Séminaire grosses données - Université Gustave Eiffel

early makers

em lyon business school

Deep learning en pratique

Jean Savinien

15 décembre 2020

Outline

- What are Neural Nets?

  "What is it ?"

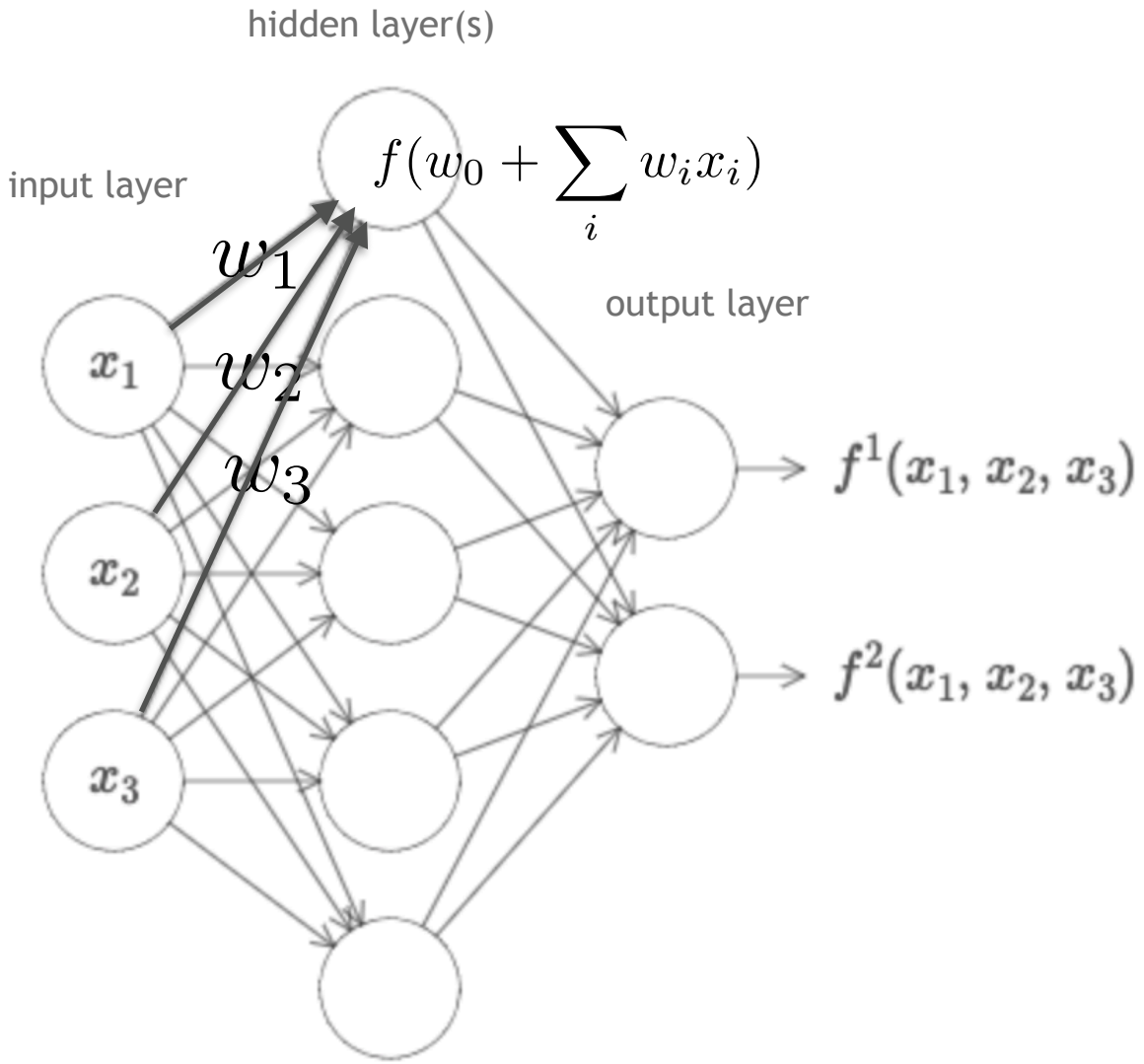
- Main Applications
  - Computer Vision
  - NLP
  - Tabular Data
  - Collaborative Filtering

  "What is it for ?"

- In Practice
  - Data Augmentation
  - Gradients Problems & Init
  - Learning Rates
  - Overfitting
  - GPUs & NN libraries

  "How do you use them ?"

Neural Networks

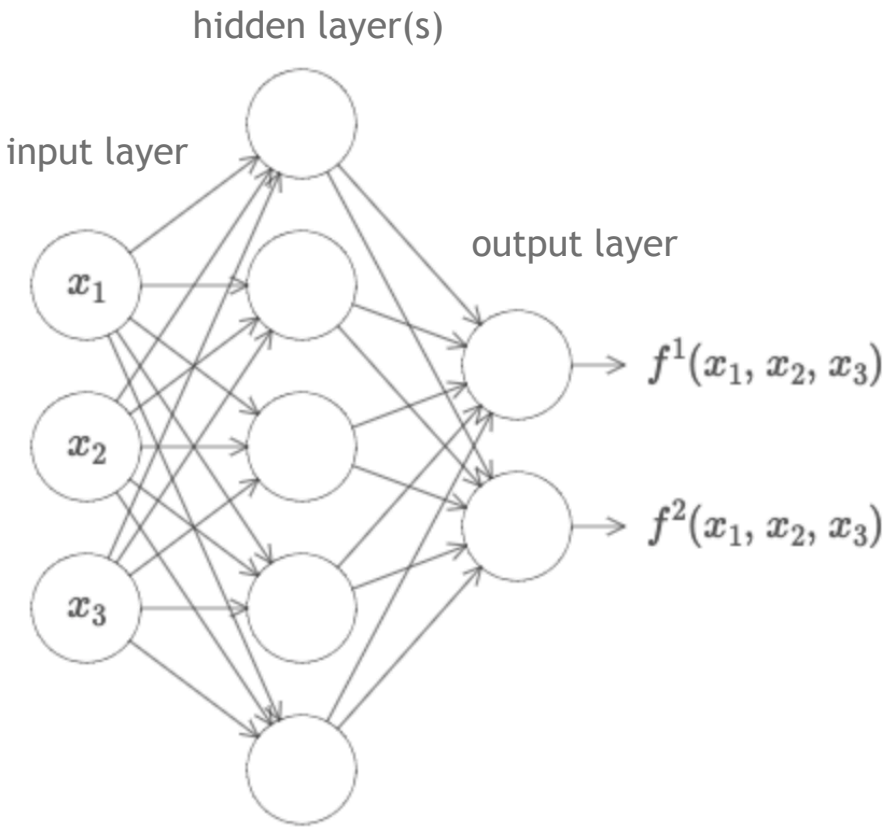


NN is ***deep*** when

#{ hidden layers }

is large

Universal Approximation Theorems

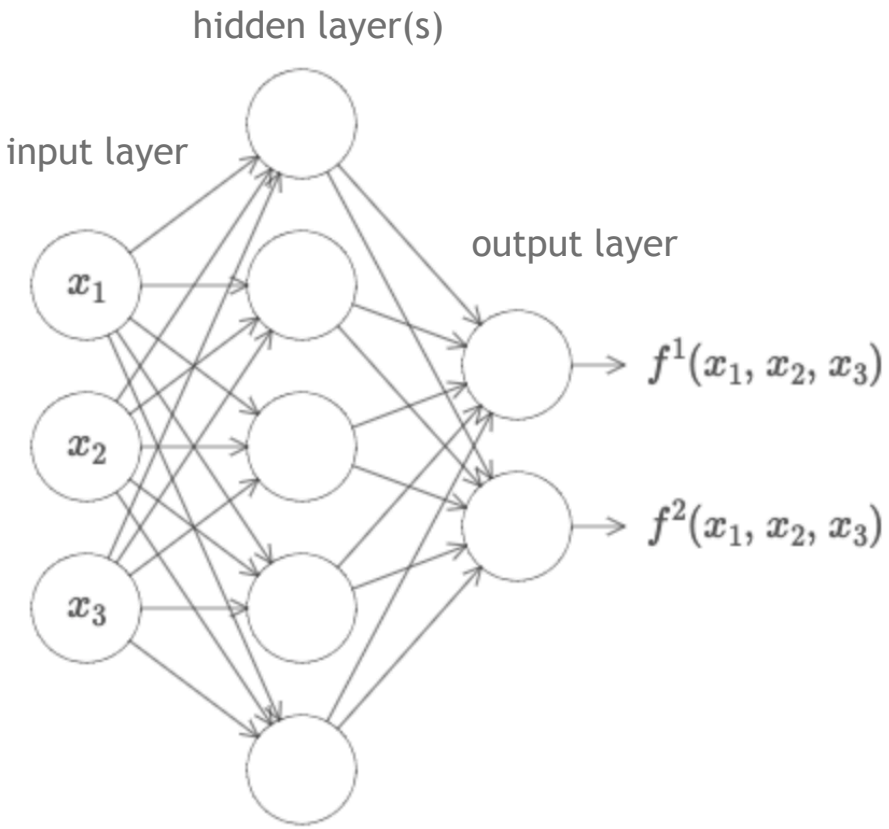


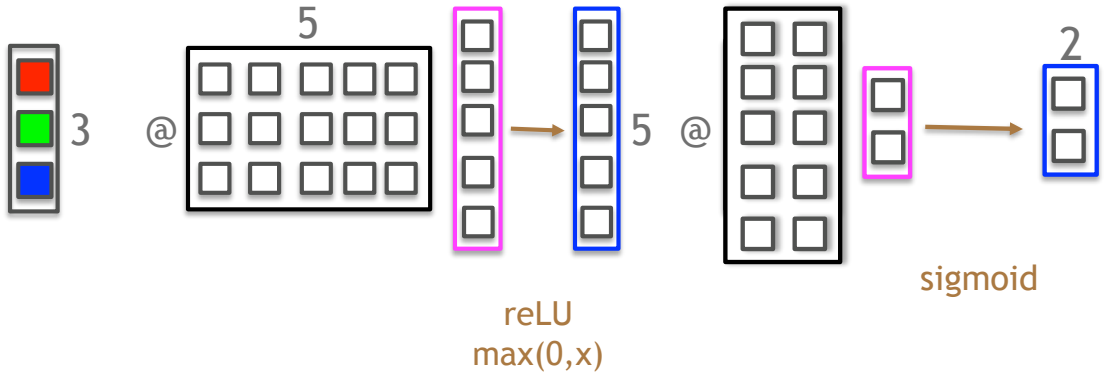
NN functions are dense in classes of functions of interest

eg:
NN of depth 1 with continuous activations are dense in $C^0(\mathbb{R}^n, \mathbb{R}^m)$ (compact topology).

ref *G. Sibenko '89, K. Hornik '91, … A. Kratsios 2020*

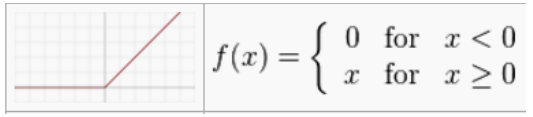
Neural Networks



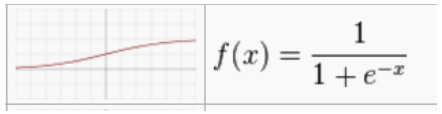


parameters (weights, biaises)

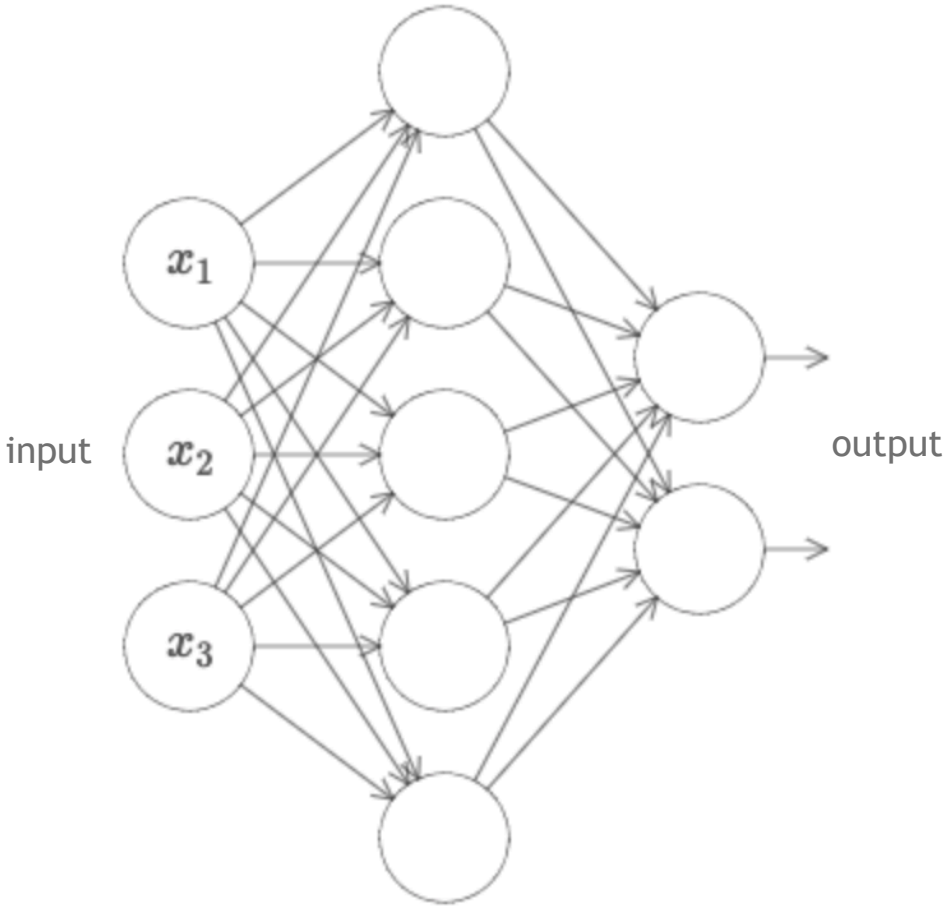
reLU, sigmoid activation functions



activations



Training



Loss $L(w_i\text{'s}) = \sum_{\text{inputs } x} l_x = \sum_{\text{batches } b} l_b$

learning rate

Gradient descend: $w_i \leftarrow w_i - \mathrm{lr}\frac{\partial L}{\partial w_i}$

performed “backwards” (*backprop*)

SGD: computations one batch at a time $w_i \leftarrow w_i - \mathrm{lr}\frac{\partial l_b}{\partial w_i}$

1 epoch : when all input/batches have been gone through

What are the NN Main Applications?

- Computer Vision
- NLP
- Tabular Data
- Collaborative Filtering

Main Applications - Computer Vision

single-label
classification

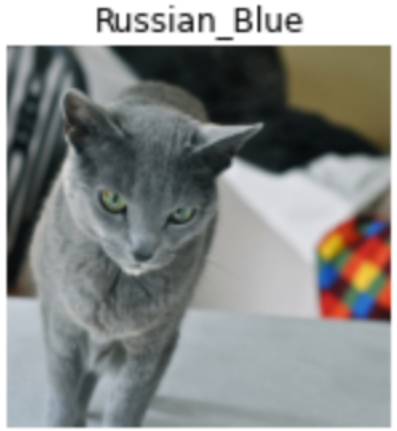
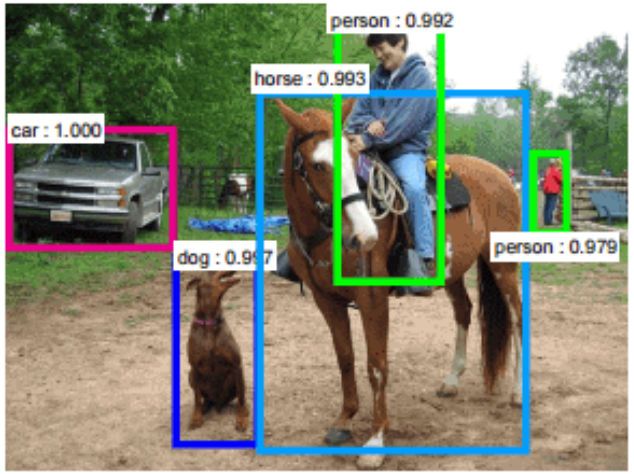


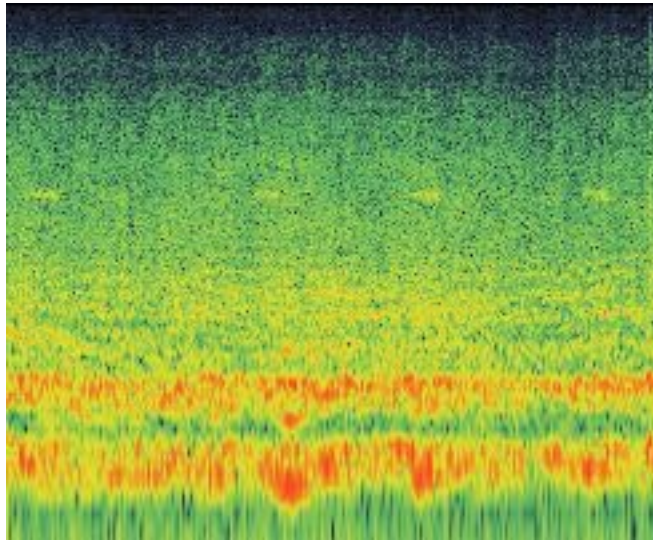
image segmentation

multi-label
classification
…
localization
object detection

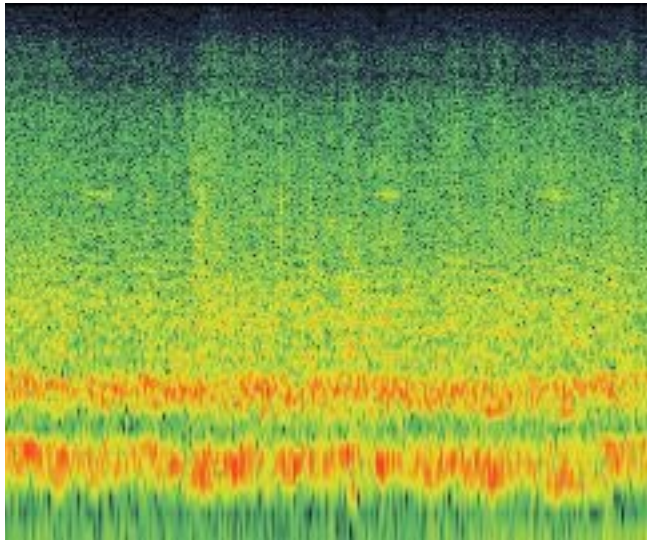


Convertion into image classification

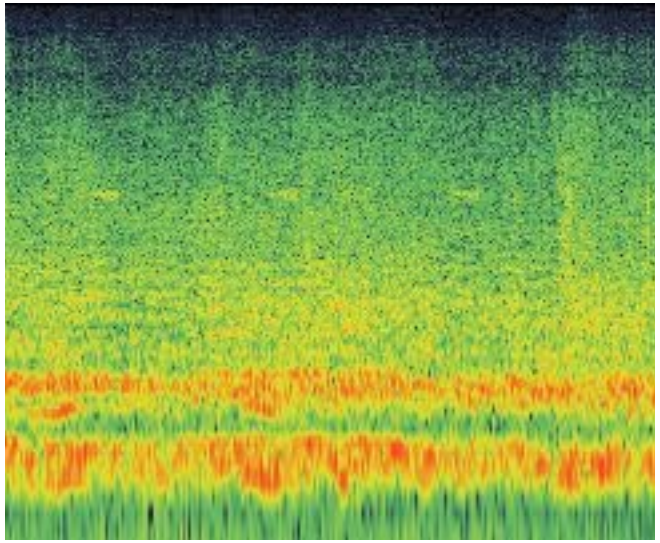
- **Detection of beehive's states…**



no queen



queen



swarming

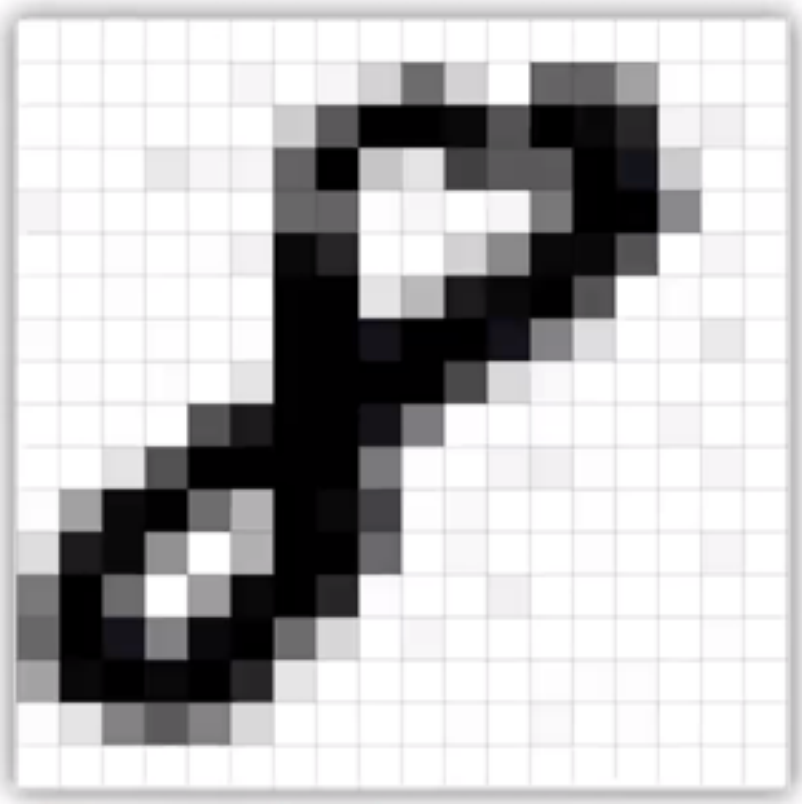
emlyon student's project with the makers lab (Ammar, Hadjur, Radisson, Savinien IOT2019)

early makers

em lyon business school

Main Applications - Computer Vision

Image recognition



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0	
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0	
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0	
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0	
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0	
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0	
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0	
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0	
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0	
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0	
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0	
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0	
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0	
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0	
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0	
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Number	Prob
0	0.01
1	0.03
2	0.03
3	0.04
4	0.07
5	0.08
6	0.10
7	0.02
8	0.59
9	0.01

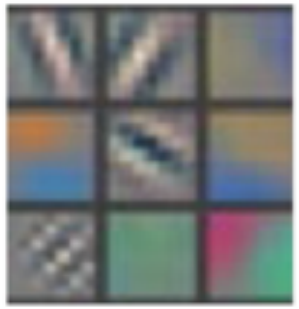
CNN - visualizing activations in CONVnets

Visualizing and Understanding Convolutional Networks

Matthew D. Zeiler and Rob Fergus

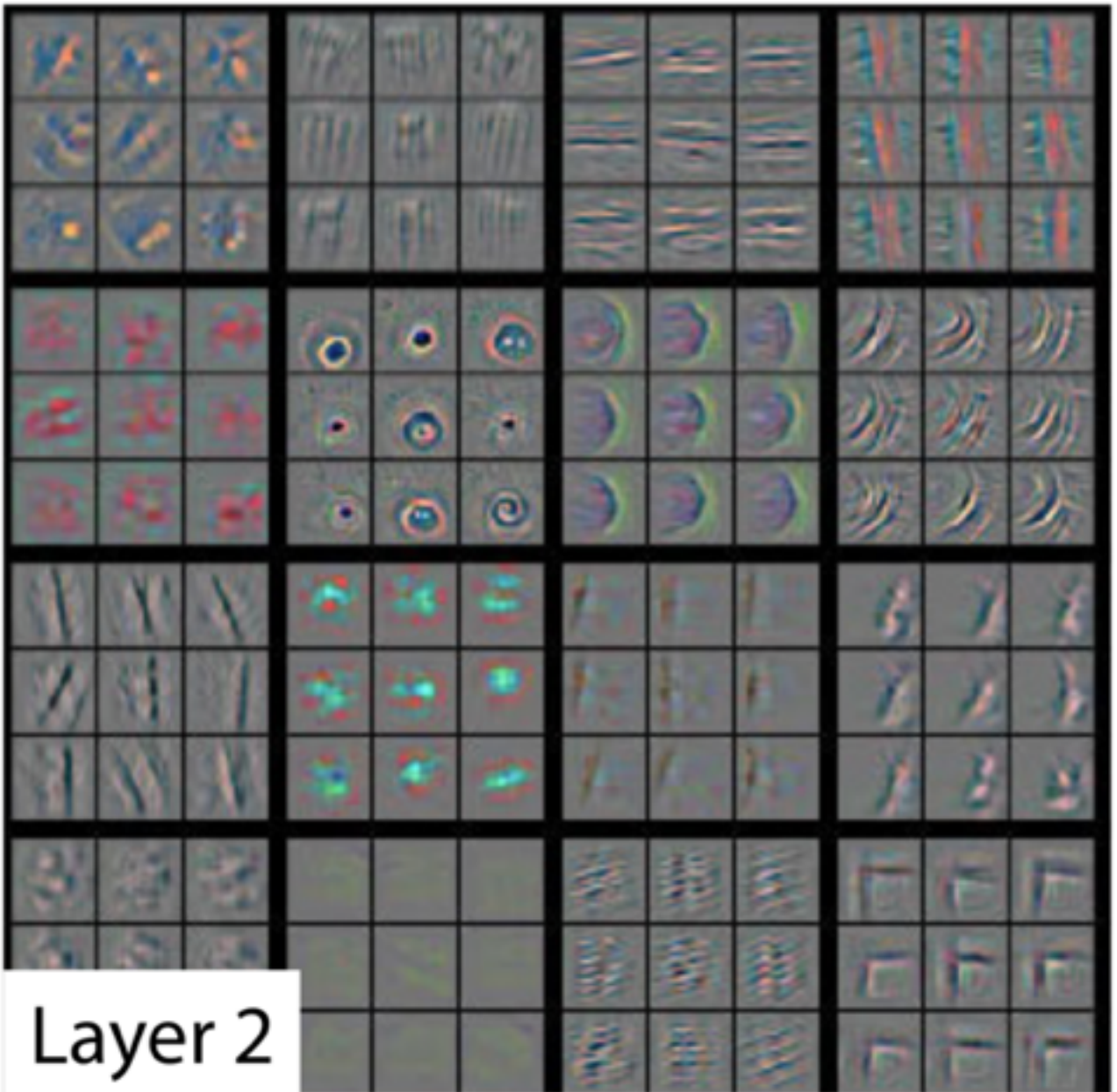
Dept. of Computer Science,
New York University, USA
{zeiler,fergus}@cs.nyu.edu

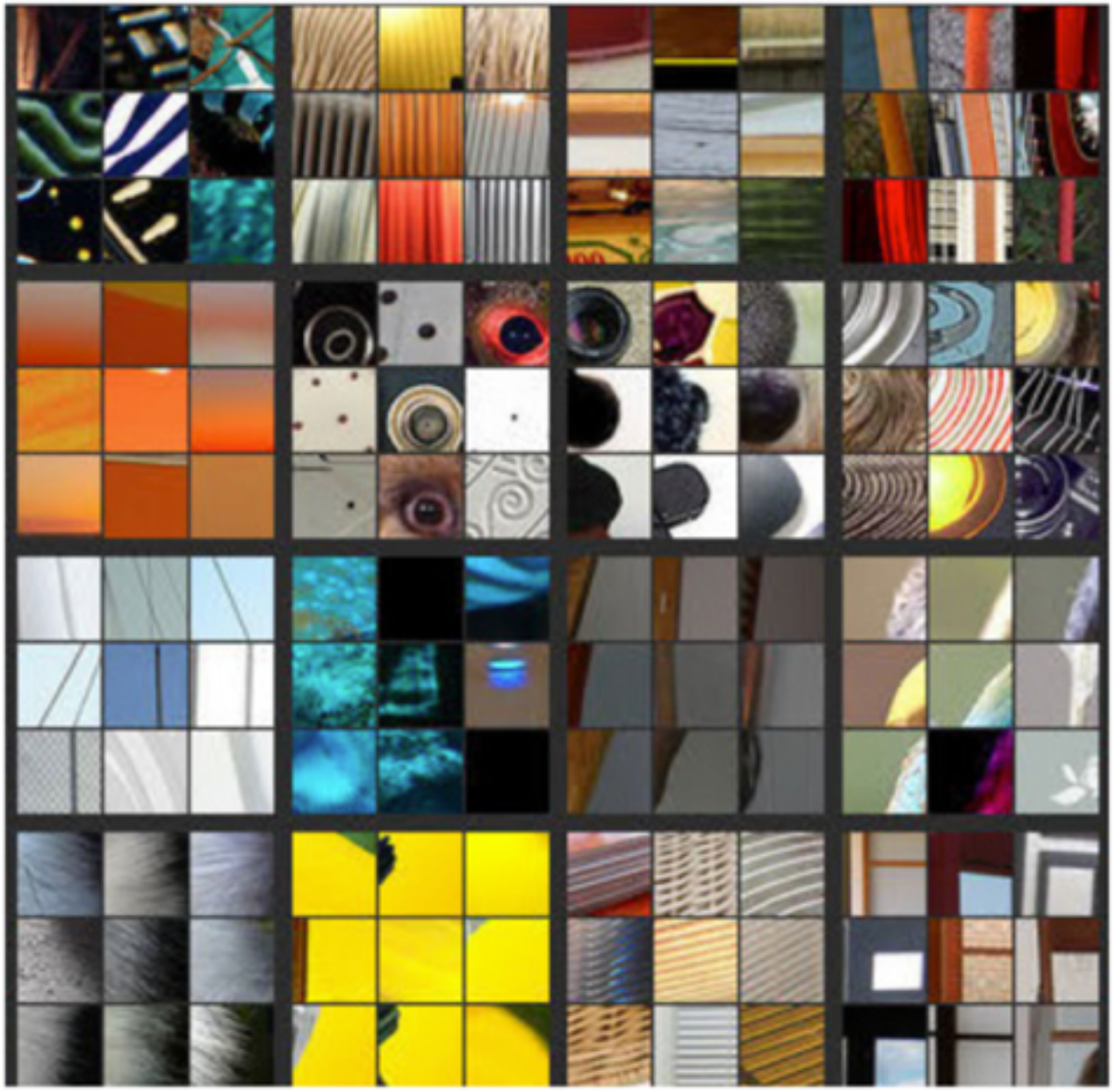
ECCV 2014



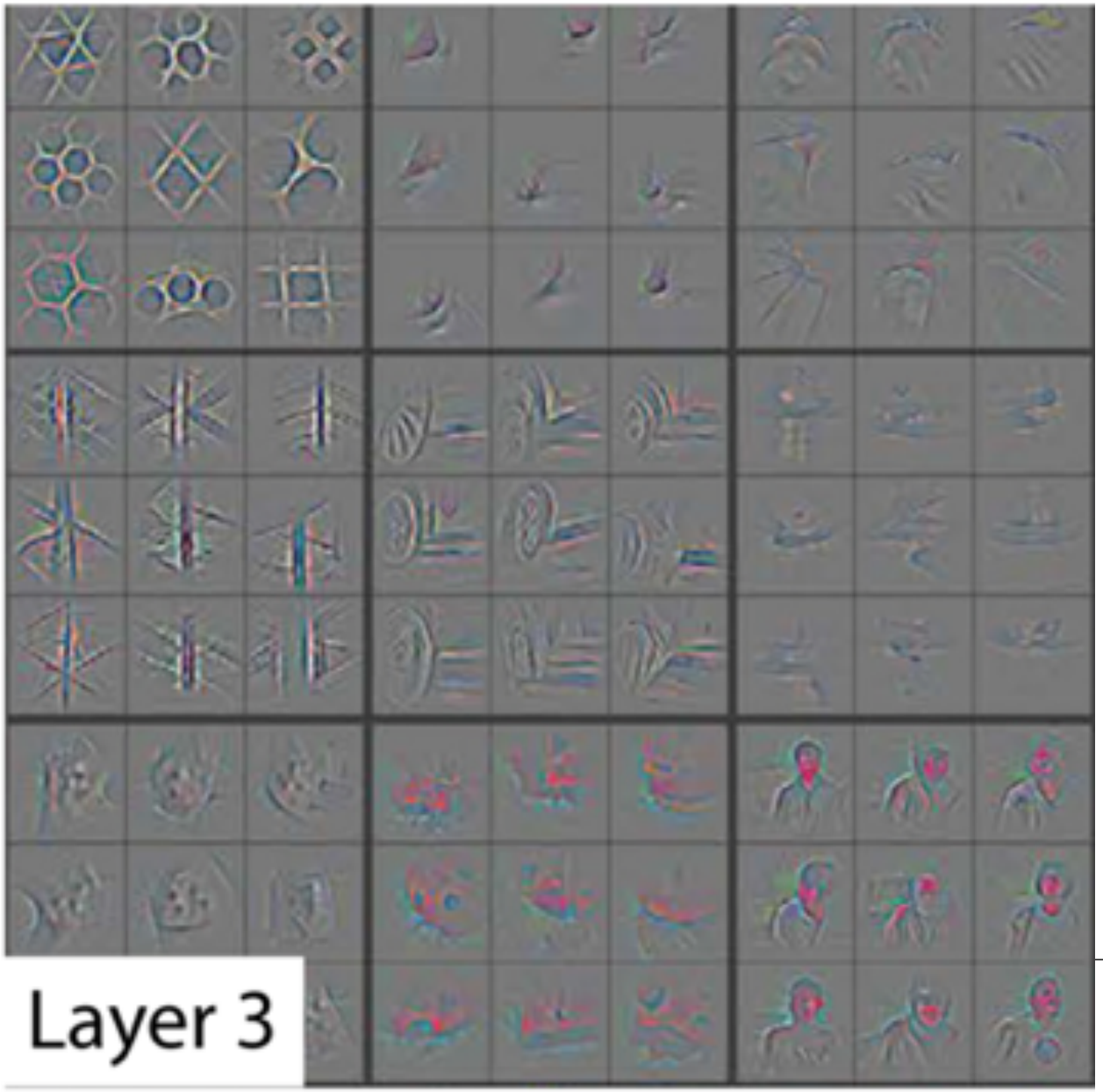
Layer 1

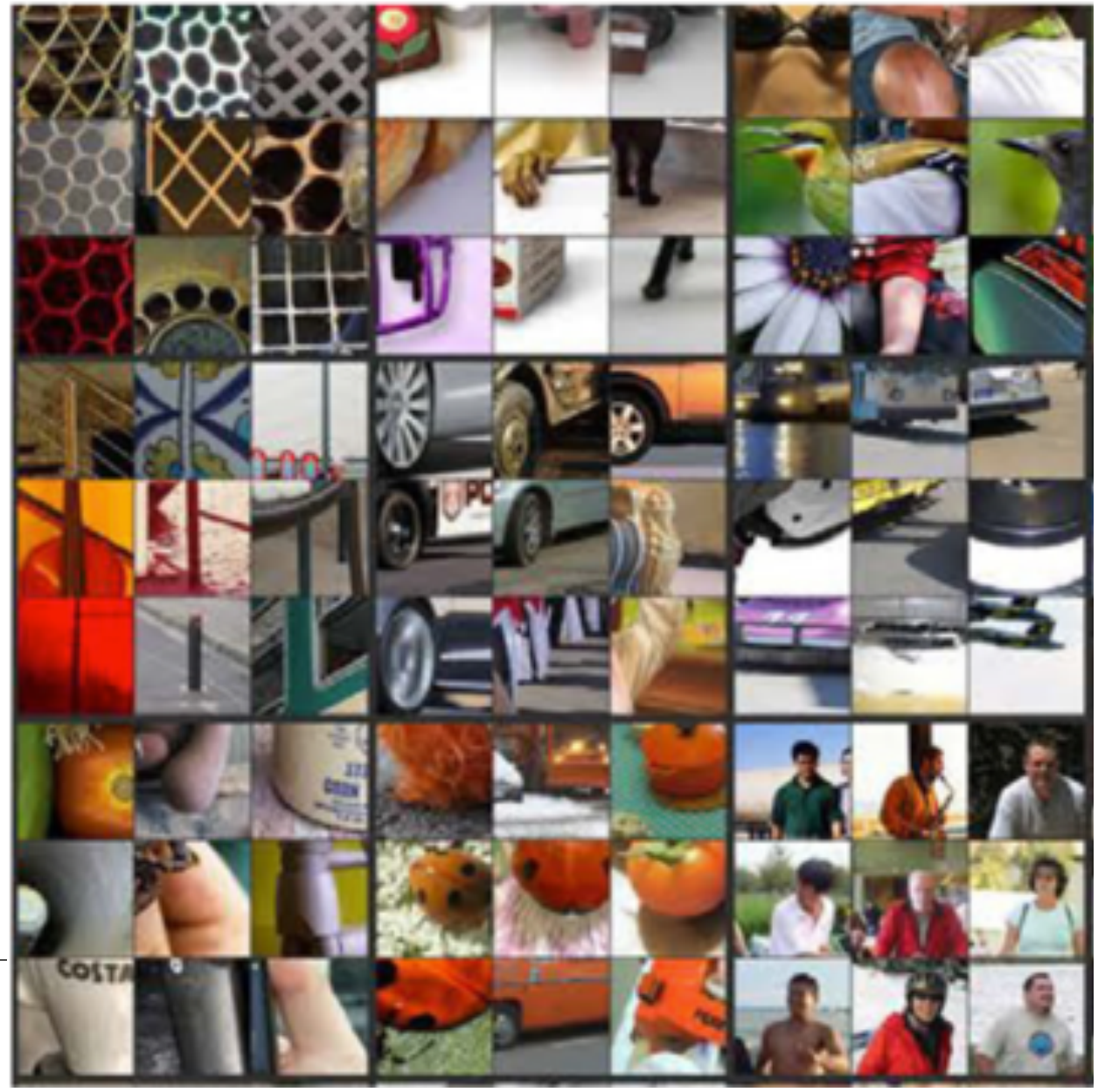
CNN - visualizing activations in CONVnets



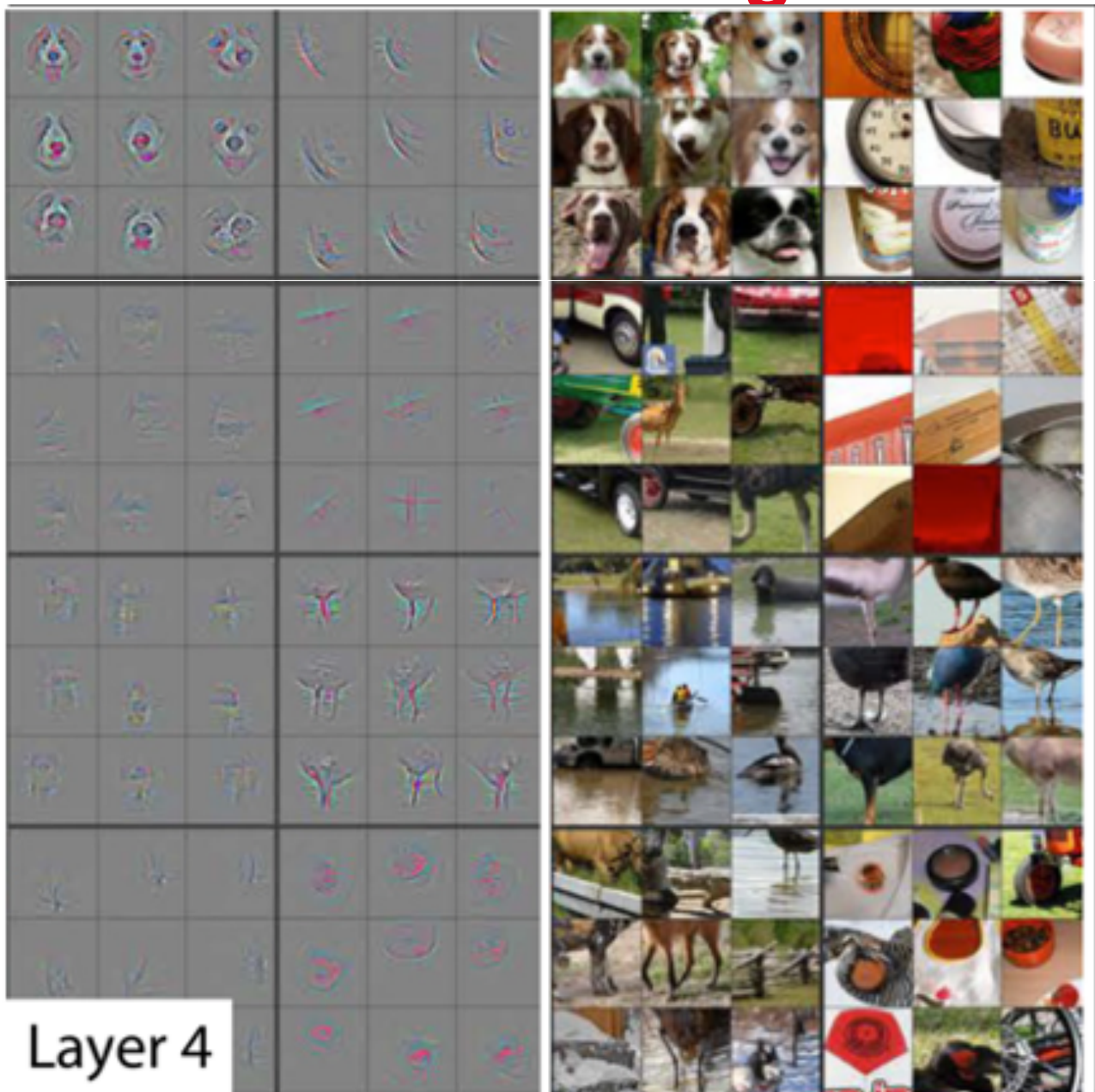


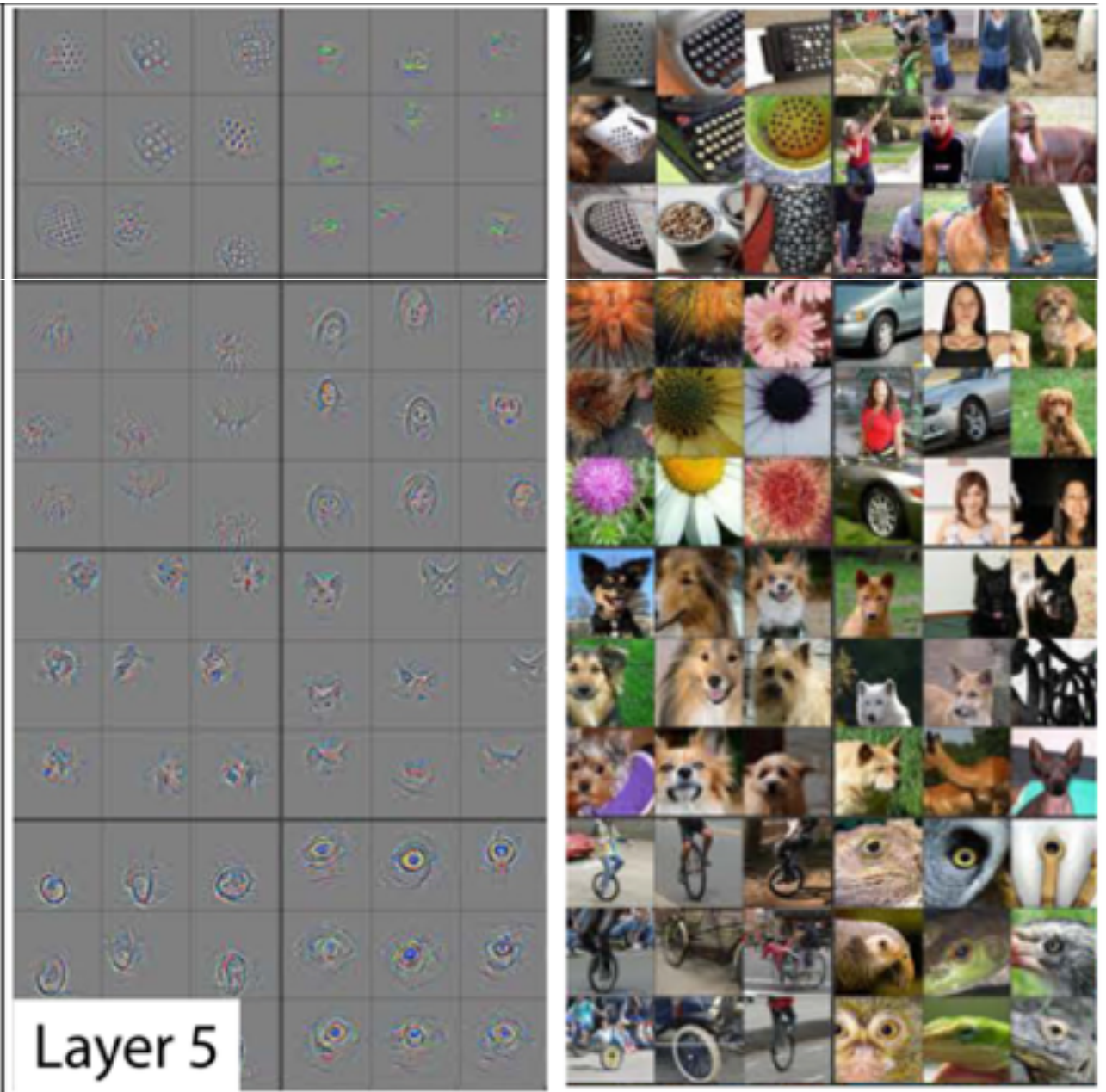
CNN - visualizing activations in CONVnets



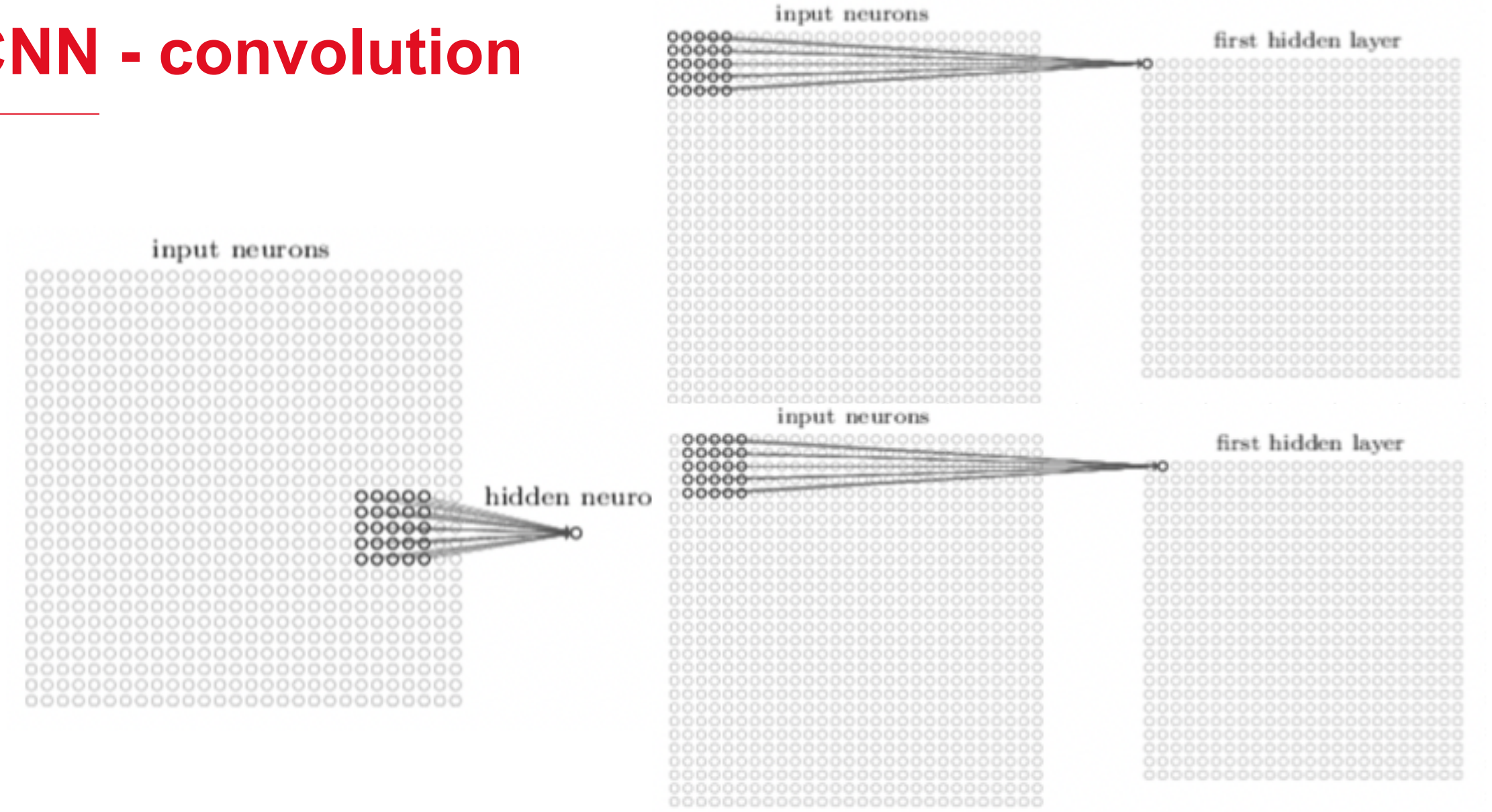


CNN - visualizing activations in CONVnets





CNN - convolution

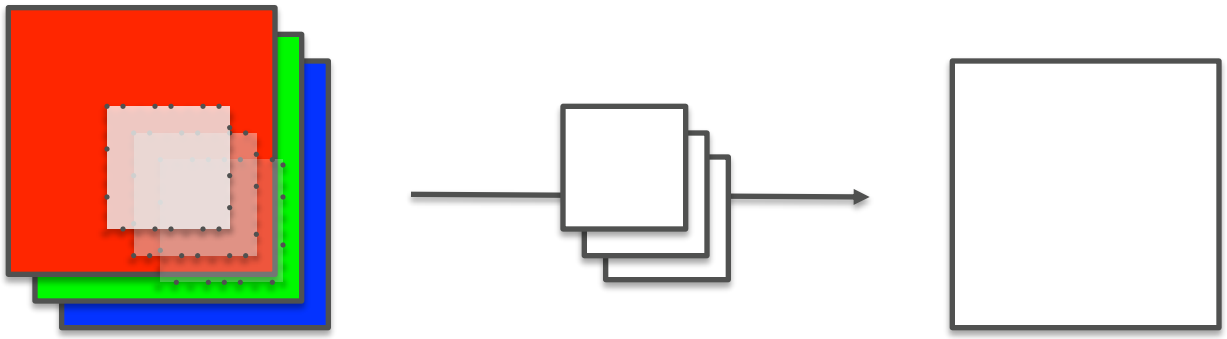


CNN - feature maps

input image

kernel

channel output



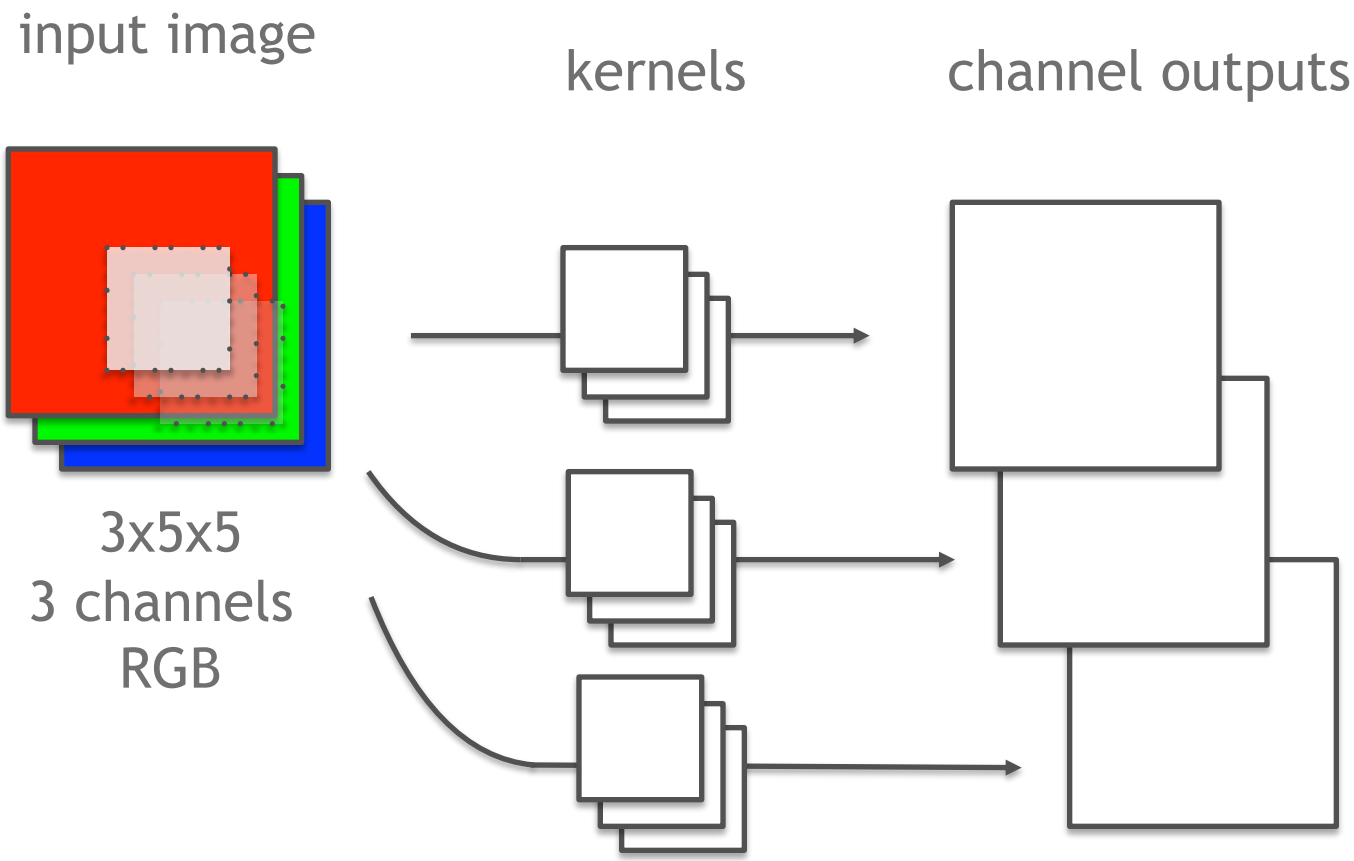
3x5x5
3 channels
RGB

3x3x3

5x5

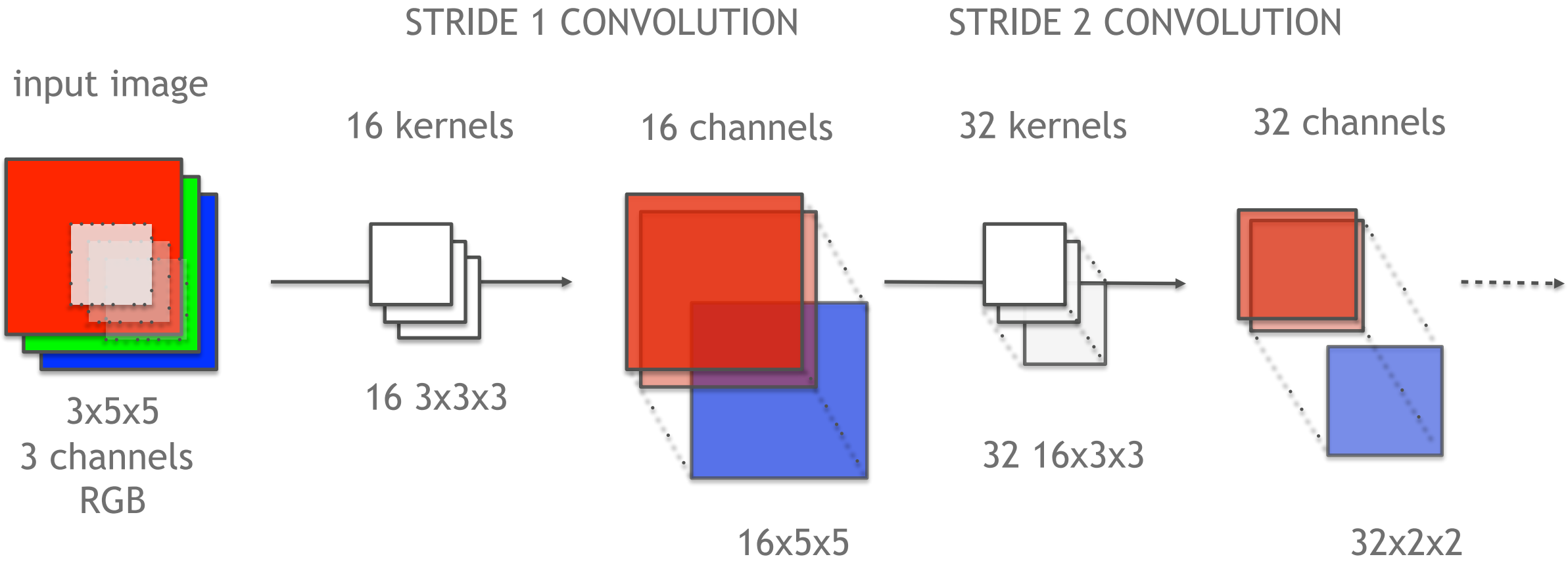
learning 1 elementary pattern,
e.g. edge, color gradient, etc.

CNN - feature maps



learning 3 elementary patterns

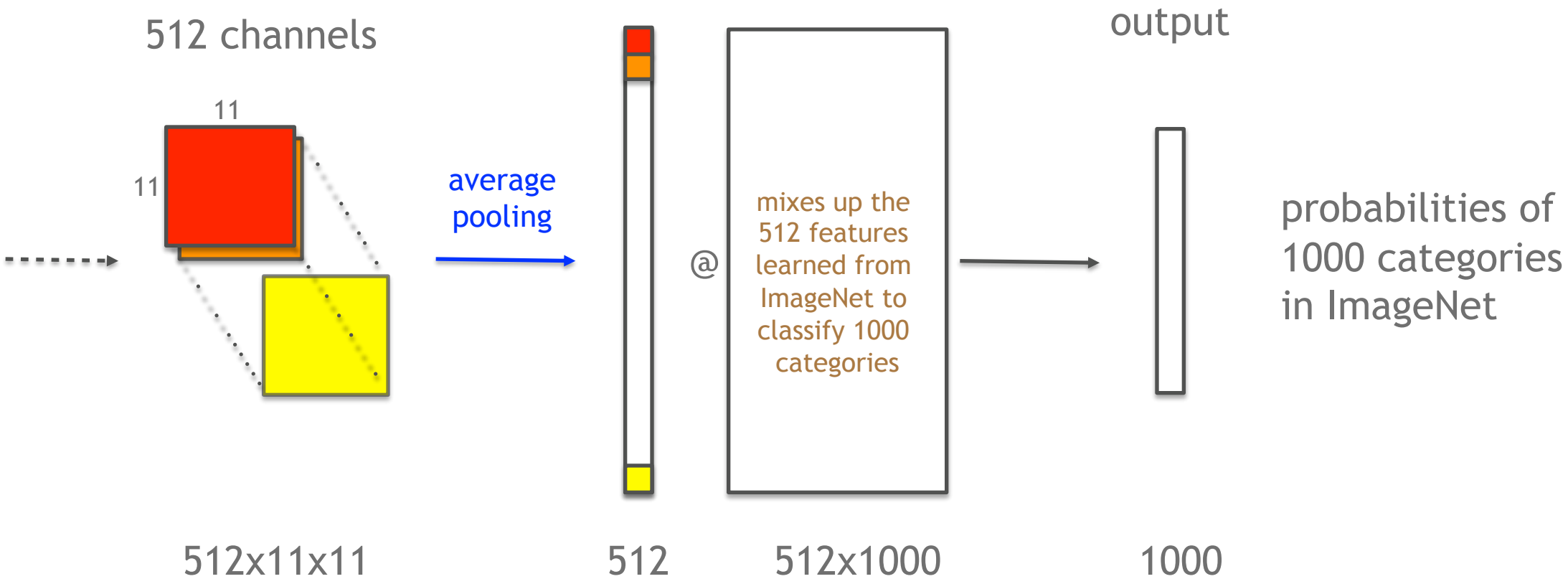
CNN - CONVnet



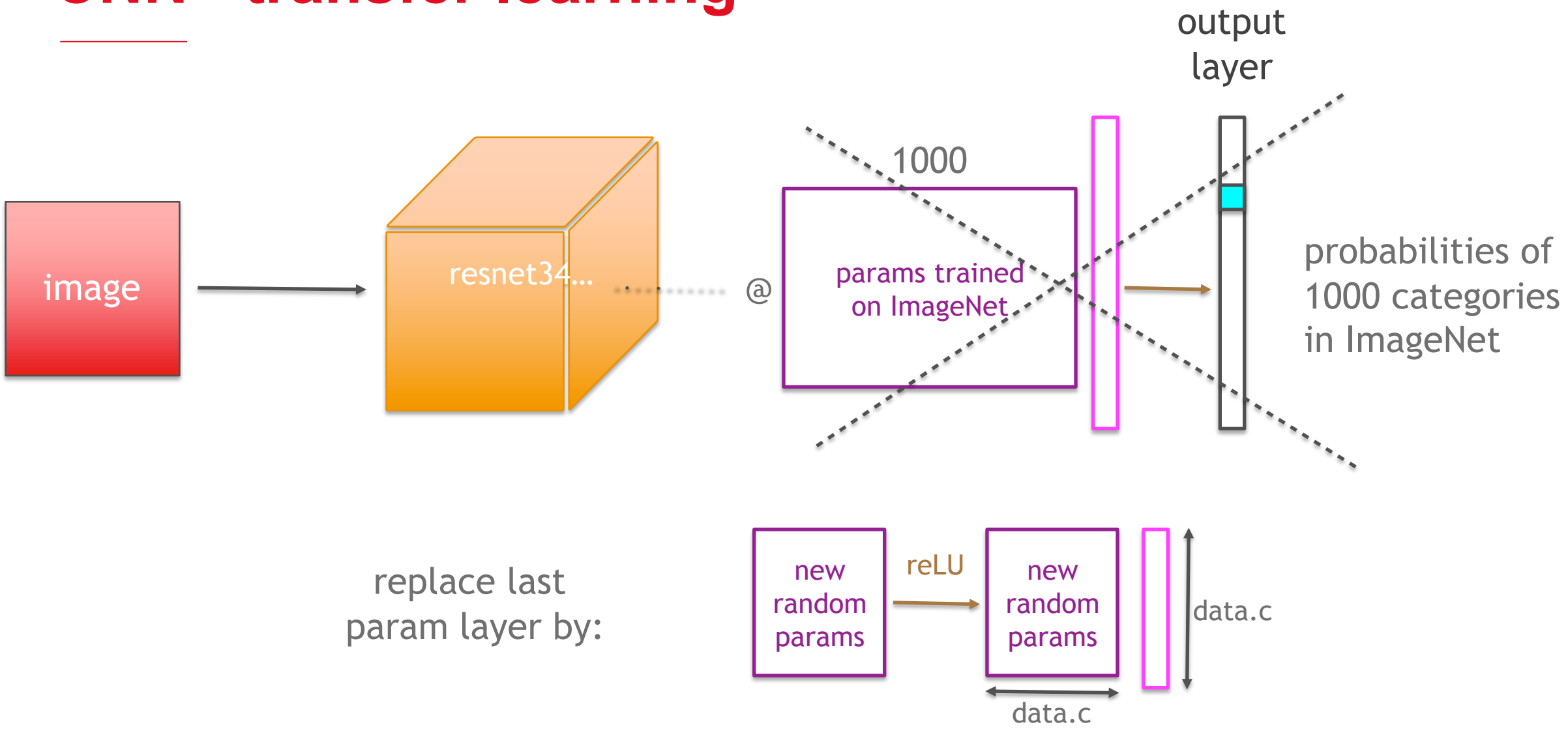
halves picture sizes
& doubles channels

CNN - CONVnet

alternate layers of stride 1 & stride 2 convs, to end up with 512 channels

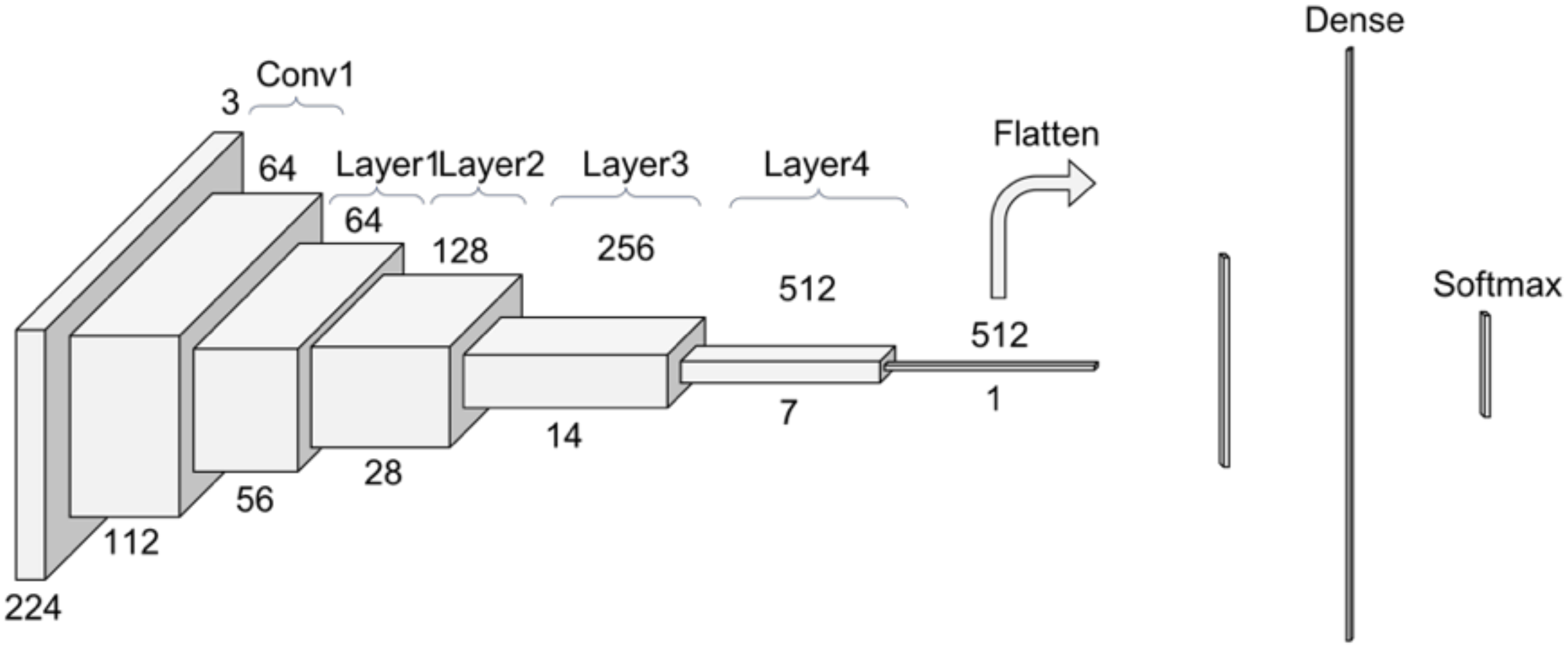


CNN - transfer learning

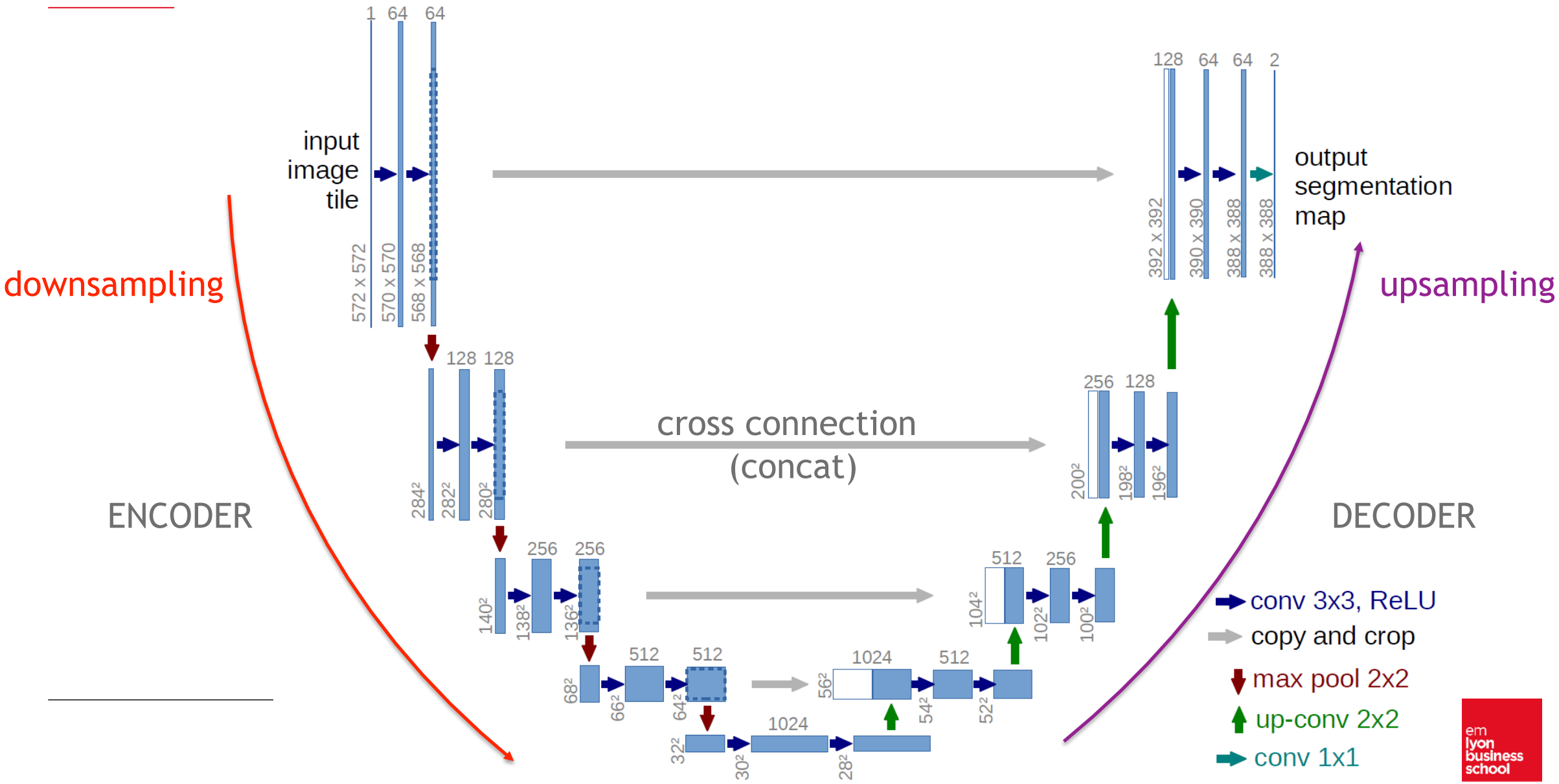


CNN - resnets

alternate layers of stride 1 & stride 2 convs, to end up with 512 channels



CNN - Unets


input image tile
output segmentation map
downsampling
upsampling
ENCODER
DECODER
cross connection (concat)
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

CNN - resnets

Deep Residual Learning for Image Recognition

IEEE CVPR 2016

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun

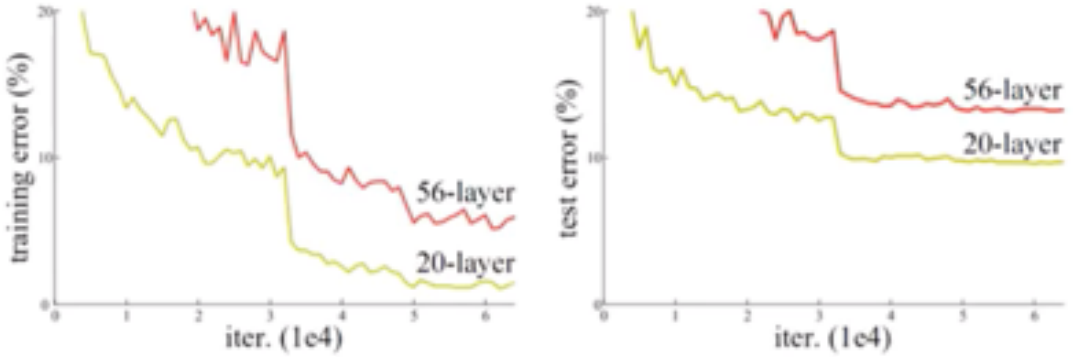


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

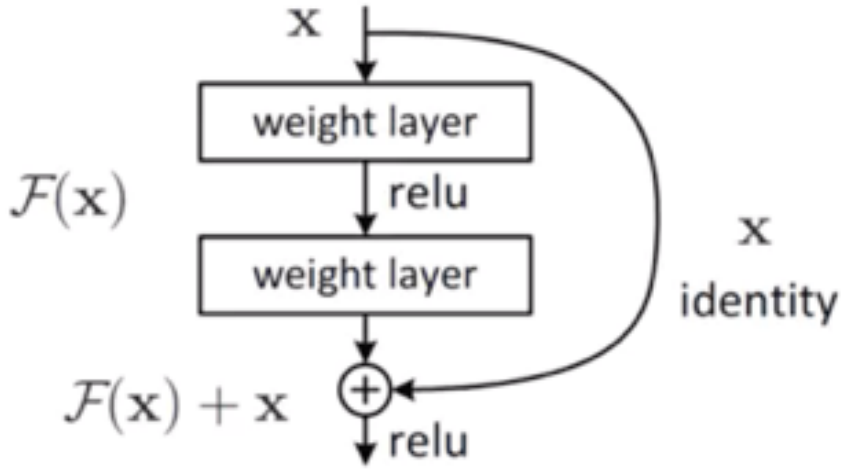
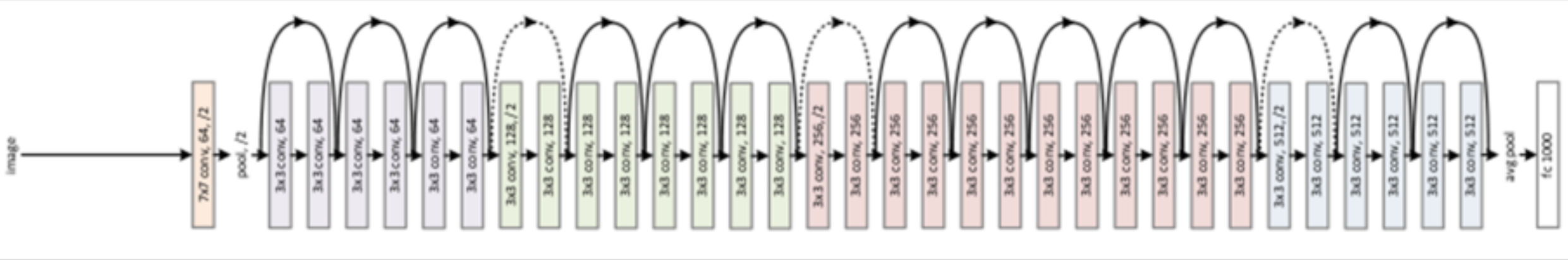


Figure 2. Residual learning: a building block.

output = x + conv2(conv1(x))

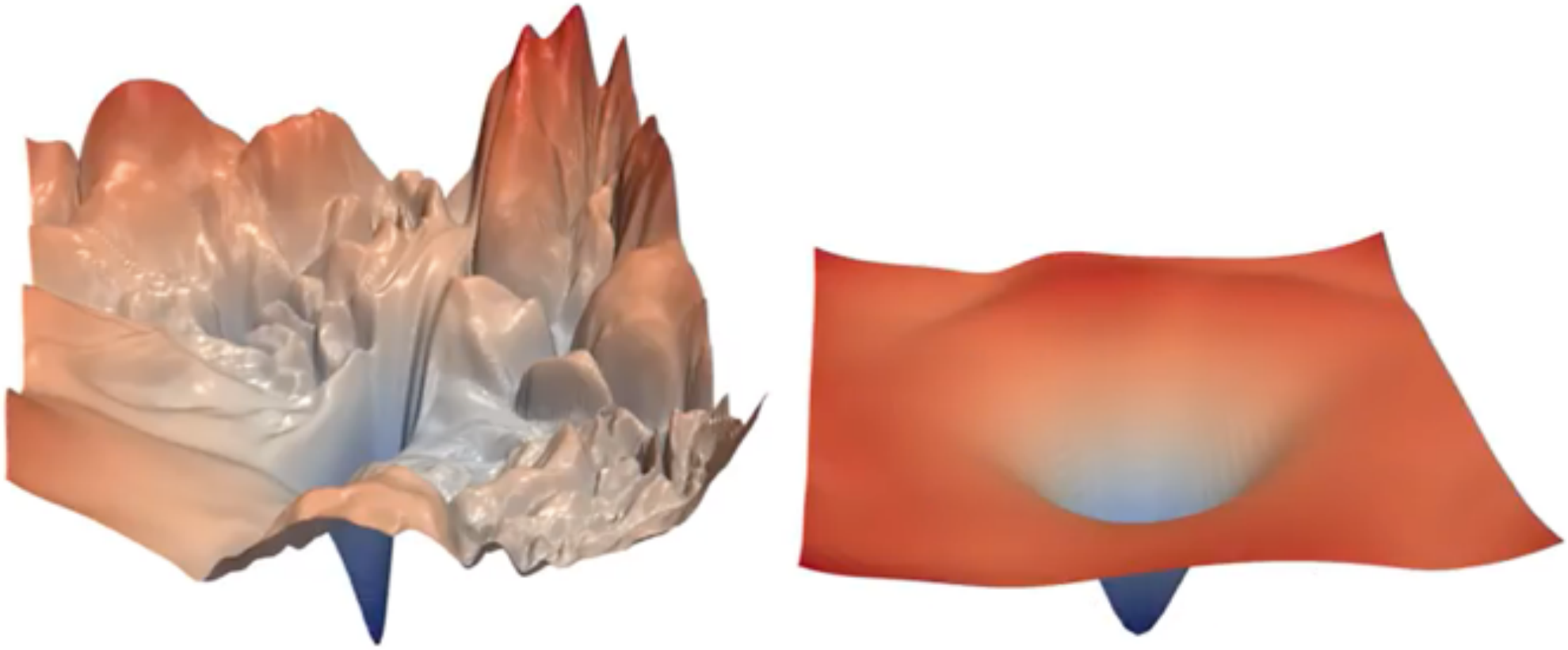


CNN - resnets

Visualizing the Loss Landscape of Neural Nets

NIPS'18

Hao Li[1], Zheng Xu[1], Gavin Taylor[2], Christoph Studer[3], Tom Goldstein[1]



(a) without skip connections

(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

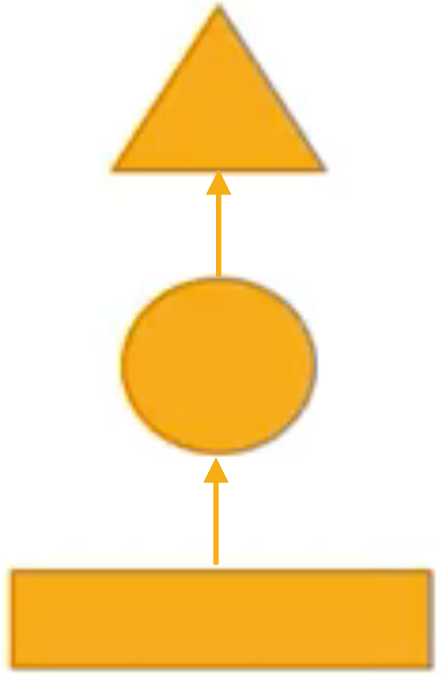
Main Applications - NLP

See Julien's talk :)

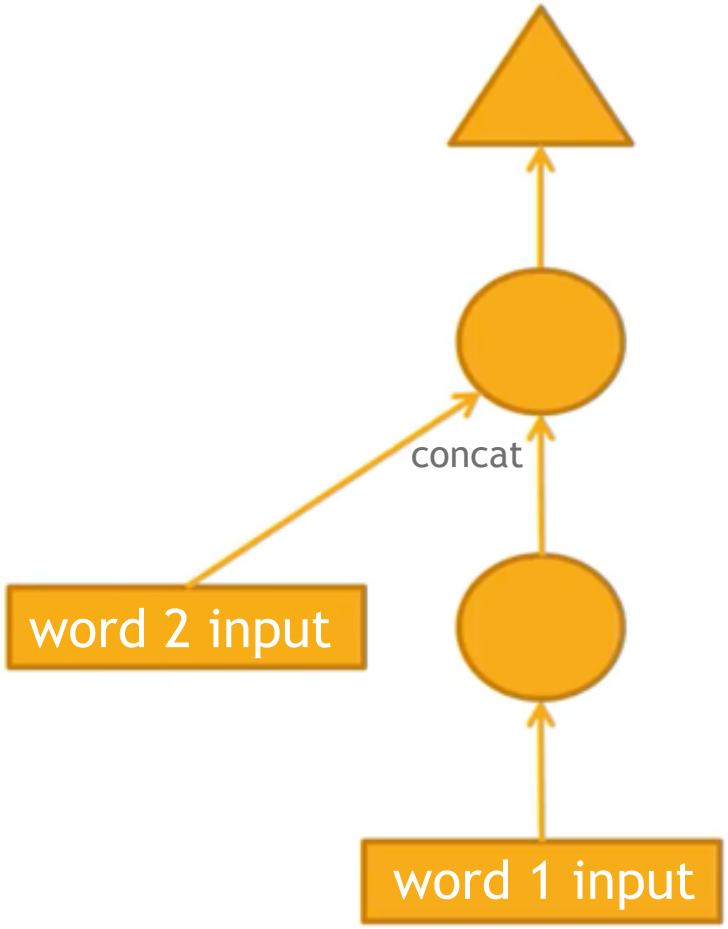
Recurrent Neural Nets (RNNs)

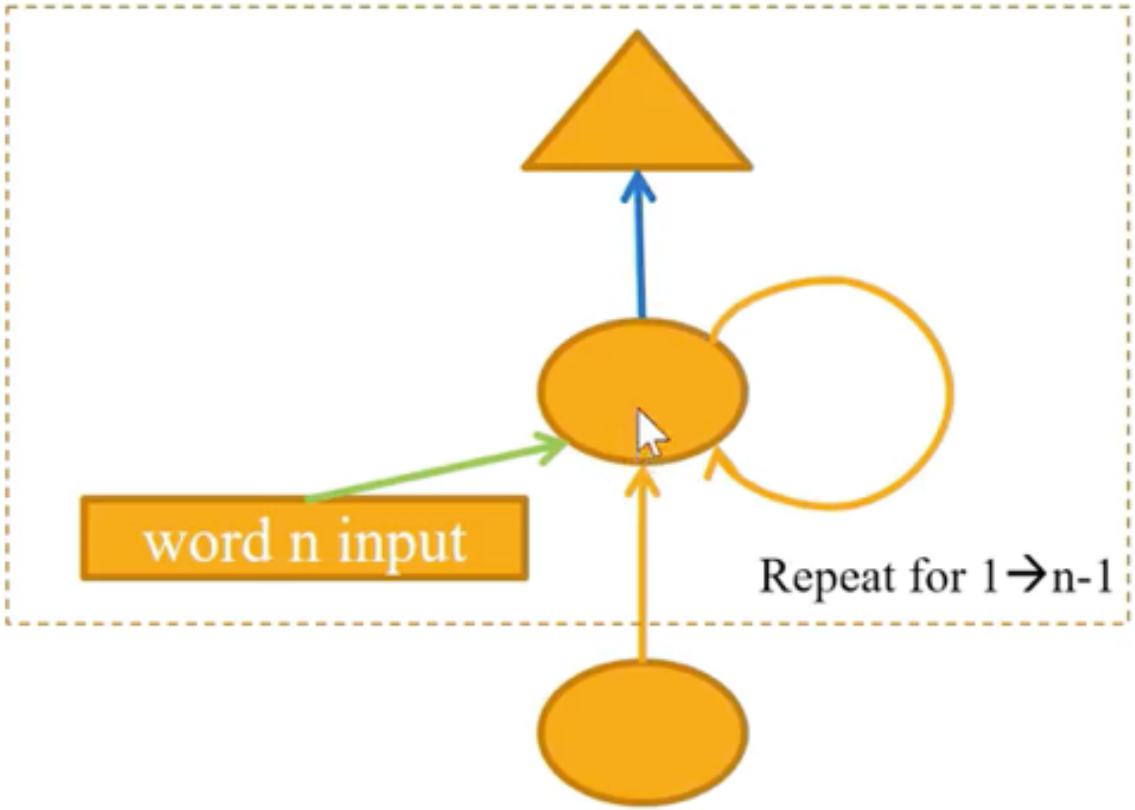
What is an RNN?

essentially a *fully connected NN*
trained in a specific way

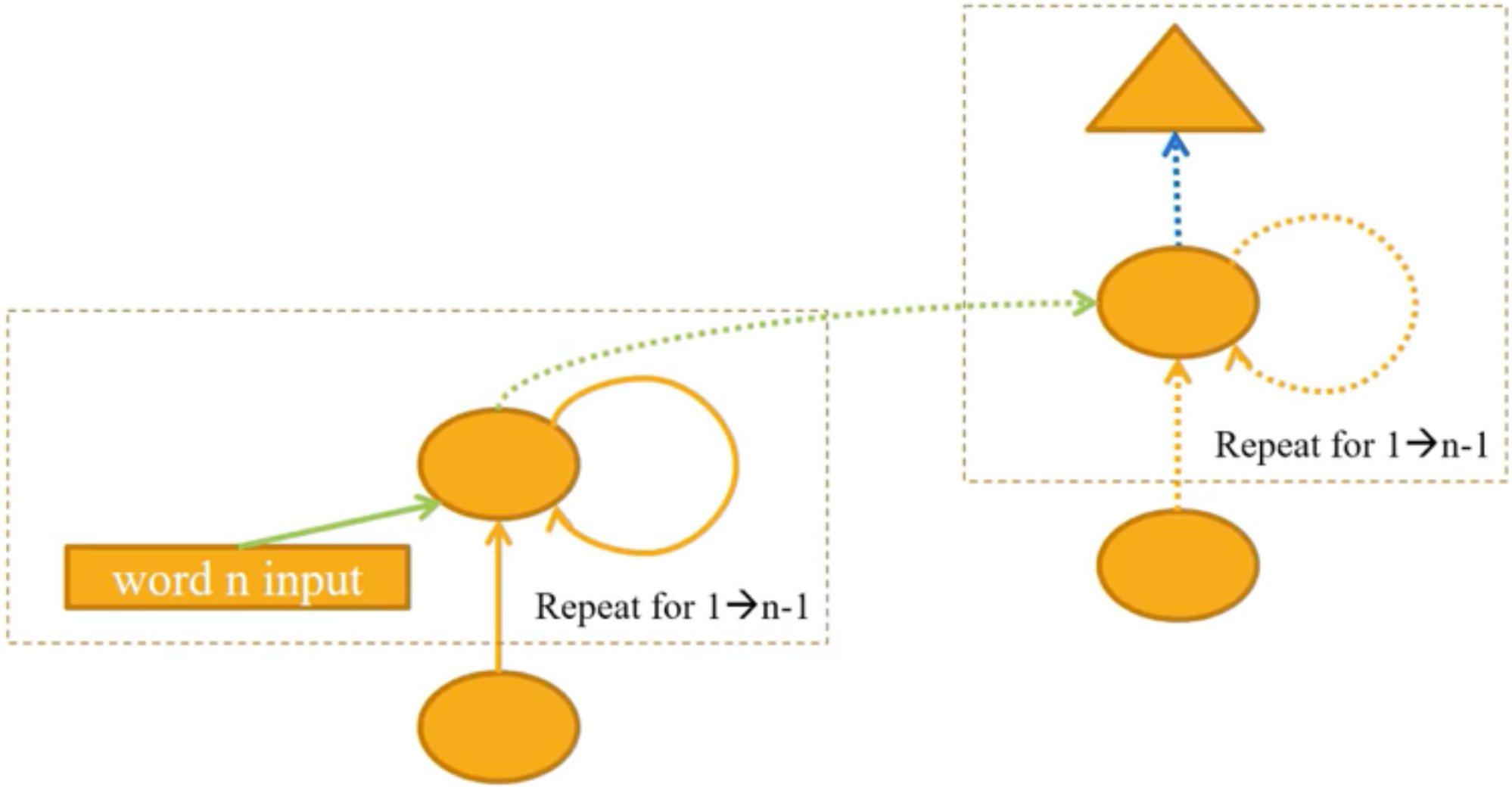


RNNs

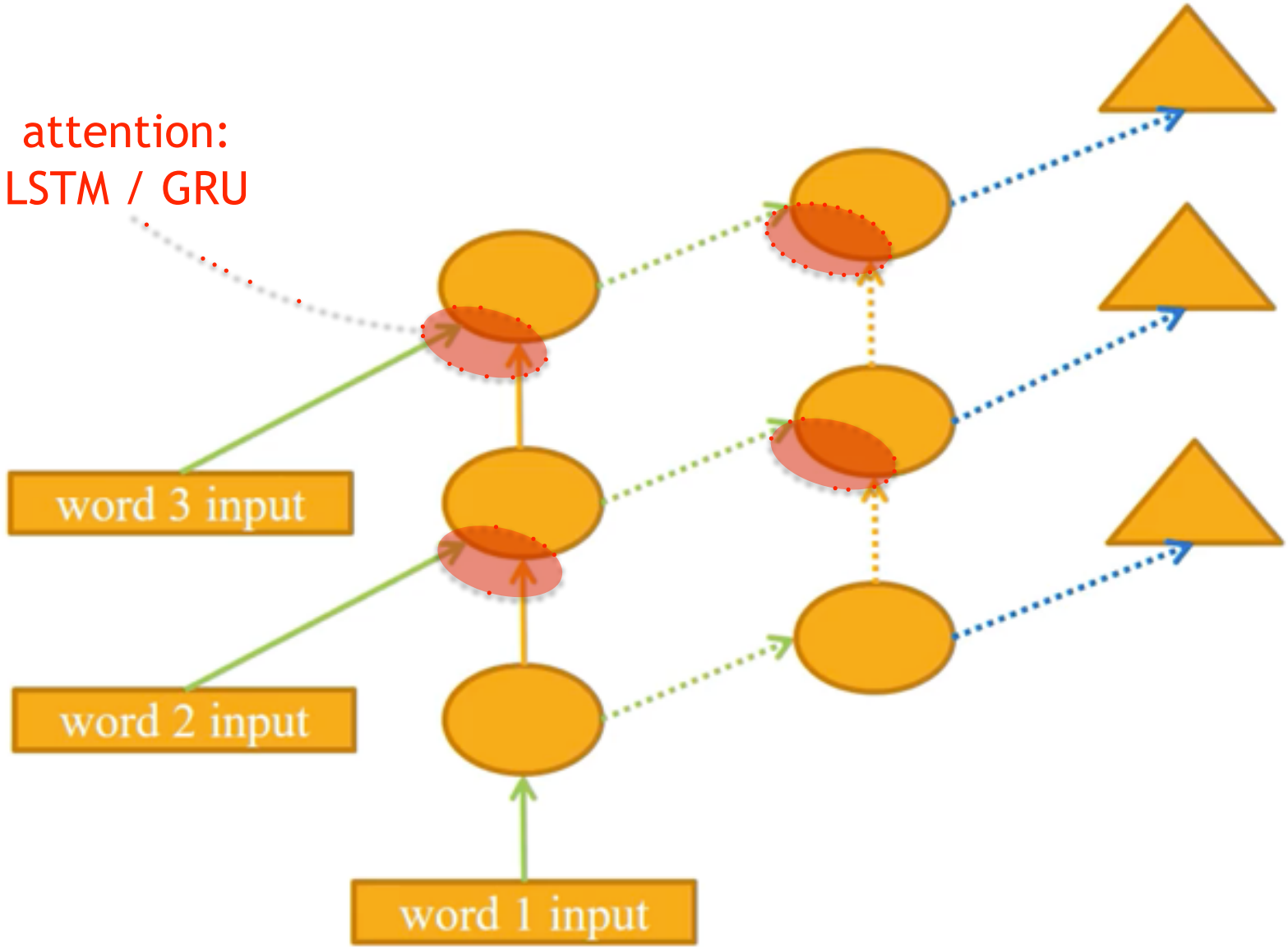




Stacked RNNs



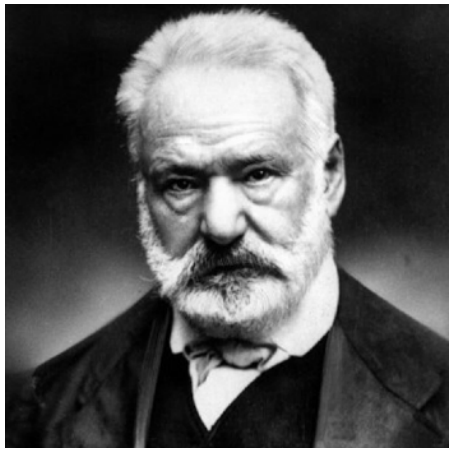
Attention mechanisms



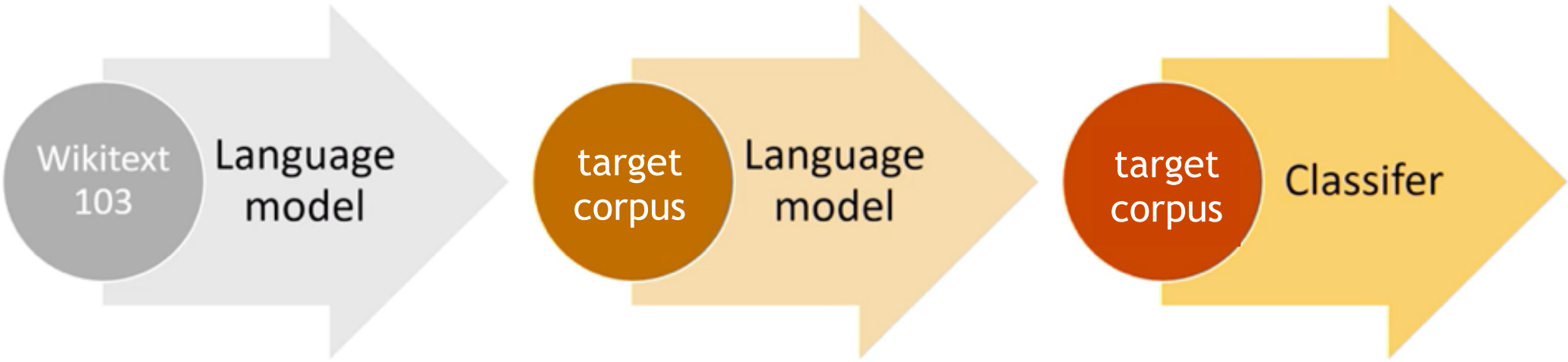
Transfer Learning for NLP

La terre a vu jadis errer des paladins;
Ils flamboyaient ainsi que des éclairs soudains,
Puis s'évanouissaient, laissant sur les visages
La crainte, et la lueur de leurs brusques passages ;

or



???



Main Applications - Tabular Data

Embeddings & NN for tabular data

Entity Embeddings of Categorical Variables

Cheng Guo* and Felix Berkhahn†
Neokami Inc.
(Dated: April 25, 2016)

ranked 2nd at Rosmann kaggle competition

method	MAPE	MAPE (with EE)
KNN	0.290	0.116
random forest	0.158	0.108
gradient boosted trees	0.152	0.115
neural network	0.101	0.093

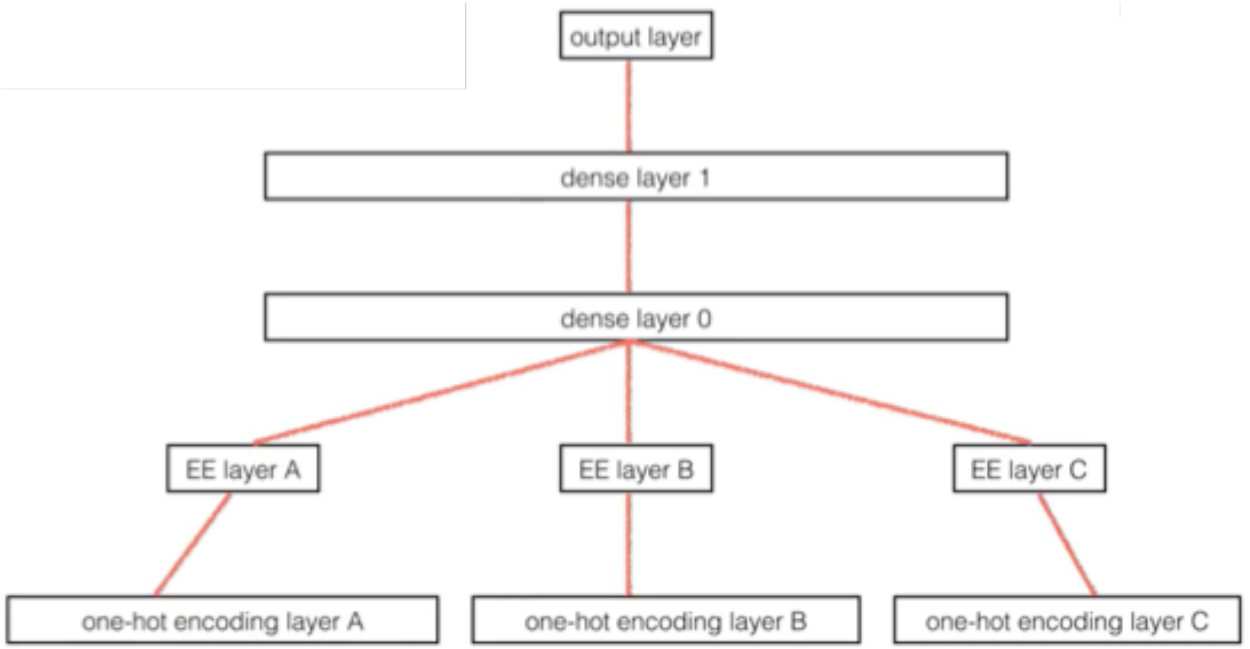
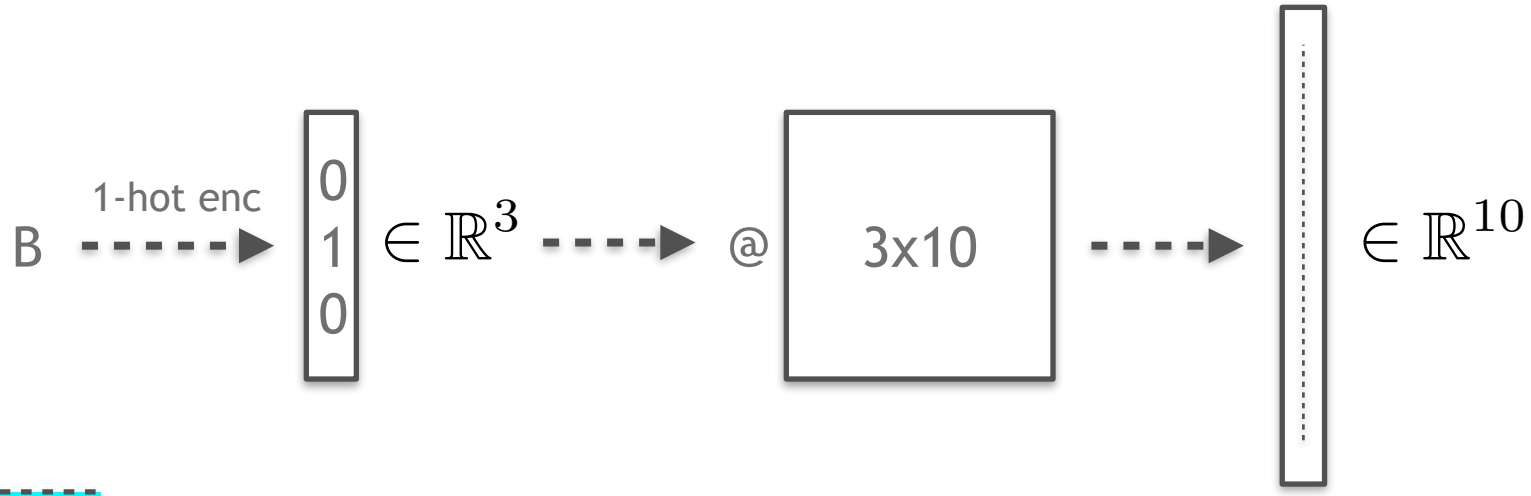
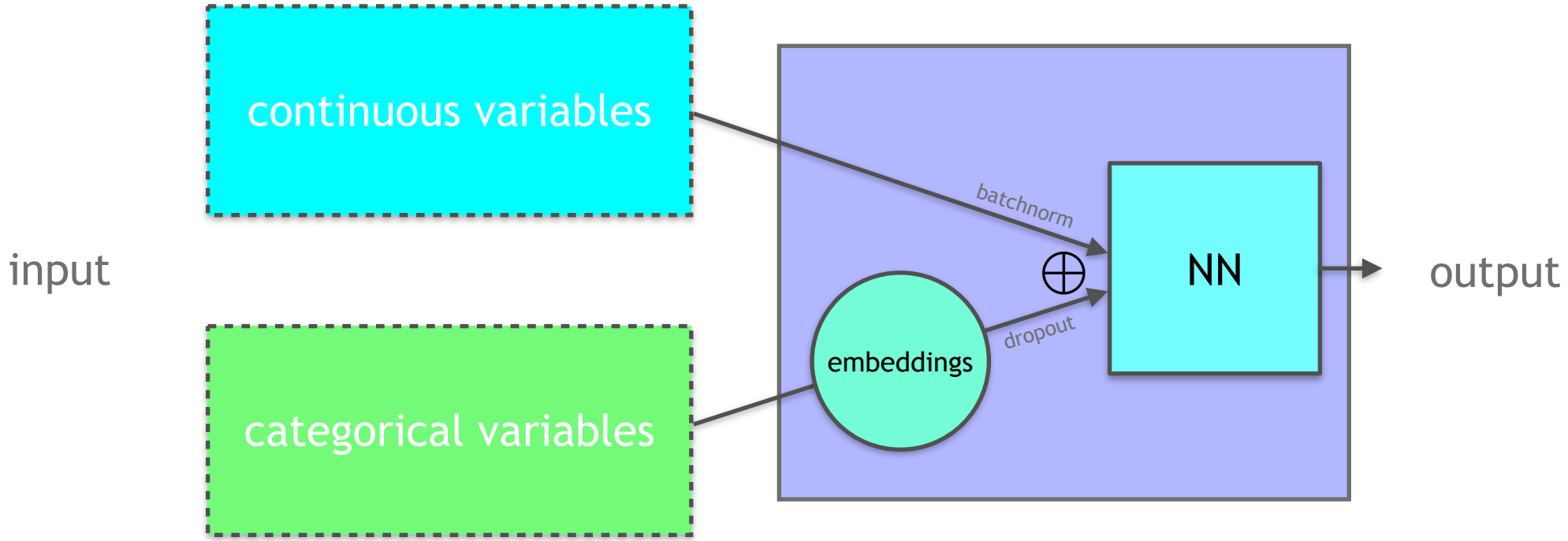


FIG. 1. Illustration that entity embedding layers are equivalent to extra layers on top of each one-hot encoded input.

Embeddings

10-dim embedding of a card 3 categorical variable (A,B,C):

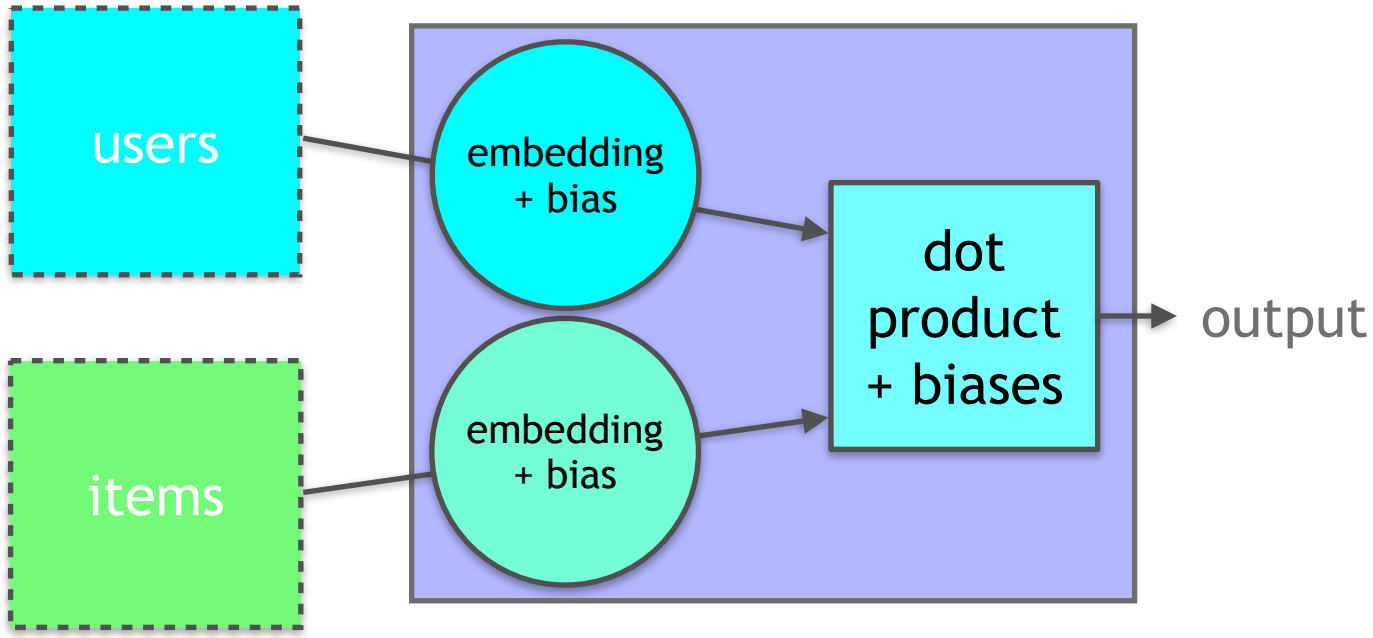




Main Applications - Collaborative Filtering

userId	movieId	review
1	302	★
2	5709	★★★
...	...	...

input

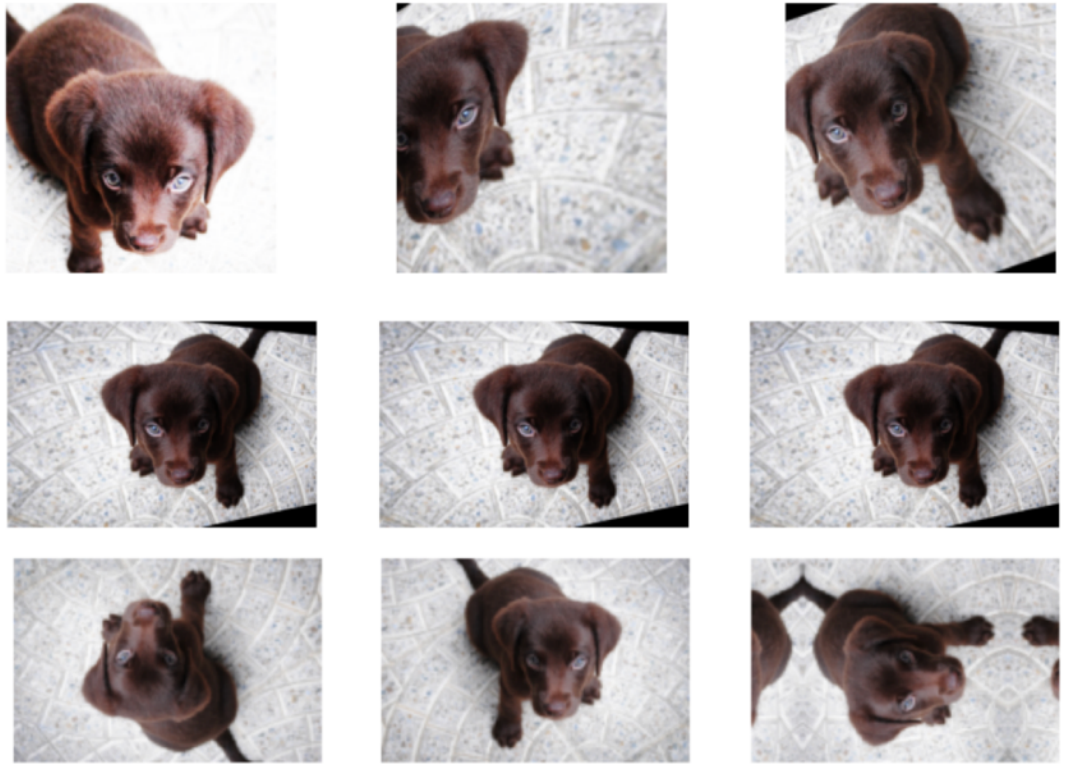


In practice

- Data Augmentation
- Gradients Problems & Init
- Learning Rates
- Overfitting
- GPUs & NN libraries

Data Augmentation

resize, crop, pad, squish, warp, zoom, flip, rotate,…
vary lightning, grayscale, contrast, blur, ***resolution***,…



Gradients Problems & Initialization

problems	culprits	cures…
vanishing & exploding signals ?	NN depth	resnets (skip connections)
vanishing & exploding gradients ?	choice of activations	LSTM
	bad init	params init

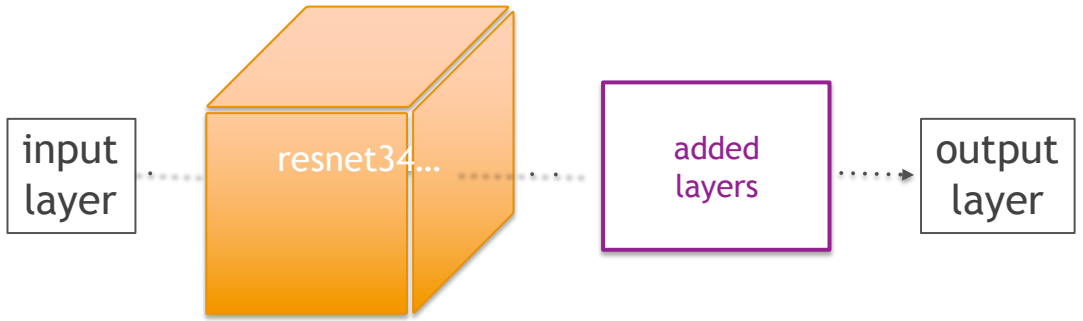
Learning rates

dynamic learning rates:

- momentum $M_t = (1-\beta)\nabla_t + \beta M_{t-1}$
- RMSprop $R_t = (1-\beta)\nabla_t^2 + \beta R_{t-1}$
- Adam:

$$w_i^{(t)} \leftarrow w_i^{(t-1)} - \mathrm{lr}\frac{M_t}{\sqrt{R_t}}$$

discriminative learning rate



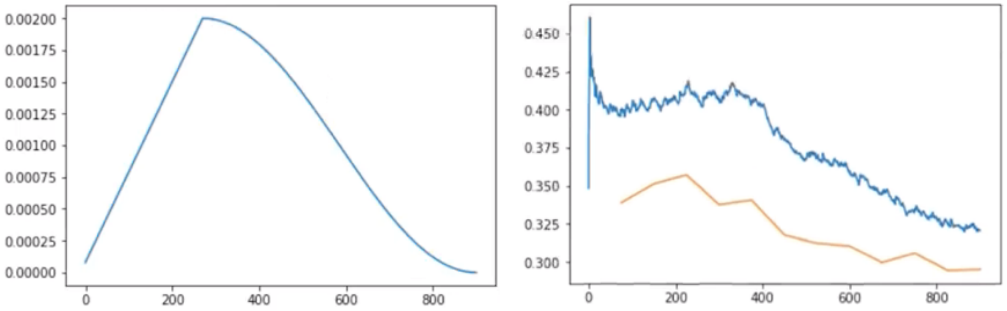
FREEZE — freeze *keep params* — learn *update params*

UNFREEZE — learn *update all params*

smaller lr
eg 1e-5

larger lr
eg 1e-3

learning rate annealing



L. N. Smith 2015

Overfitting

Deep NN have **HUGE** numbers of learnable parameters !

eg models widely used today:

- resnet50 model for image recognition: over **23 millions**
- “BERT base” model for NLP: over **110 millions**

Standard ways to reduce overfitting:

- weight decay (typically L2-reg)

- dropout

Dropout

"I went to my bank. The tellers kept changing and I asked one of them why. He said he didn't know but they got moved around a lot. I figured it must be because it would require cooperation between employees to successfully defraud the bank. This made me realize that randomly removing a different subset of neurons on each example would prevent conspiracies and thus reduce overfitting"

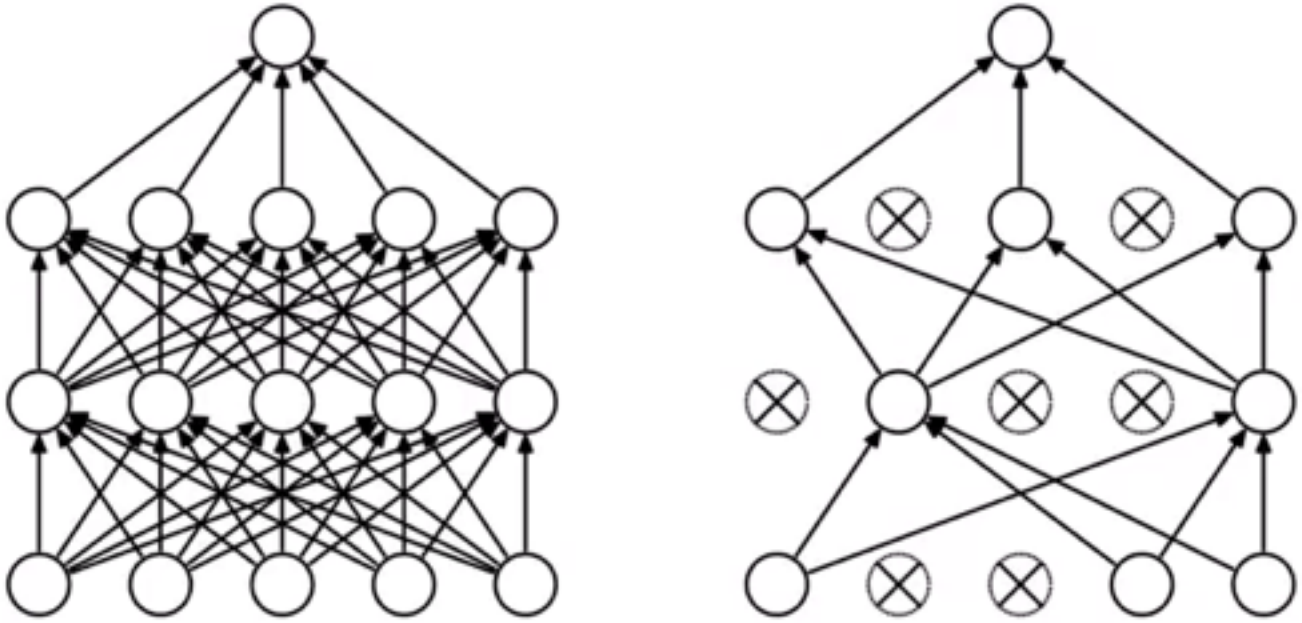
Hinton: Reddit AMA

Dropout

JMLR 2014

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava NITISH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
Alex Krizhevsky KRIZ@CS.TORONTO.EDU
Ilya Sutskever ILYA@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Batchnorm

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

ICML'2015

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

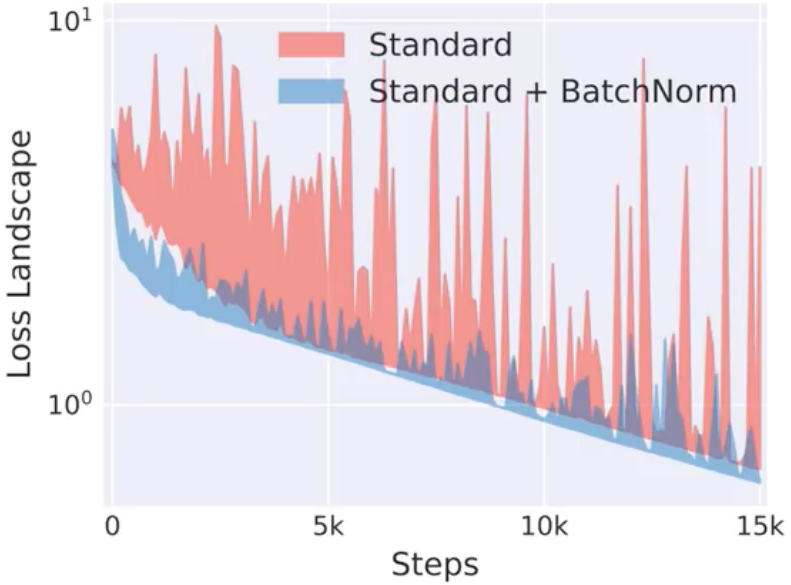
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



S. Santukar *et al.* MIT

batchnorm
flattens out
loss surface
=> higher lr
=> faster train

Eg at current stage of learning, last activations predict outputs

$$\hat{y} = f(w_i\text{'s}; x)$$

in (-1,1)… but target is in range 1 to 5!

BN scales and shifts output
via 2 extra learnable parameters

$$\hat{y} = \gamma f(w_i\text{'s}; x) + \beta$$

GPUs & NN libraries

GPUs

free

professional solutions

Google Cloud Platform

300$ free credit
education packs
0.3-1.2€/hour

NN libraries

vs


early
makers
em
lyon
business
school