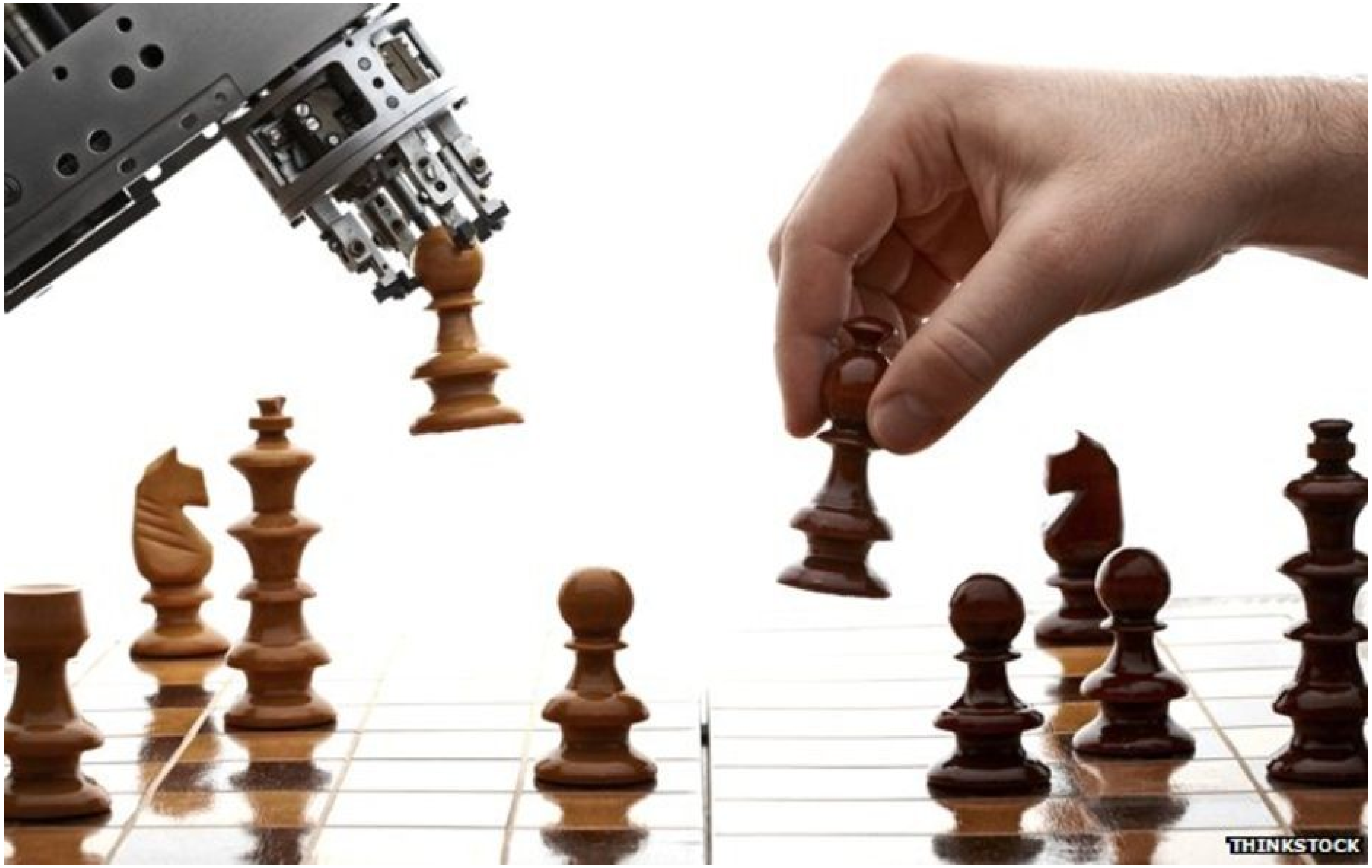


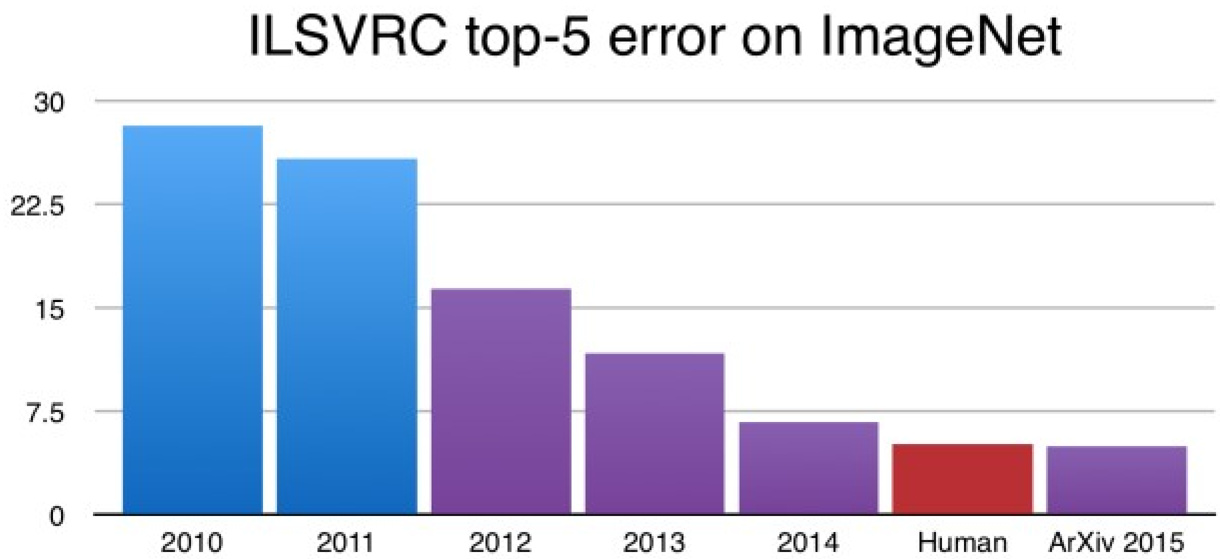
Introduction to Artificial Intelligence

COSC 4550 / COSC 5550

Professor Cheney

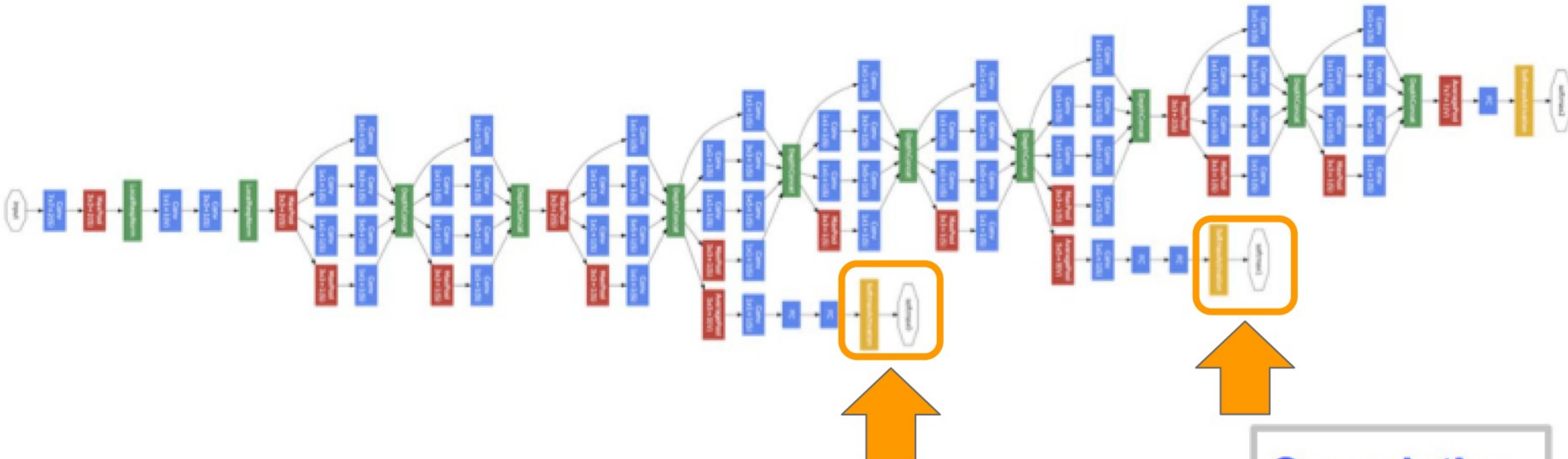
11/8/17

deep learning architectures cont.


ILSVRC top-5 error on ImageNet
30
22.5
15
7.5
0
2010
2011
2012
2013
2014
Human
ArXiv 2015

GoogLeNet (Inception)

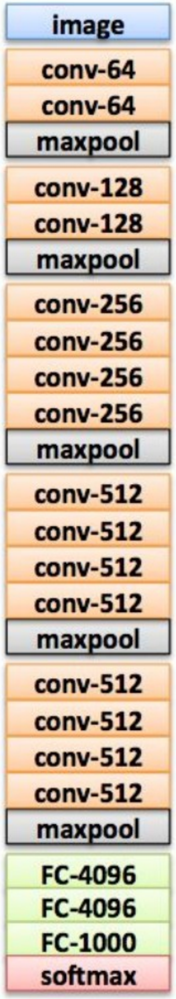
Two Softmax Classifiers at intermediate layers combat the vanishing gradient while providing regularization at training time.



Convolution
Pooling
Softmax
Other

...and no fully connected layers needed !

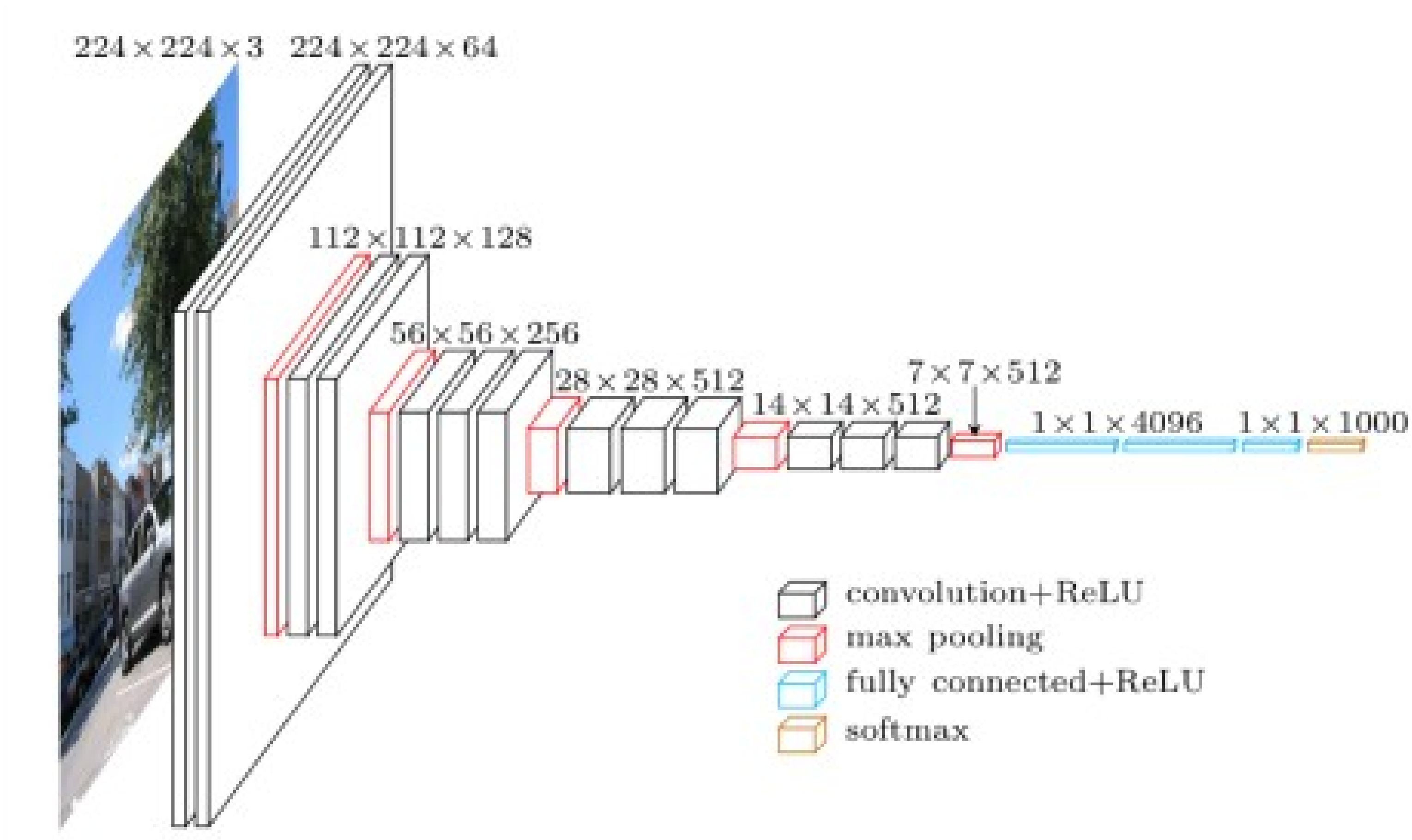
E2E: Classification: VGG



- No poolings between some convolutional layers.
- Convolution strides of 1 (no skipping).

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *International Conference on Learning Representations (2015)*. [video] [slides] [project]

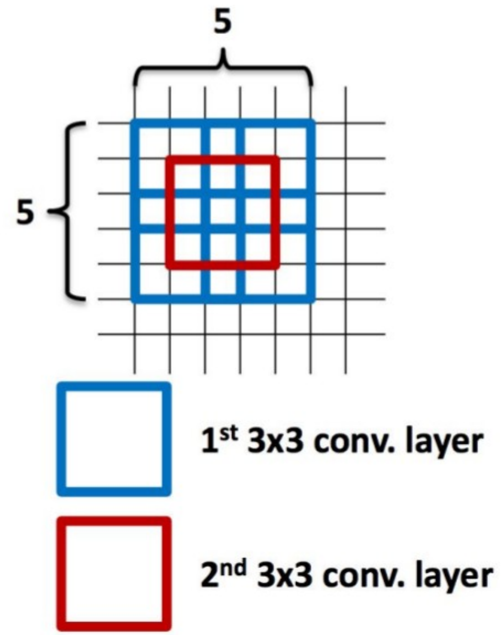
VGG (2014)


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

E2E: Classification: VGG: 3x3 Stacks

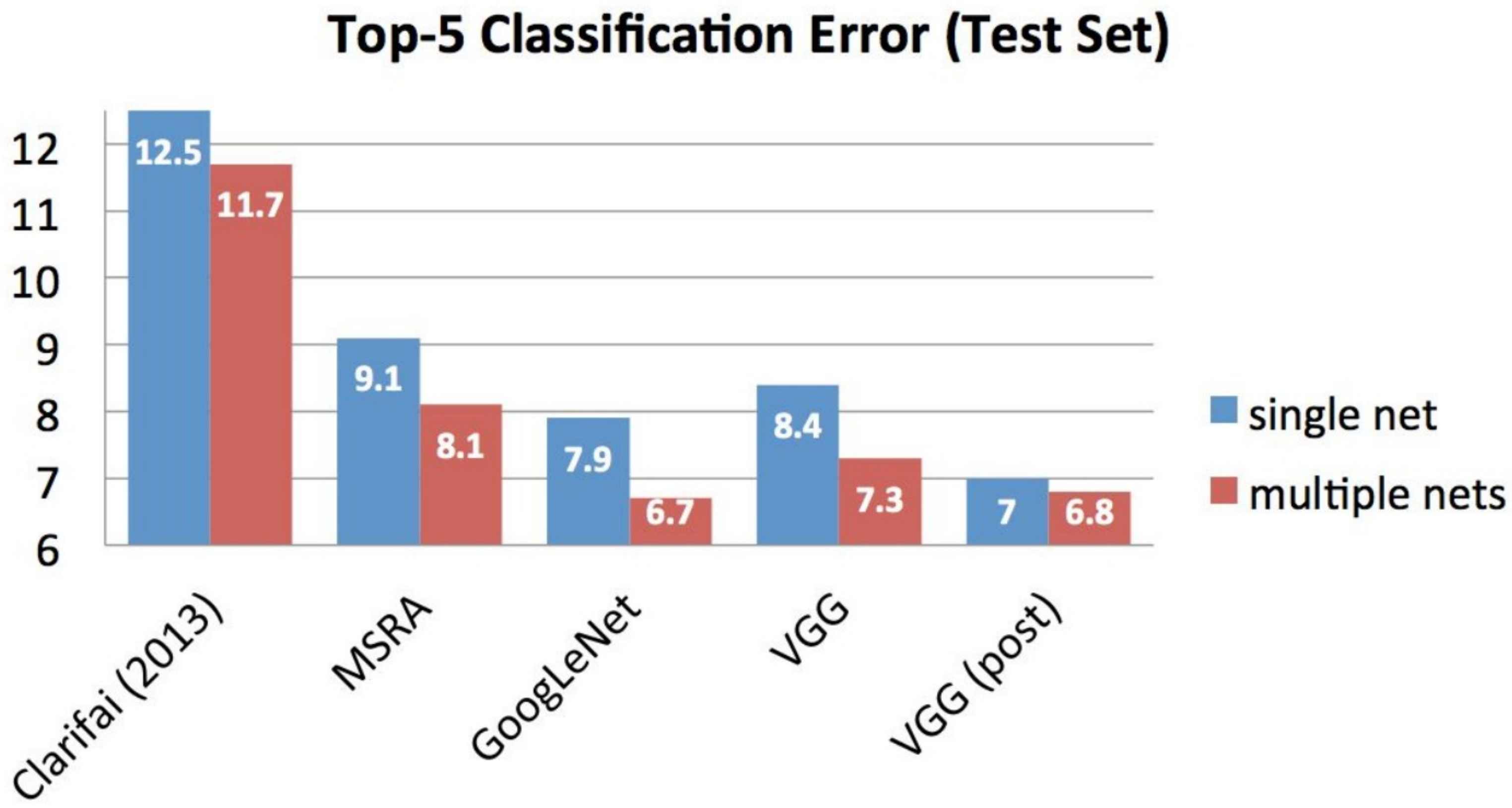
Why 3x3 layers?

- Stacked conv. layers have a large receptive field
  - two 3x3 layers – 5x5 receptive field
  - three 3x3 layers – 7x7 receptive field
- More non-linearity
- Less parameters to learn
  - ~140M per net



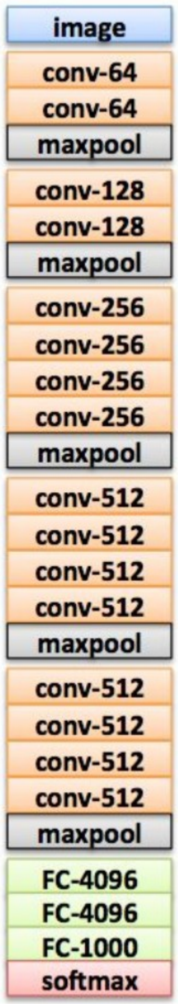
Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *International Conference on Learning Representations (2015)*. [video] [slides] [project]

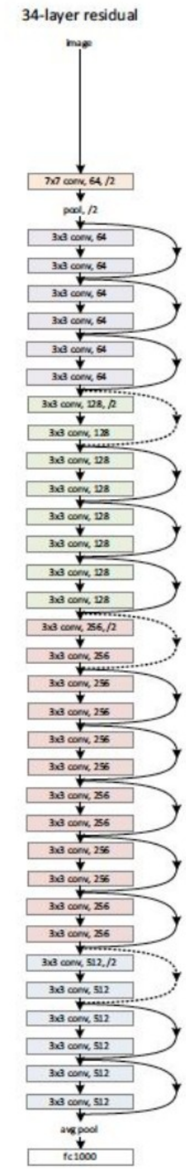
E2E: Classification: VGG



Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *International Conference on Learning Representations (2015)*. [video] [slides] [project]

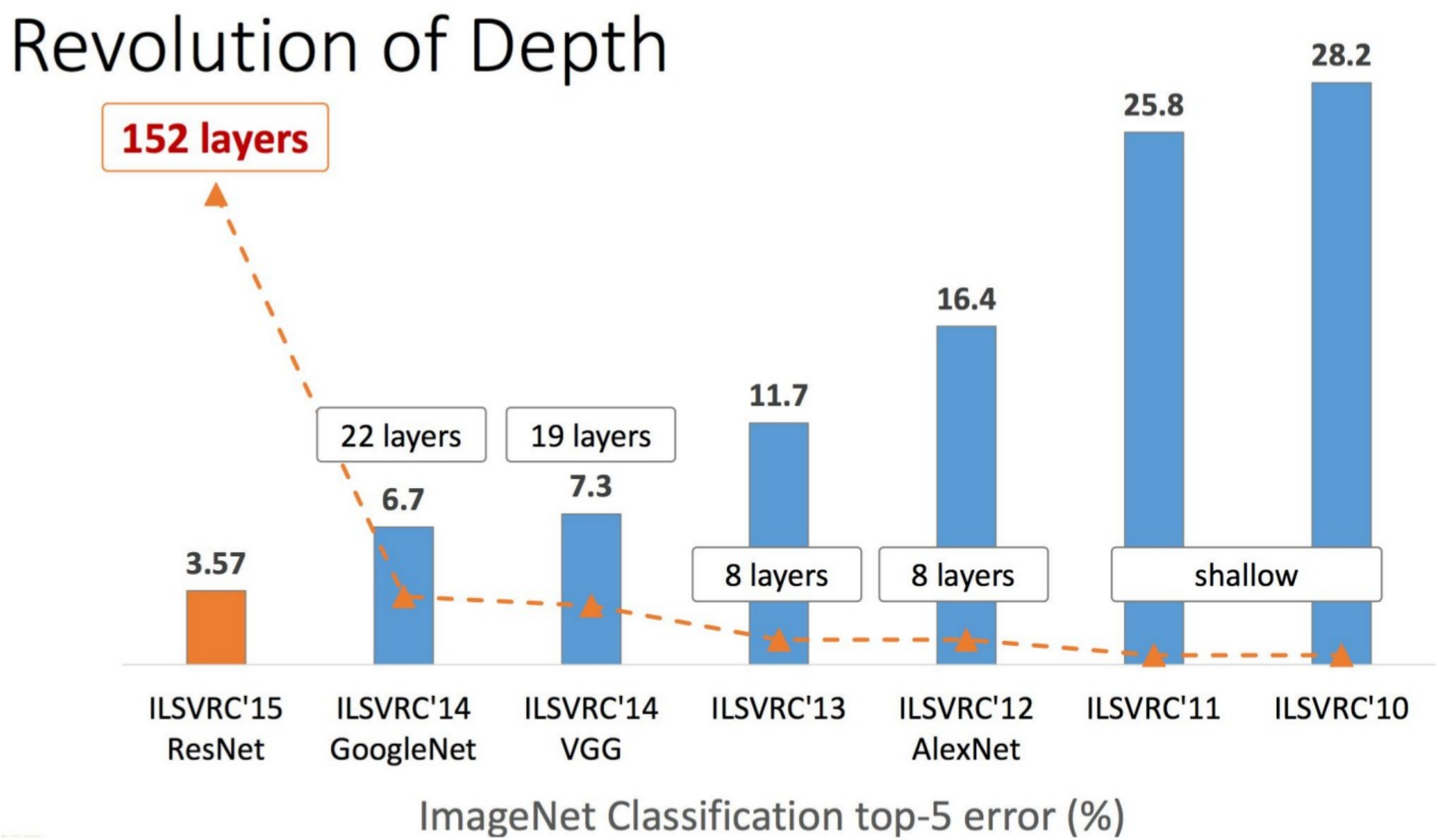
ImageNet Challenge: 2015





3.6% top 5 error…
with 152 layers !!

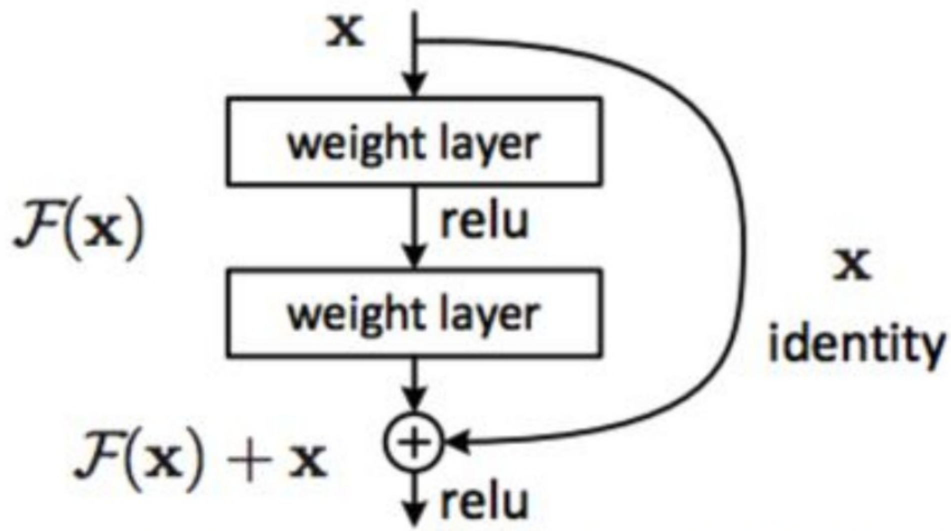
E2E: Classification: ResNet



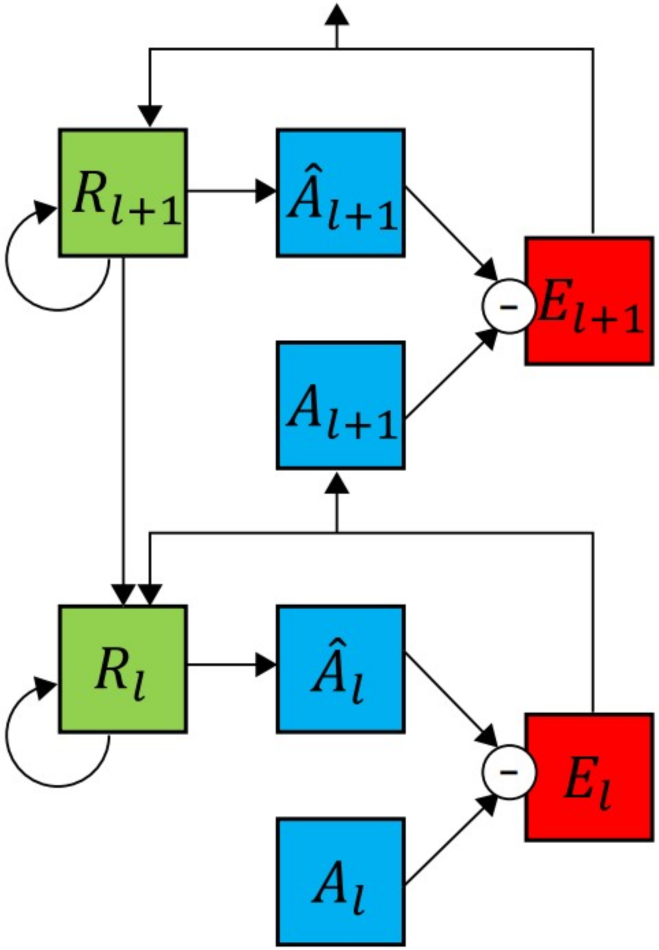
He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." *arXiv preprint arXiv:1512.03385* (2015). [slides]

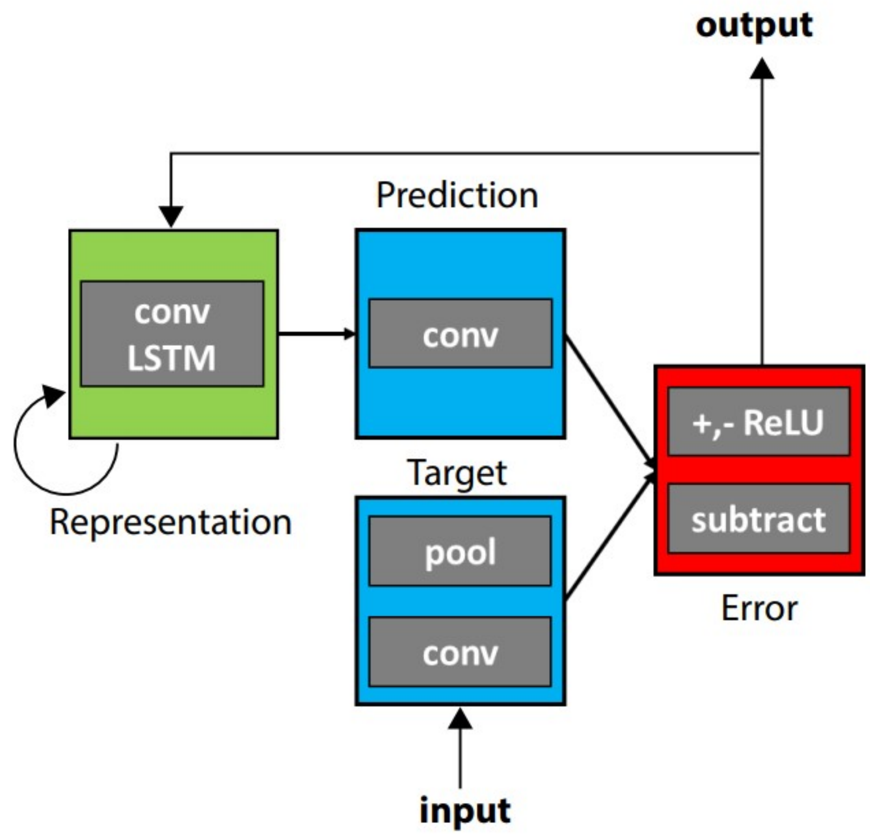
ResNet

- Residual learning: reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions



He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." *arXiv preprint arXiv:1512.03385* (2015). [slides]

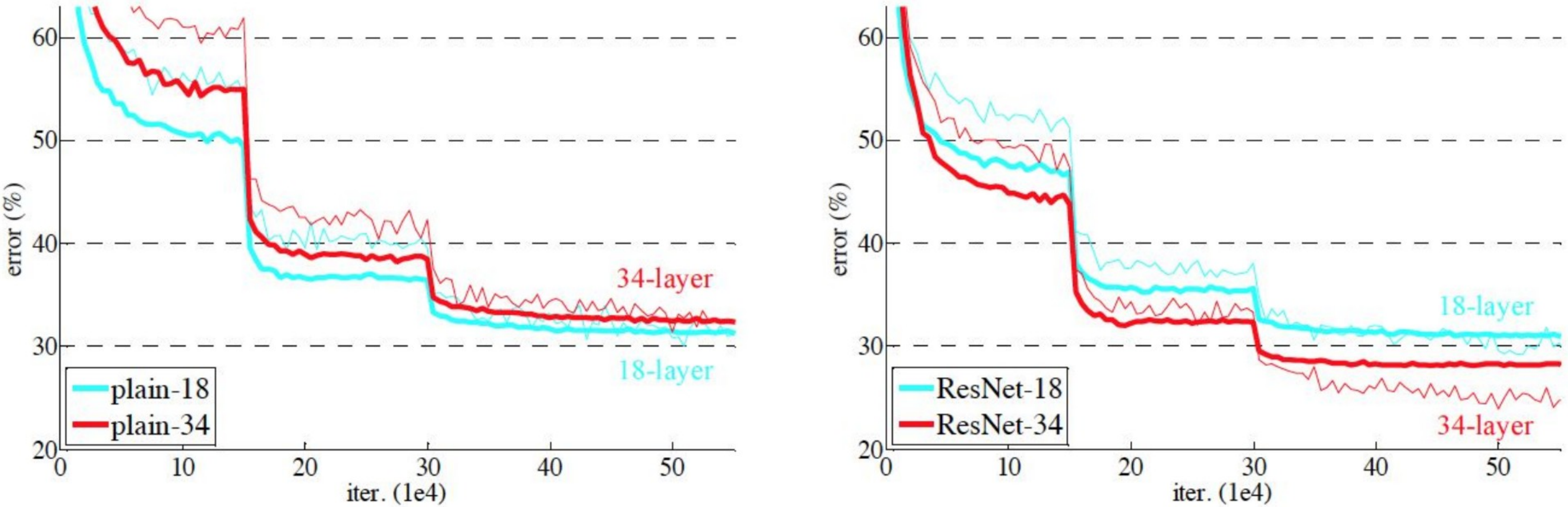




Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning

William Lotter, Gabriel Kreiman & David Cox
Harvard University
Cambridge, MA 02215, USA
{lotter,davidcox}@fas.harvard.edu
gabriel.kreiman@tch.harvard.edu

E2E: Classification: ResNet



He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." *arXiv preprint arXiv:1512.03385* (2015). [slides]

Learn more

Li Fei-Fei, [“How we’re teaching computers to understand pictures”]() TEDTalks 2014.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*. [web]

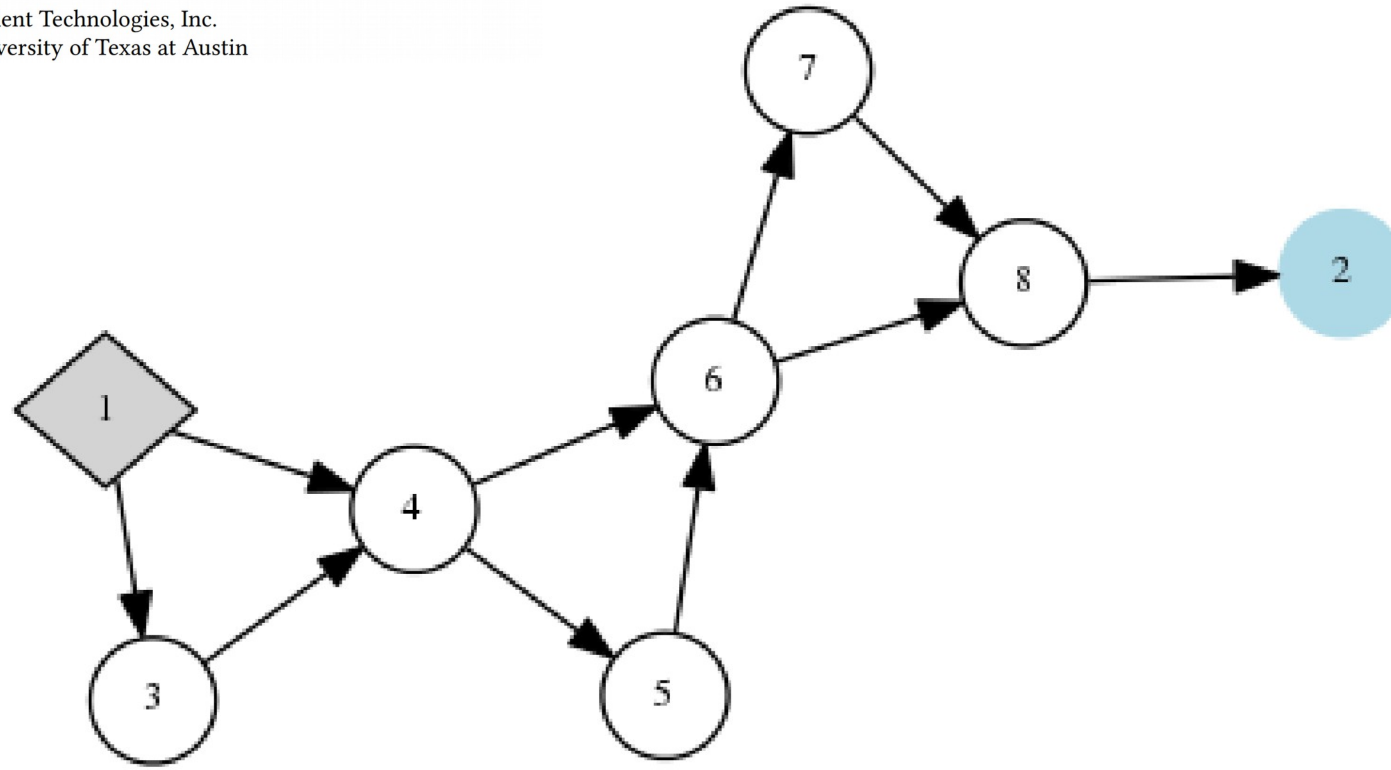
most network architecture are designed by hand

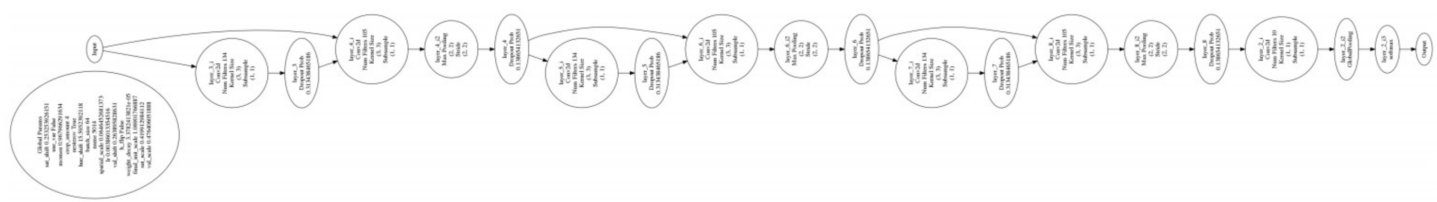
but there are efforts to optimize the topologies as well...

Evolving Deep Neural Networks

Risto Miikkulainen[1,2], Jason Liang[1,2], Elliot Meyerson[1,2], Aditya Rawal[1,2], Dan Fink[1], Olivier Francon[1], Bala Raju[1], Hormoz Shahrzad[1], Arshak Navruzyan[1], Nigel Duffy[1], Babak Hodjat[1]

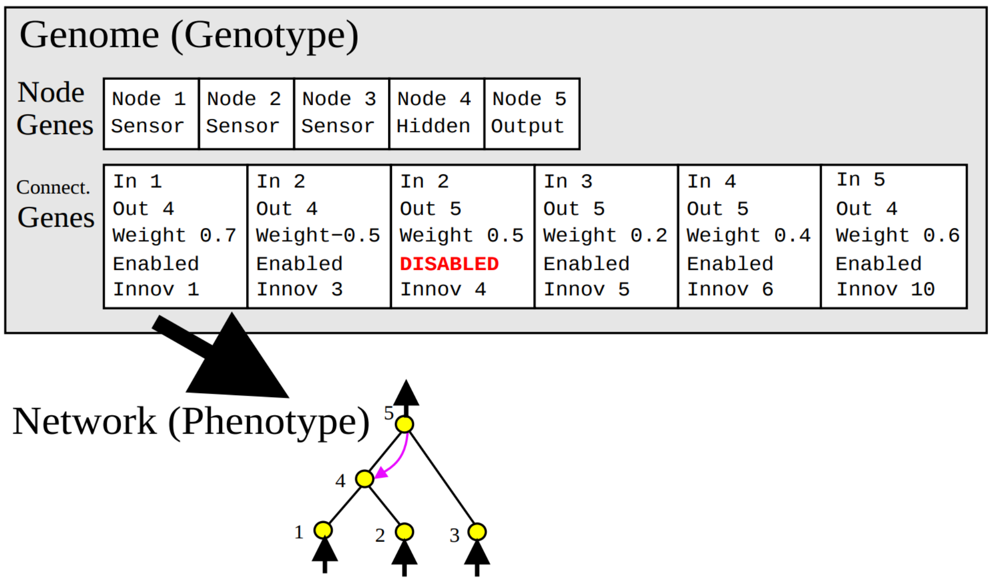
[1]Sentient Technologies, Inc.
[2]The University of Texas at Austin

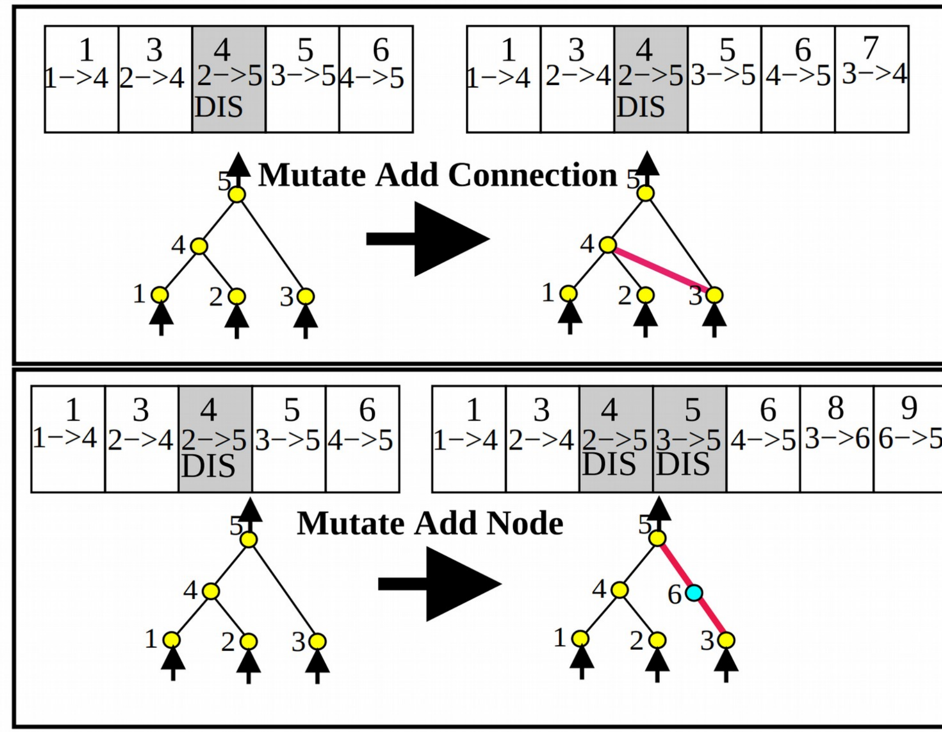


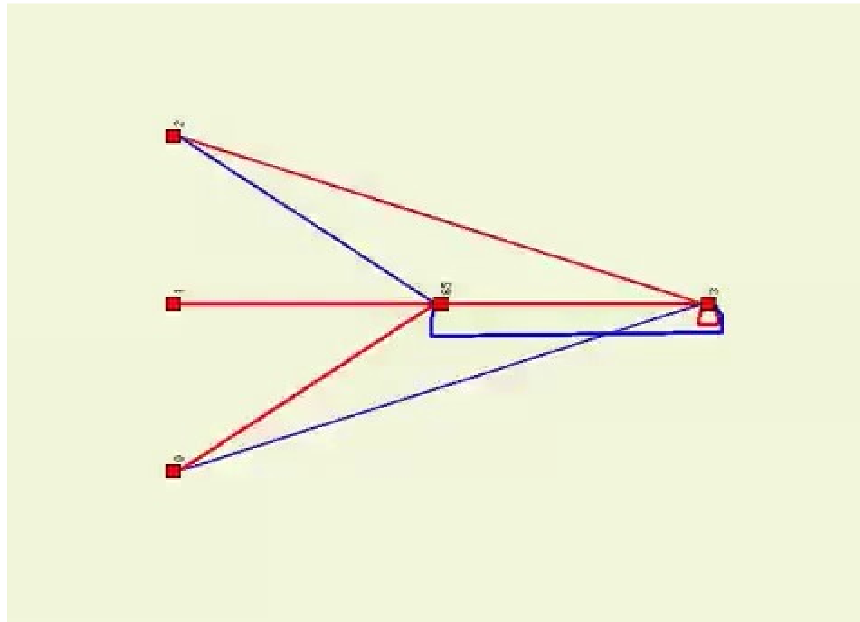


Evolving Neural Networks through Augmenting Topologies

Kenneth O. Stanley and Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
{kstanley, risto}@cs.utexas.edu







Node Hyperparameter	**Range**
Number of Filters	$[32, 256]$
Dropout Rate	$[0, 0.7]$
Initial Weight Scaling	$[0, 2.0]$
Kernel Size	$\{1, 3\}$
Max Pooling	$\{\text{True, False}\}$
Global Hyperparameter	**Range**
Learning Rate	$[0.0001, 0.1]$
Momentum	$[0.68, 0.99]$
Hue Shift	$[0, 45]$
Saturation/Value Shift	$[0, 0.5]$
Saturation/Value Scale	$[0, 0.5]$
Cropped Image Size	$[26, 32]$
Spatial Scaling	$[0, 0.3]$
Random Horizontal Flips	$\{\text{True, False}\}$
Variance Normalization	$\{\text{True, False}\}$
Nesterov Accelerated Gradient	$\{\text{True, False}\}$

moral of the story, the deeper the better…

so why haven't neural networks always been deep?

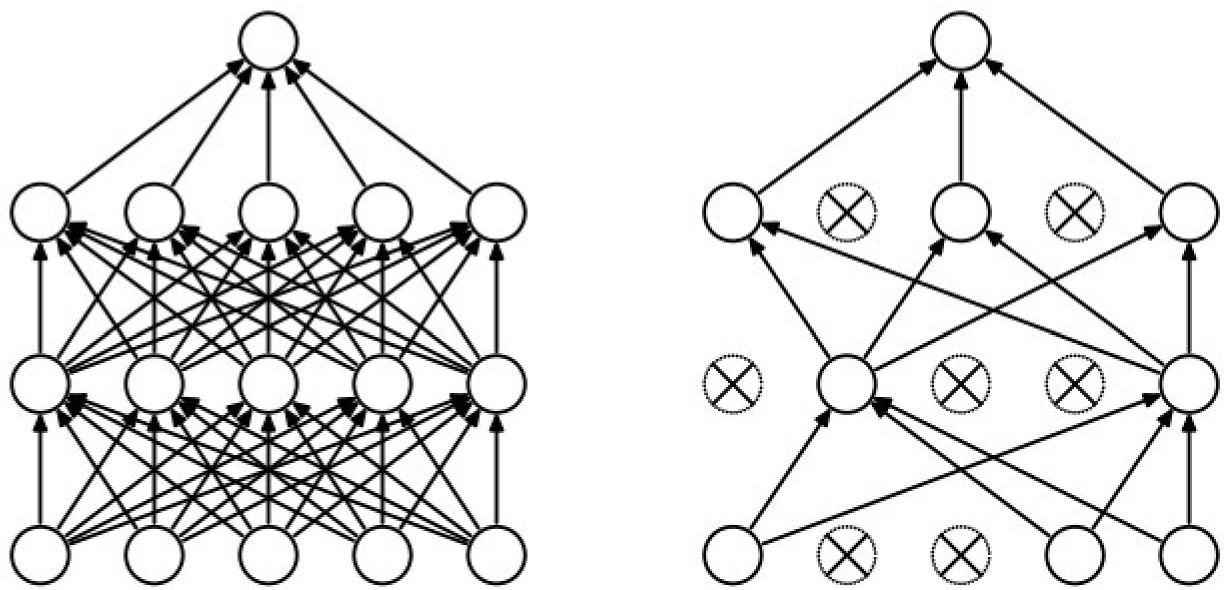
it's not easy… turns out deep neural network
have problems that shallow ones don't

recent tips and tricks for going deep!

with so many parameters,
deep neural networks have the potential
to overfit to the training data

dropout (2014)

only use a random subset of weights
for each forward/backwards pass



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

helps prevent overfitting, with relatively little determent to learning or performance (given a big enough network)

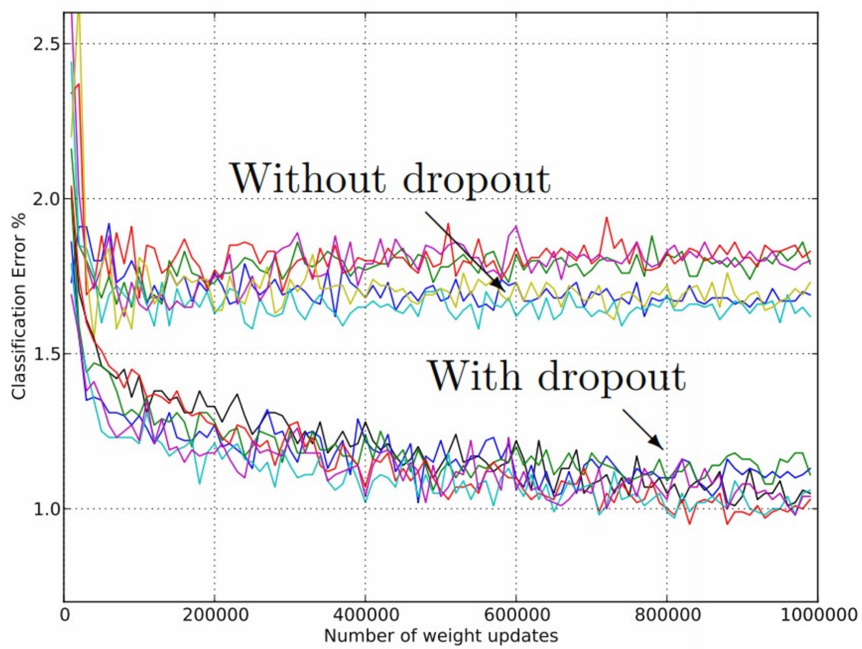


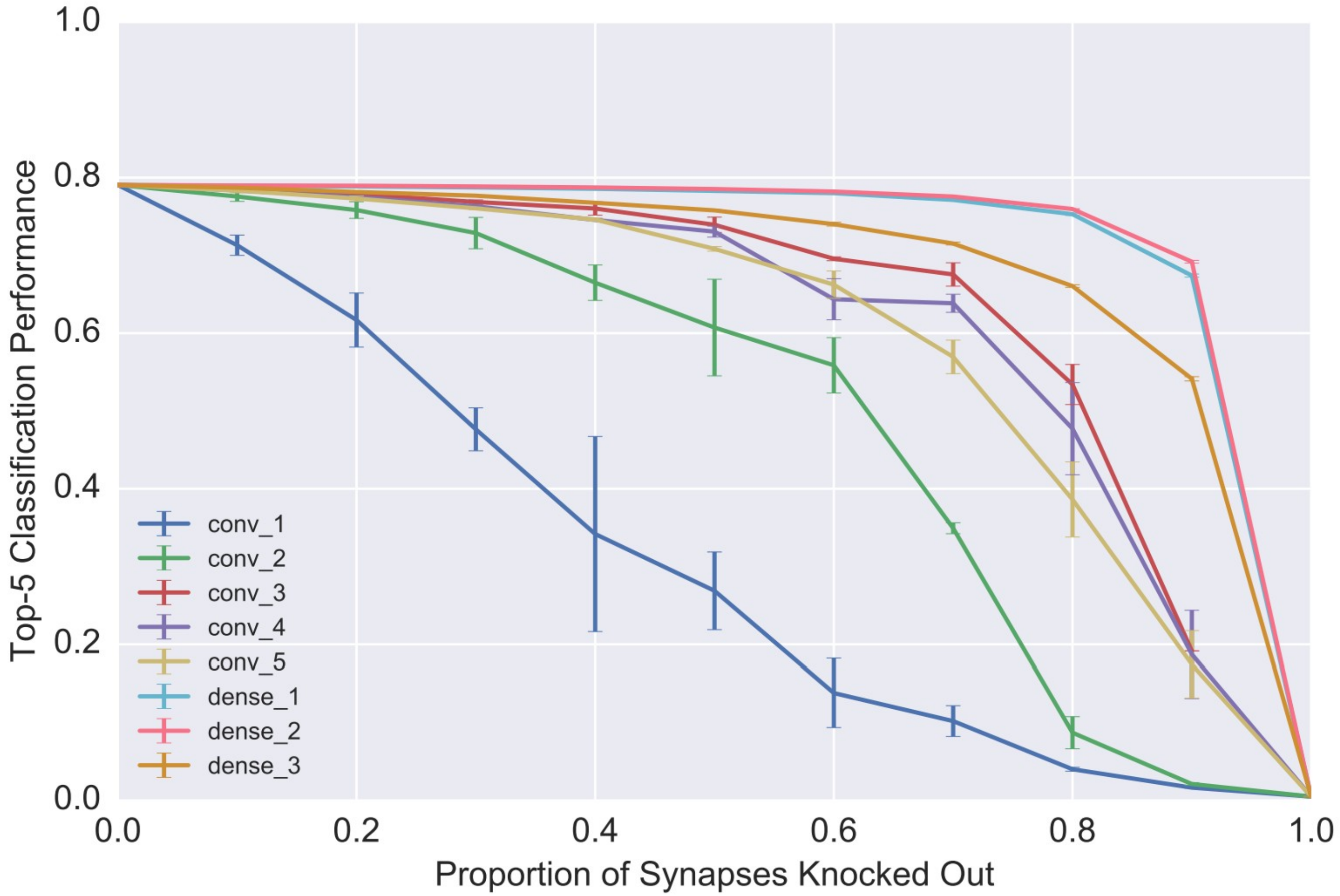
Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

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
Department of Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.

Editor: Yoshua Bengio

helps prevent overfitting, with relatively little determent to learning or performance (given a big enough network)



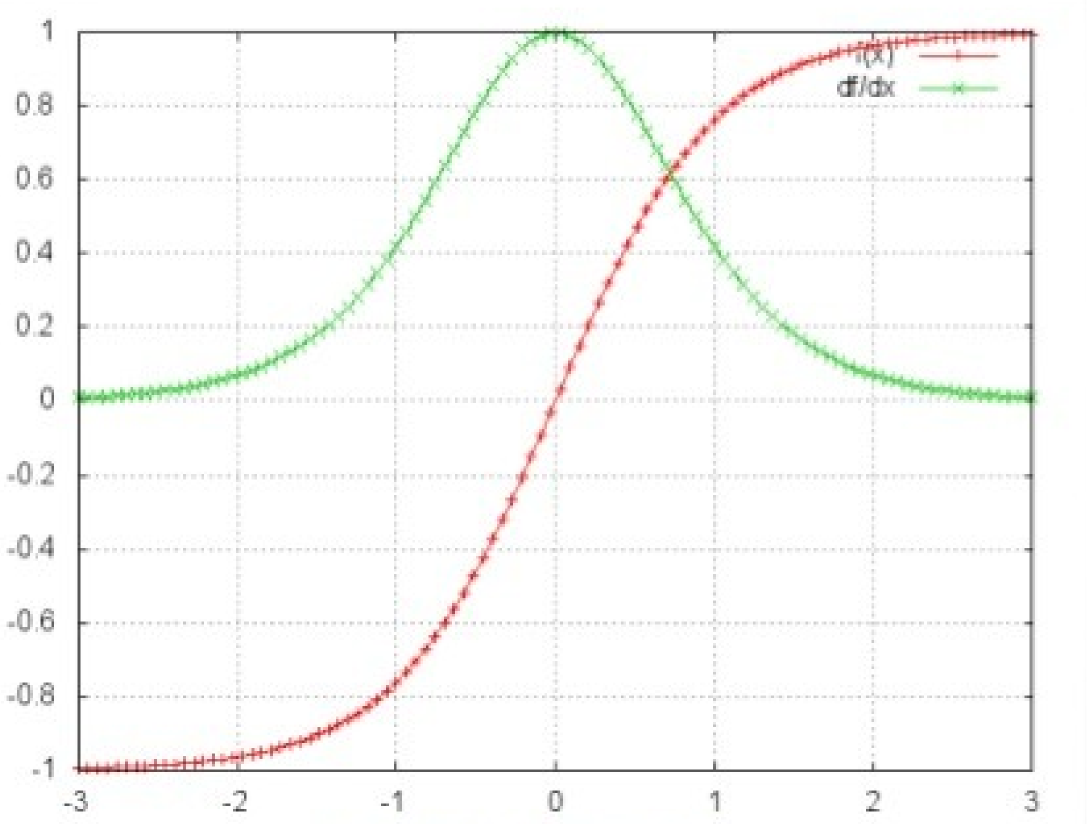
On the Robustness of Convolutional Neural Networks to Internal Architecture and Weight Perturbations

Nicholas Cheney [*1] **Martin Schrimpf** [*2 3] **Gabriel Kreiman** [3]

with so many layers,
error signals may similarly be continually decreased
by each layer during backpropogation,
leading to vanishings gradients!

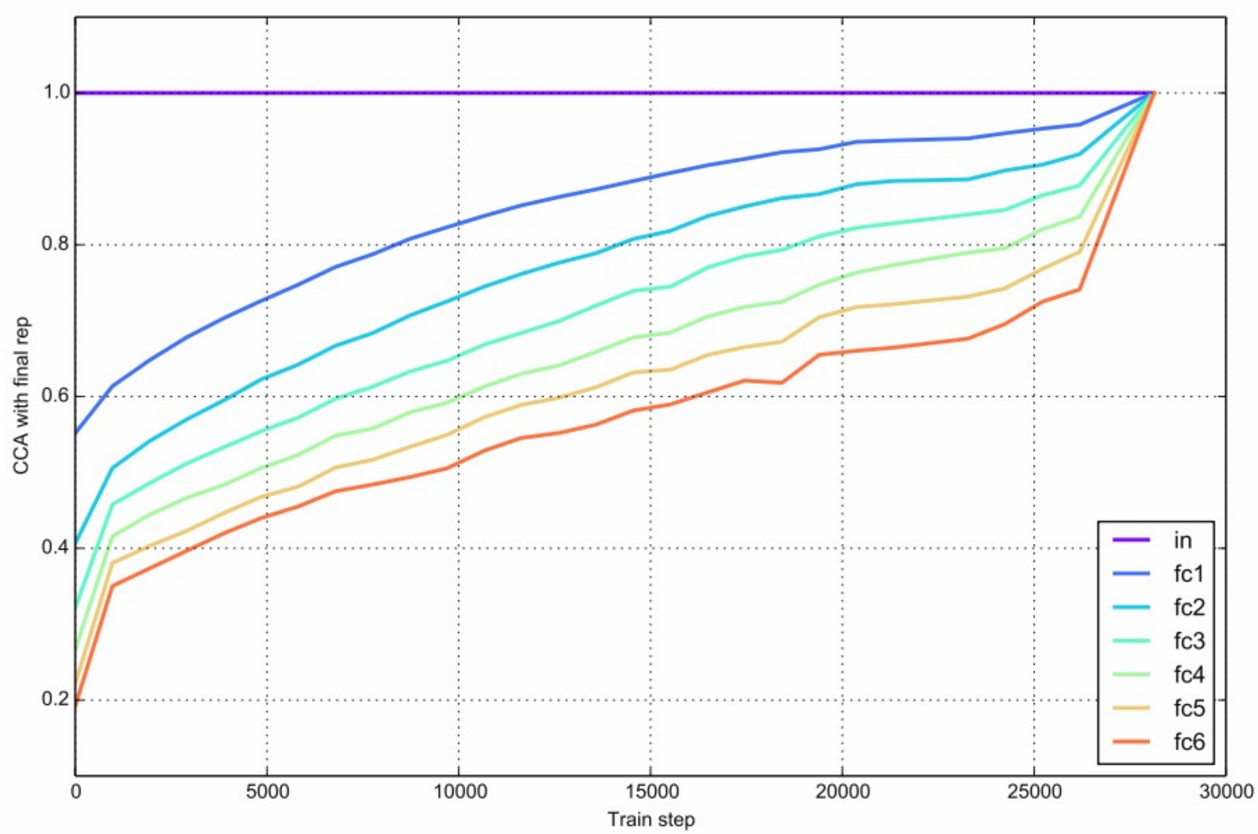
this problem is especially bad when activations
are extreme (very high or very low)

tanh(x) activation function, and its derivative



for activation values at the extremes, derivative is small!
(this is less of a problem when inputs are around 0,
but all are ≤ 1, i.e. shrinking error signal)

to complicate things further… error signals are strongest near the outputs, but layers must be built up from inputs first



BOTTOM-UP OR TOP-DOWN? DYNAMICS OF DEEP REPRESENTATIONS VIA CANONICAL-CORRELATION ANALYSIS

Maithra Raghu,[1] Jason Yosinski,[2] & Jascha Sohl-dickstein[1]
[1]Google [2]Uber AI Labs
maithra@google.com, yosinski@uber.com, jaschasd@google.com

batch normalization (2015)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

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

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

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

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

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

for a mini-batch of training examples,

normalize (scale and shift) all of the inputs at each layer

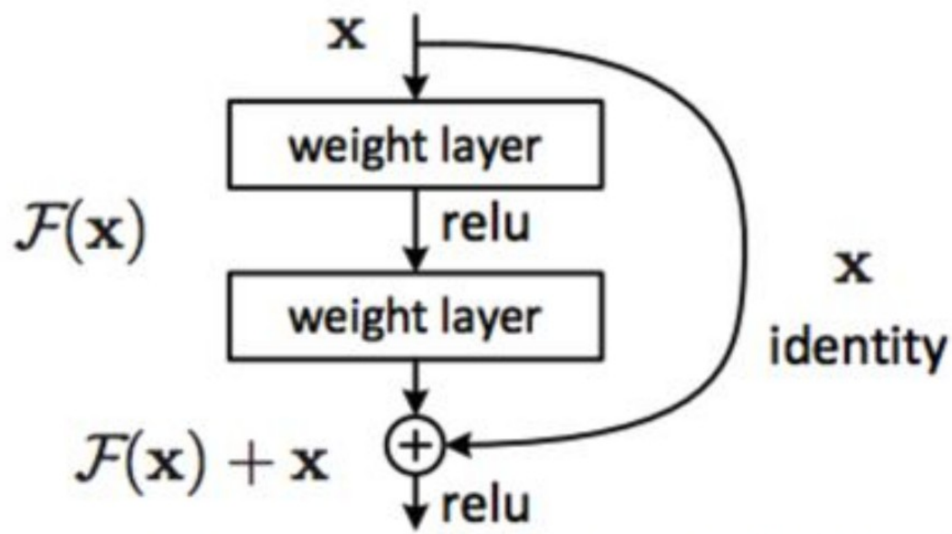
(i.e. so they are centered around 0)

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

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

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

ResNets (2015) help to overcome this problem too by calculating the residual (difference) between activation layers, which tend to be centered around 0

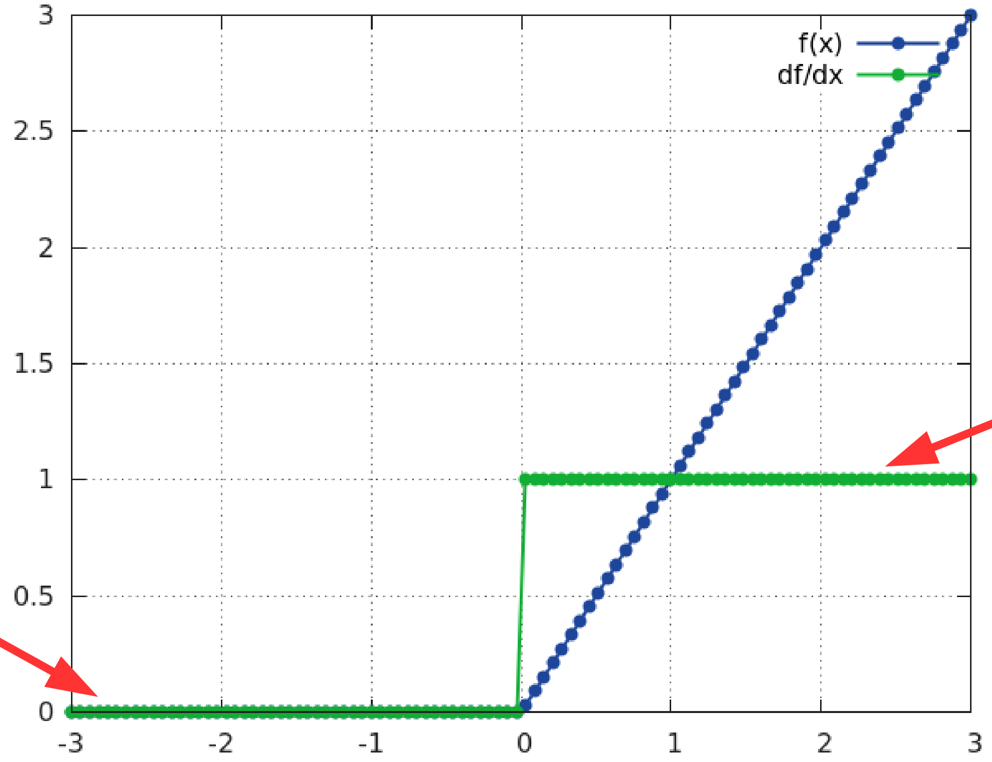


ReLu activation function (2000… popularized around 2015)

rectified linear unit activation function, and its derivative

removal of negative activations creates non-linearity, but...

zero gradient for negative activations disallows learning on inhibitory activations



constant derivative for positive activations reduces vanishing gradients (and activations)

as an aside… why do we even need non-linearities?

with non-linearity:

$$l_2 = f(w_{1,2} \circ l_1)$$
$$l_3 = f(w_{2,3} \circ l_2)$$

$$l_3 = f(w_{2,3} \circ f(w_{1,2} \circ l_1))$$

without non-linearity:

$$l_2 = (w_{1,2} \circ l_1)$$
$$l_3 = (w_{2,3} \circ l_2)$$

$$l_3 = (w_{2,3} \circ (w_{1,2} \circ l_1))$$
$$l_3 = (w_{2,3} \circ w_{1,2}) \circ l_1$$
$$l_3 = w_{all} \circ l_1$$

condenses to the equivalent of a single layer!