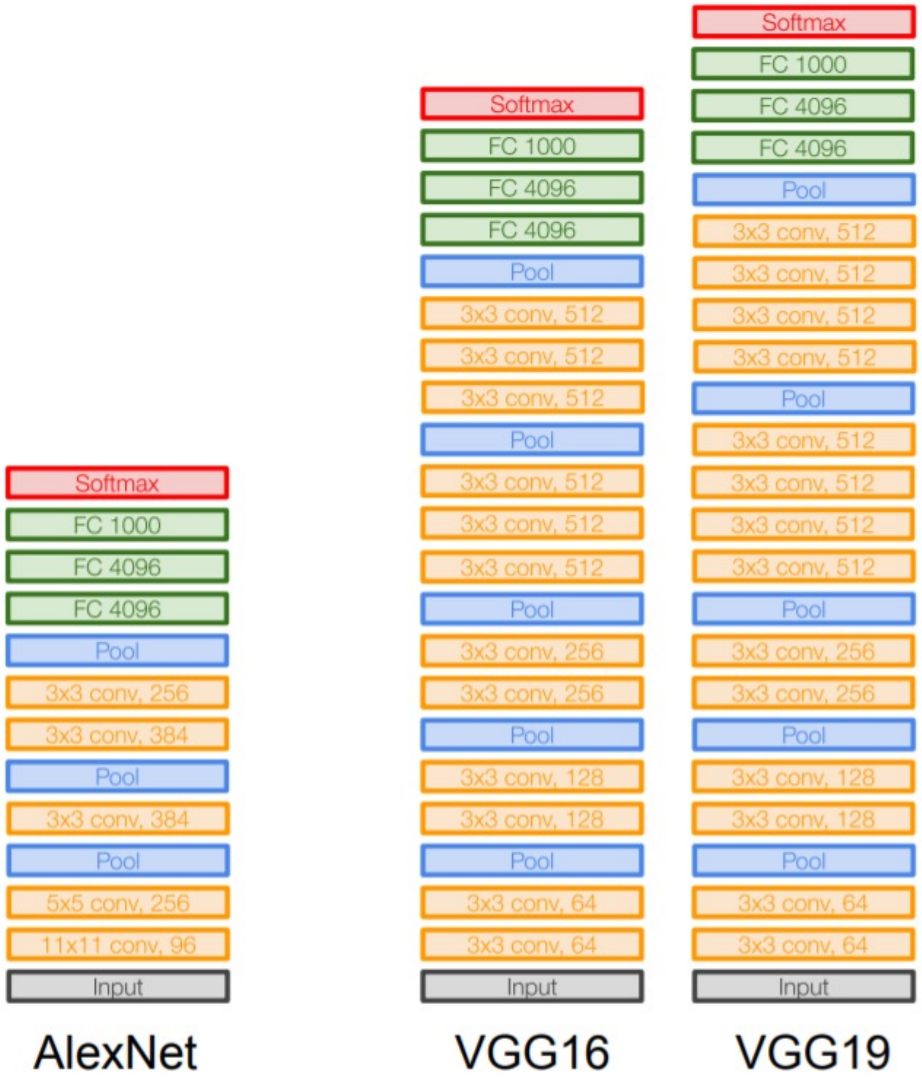
First really «deep» network

VGG

VGG (2014):

- many small convolutional filters stacked together before pooling

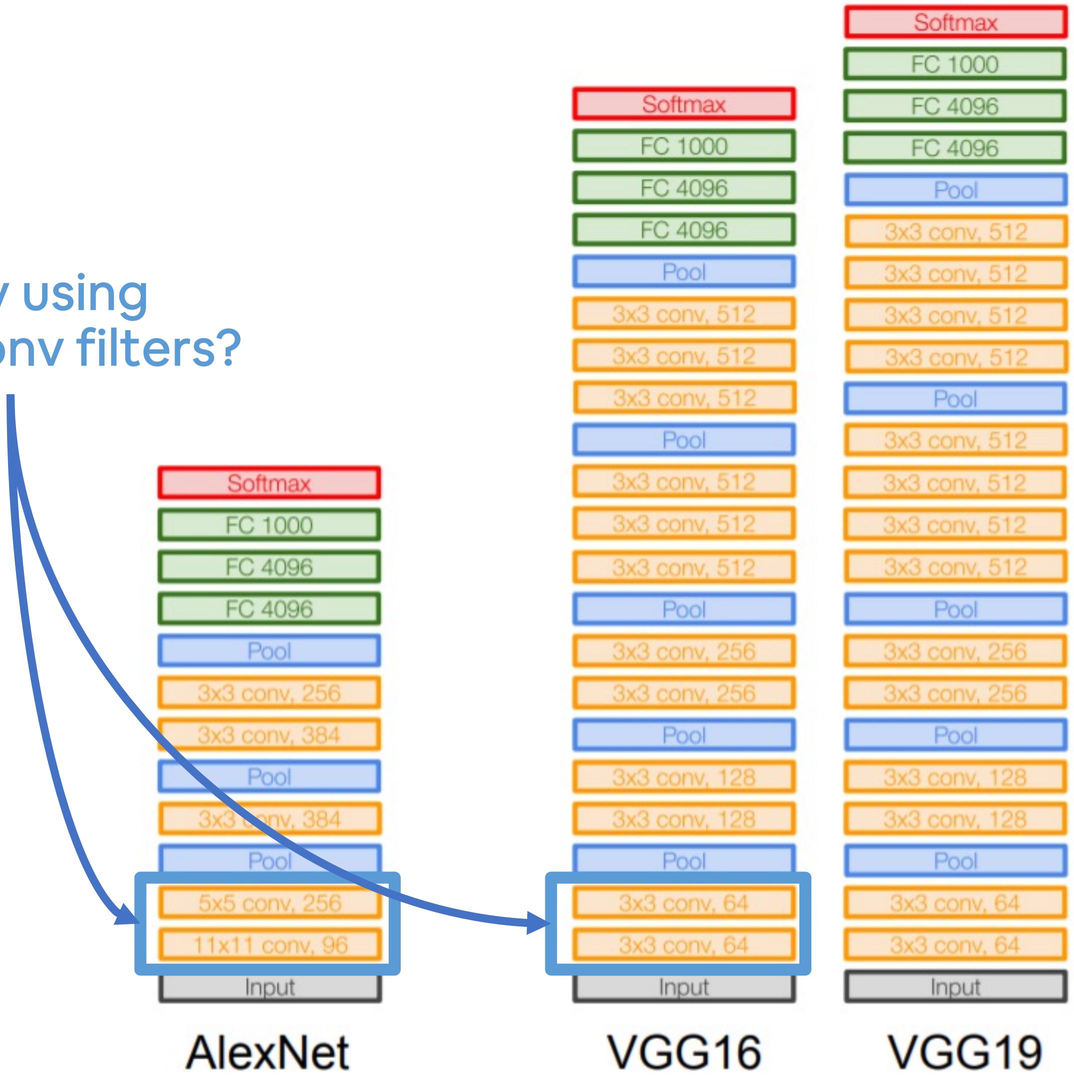
- very deep (at the time) with 16/19 layers



From: http://cs231n.stanford.edu

VGG

Wait. Why using smaller conv filters?

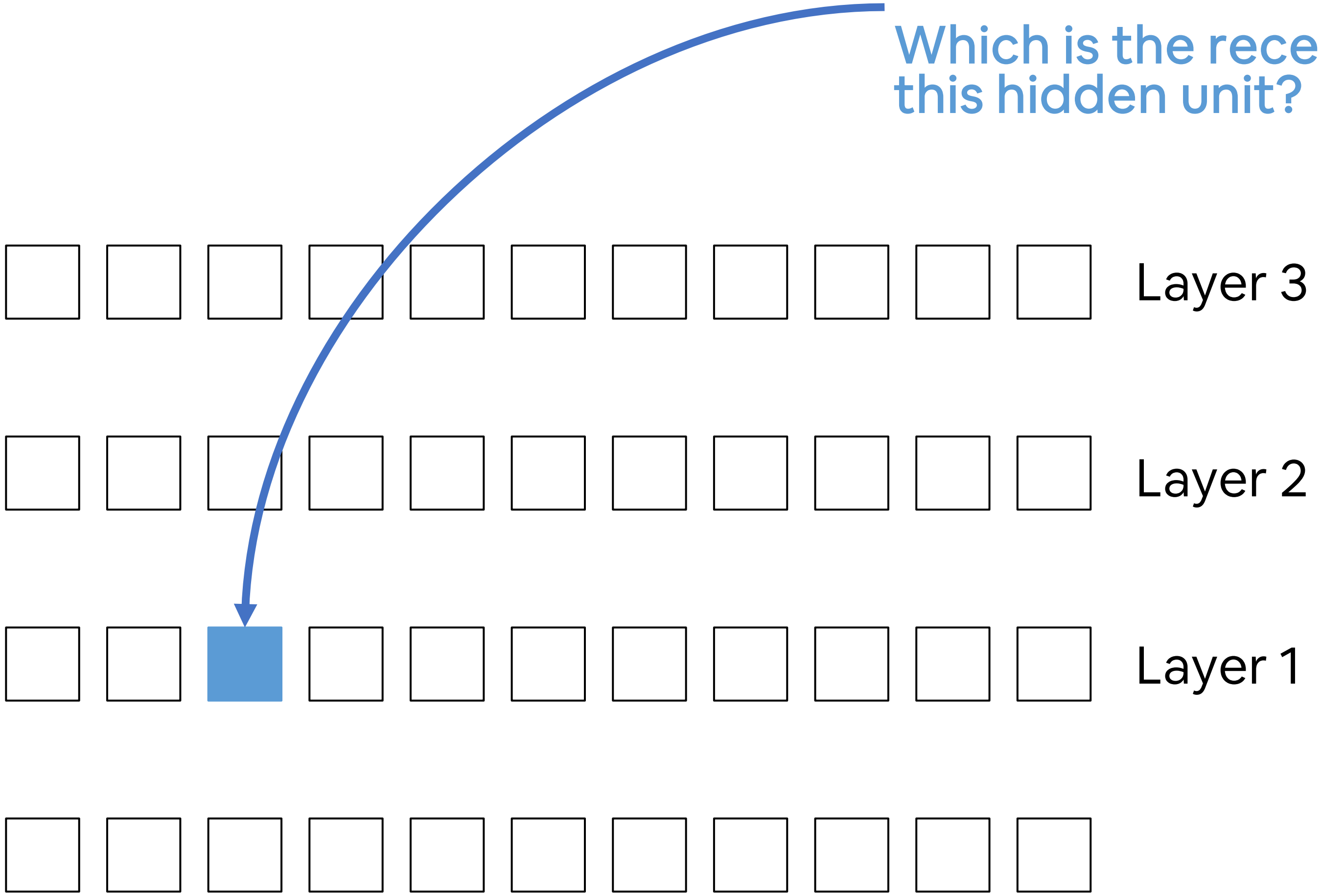


From: http://cs231n.stanford.edu

VGG

Consider a stack of three 3x3 conv layers.

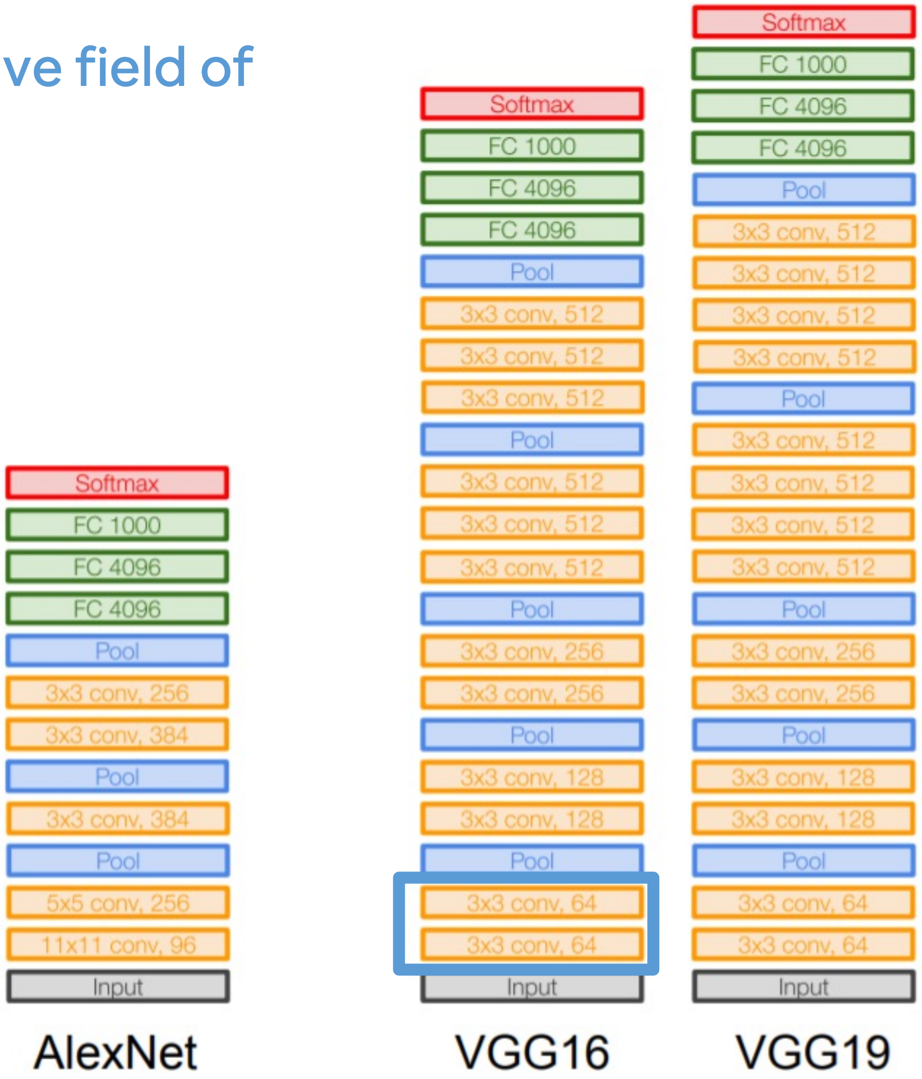
Which is the receptive field of this hidden unit?



Layer 3

Layer 2

Layer 1



Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 256
3x3 conv, 384
Pool
3x3 conv, 384
Pool
5x5 conv, 256
11x11 conv, 96
Input

AlexNet

Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input

VGG16

Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input

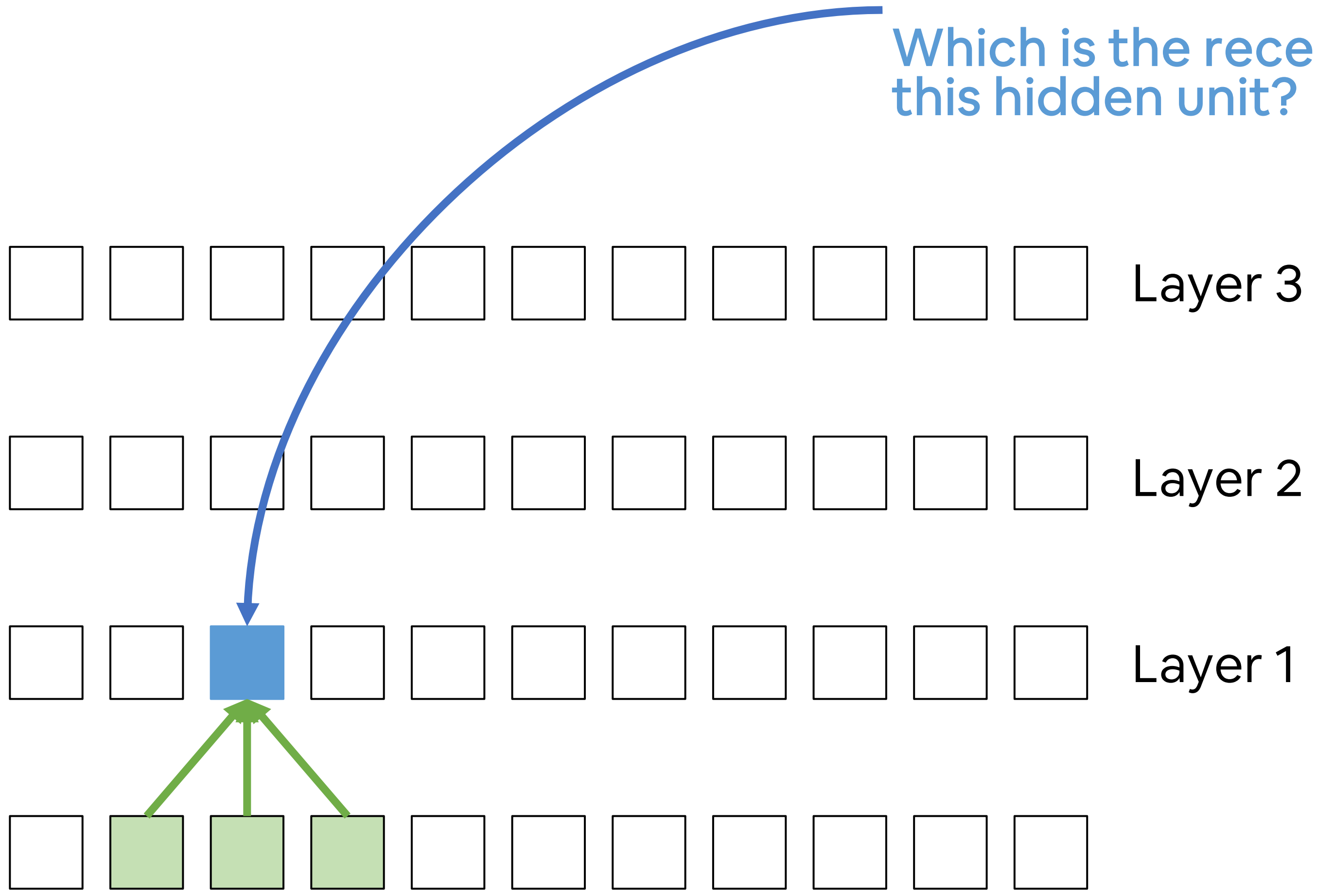
VGG19

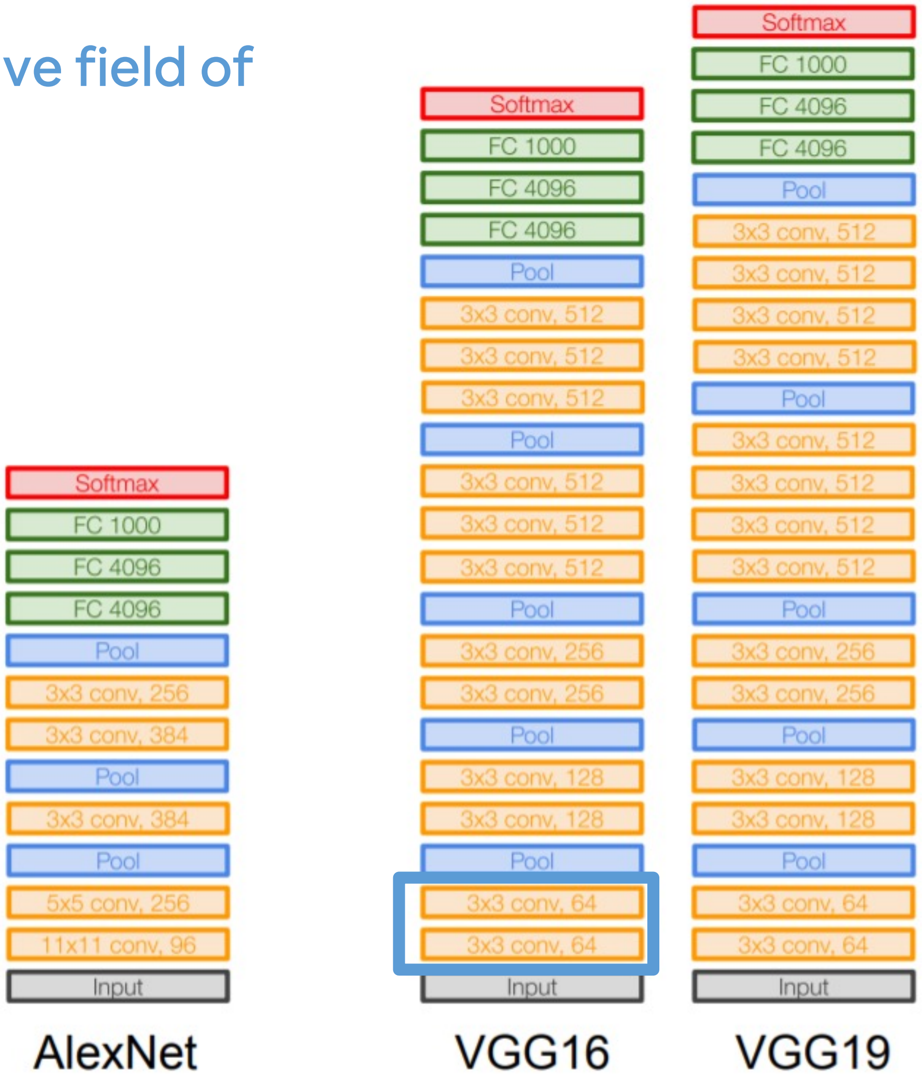
From: http://cs231n.stanford.edu

VGG

Consider a stack of three 3x3 conv layers.

Which is the receptive field of this hidden unit?



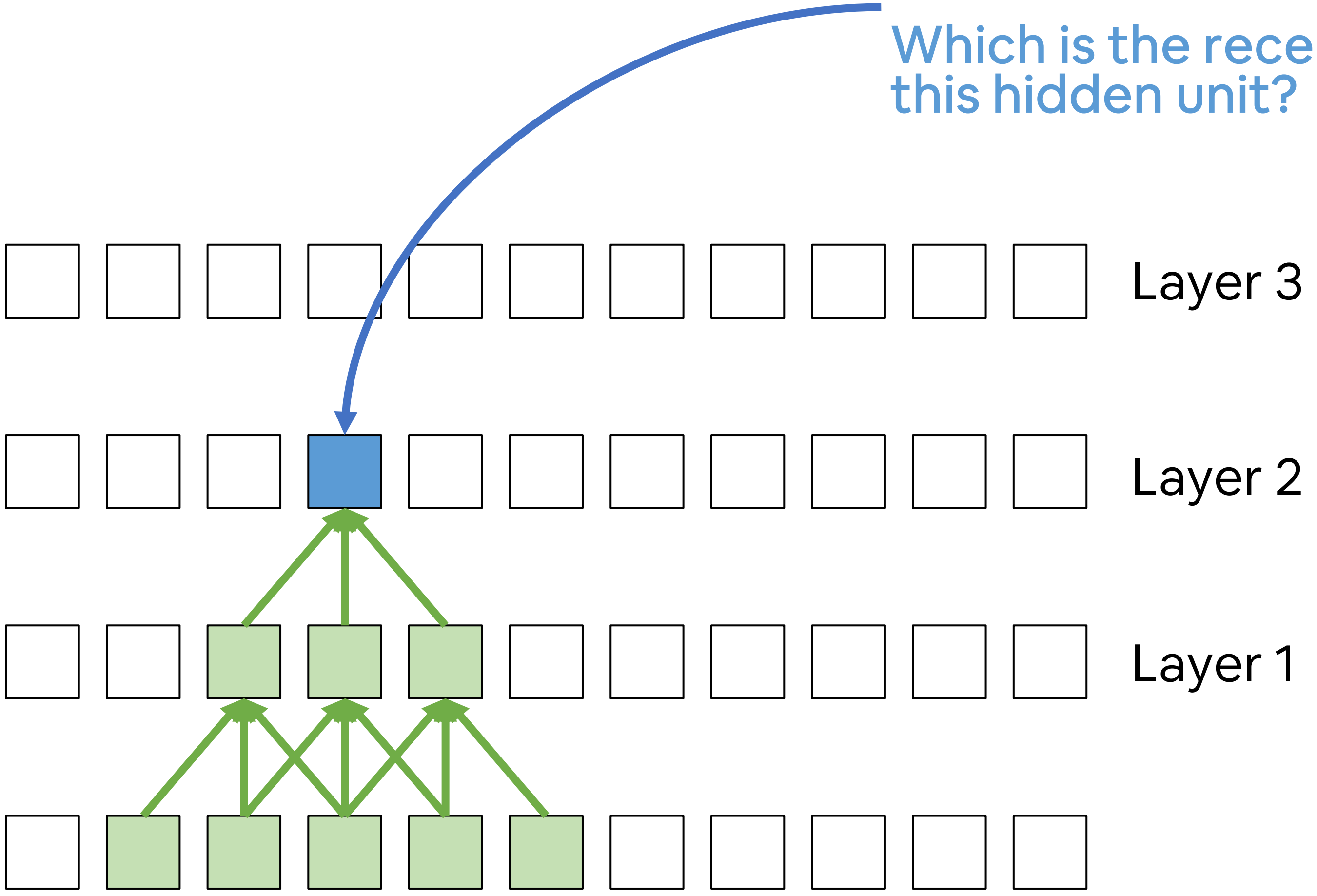


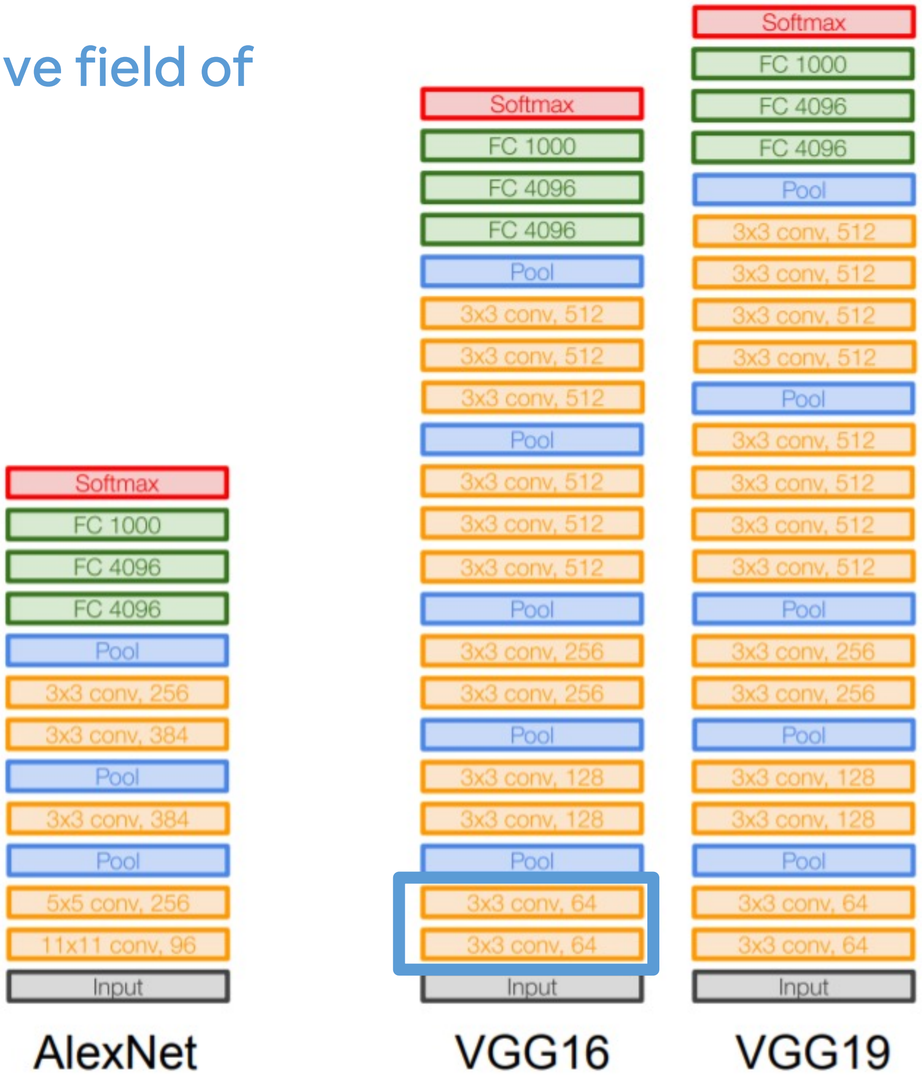
From: http://cs231n.stanford.edu

VGG

Consider a stack of three 3x3 conv layers.

Which is the receptive field of this hidden unit?



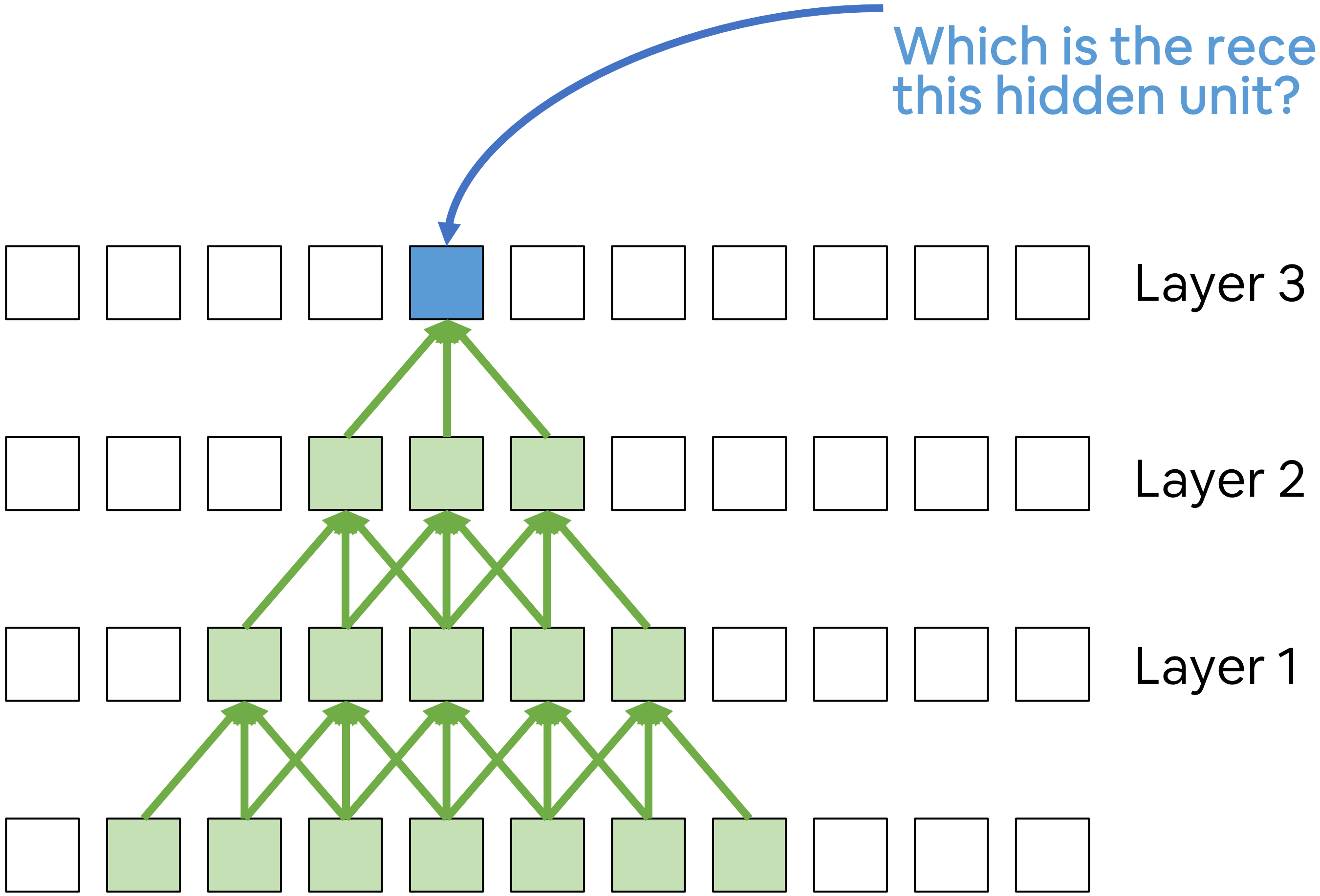


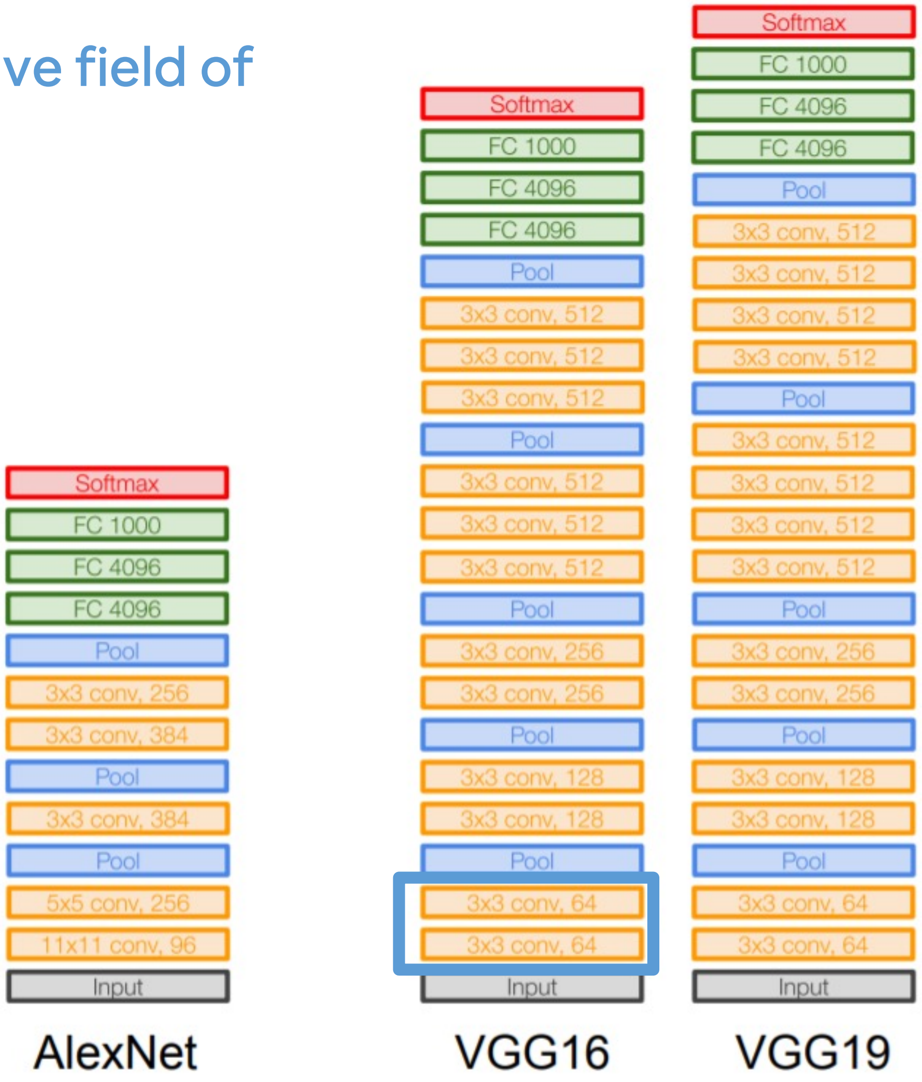
From: http://cs231n.stanford.edu

VGG

Consider a stack of three 3x3 conv layers.

Which is the receptive field of this hidden unit?





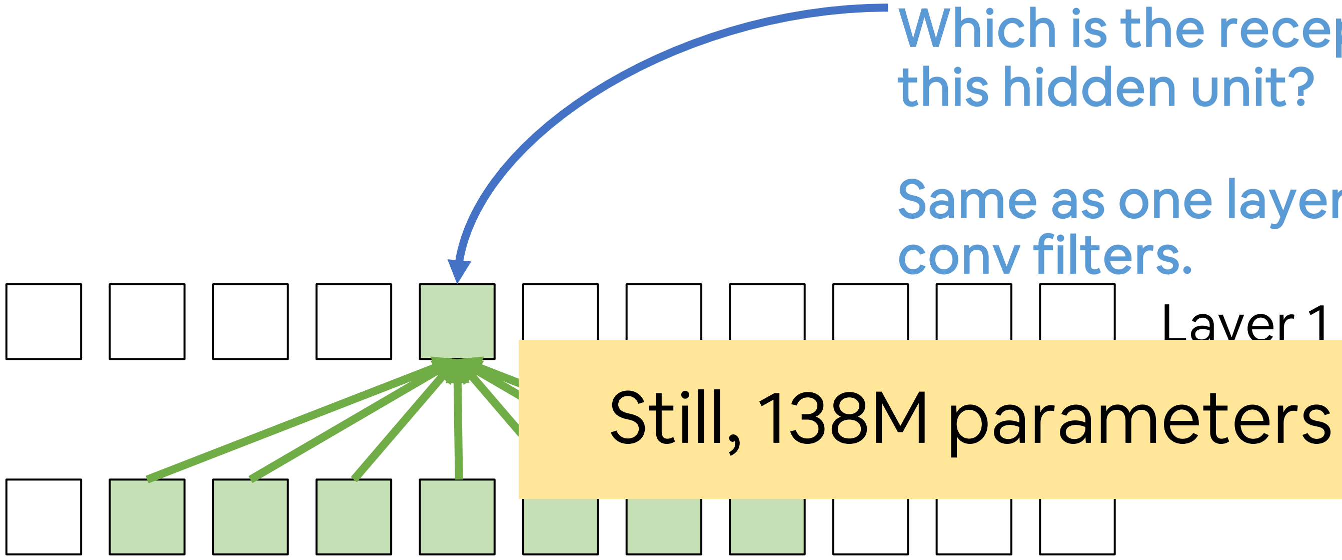
From: http://cs231n.stanford.edu

VGG

Consider a stack of three 3x3 conv layers.

Which is the receptive field of this hidden unit?

Same as one layer with 7x7 conv filters.



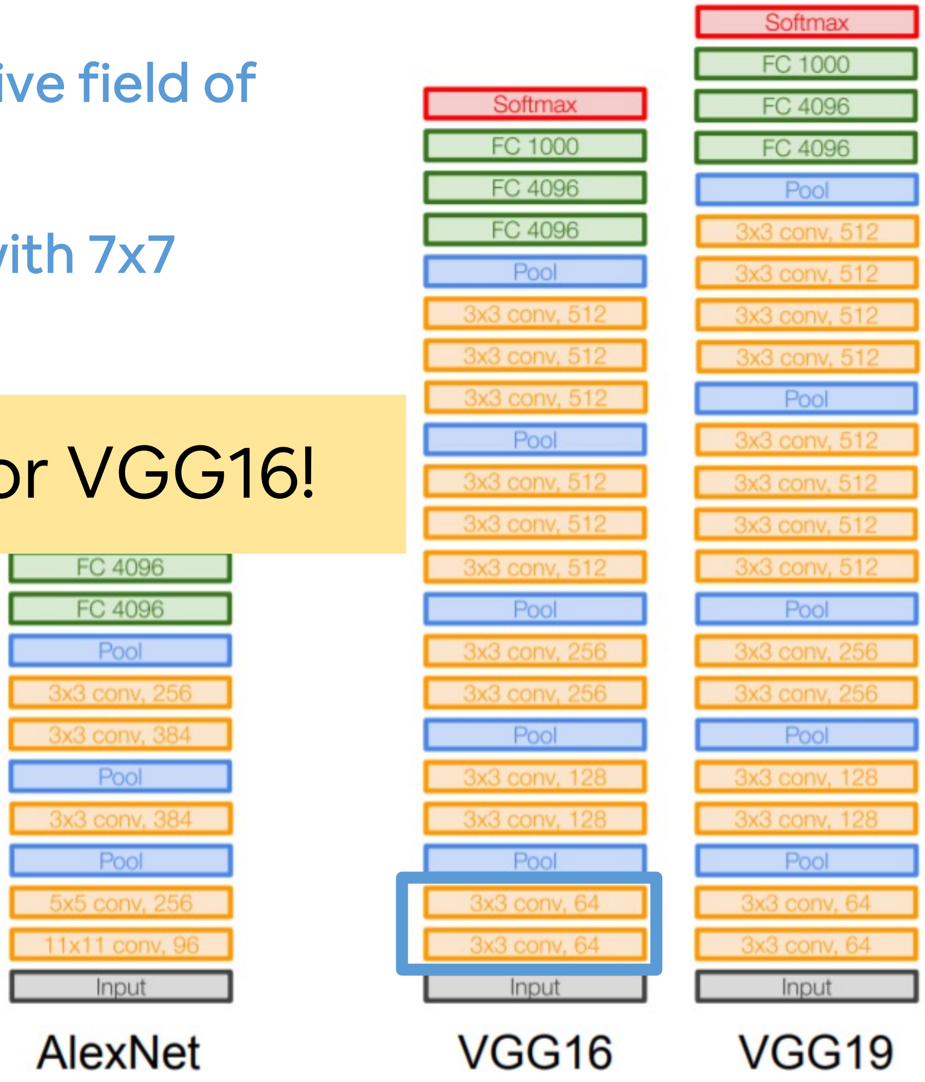
Still, 138M parameters for VGG16!

But three layers mean more non-linearities, i.e. more complex features...

... and fewer parameters!

C channels (e.g. C=3 for RGB images):

- 7x7 conv has 7^2 x C = 147 parameters
- 3 layers of 3x3 conv have 3 x (3^2 x C) = 81

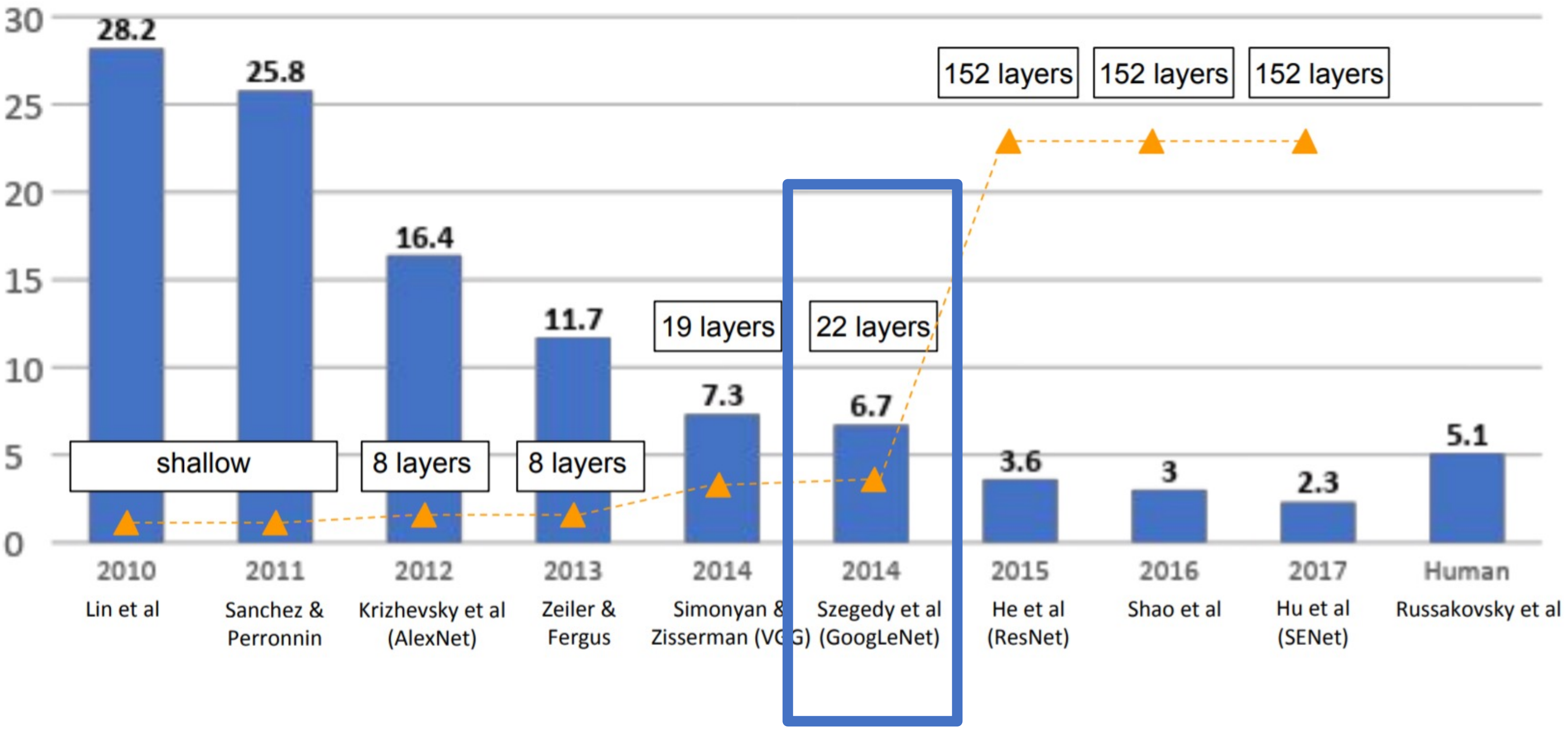


From: http://cs231n.stanford.edu

An overview on the most famous architectures

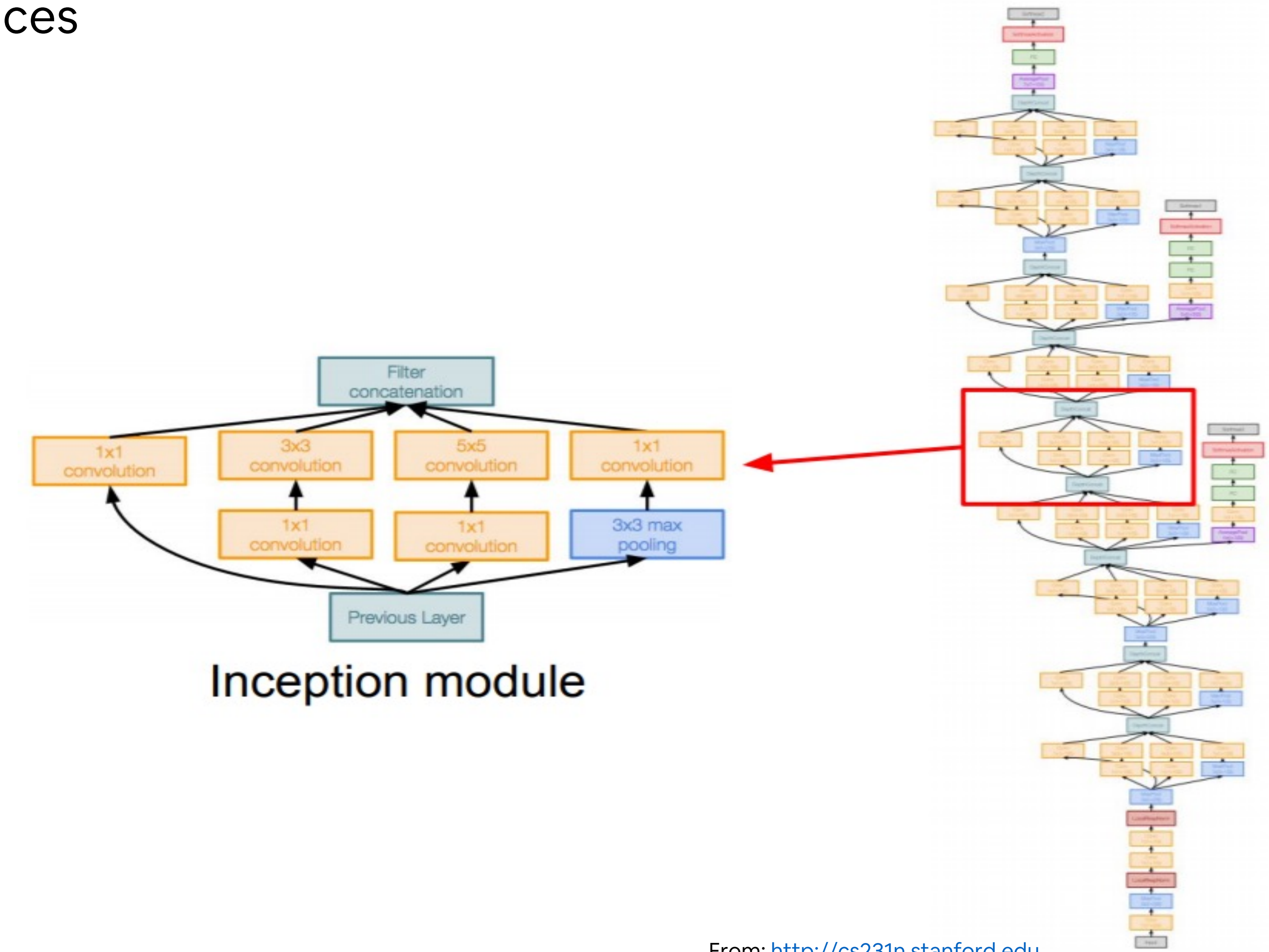
Imagenet – visual recognition challenge with 1000 classes.

Winners:



GoogLeNet

Inception + GoogLeNet (2015)– introduces parallel conv blocks (inception)



Inception module

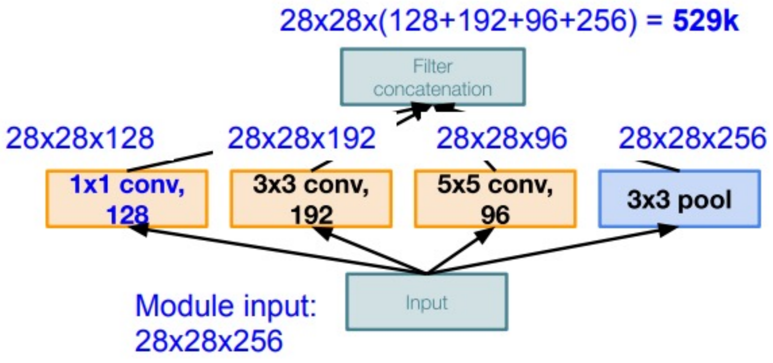
From: http://cs231n.stanford.edu

From: http://cs231n.stanford.edu

GoogLeNet

Inception + GoogLeNet (2015)– introduces parallel conv blocks (inception)

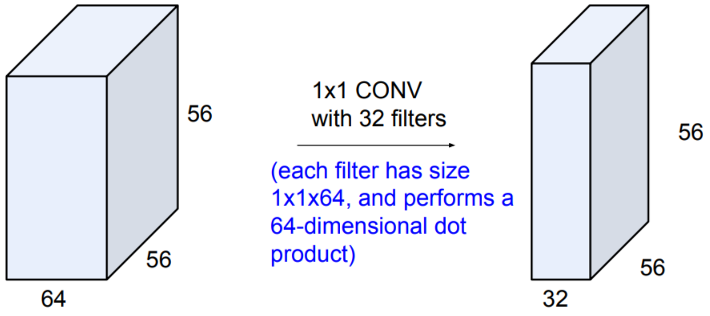
Cleverly uses 1x1 convolutions

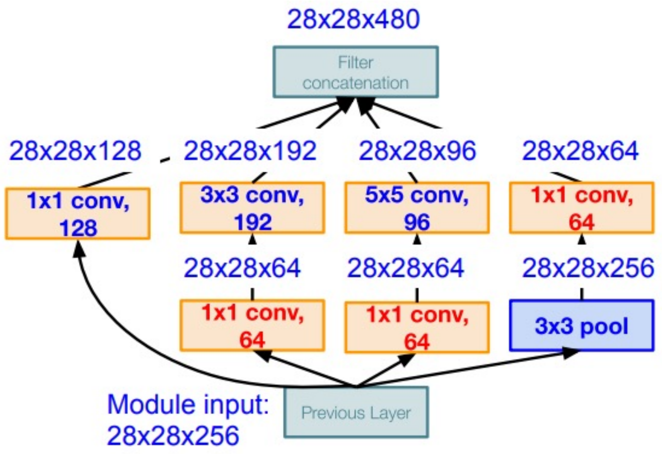


Naive Inception module

From: http://cs231n.stanford.edu

Preserves spatial dimensions, but reduces depth! Feature maps (depth) are projected to lower dimension


56
56
64
1x1 CONV
with 32 filters
(each filter has size 1x1x64, and performs a 64-dimensional dot product)
56
56
32



Inception module with dimension reduction

From: http://cs231n.stanford.edu

GoogLeNet

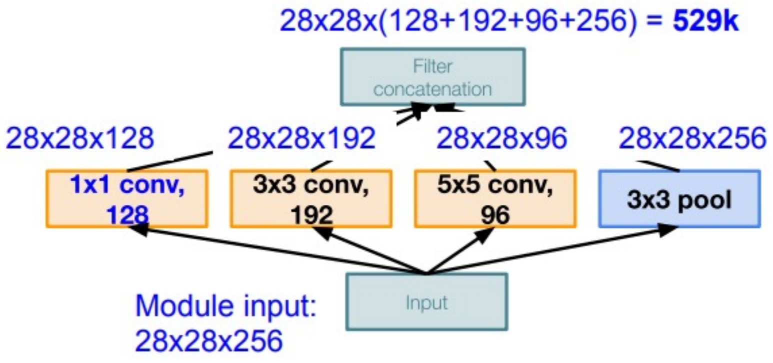
Preserves spatial dimensions, but reduces depth! Feature maps (depth) are projected to lower dimension

Inception + GoogLeNet (2015)– introduces parallel conv blocks (inception)

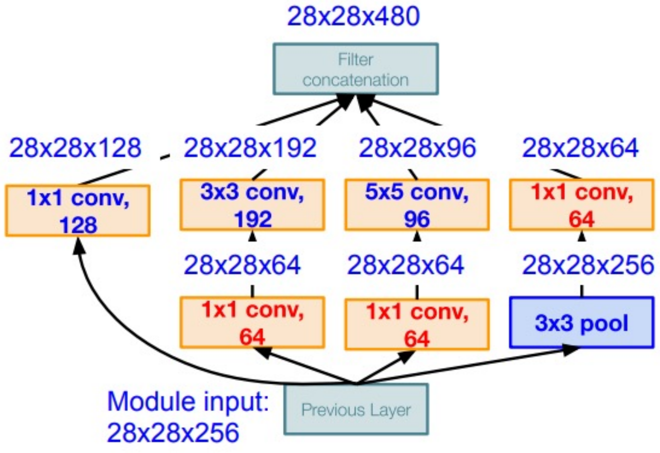
Cleverly

Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

Conv Ops:
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
Total: 358M ops



Naive Inception module



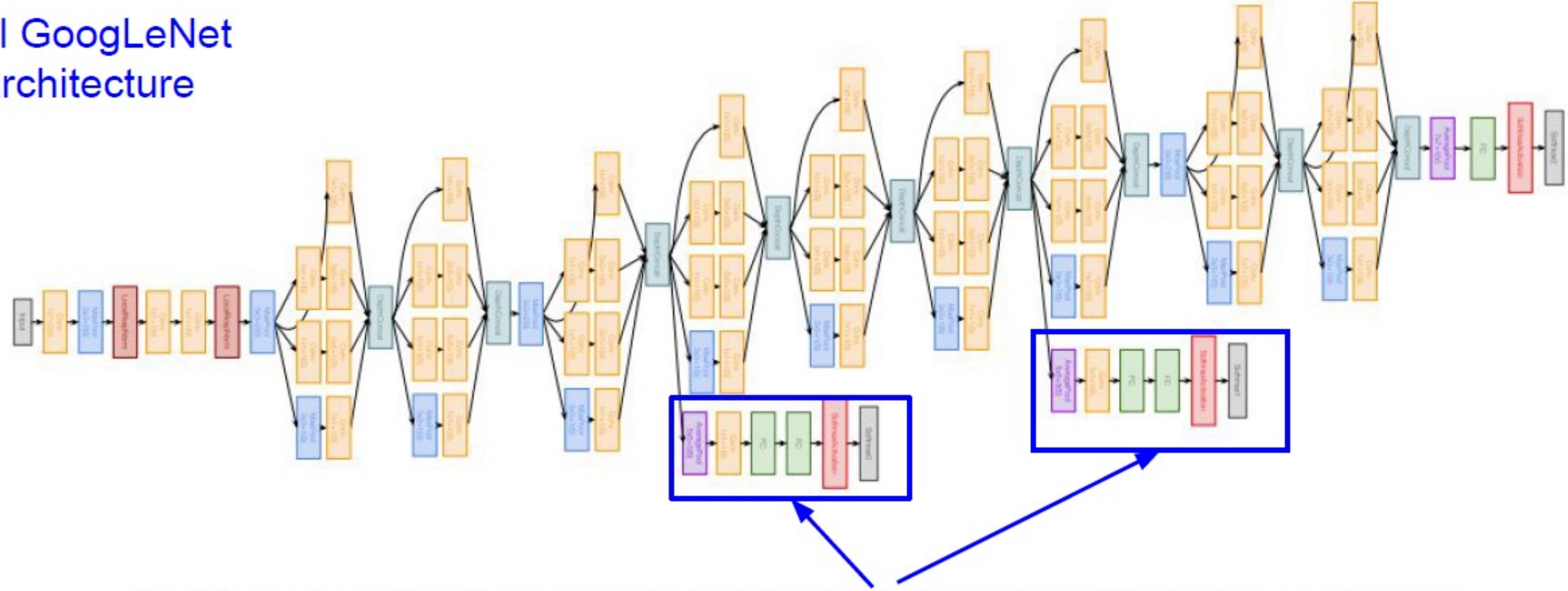
Inception module with dimension reduction

From: http://cs231n.stanford.edu

From: http://cs231n.stanford.edu

GoogLeNet

Full GoogLeNet architecture



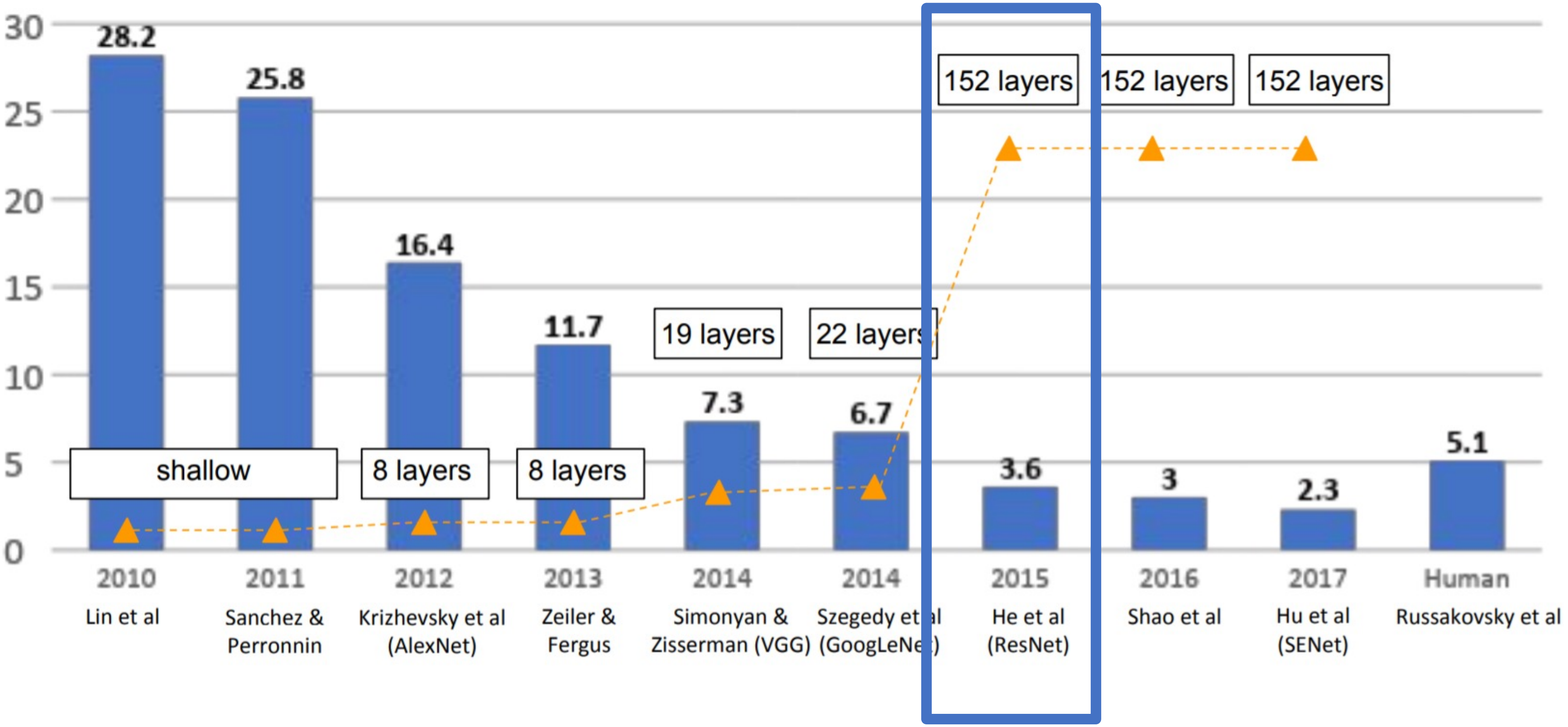
Auxiliary classification outputs to inject additional gradient at lower layers
(AvgPool-1x1Conv-FC-FC-Softmax)

From: http://cs231n.stanford.edu

An overview on the most famous architectures

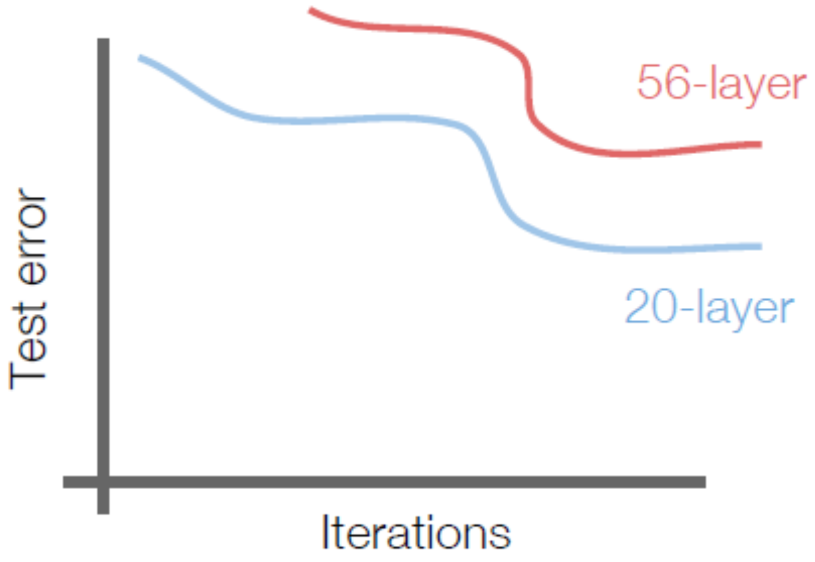
Imagenet – visual recognition challenge with 1000 classes.

Winners:



ResNet

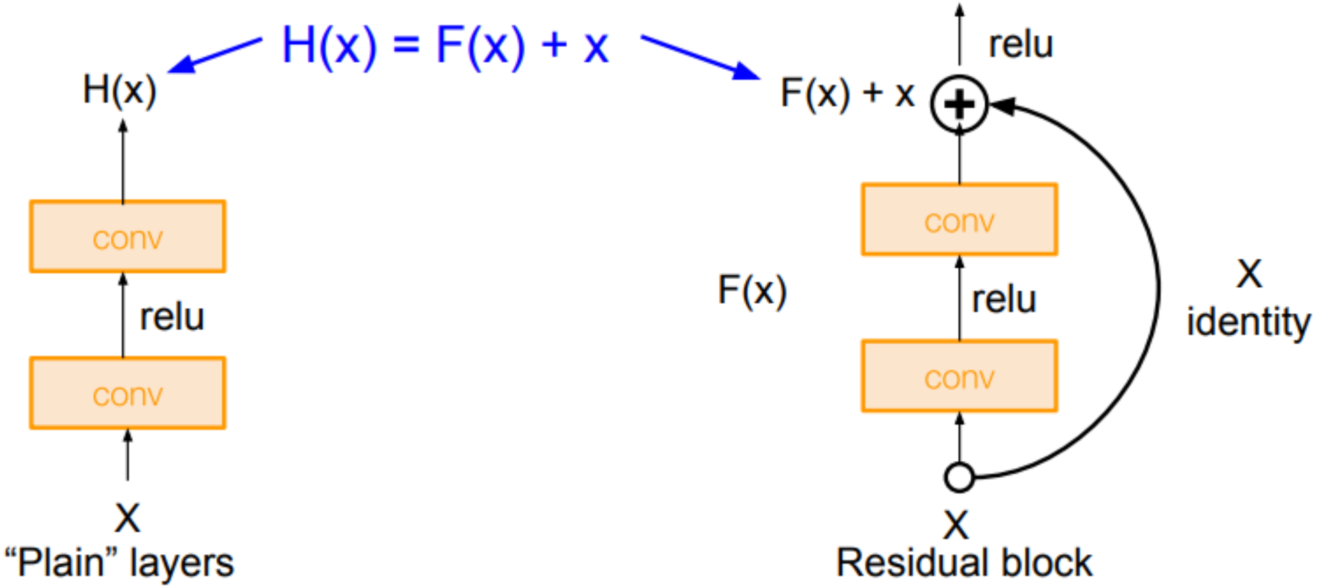
Why not stacking more and more layers?



From: http://cs231n.stanford.edu

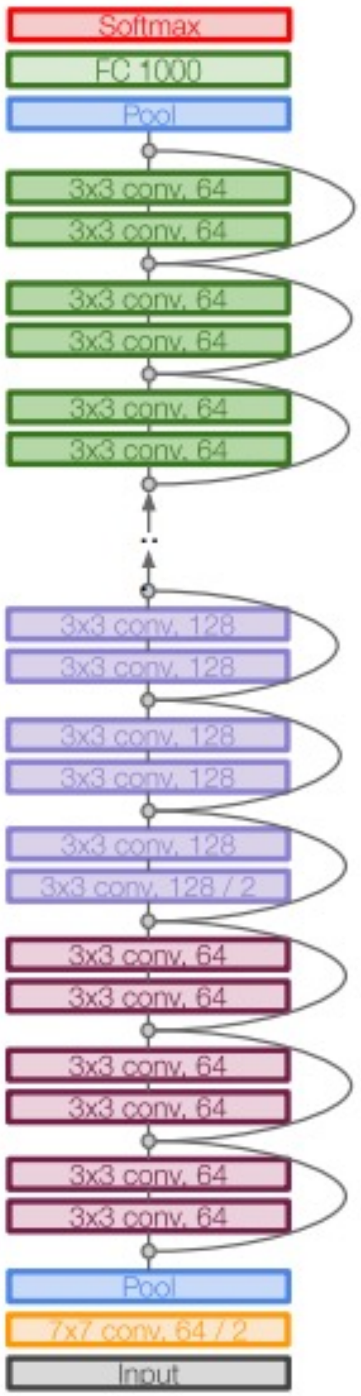
ResNet

ResNet (2015) – very deep model (152 layer) with shortcut connections



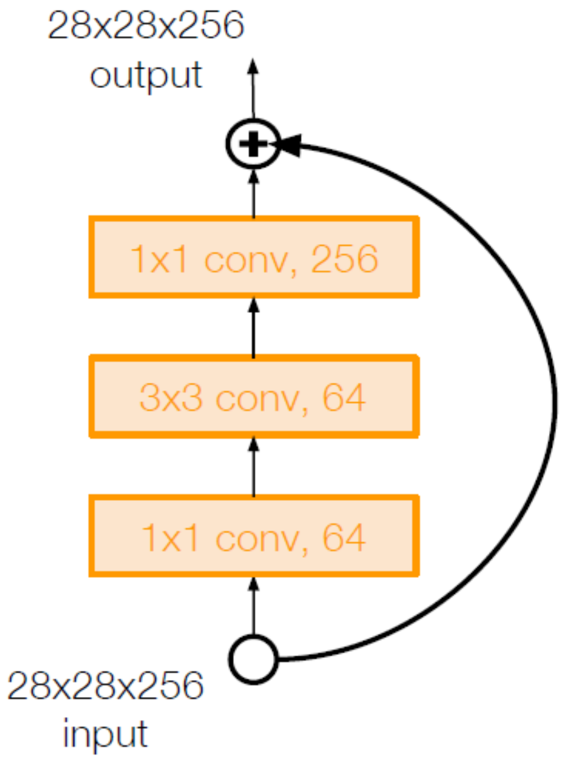
Use layers to fit residual F(x) = H(x) - x instead of H(x) directly

From: http://cs231n.stanford.edu



From: http://cs231n.stanfo

ResNet



For ResNet with more than 50 layers

From: http://cs231n.stanford.edu

ResNet training:

- Batch Normalization after every CONV layer
- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

State of the art is always on the move...

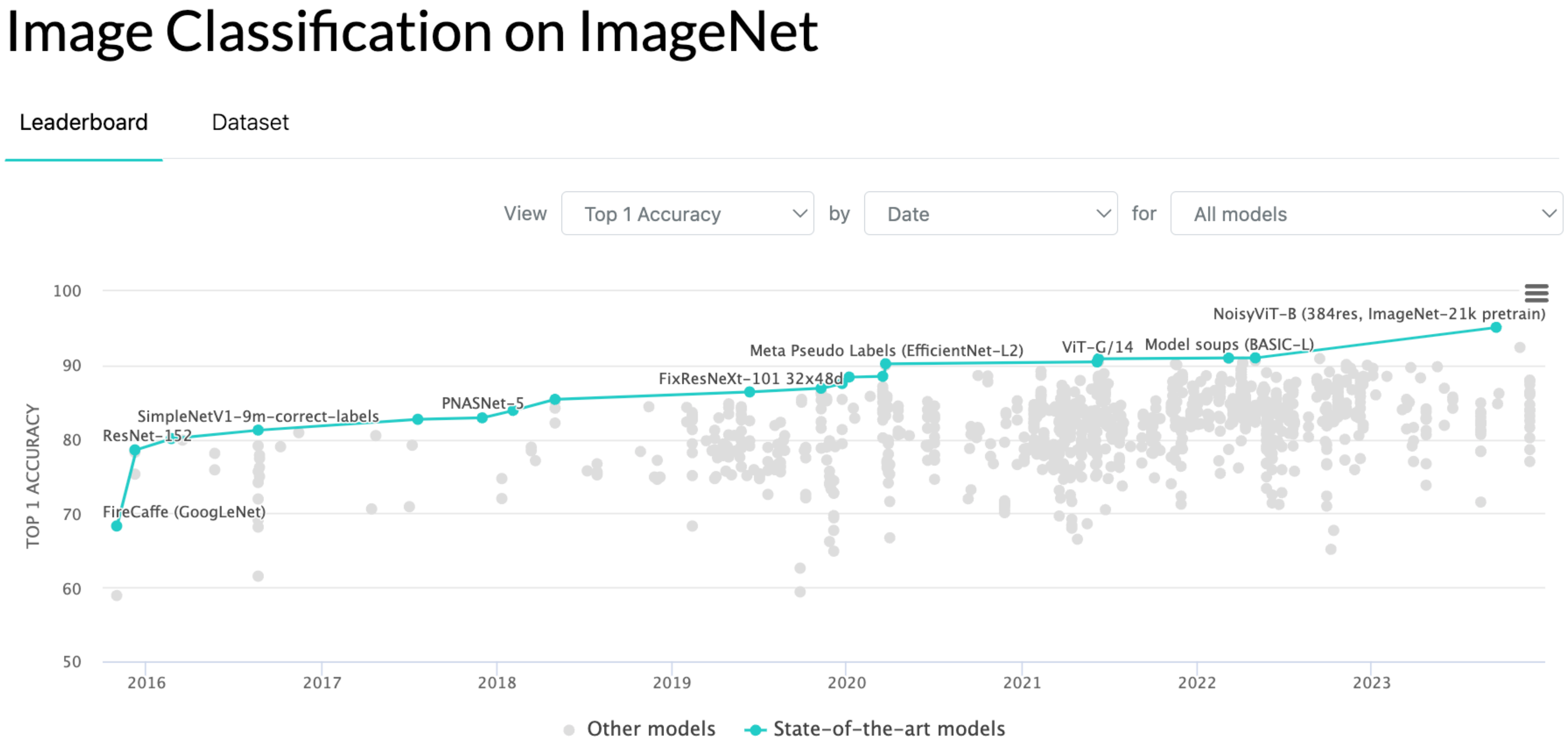


Image Classification on ImageNet

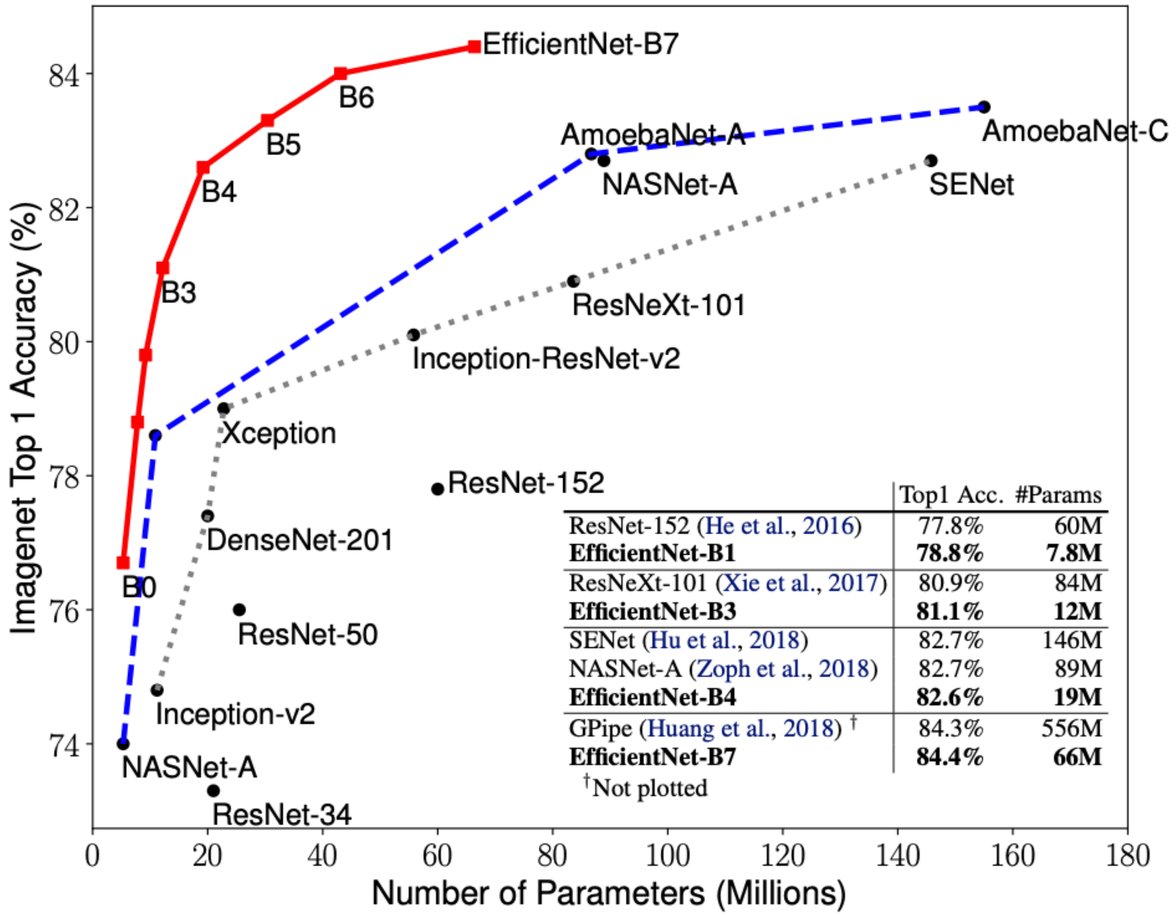
Leaderboard   Dataset

View Top 1 Accuracy by Date for All models

https://paperswithcode.com/sota/image-classification-on-imagenet

Is not all about accuracy… EfficientNet

Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning* (pp. 6105-6114). PMLR.



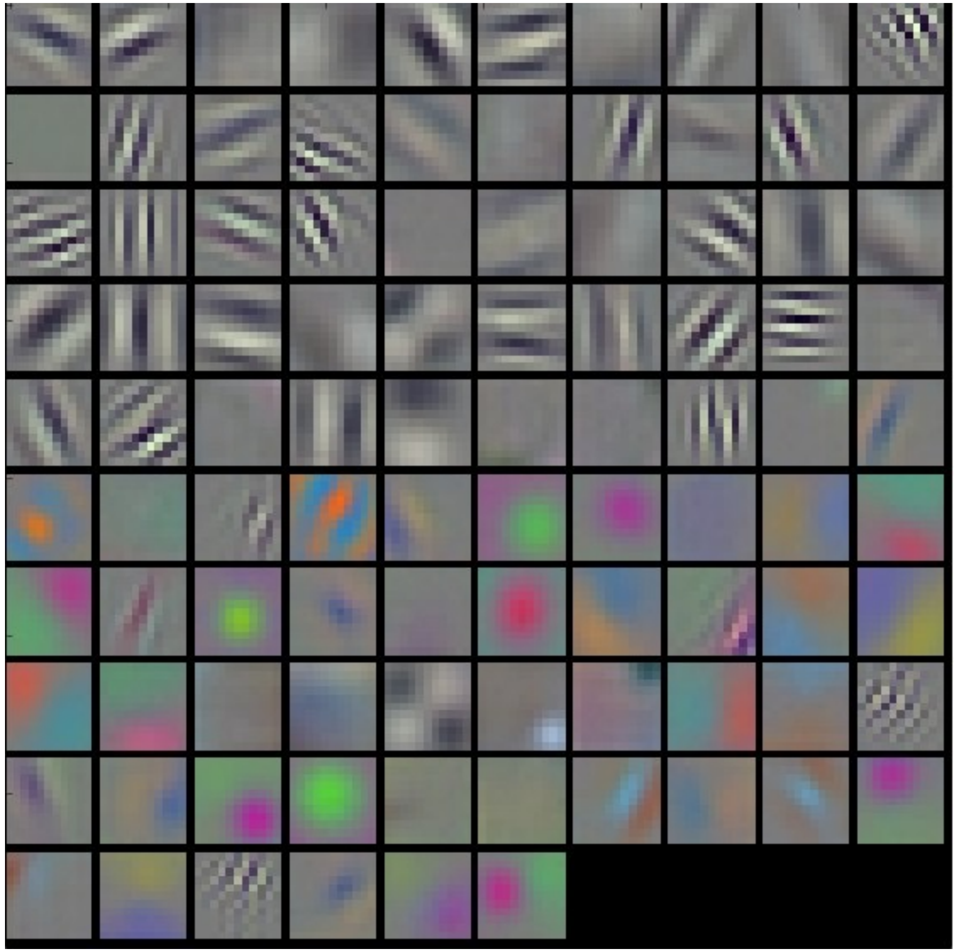
	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

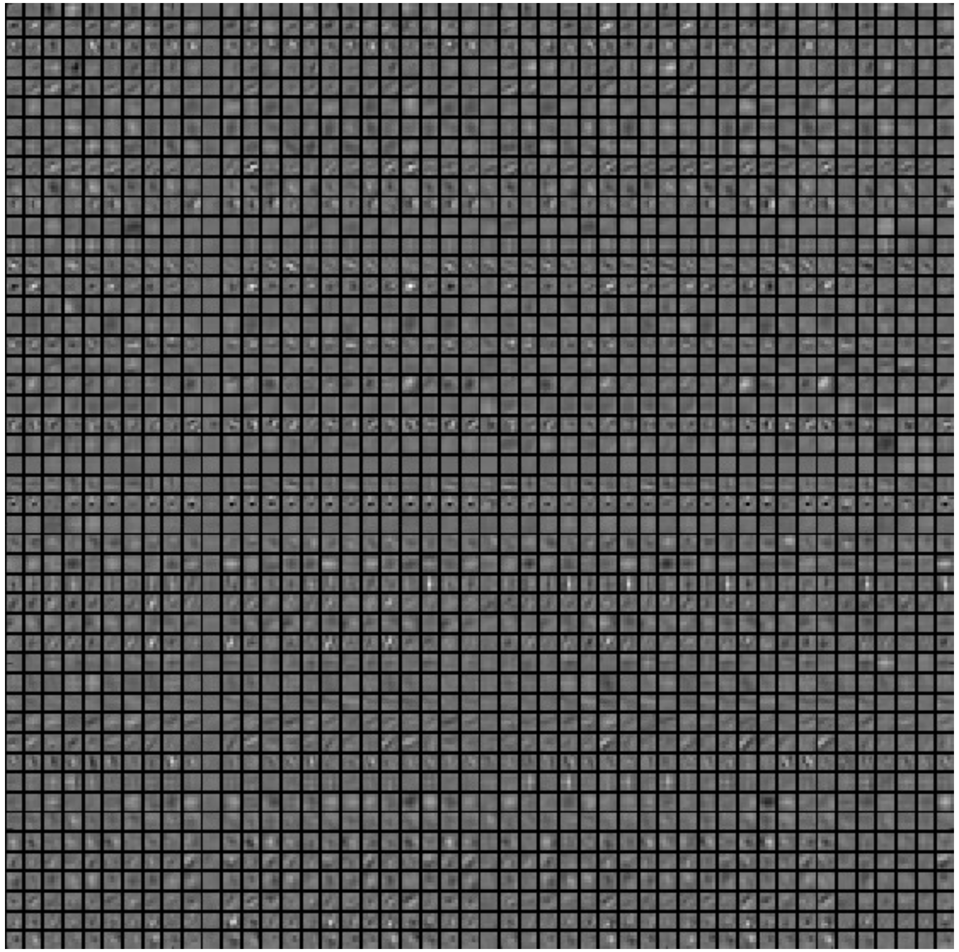
Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.4% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152. Details are in Table 2 and 4.

What do CCN see?

What do CNN see? Visualize the layers



AlexNet 1CONV layer

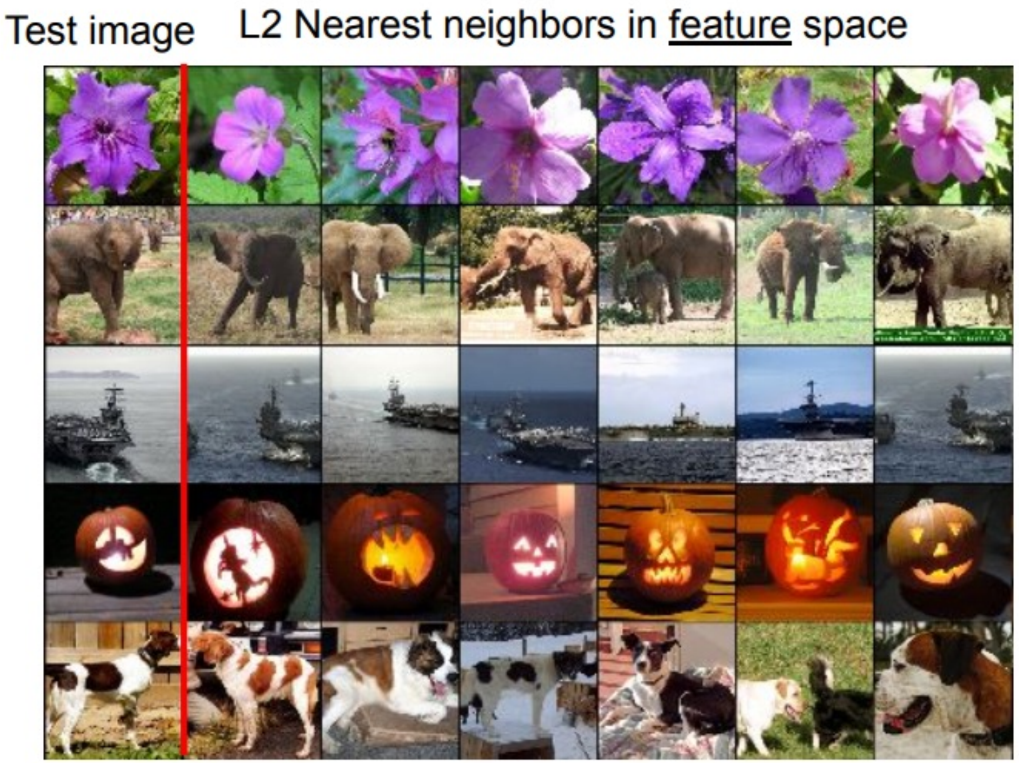


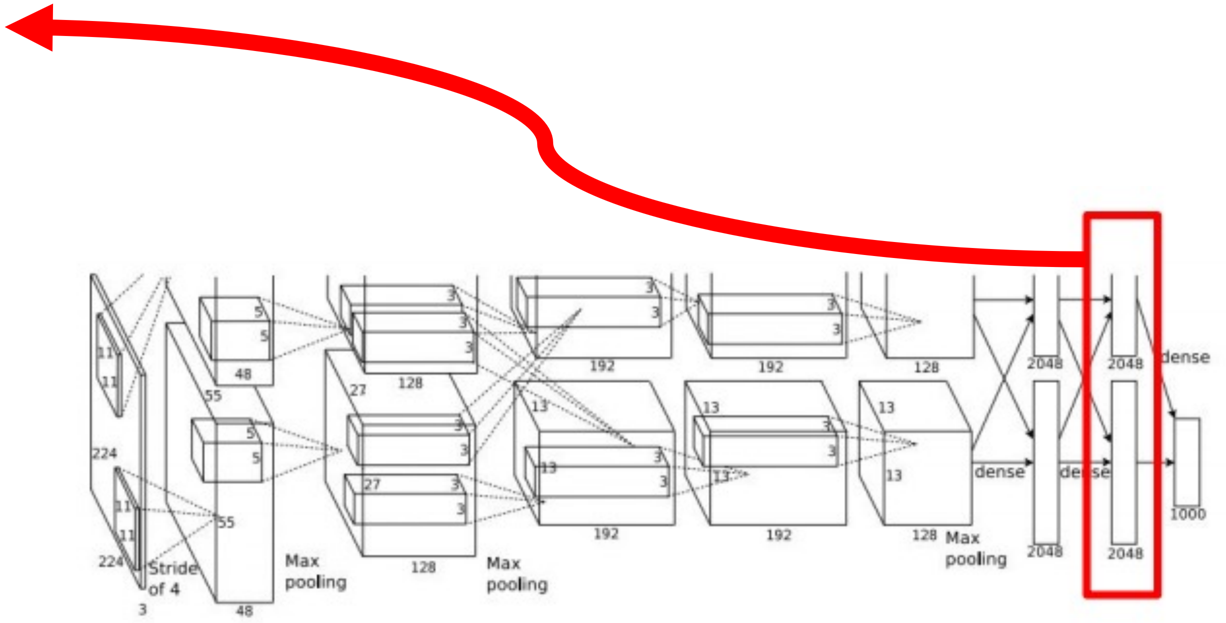
AlexNet 2CONV layer

From: http://cs231n.stanford.edu

What do CNN see? Embedding space for features

We can **consider k-nearest neighbors in embedding space for last FC layer:**



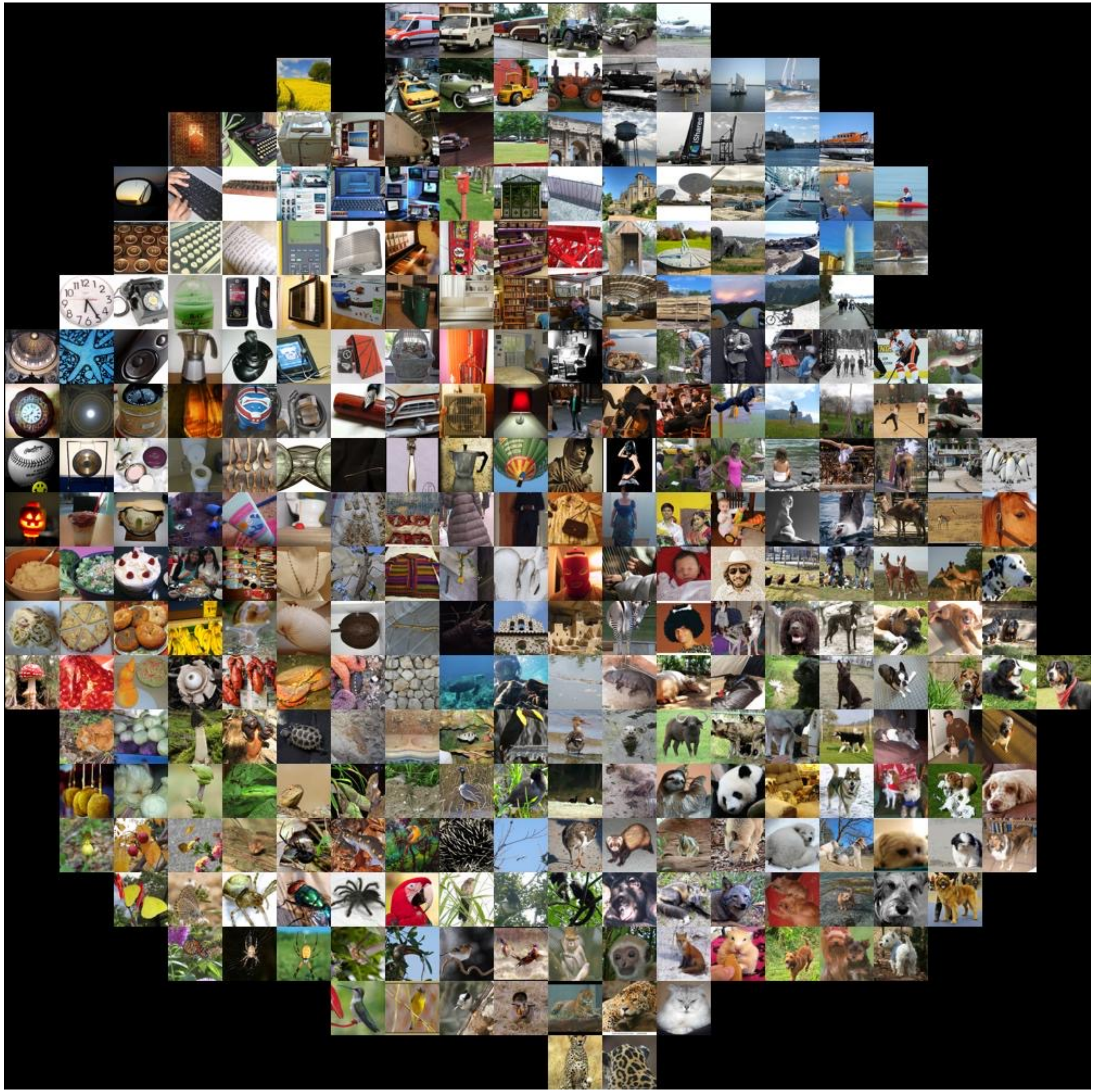


Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.

From: http://cs231n.stanford.edu

What do CNN see? Embedding space for features

We can **plot final FC embedding layer by means of dimensionality reduction**, e.g. tSNE (more powerful than PCA) or UMAP



Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

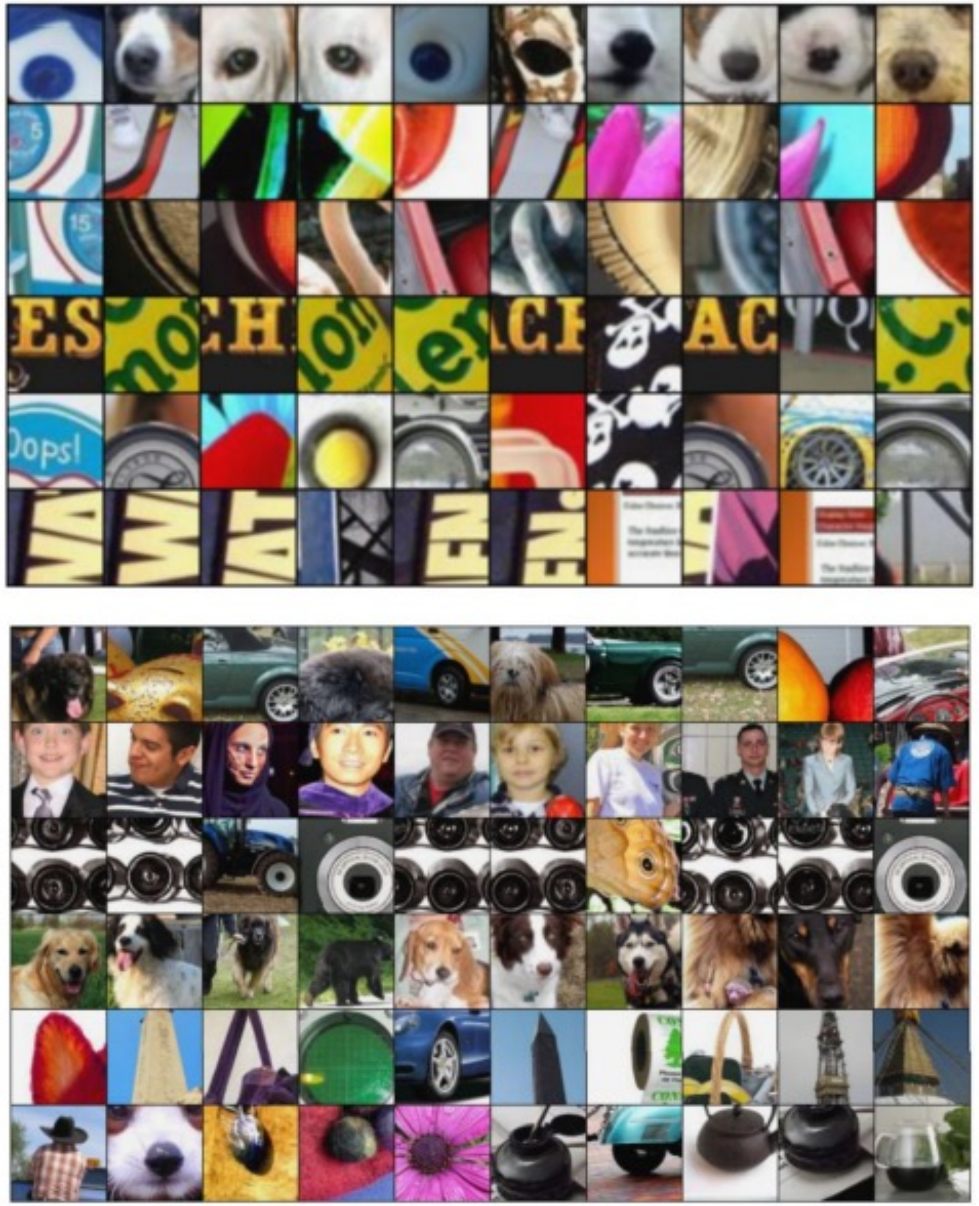
From: http://cs231n.stanford.edu

What do CNN see? Maximally activating neuros

We can **compute maximally activating patches.**

Run many images through the network, record values of chosen channel (e.g. channel 17/128 in conv5).

Visualize image patches that correspond to maximal activations.

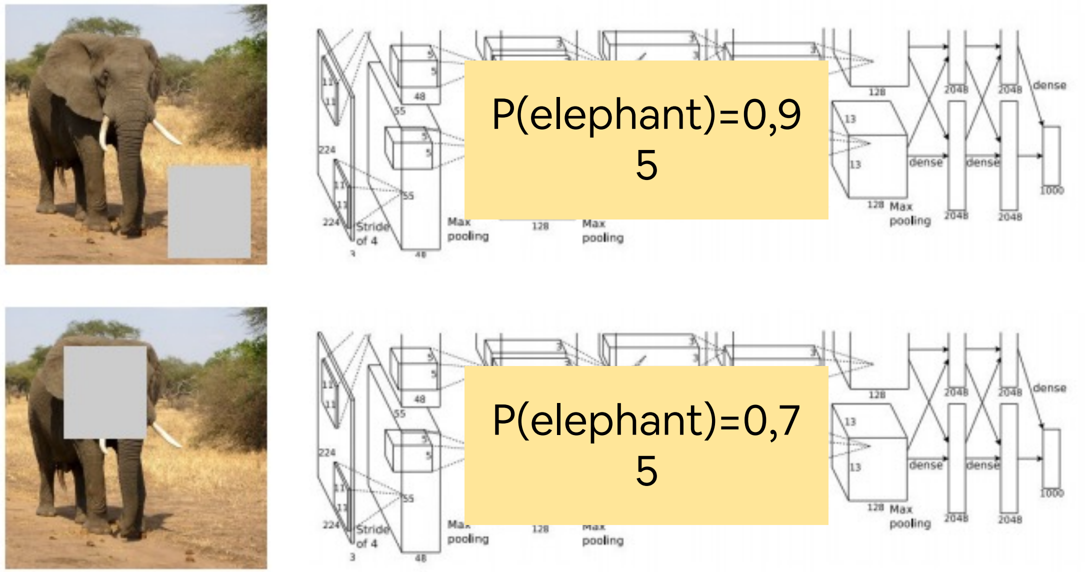


Springenberg et al, “Striving for Simplicity: The All Convolutional Net”, ICLR Workshop 2015 Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015;

From: http://cs231n.stanford.edu

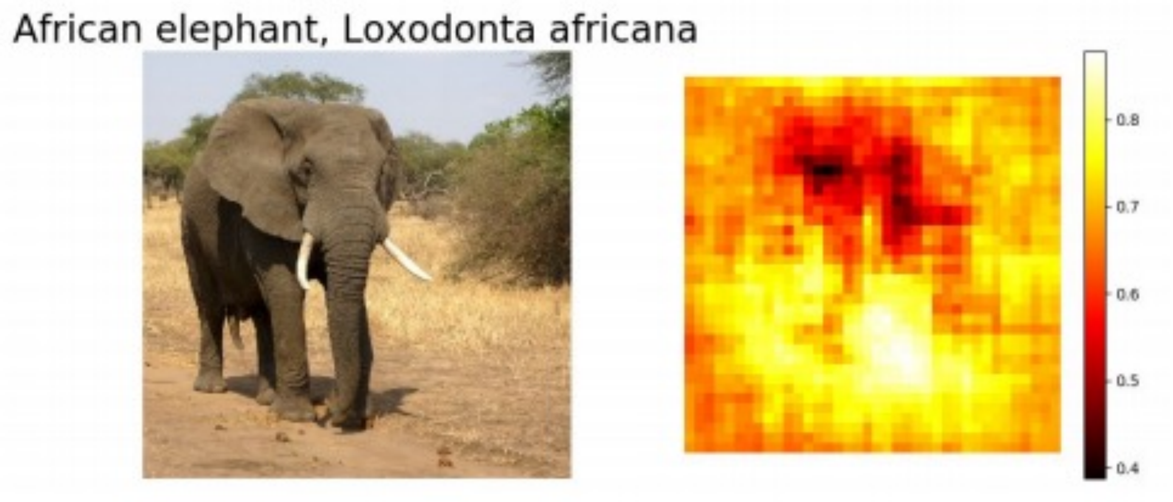
What do CNN see? Most relevant pixels

Saliency maps, e.g. by occlusion:



P(elephant)=0,95

P(elephant)=0,75



Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

From: http://cs231n.stanford.edu

Thank you!

Gian Antonio Susto