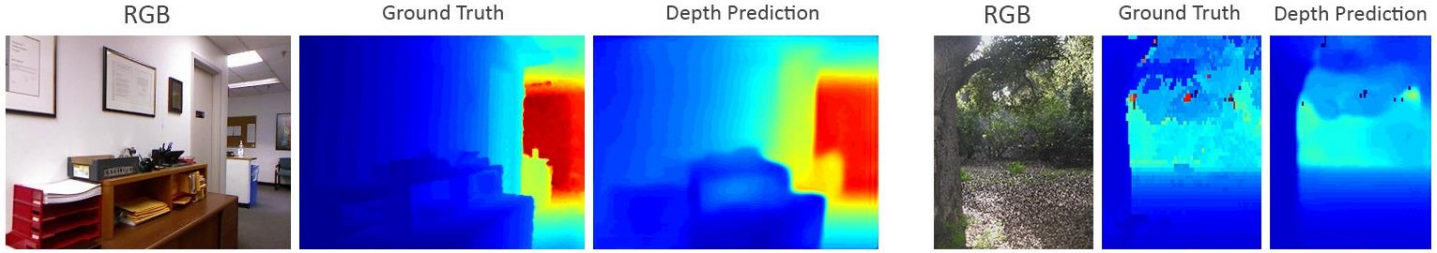
Convolutional Neural Networks

What can CNNs do?

- Image restoration

  Eigen et al. “Restoring an image taken through a window covered with dirt or rain.” *IEEE ICCV (*2013).

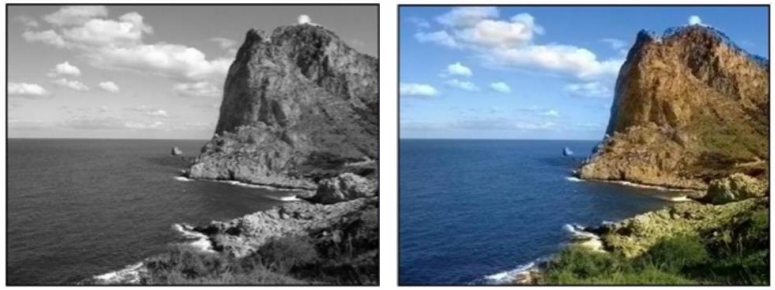
- Depth



Laina et al., “Deeper Depth Prediction with fully convolutional residual networks” *IEEE 3DV (2016).*

- Colorization

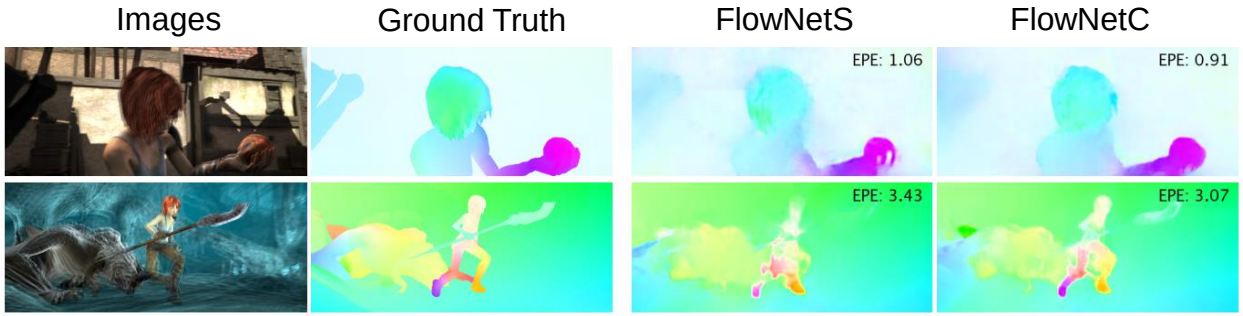
  Zhang et al.,“Colorful Image Colorization.“ *arXiv preprint arXiv:1603.08511* (2016).



What can CNNs do?

- Optical flow (FlowNet)

  Fischer, Philipp, et al. “Flownet: Learning optical flow with convolutional networks.” *arXiv preprint arXiv:1504.06852* (2015).



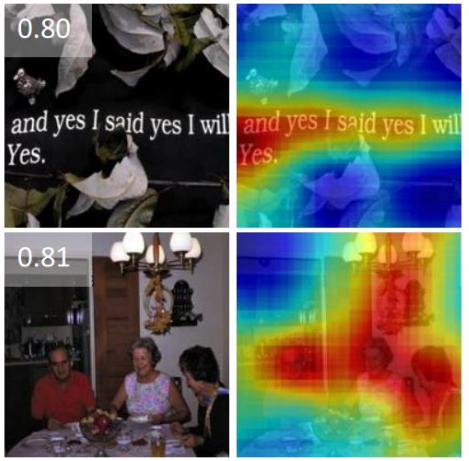
- DeepStereo (Google)

  Flynn et al. “DeepStereo: Learning to Predict New Views from the World's Imagery.” *arXiv preprint arXiv:1506.06825* (2015).

Result (synthesized) Reference

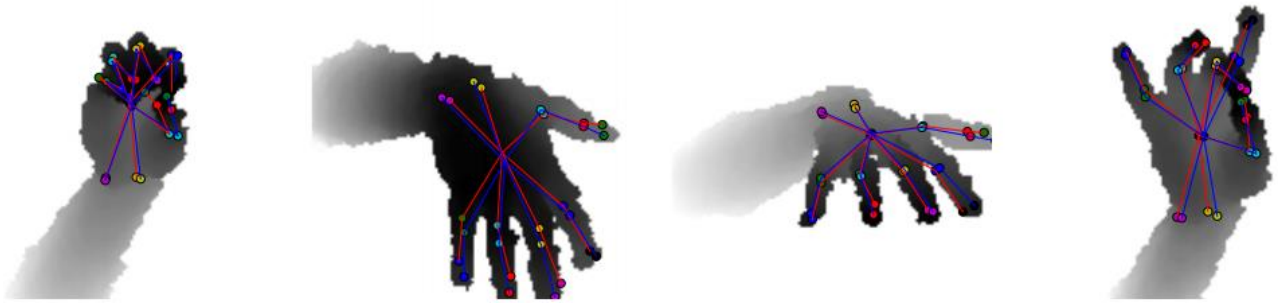
What can CNNs do?

- Image Memorability (MemNet)
  Khosla et al., “Understanding and predicting image memorability at a large scale.” *IEEE ICCV* (2015).



Input Image Memorability Map

- Hand pose estimation (HandNet)
  Oberweger et al., “Hands deep in deep learning for hand pose estimation.„ *arXiv preprint arXiv:1502.06807* (2015).



What can CNNs do?

- Human Pose Estimation

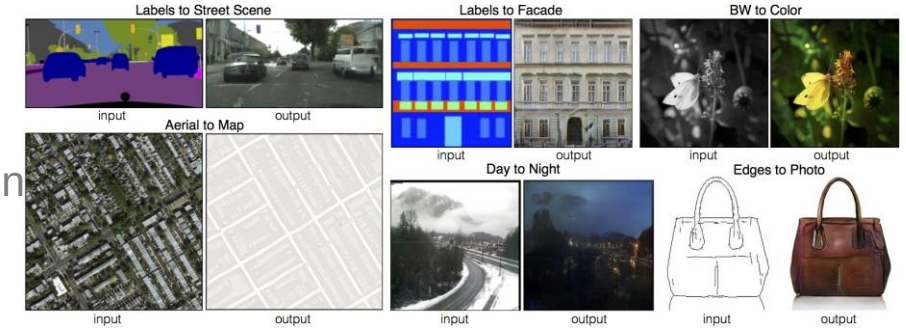
  Belagiannis et al. “Robust Optimization for Deep Regression.” *IEEE ICCV (*2015).

- Video Prediction

Vondrick et al., “Generating Videos with Scene Dynamics.” *NIPS (2016).*

- Adversarial Learning

  Isola et al.,“Image-to-Image Translation with Conditional Adversarial Nets“ *arXiv:1611.07004* (2016).



Currently „Famous“ People

Yoshua Bengio

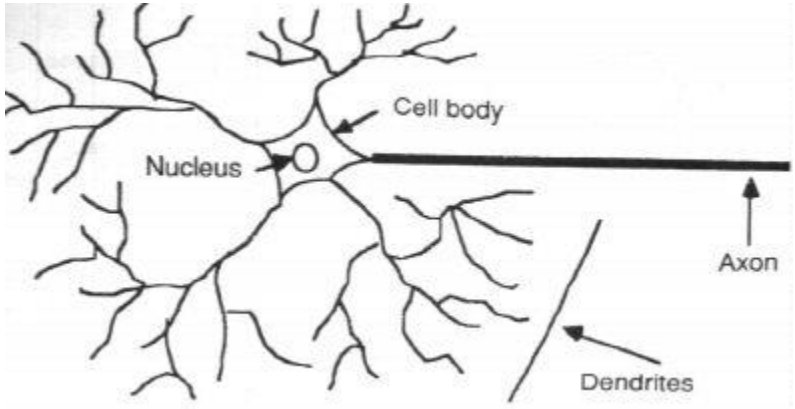
Geoffrey Hinton

Yann LeCun

Jürgen Schmidhuber

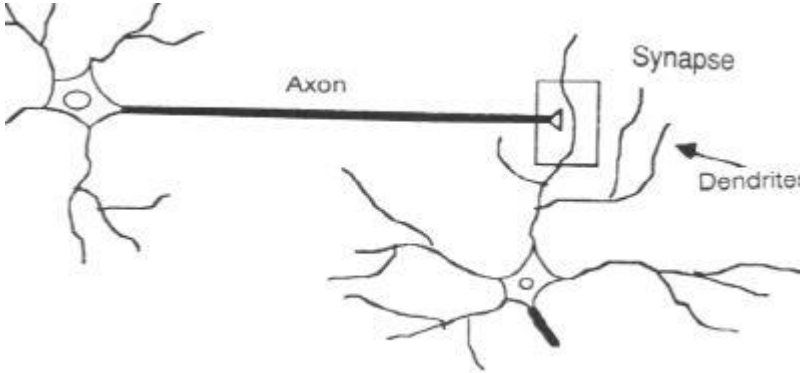
Title 1–14	Cited by	Year
Imagenet classification with deep convolutional neural networks A Krizhevsky, I Sutskever, GE Hinton Advances in neural information processing systems, 1097-1105	8476	2012

Neural Networks



a (biological) neuron collects signals from other neurons through its dendrites

the neuron‘s output signal is sent through the axon that is connected via synapses to the dendrites of other neurons



Artificial Neural Networks (ANNs)

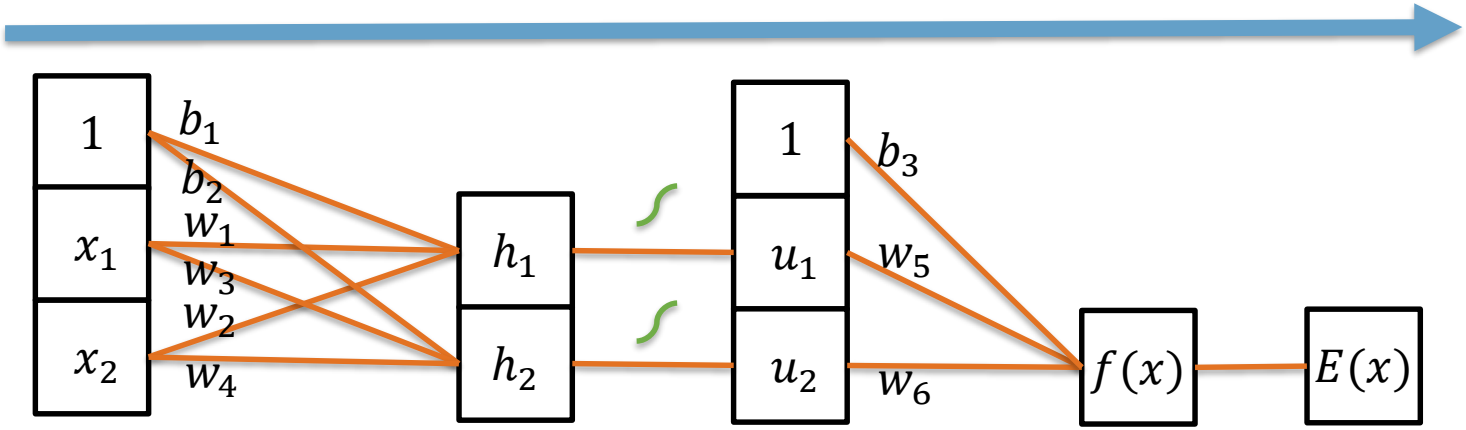
"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."

Dr. Robert Hecht-Nielsen in "Neural Network Primer: Part I" by Maureen Caudill, Feb. 1989

How To Train Your Network

- Set of N samples $\left(x^{(i)}, y^{(i)}\right)$
- Define loss $\mathrm{E}(x) \in \mathbb{R}$ to measure error for a sample
- Training: finding weights that minimize $\sum_i E(x^{(i)})$ for the samples

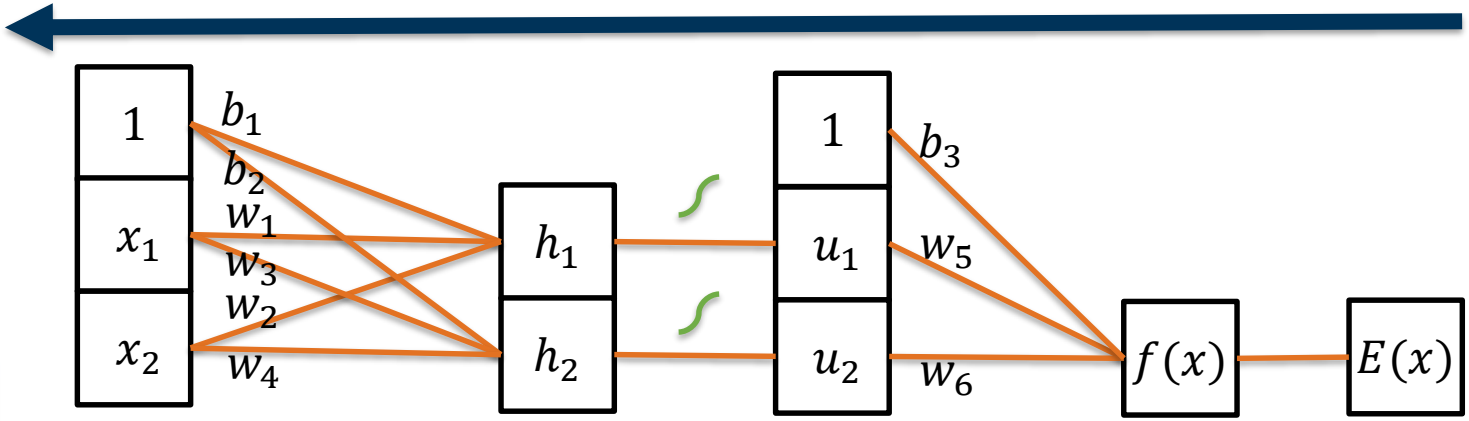
- **Forward pass:** Compute all activations and estimate $f(x)$



How To Train Your Network

- Set of N samples $\left(x^{(i)}, y^{(i)}\right)$
- Define loss $\mathrm{E}(x) \in \mathbb{R}$ to measure error for a sample
- Training: finding weights that minimize $\sum_i E(x^{(i)})$ for the samples

- **Backpropagation:** a smart way to compute gradients for all parameters, using the (stored) activations and chain rule for derivatives.



Gradient Descent

- Initialize all weights (randomly, close to solution, …)

- How to use the derivatives?

  - Gradient descent to minimize error function

$$w^{(t+1)} = w^{(t)} - \lambda \frac{\partial E}{\partial w}$$

- Update the weights in every iteration of training with a small gradient step
- Learning rate λ adjusts the step-size

Stochastic Gradient Descent

- Error was defined over the whole training set: $\sum_i E(x^{(i)})$
- Need to compute and sum the derivatives of *all* samples before gradient step
- Slow but accurate updates

- Stochastic Gradient Descent (SGD): approximate derivative from small, random subset ([mini]batch) of training set
- Noisy but faster
- Usually: make sure to see every sample the same amount of times (epochs)

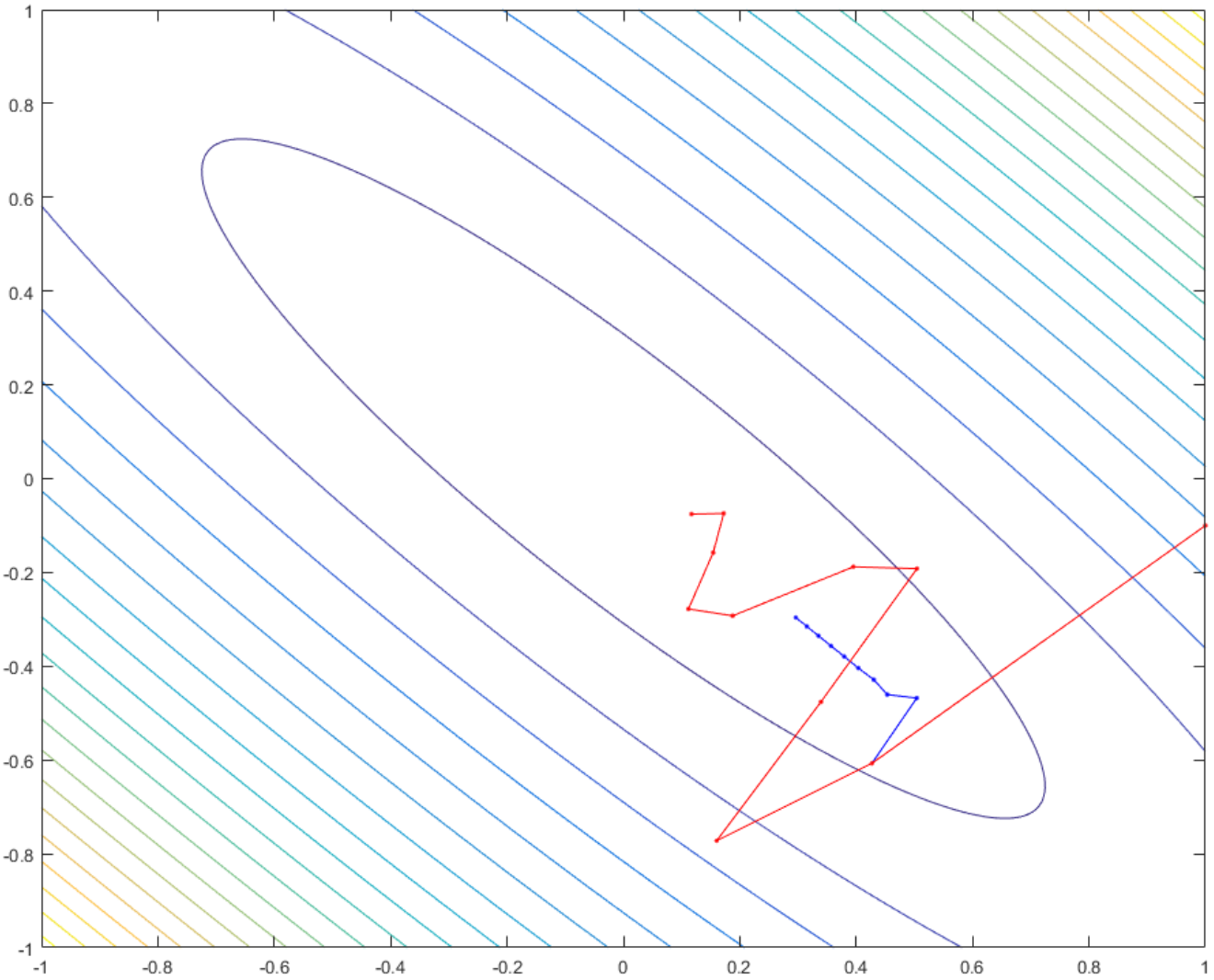
Momentum

- Narrow ravines in error function make gradient descent methods oscillate
- Slow convergence
- Momentum: velocity v that accumulates gradients

$$v^{(t+1)} = \alpha v^{(t)} + \lambda \Delta E$$
$$w^{(t+1)} = w^{(t)} - v^{(t+1)}$$

- $0 \leq \alpha < 1$: how much velocity is kept each step (usually between 0.5 and 0.9)
- Also accelerates learning on plateaus

Momentum

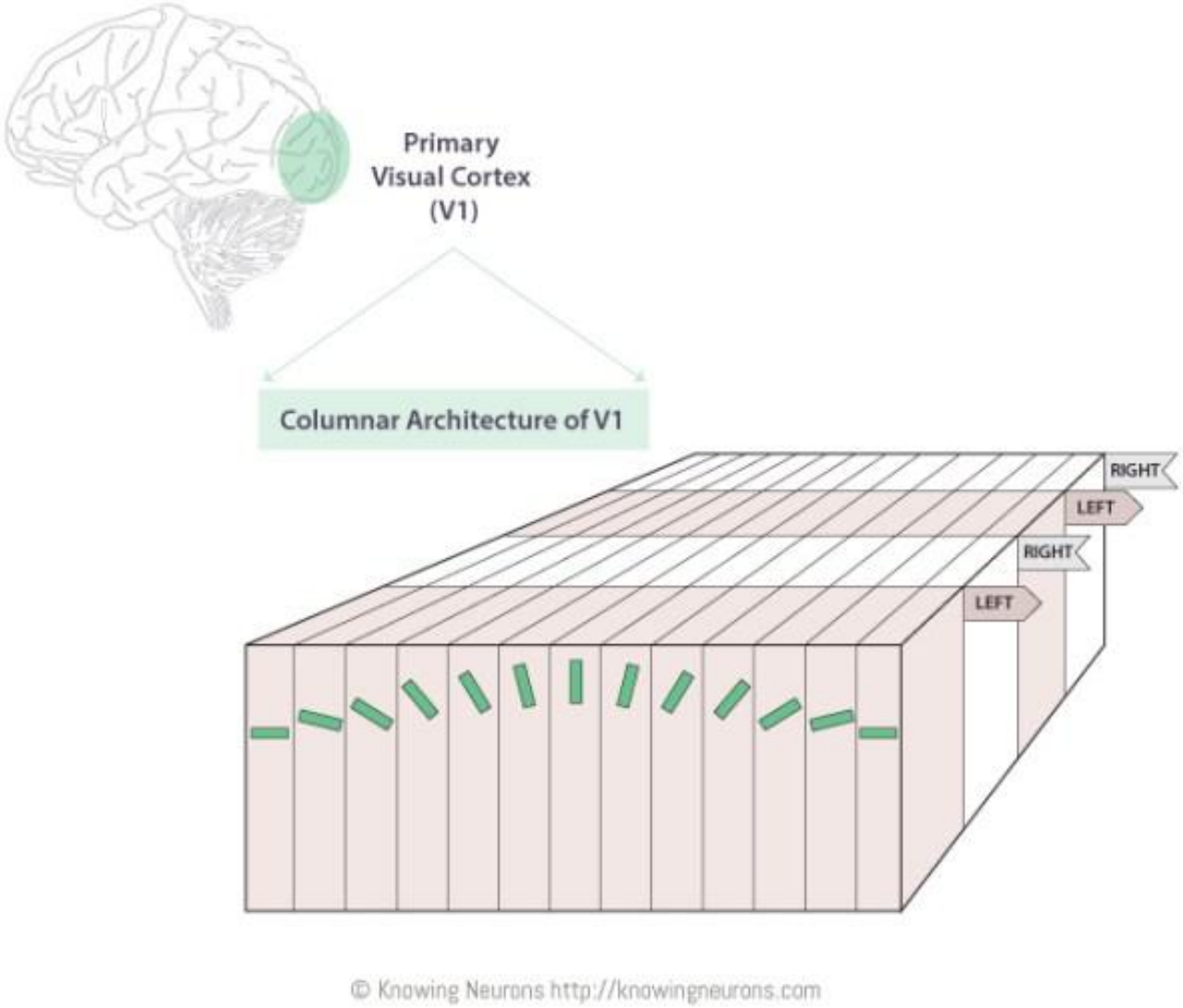


Towards CNNs

https://www.youtube.com/watch?v=IOHayh06LJ4

Hubel et al. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." *The Journal of physiology* (1962)

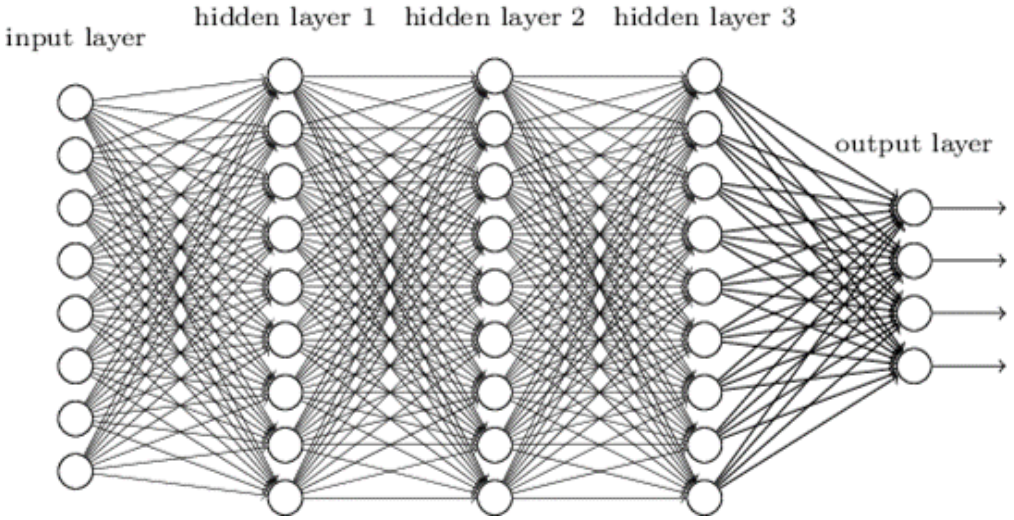
Towards CNNs



From ANN to CNN

- ANNs
  - Learn complex mappings from examples
  - Learn features instead of engineering features
  - Suitable for image recognition
  - Full connections among all neurons

What happens when the input is an image?



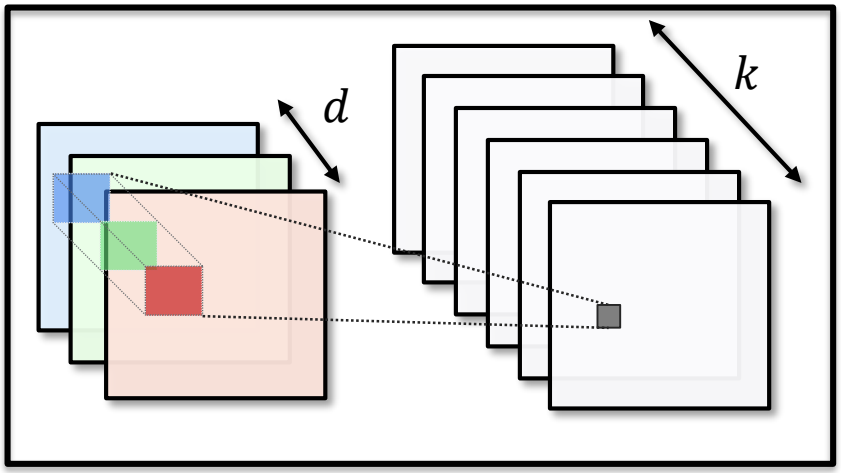
From ANN to CNN

- ANNs
  - Large inputs → Exploding number of parameters
    - Need for large data sets
    - Memory requirements
  - No invariance with respect to translations or local distortions
  - Similar weight patterns appear for different locations of the input
  - Topology of the input is ignored

- CNNs
  - Replication of weight configuration over space (*shared weights*)
  - Local connections (*receptive fields*) – Hubel and Wiesel
  - Sub-sampling

Convolutional Neural Networks

- **Convolutional Layer**



- Convolutional filter $n \times m \times d$
- Learnable parameters
- k filters → k feature maps
- Local connectivity
- Shared weights

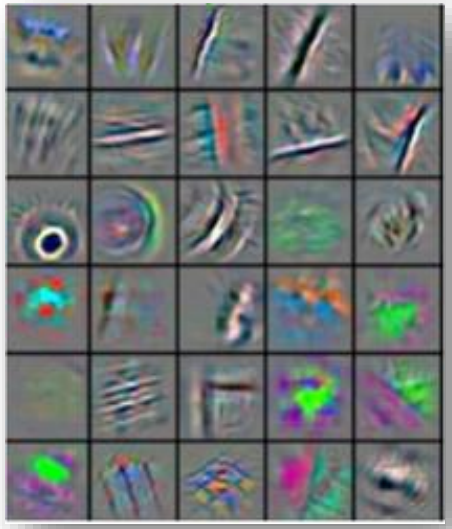
Hyper-parameters: number and size of filters, padding, stride (step)

Convolutional Neural Networks

- **Convolutional Layer**

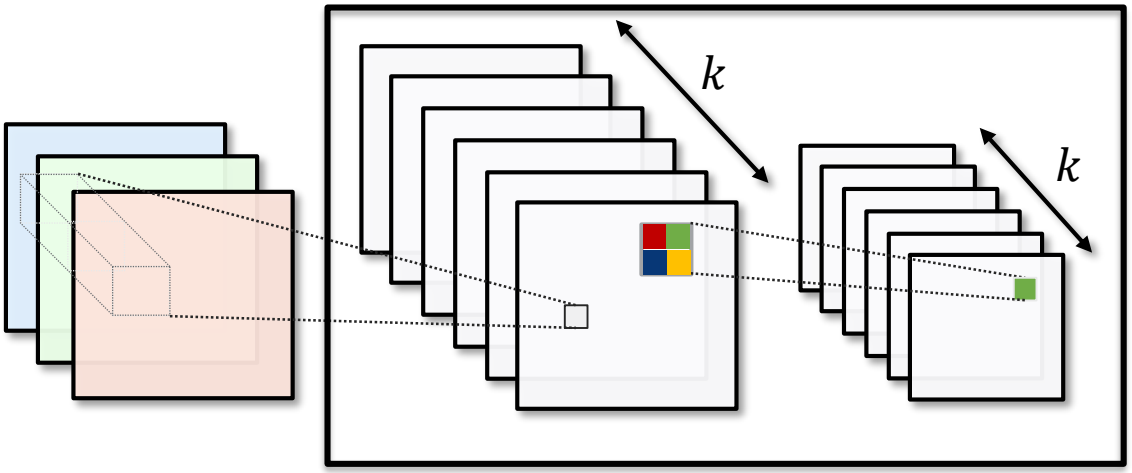
  Stacking convolutional layers → higher levels of abstraction

Learned filters:



Convolutional Neural Networks

- **Pooling Layer**

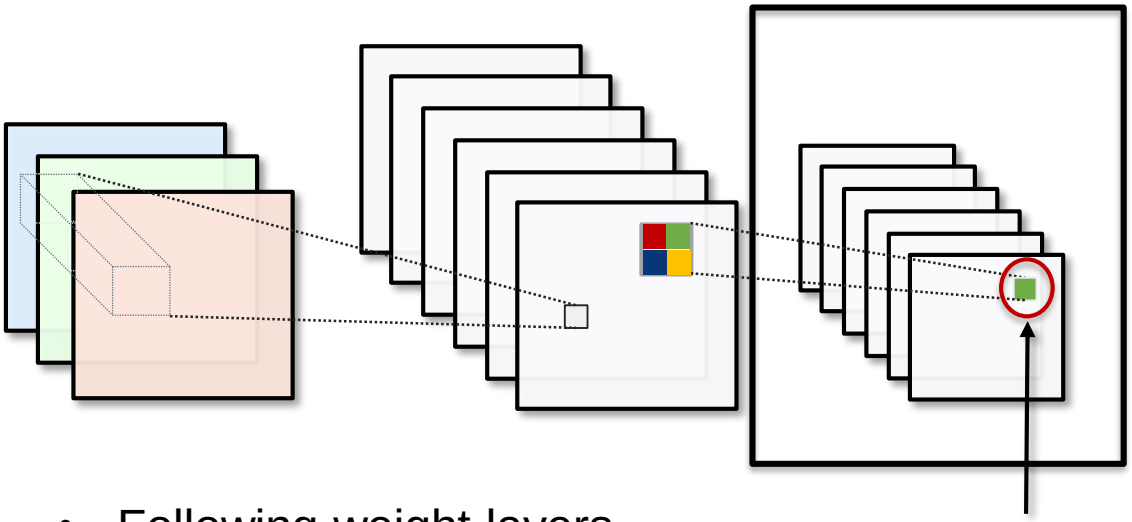


- Max or average operation
- Parameter-free
- Reduce dimensionality
- Invariance to translations

Hyper-parameters:
operation
window size,
padding,
stride (step)

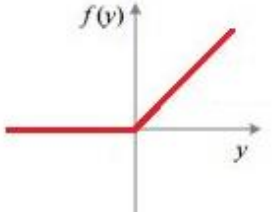
Convolutional Neural Networks

- **Activation Function**



- Following weight layers
- In CNNs typically use Rectified Linear Units (ReLU)
- Non-linearity → necessary

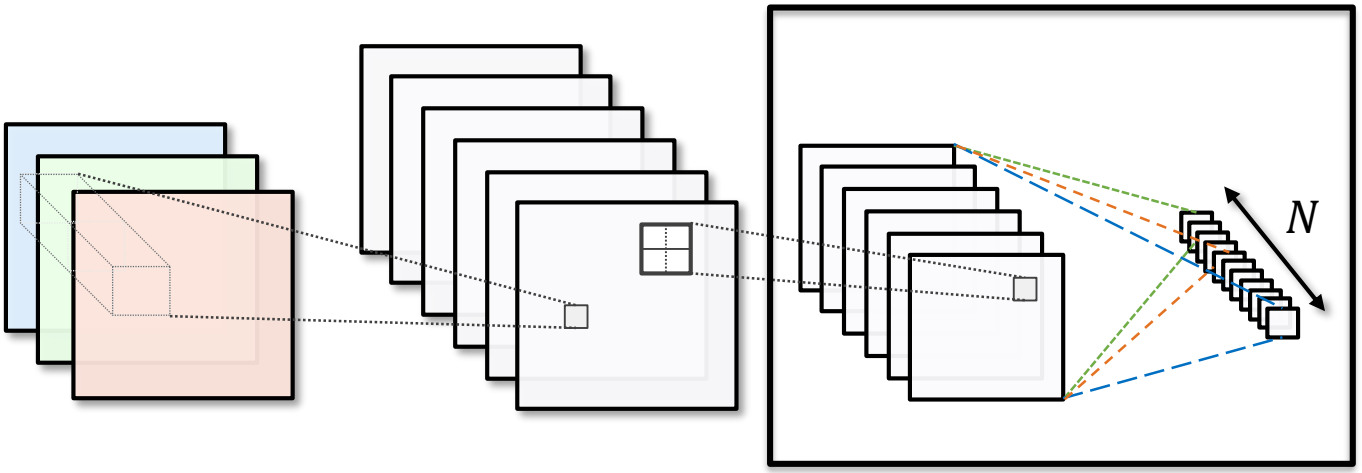
$$f(y) = \max(0, y)$$



Convolutional Neural Networks

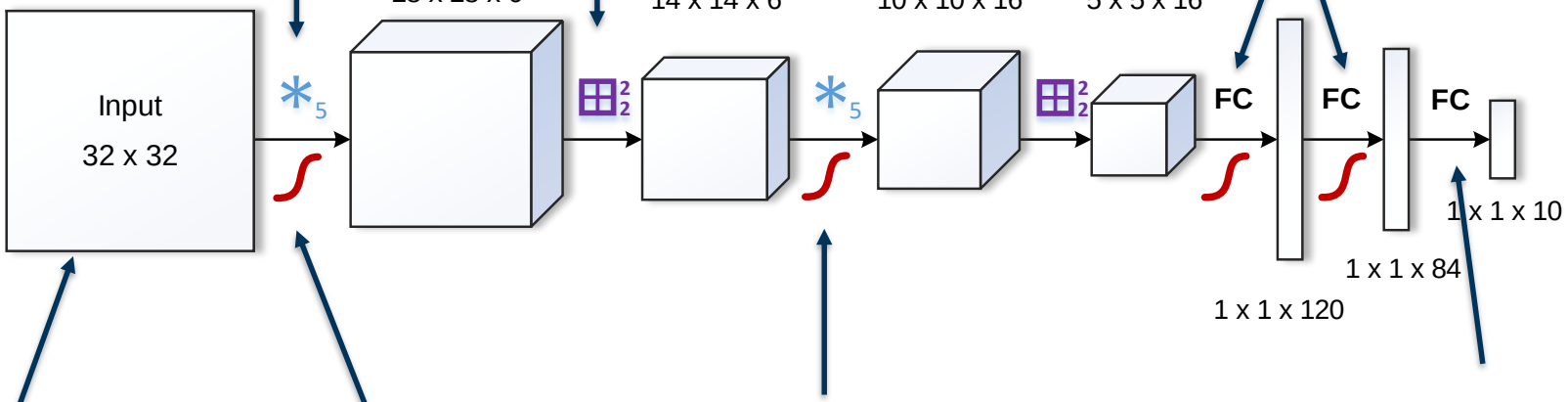
- **Fully-connected layer**

FC



- *All* neurons of the previous layer connect to a *single output neuron*
- Can be seen as a convolution with the same size as the previous layer
- Usually at the end of the network

LeNet – 5



handwritten digit

sigmoid activation $(tanh)$

connects only to a subset of the 6 feature maps of the previous layer

modeled as Radial Basis Function (RBF) units:

$$y_i = \sum_{j=1}^{84} \left(x_i - w_{ij}\right)^2,$$

i classes (total 10)

j inputs per class (total 84)

LeCun et al., “Gradient-Based Learning Applied to Document Recognition”, Proc. IEEE, 1998

Parameter Initialization

- Proposed methods for weight initialization:
  Random Gaussian distribution with zero mean and

  1. fixed variance (e.g. 0.01)
  2. Xavier initialization: $Var(w) = \frac{2}{n_{in} + n_{out}}$,

     n: number of neurons connecting to given neuron

     Glorot et al., "Understanding the difficulty of training deep feedforward neural networks." *International conference on artificial intelligence and statistics.* 2010.
  3. Latest simplification: $Var(w) = \frac{2}{n_{in}}$

     (suitable for rectifying linear activation functions)

     He et al., "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." *IEEE ICCV* (2015).

Hyperparameters

- Adjusting learning rate, weight decay
  - Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” Neural Networks: Tricks of the Trade, Springer, 2012, pp. 437–478
- Batch size, epochs
  - Depends on problem, data, network
  - also on network size & current hardware
- Losses, (global) minima, (higher order) optimization

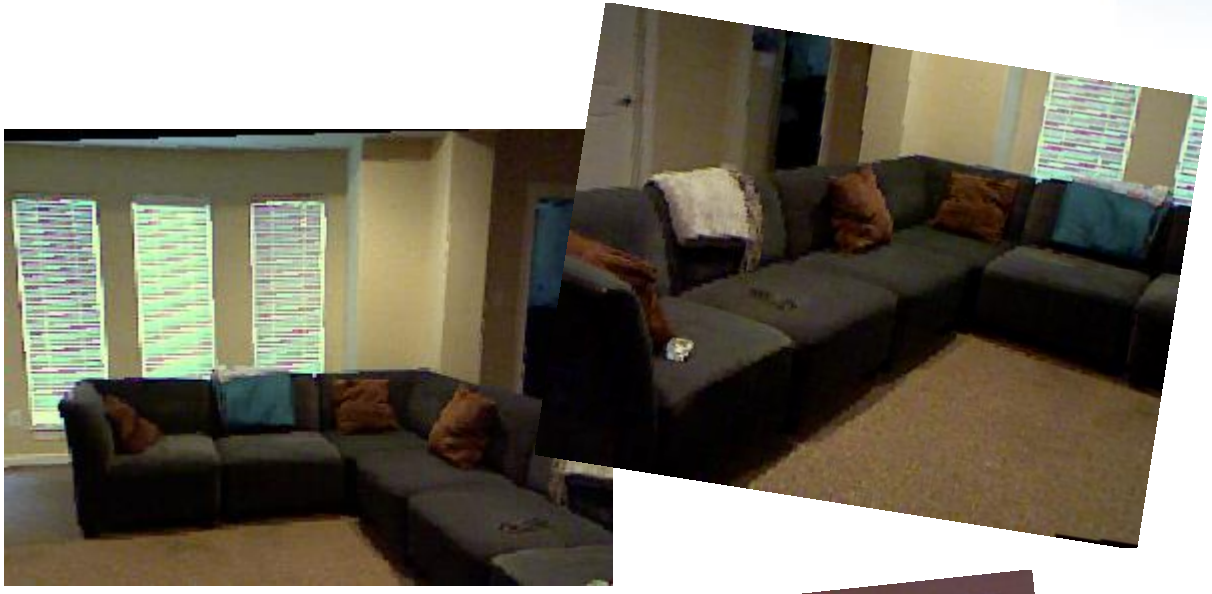
  https://www.cs.nyu.edu/~yann/talks/lecun-20071207-nonconvex.pdf

  Choromanska et al., “The Loss Surfaces of Multilayer Networks”, arxiv, 1412.0233v3, 2015
- Rule of thumb: the higher the number of parameters, the more data is required.

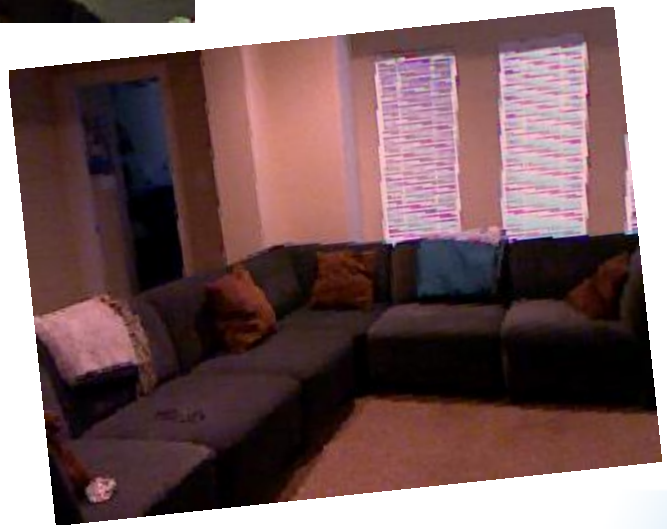
Data Augmentation

Randomly

- rotate
- crop
- shift colors
- change contrast
- …

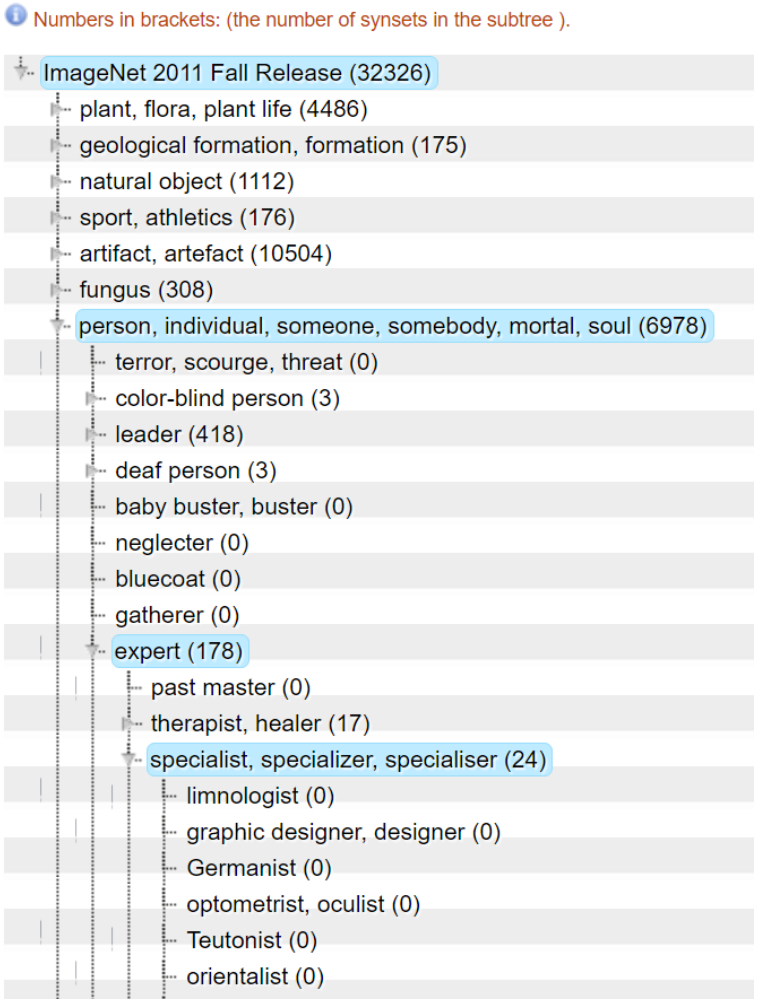


of the image to artificially increase training set size and to learn these invariances.



ImageNet

- Collection of images
- Hierarchical structure from WordNet
- Yearly: Large Scale Visual Recognition Challenge (ILSVRC)
- Since 2012: CNNs win everything

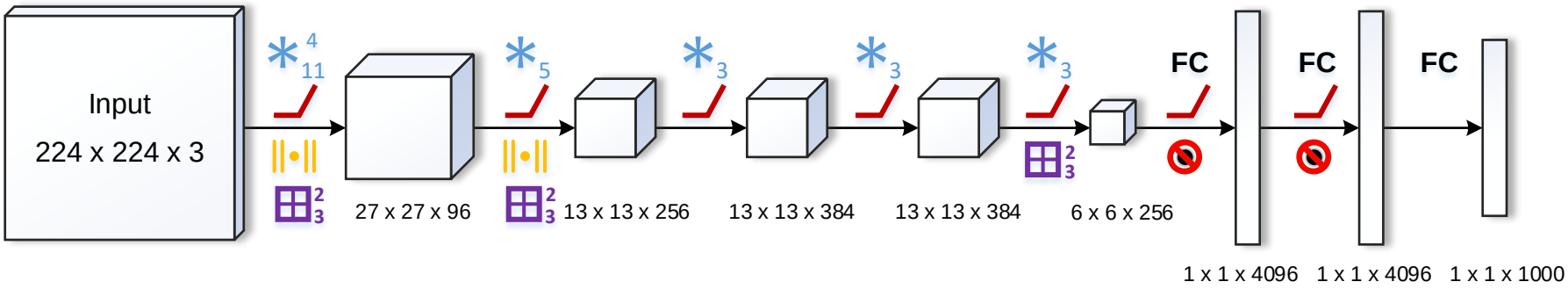


Learning from ILSVRC

What can we learn from the entries to this challenge?

- Winners usually from large groups or big companies (Stanford, Oxford, Google, Microsoft, …)
- High computational power, hardware, etc.
- We can assume they tried many different variants and submitted the best
- Usually publicly accessible models
- So far: every year an innovation (by a different group)

AlexNet (2012)



(s) n nxn Convolution (stride s)
(s) n nxn max-Pool (stride s)
(s) n nxn avg-Pool (stride s)
LRN
Batch Norm
ReLU
Dropout

Krizhevsky et al., “Imagenet classification with deep convolutional neural networks”, NIPS12

AlexNet (2012)

- Made modern CNNs popular
- Architectural details
  - 60 million parameters
  - Trained in parallel on two GPUs with custom code
  - ReLU instead of sigmoid
  - Overlapped pooling (3x3, stride 2)
  - Local Response Normalization (LRN)
- Reduce overfitting
  - Dropout
  - Data augmentation

top-1 test error: 36.7%
top-5 test error: 15.3%

Krizhevsky et al., “Imagenet classification with deep convolutional neural networks”, NIPS12

Local Response Normalization

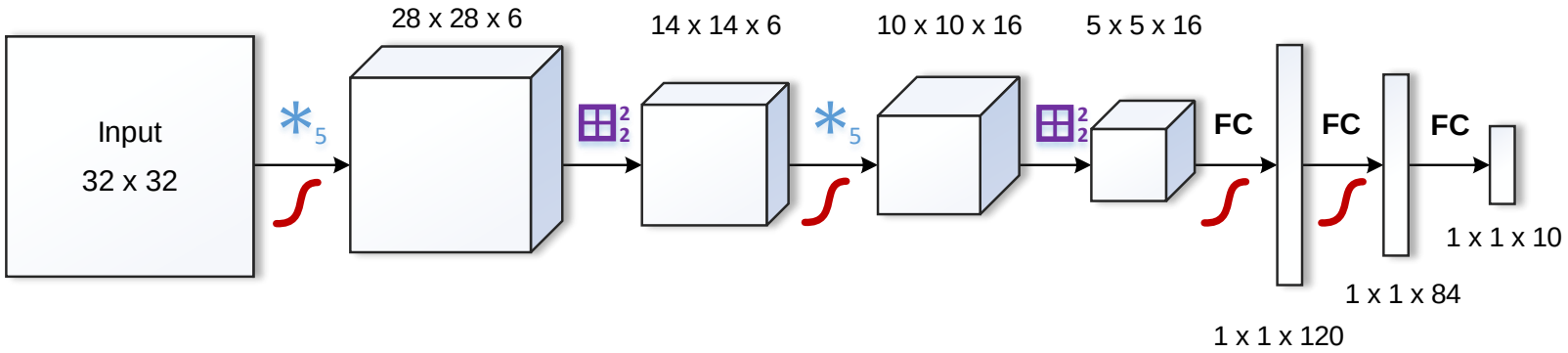
Improves generalization

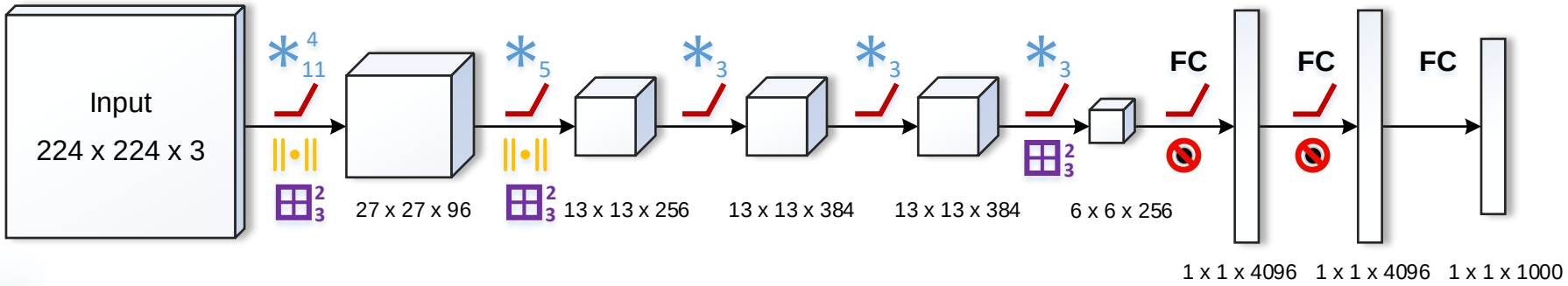
$$LRN\left(a^{(l)}\right) = a^{(l)}\left(k + \alpha \sum_{j \in adj(l,n)} \left(a^{(j)}\right)^2\right)^{-\beta}$$

$$k = 2, \qquad n = 5, \qquad \alpha = 10^{-4}, \qquad \beta = \frac{3}{4}$$

Activation a in feature map l, looking at n adjacent feature maps j at this spatial location.

AlexNet vs. LeNet - 5





(s) n nxn Convolution (stride s)
(s) n nxn max-Pool (stride s)
(s) n nxn avg-Pool (stride s)
LRN
Batch Norm
ReLU
Dropout

VGG (2014)

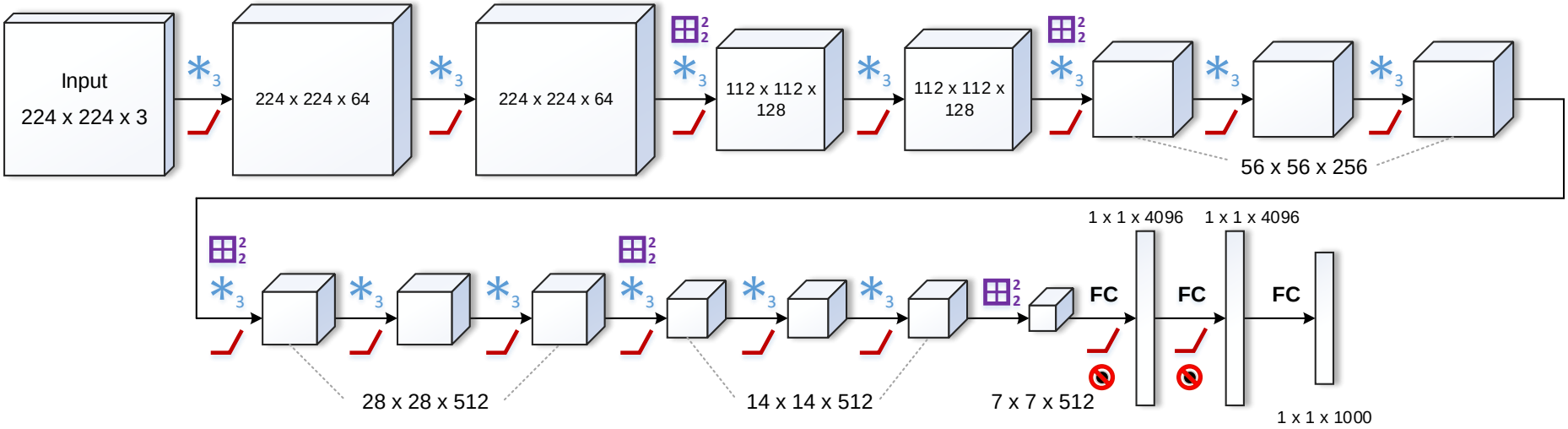
- Oxford
- Much deeper (11 to 19 convolutional layers)
- 133+ million parameters (double compared to AlexNet)

- Only small convolutions (3x3)
- No LRN (does not improve)
- More, but smaller convolutions
  - increase receptive fields
  - more non-linearities

top-1 test error: 23.7%
top-5 test error: 6.8%

Simonyan et al., “Very deep convolutional networks for large-scale image recognition”, *arXiv preprint arXiv:1409.1556*(2014).

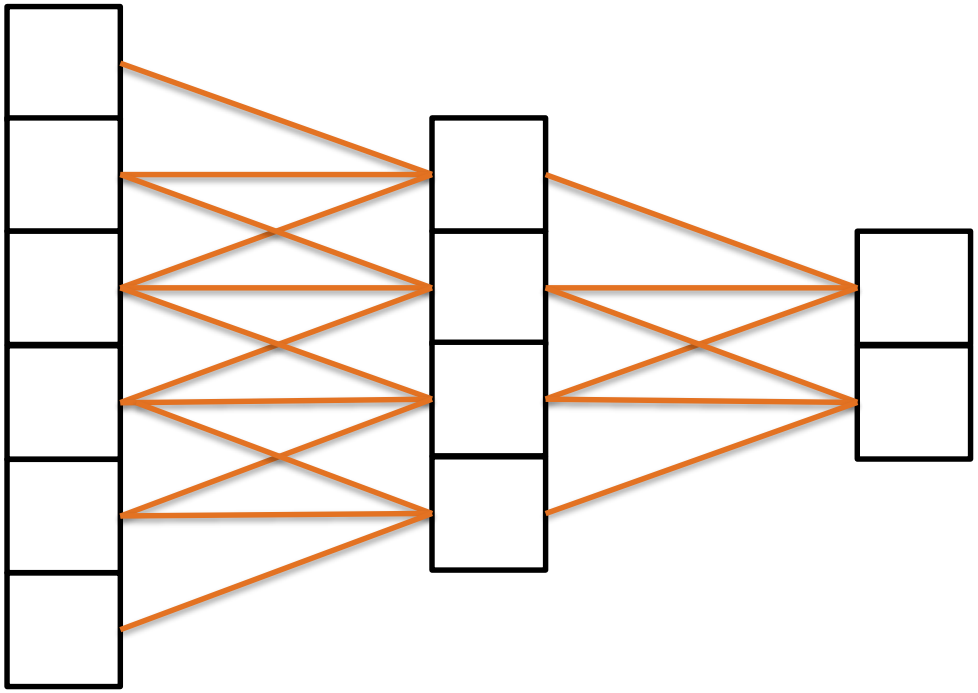
VGG – 16



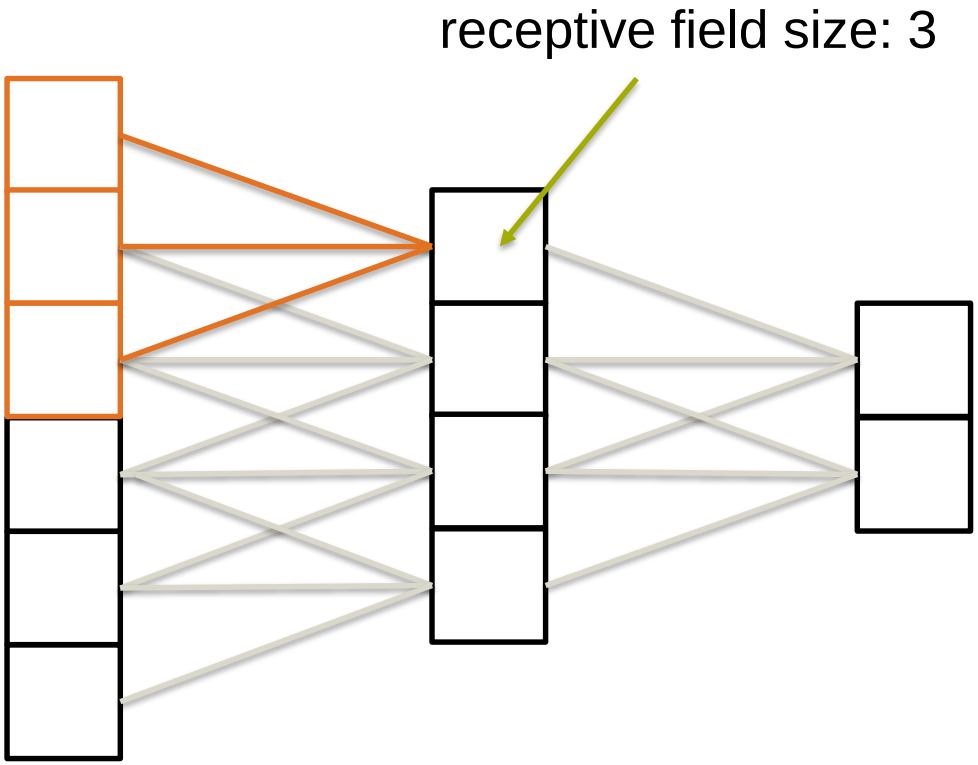
(s) n nxn Convolution (stride s) | (s) n nxn max-Pool (stride s) | (s) n nxn avg-Pool (stride s) | LRN | Batch Norm | ReLU | Dropout

Simonyan et al., "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*(2014).

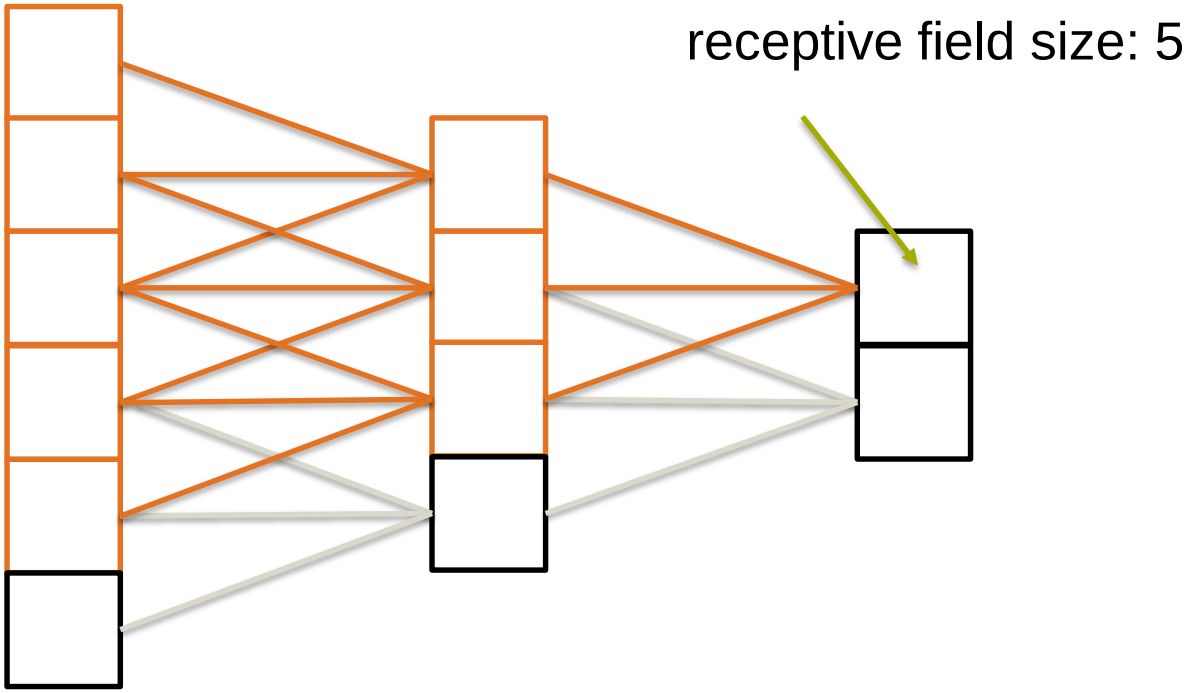
Receptive Field



Receptive Field



Receptive Field



Receptive Field

- Size of the maximal area in an earlier layer that a neuron can receive inputs from
- Convolutions increase the receptive field by half their filter size (no stride)
- Fully-connected layers ensure a full receptive field
- Receptive field sizes give us an upper bound on the size of concepts/patterns/objects the CNN can learn (before the fully connected layer)

- AlexNet (before FC): 151 x 151
- VGG (before FC): 276 x 276

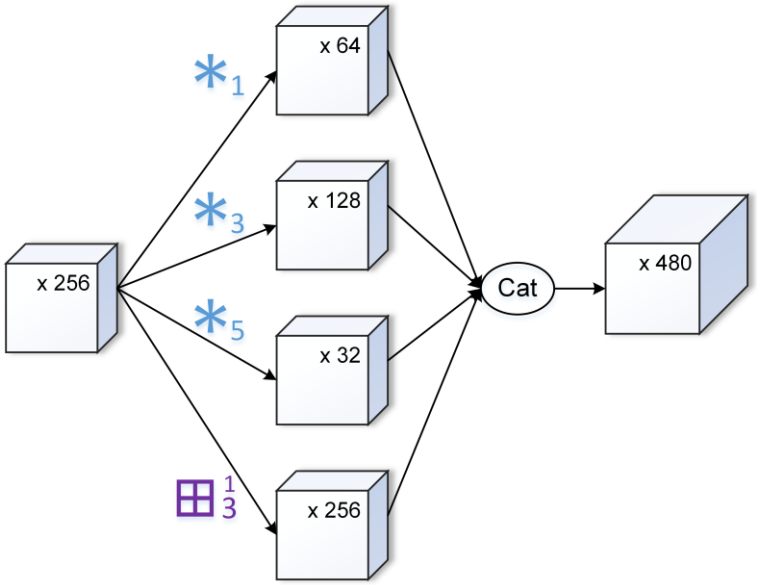
GoogLeNet (2014)

- Increasing depth (22 layers) and width (Inception module)
- 9x fewer parameters than AlexNet (6.8 million)
- 1x1 convolutions to reduce number of filters
- Adding auxiliary classification losses
  - encourage discrimination at lower stages of the network
  - increase the gradient signal that gets back-propagated
  - provide additional regularization

top-5 test error: 6.67%

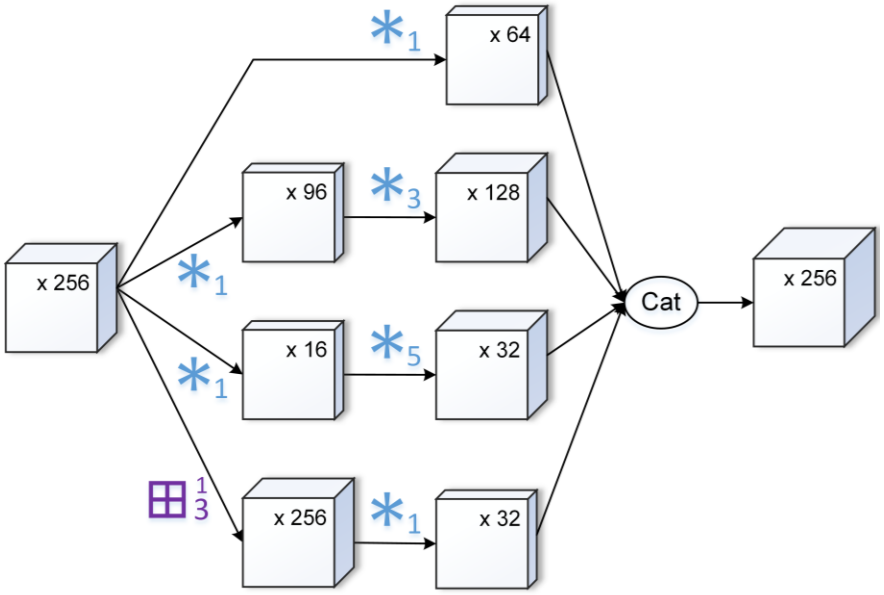
Szegedy et al., “Going deeper with convolutions”, CVPR15

Simple Inception Blocks



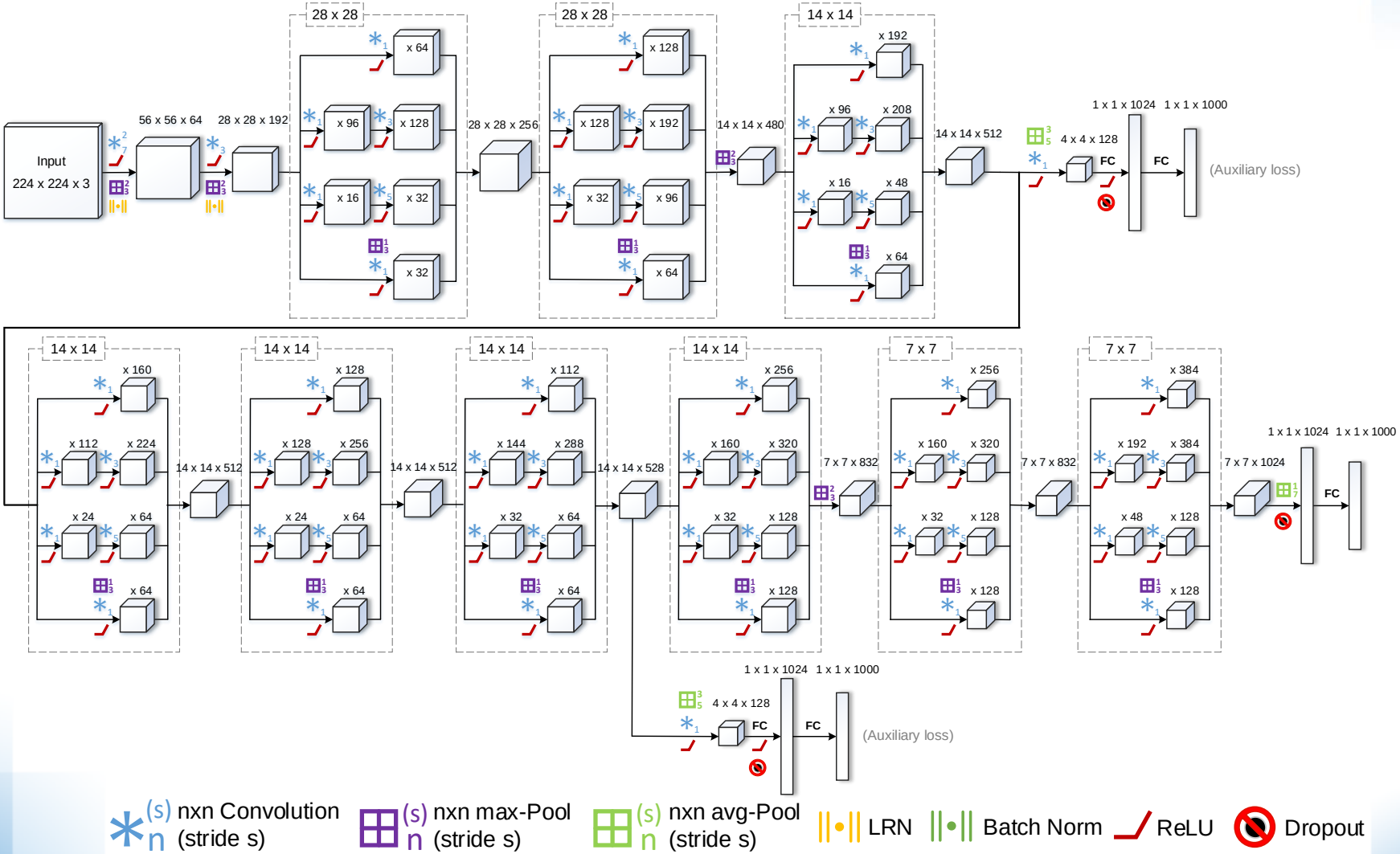
- More width (filters) by concatenating different sized convolutions (1x1 is 25 times less expensive than 5x5)
- “visual information should be processed at various scales”

Inception Blocks



- Same idea: use 1x1 convolution to reduce number of channels for larger, more expensive convolutions
- save parameters while increasing depth

GoogLeNet



Szegedy et al., “Going deeper with convolutions”, CVPR15

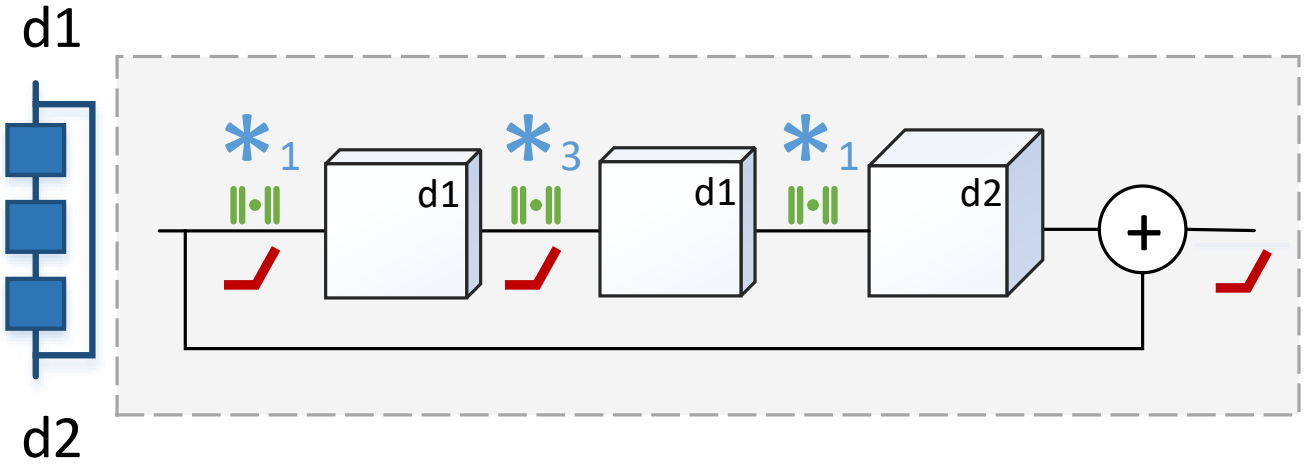
ResNet (2015)

- Residual learning
  - "It is easier to optimize a residual mapping than the original unreferenced mapping."
- Identity mapping: parameter-free shortcut connection
  - idea: $g(x) + x$
- Projection shortcuts (with 1x1 conv): when downsampling
- Batch normalization solves vanishing gradients problem
- No dropout
- Up to 1000 layers

top-5 test (ResNet-152): 4.49%
top-5 test (ensemble): 3.57%

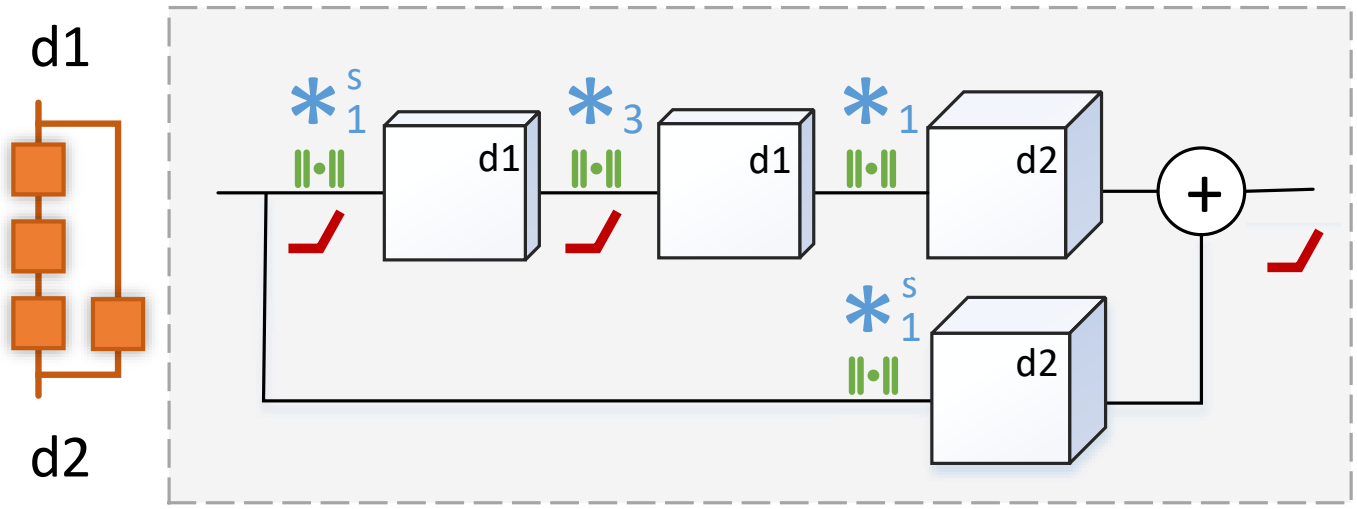
He et al., "Deep Residual Learning for Image Recognition", *arXiv preprint arXiv:1512.03385* (2015).

Shortcut Blocks



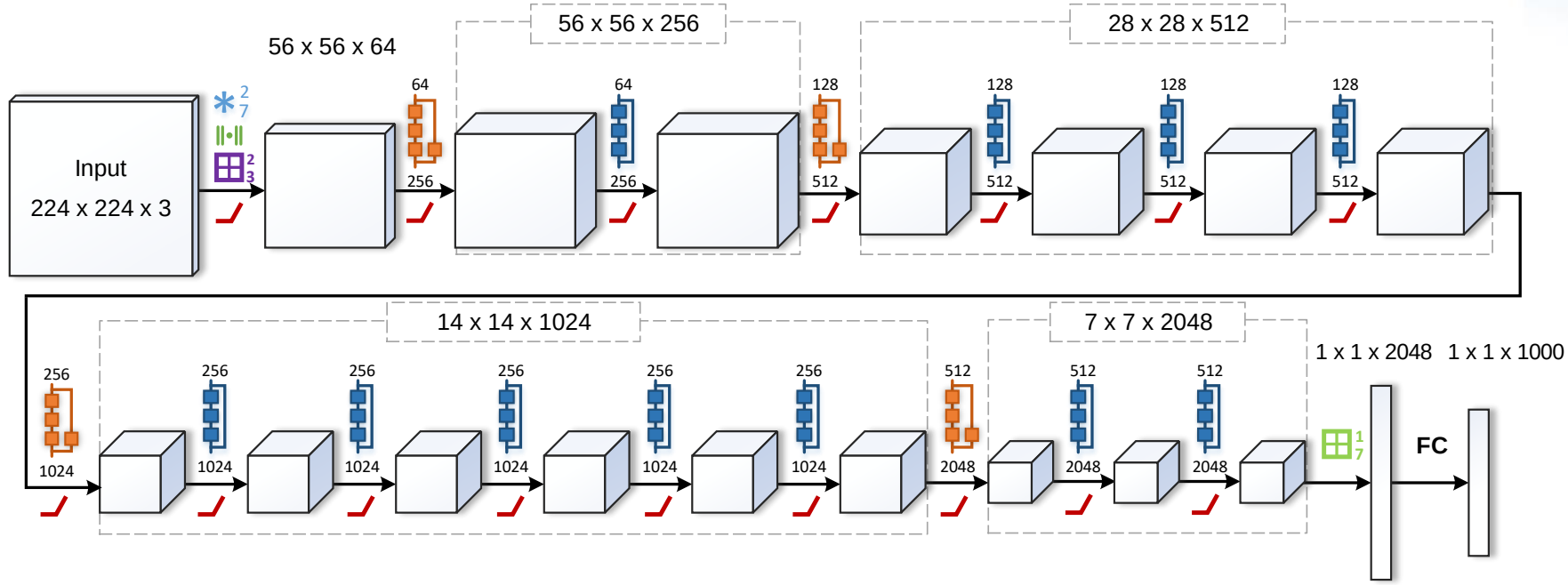
- The first 1x1 conv. reduces the number of filters, the second increases it again
- Makes 3x3 conv. less heavy on resources
- Each convolution is followed by batch normalization

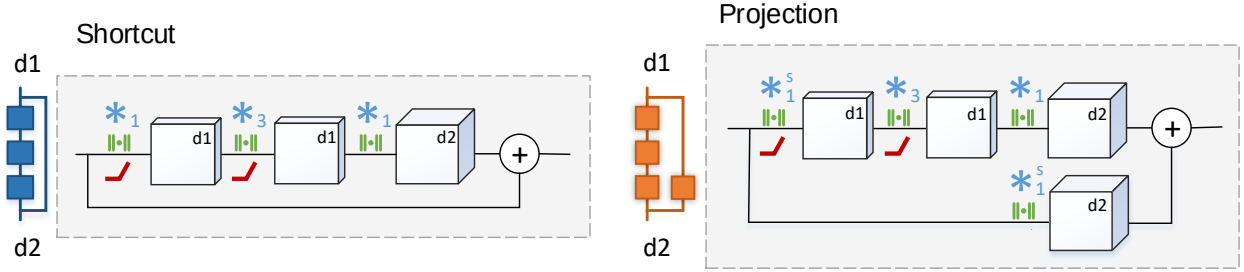
Projection Blocks



- Used for downsampling
- First convolution in each branch is with stride
- Can also change the number of channels

ResNet – 50





(s) n nxn Convolution (stride s) | (s) n nxn max-Pool (stride s) | (s) n nxn avg-Pool (stride s) | LRN | Batch Norm | ReLU | Dropout

He et al., "Deep Residual Learning for Image Recognition", *arXiv preprint arXiv:1512.03385* (2015).

Batch Normalization

Normalize activations of a neuron over the current batch

For every neuron activation x individually over the batch $x^{(1)} \dots x^{(m)}$:

1. Mean: $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$
2. Variance: $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \mu\right)^2$
3. Normalize: $\hat{x}^{(i)} = \frac{x^{(i)}-\mu}{\sqrt{\sigma^2+\epsilon}}$
4. Scale and Shift: $y^{(i)} = \gamma\hat{x}^{(i)} + \beta$

Learned parameters, for every neuron: γ, β

Ioffe et al., "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015)

Batch Normalization

- Normalize over the batch instead of the feature map
- Mean subtraction cancels biases of previous layer
- Reintroduced through offset β
- Also learn scaling γ
- Before ReLU

- *Increase learning rate*
- *Remove Dropout*
- *Reduce L2 weight regularization*
- *Accelerate learning rate decay*
- *Remove Local Response Normalization*
- *Shuffle training examples more thoroughly*

Ioffe et al., "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015)

What did we learn?

- Every year the CNNs became deeper
- Innovations how to make training possible
  - save parameters
  - batch normalization
  - skip connections

- Keep checking arxiv for current trends
  - Szegedy et at., "*Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*", arXiv:1602.07261, Feb 23, 2016
  - He et al., "*Identity Mappings in Deep Residual Networks*", arXiv:1603.05027, Mar 16, 2016

Feature Visualization

Feature Visualization

- Input Reconstruction
  - The network as black-box
  - Using the output
  - Computationally expensive
- Deconvolution
  - The network as white-box
  - Using the inherent network structure
  - Computationally cheap
- Input modification
  - The network as black-box
  - Using the input
  - Varying costs

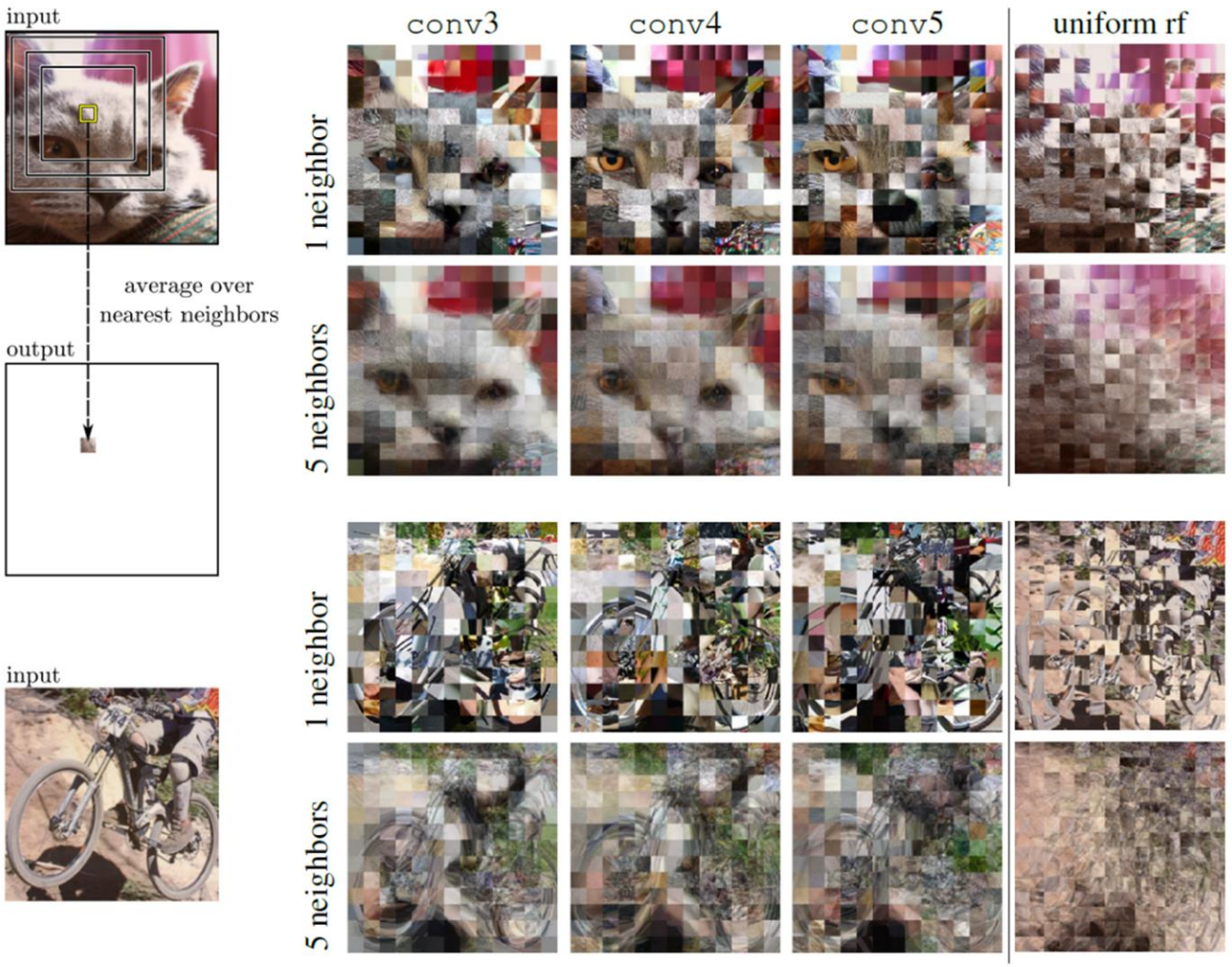
Input Reconstruction

Find input(s) (patches), that match the desired output

Do Convnets Learn Correspondence? (Long et al., NIPS14)

- Do forward pass compute activations
- For each unit in a layer find other inputs the best reproduce the same activation
- Compute average
- Visualize patch-based

Do Convnets Learn Correspondence?

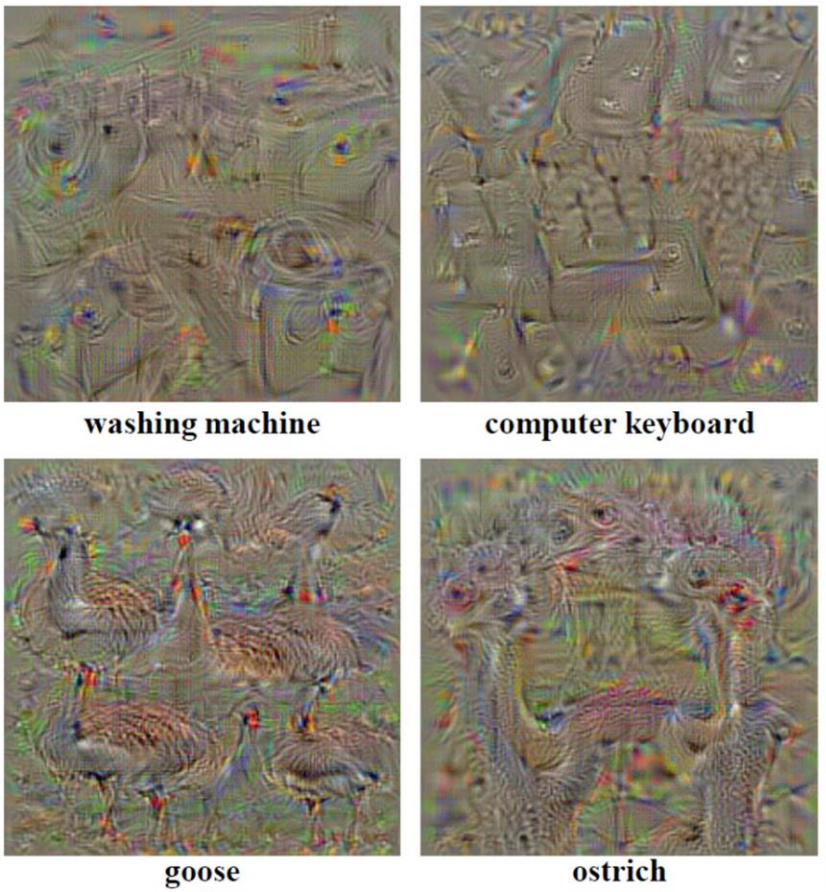


Long, J. et al.,"Do Convnets Learn Correspondence?", NIPS 2014

Deep Inside Convolutional Networks Visualising Image Classification Models and Saliency Maps

- Choose desired output
- Input noise Image
- Back-propagation errors into the input image
- Gradient step in image
- Repeat

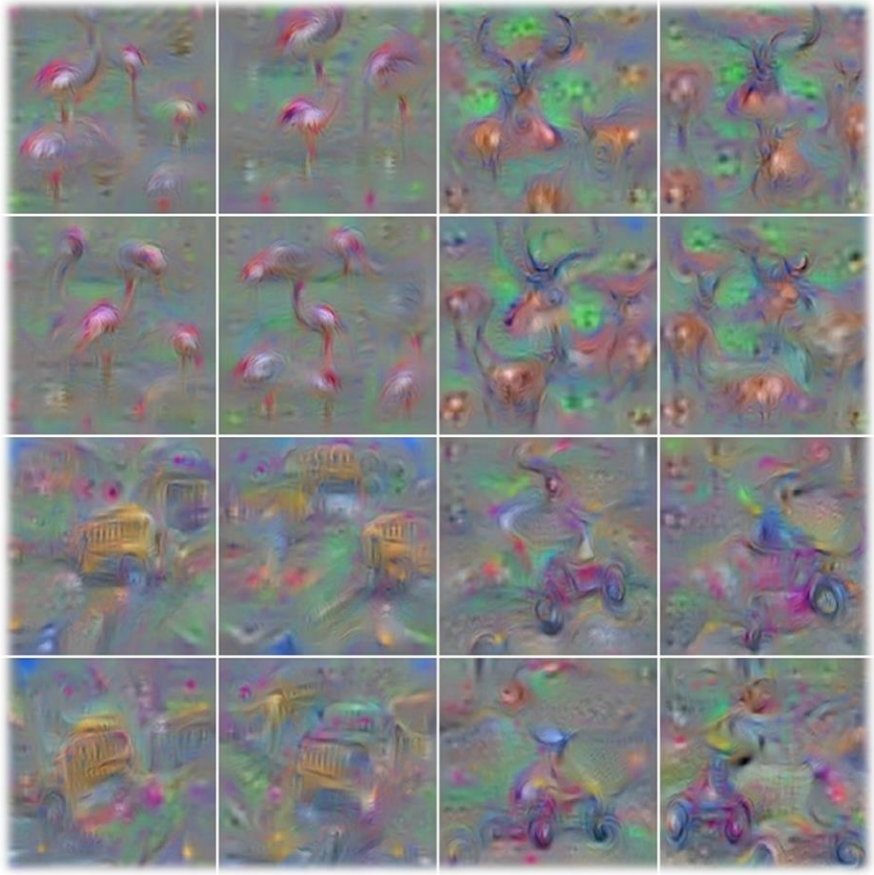
- Similar: Mahendran, A. et al., „*Understanding Deep Image Representations by Inverting Them*”, CVPR 2015



Simonyan, K., “Deep Inside Convolutional Networks Visualising Image Classification Models and Saliency Maps”, arXiv:1312.6034, 2013

Deep Inside Convolutional Networks Visualising Image Classification Models and Saliency Maps

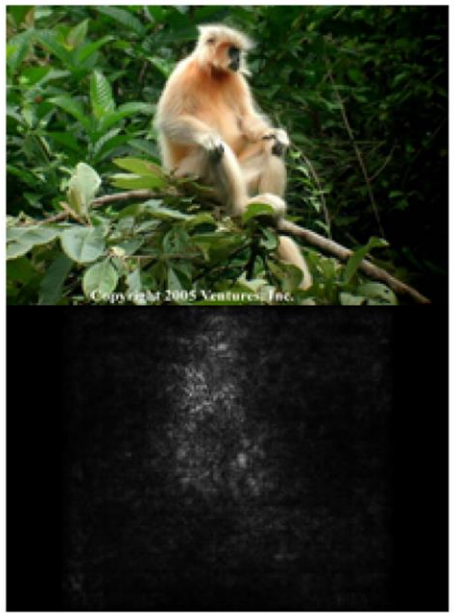
- Reconstruction quality depends strongly on regularizers

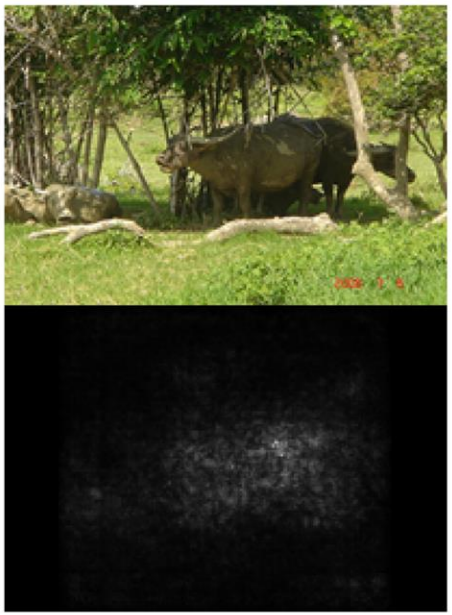


Yosinski, J., “Understanding Neural Networks through Deep Visualization”, arXiv:1506.06579, 2015

Deconvolution

- Back-propagate activations until the input image and visualize them there

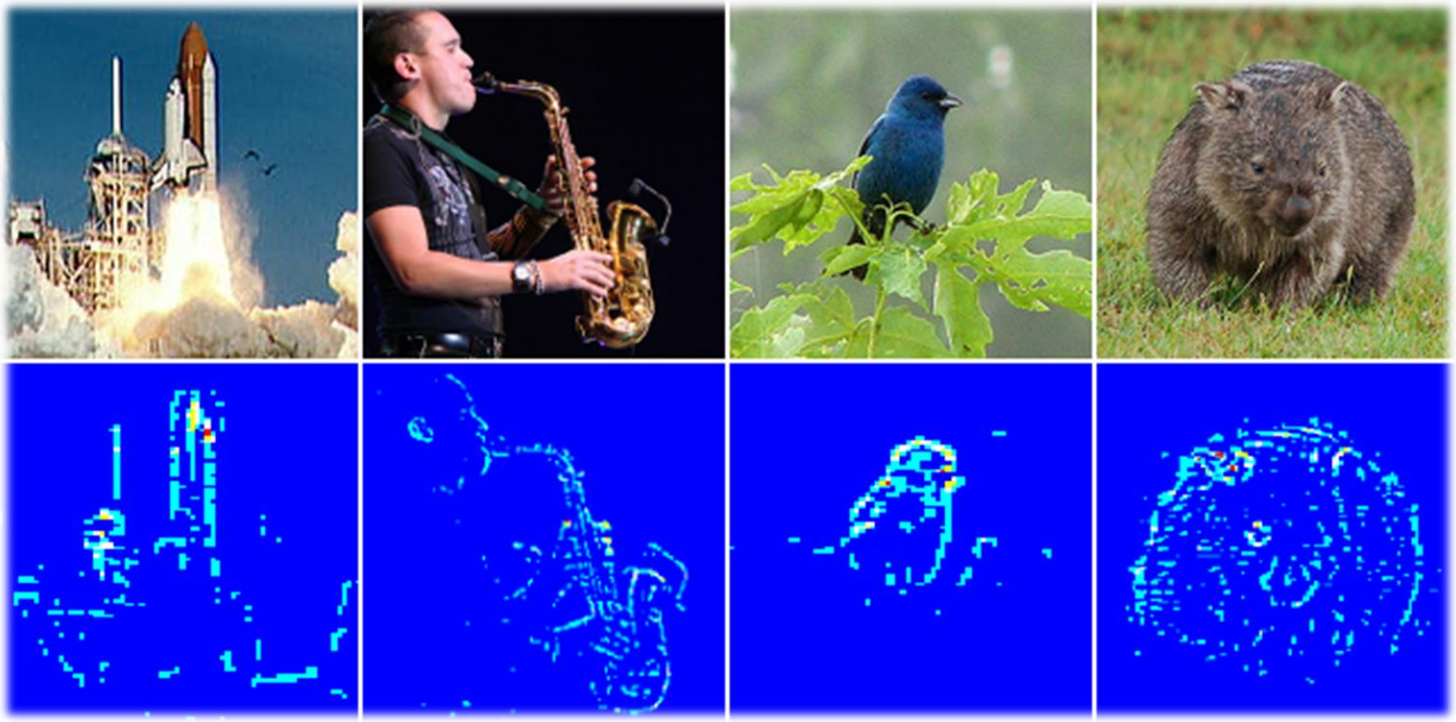




Simonyan, K., “Deep Inside Convolutional Networks Visualising Image Classification Models and Saliency Maps”, arXiv:1312.6034, 2013

Deconvolution

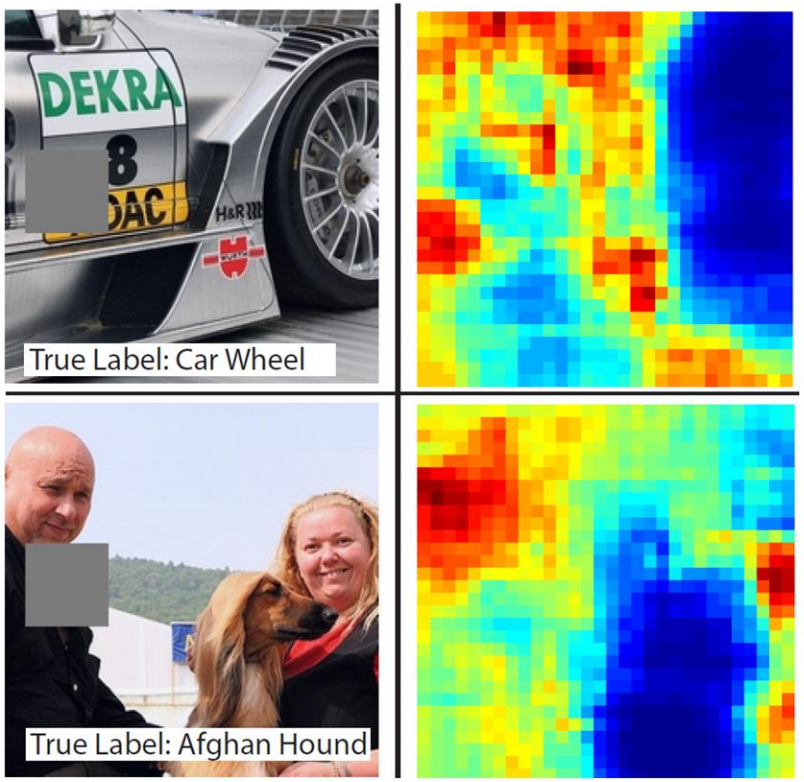
- Guided packpropagation changes the way pooling is traversed



Springenberg, J.T., “Striving for Simplicity: The all Convolutional Net”, arXiv:1412.6806, 2014

Input Modification

- Modify input and analyse changes in output
- Slide grey patch over input and measure difference in output
- Generates heat-map of importance



Zeiler, M.D. et al., “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Feature Visualization

- Helps understanding what a network has learned
- Easy to implement
- Many of them work also for regression

- also allow artistic escapades

https://github.com/google/deepdream

- see also

Gatys et al., „A Neural Algorithm of Artistic Style”, arxiv: 1508.06576, 2015

THANK YOU

QUESTIONS?