Deep learning: Convolutional Neural Networks (CNN)

Announcements

- A2 grades will be uploaded tonight
- A3 due on December 2nd
- Optional A4 will be released this week (it will be due during finals week)
- Final exam is on December 16
- Let us know if there is any outstanding regrade request
- Bump in points for A0 readme

Totals

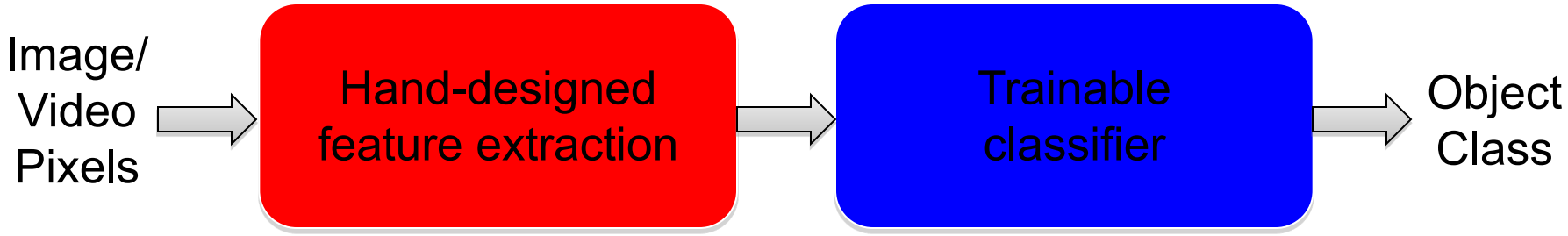
22.87 / 33

31.50	/33
31.00	/33
30.50	/33
30.00	/33
30.00	/33
30.00	/33
29.50	/33
29.50	/33
29.50	/33
29.00	/33
29.00	/33
28.50	/33
28.50	/33
28.00	/33
28.00	/33
28.00	/33
28.00	/33
28.00	/33
28.00	/33
28.00	/33
27.50	/33
27.50	/33
27.50	/33
27.50	/33
27.00	/33

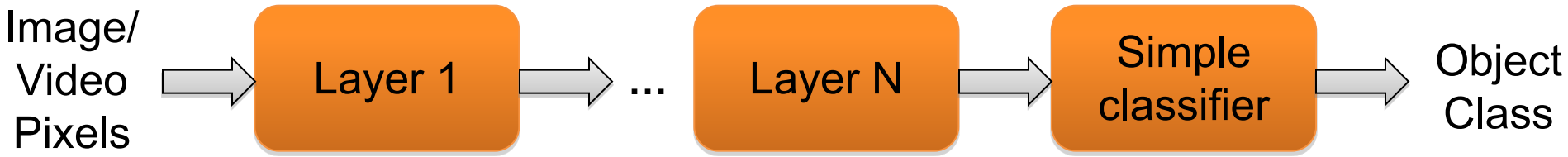
First place: automatic A+
Second and third place: bump in the grade (e.g. A to A+)

“Shallow” vs. “deep” learning

“Shallow” architecture



Deep learning: “Deep” architecture



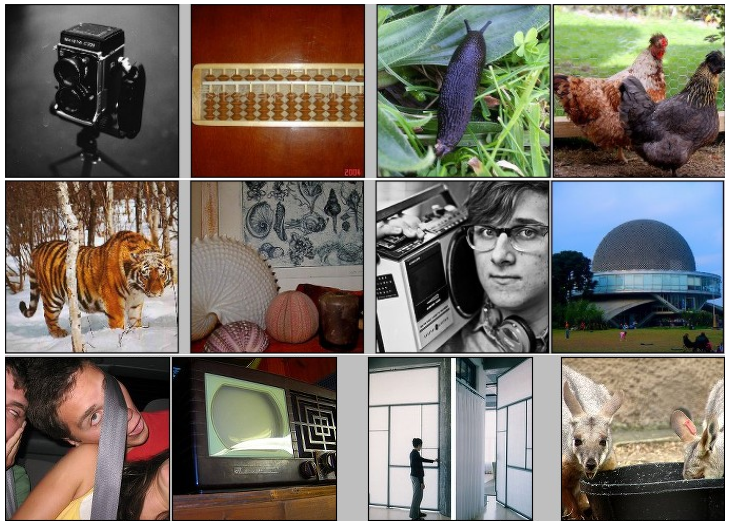
Slide credit: Rob Fergus

Regularization and optimization in Neural Networks

- Regularization and optimization in Neural Networks
  - Parameter sharing: e.g. Convolutional Neural Networks (CNNs)
  - Weight decay
  - Dropout
  - Early stopping
  - Change of learning rate during learning
  - Adding momentum
  - Dataset augmentation
  - Activation function selection
- Based on:
  - Deep learning book (http://www.deeplearningbook.org/), chapter 9 (CNN)

ImageNet Challenge 2012

IMAGENET



[Deng et al. CVPR 2009]

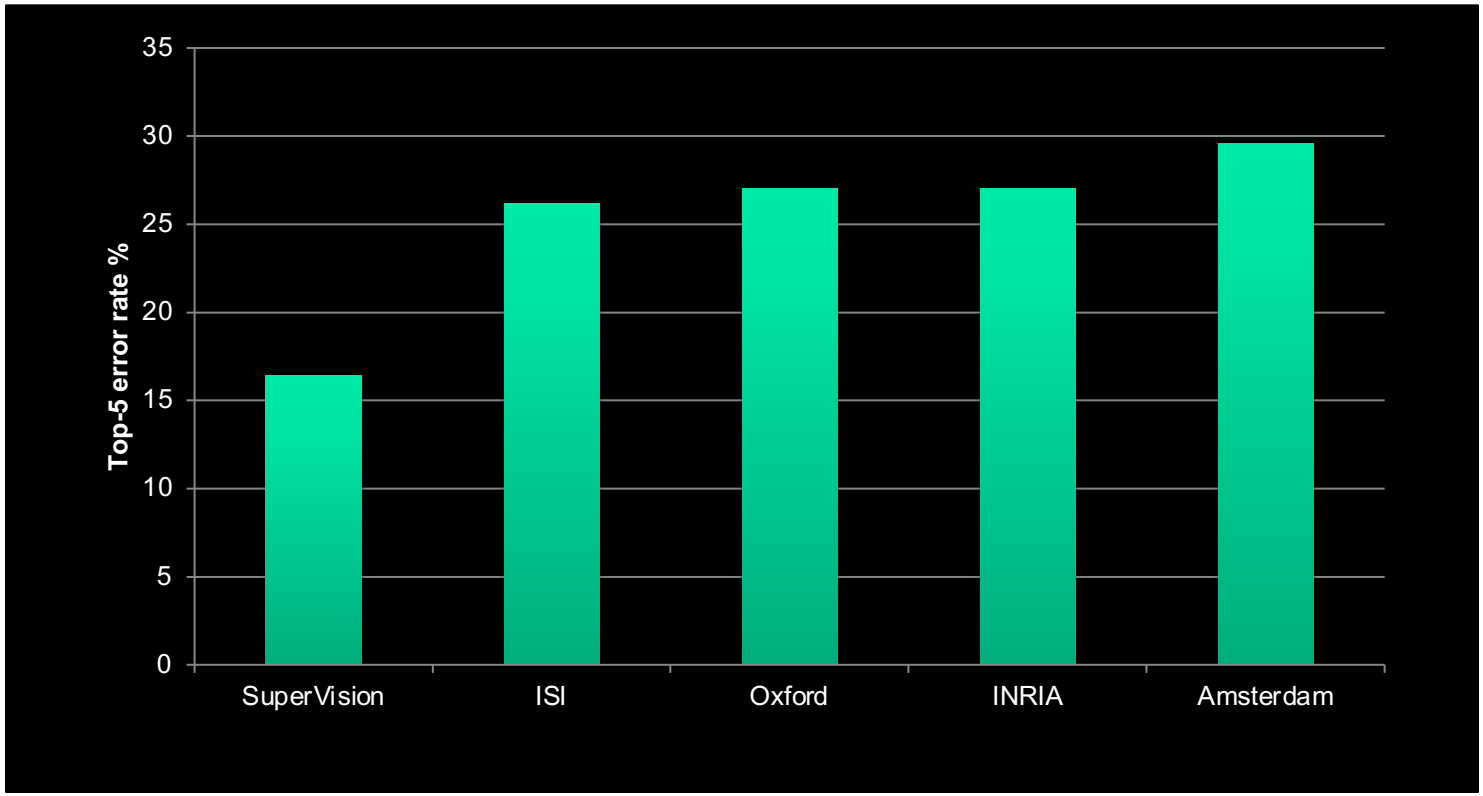
- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk
- Challenge: 1.2 million training images, 1000 classes

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Slide credit: Rob Fergus

ImageNet Challenge 2012

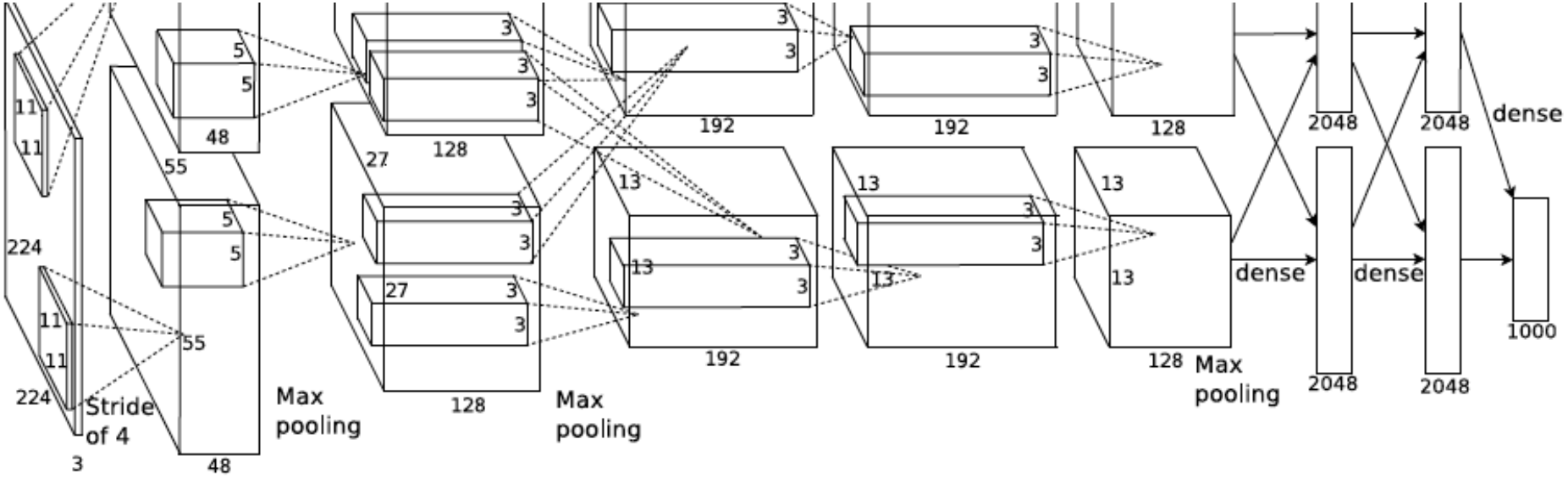
- Krizhevsky et al. -- **16.4% error** (top-5)
- Next best (non-convnet) – **26.2% error**



Slide credit: Rob Fergus

ImageNet Challenge 2012

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
  - Better regularization for training (DropOut)



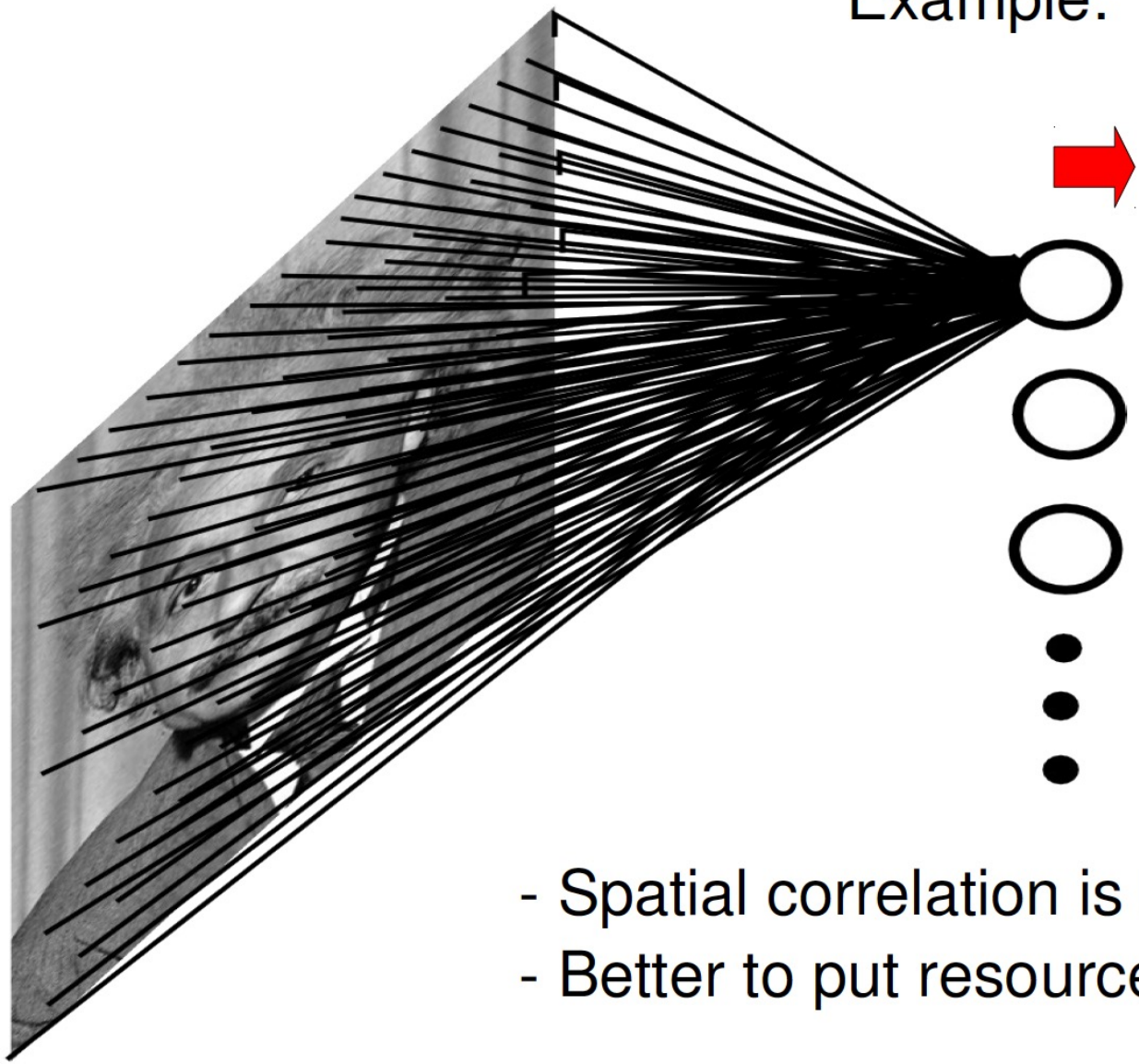
A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

FULLY CONNECTED NEURAL NET

Example: 1000x1000 image

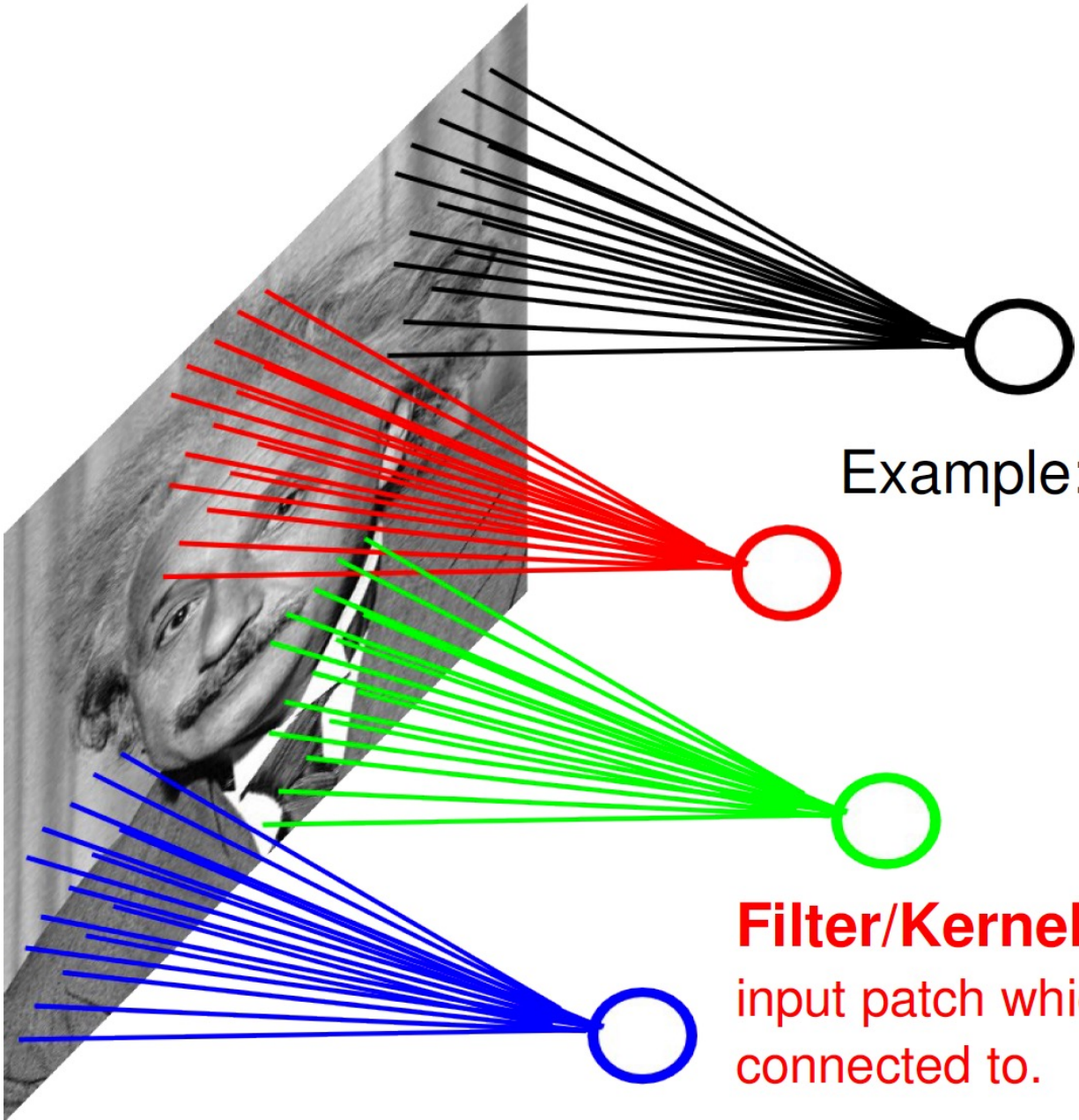
1M hidden units

10^12 parameters!!!



- Spatial correlation is local
- Better to put resources elsewhere!

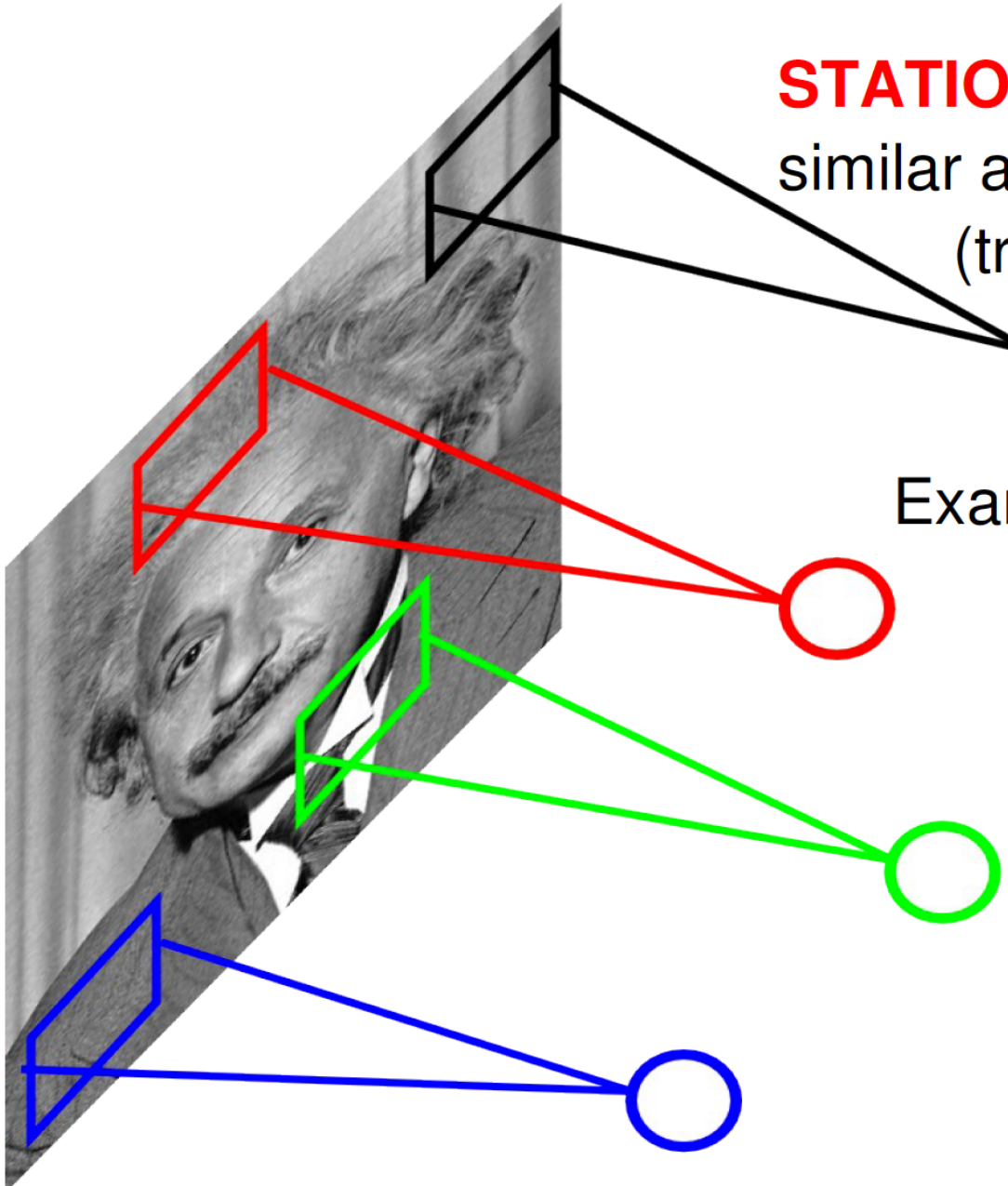
LOCALLY CONNECTED NEURAL NET



Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

Filter/Kernel/Receptive field:
input patch which the hidden unit is connected to.

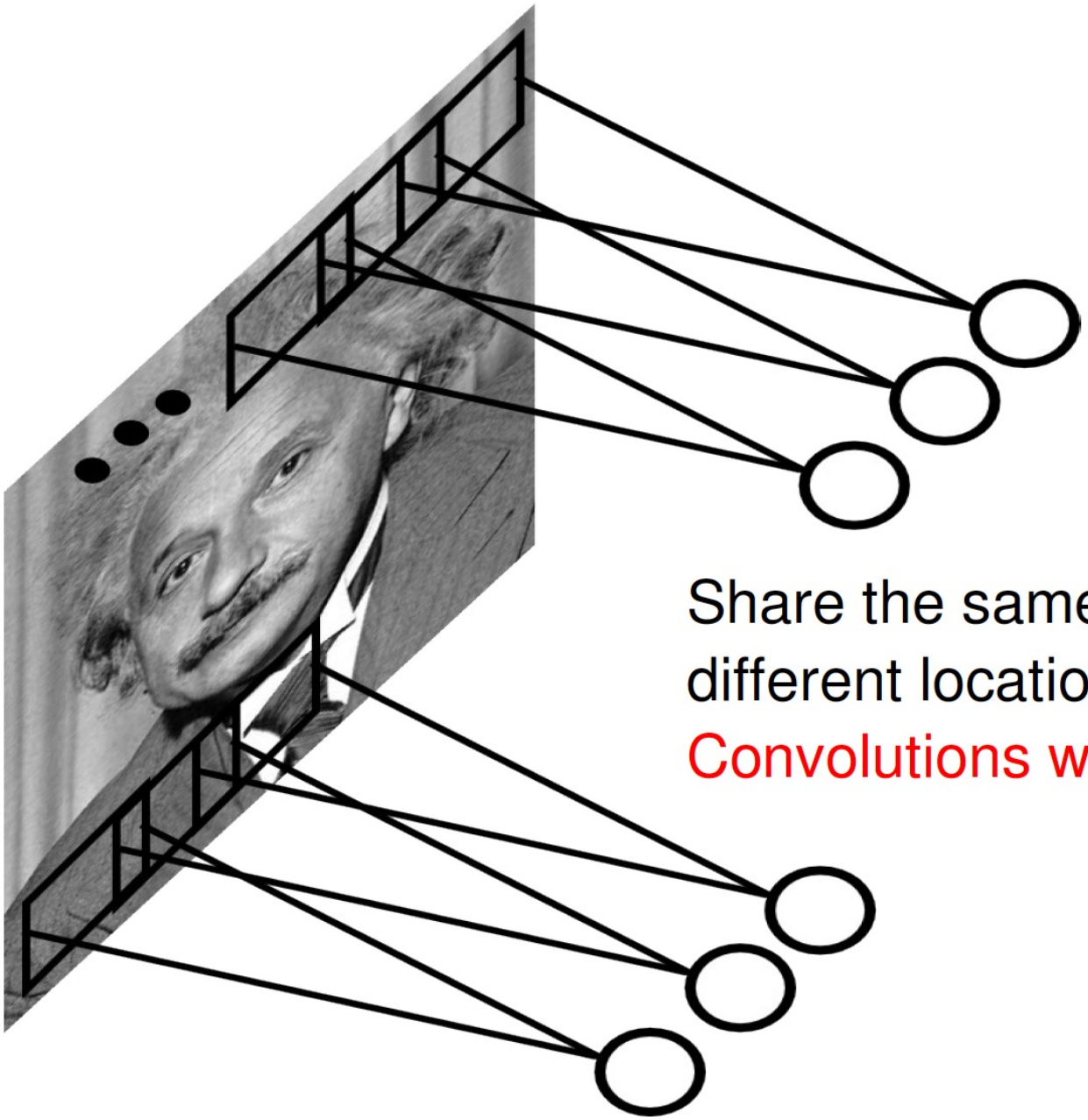
LOCALLY CONNECTED NEURAL NET



STATIONARITY? Statistics are similar at different locations (translation invariance)

Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

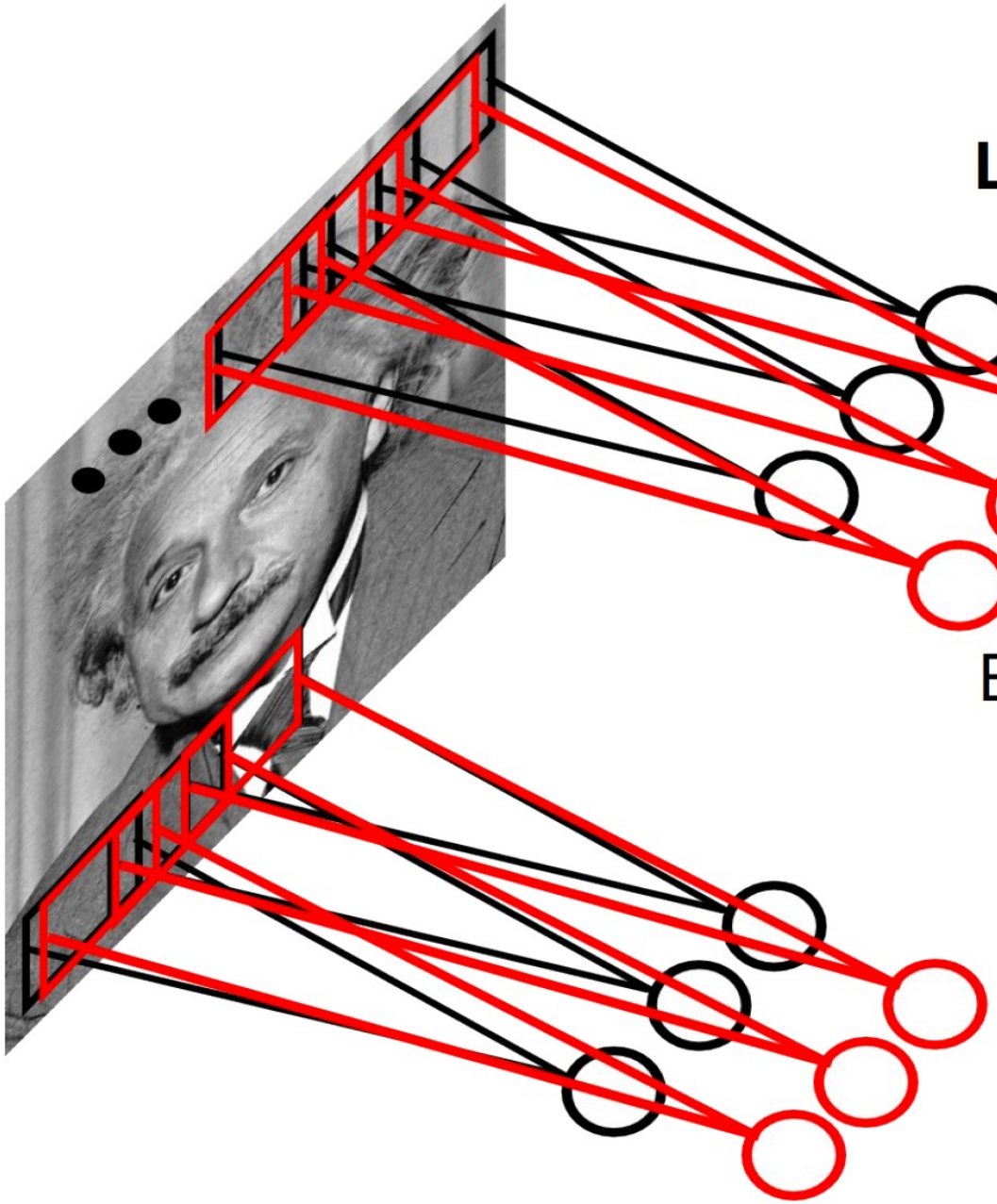
CONVOLUTIONAL NET



Share the same parameters across different locations:

Convolutions with learned kernels

CONVOLUTIONAL NET

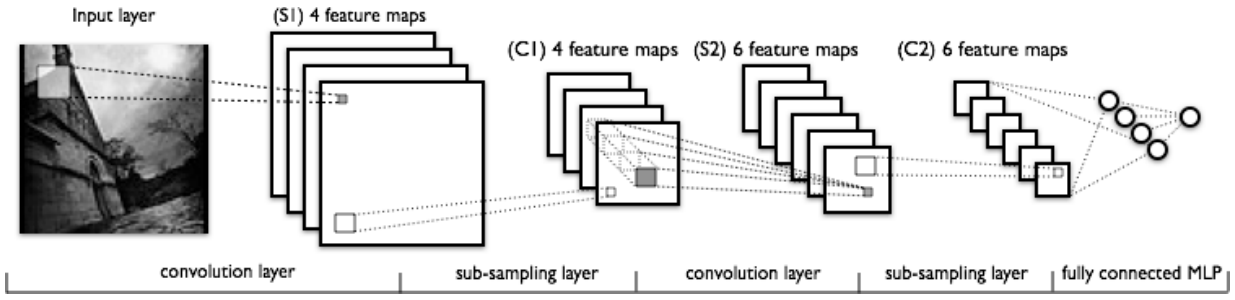


Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

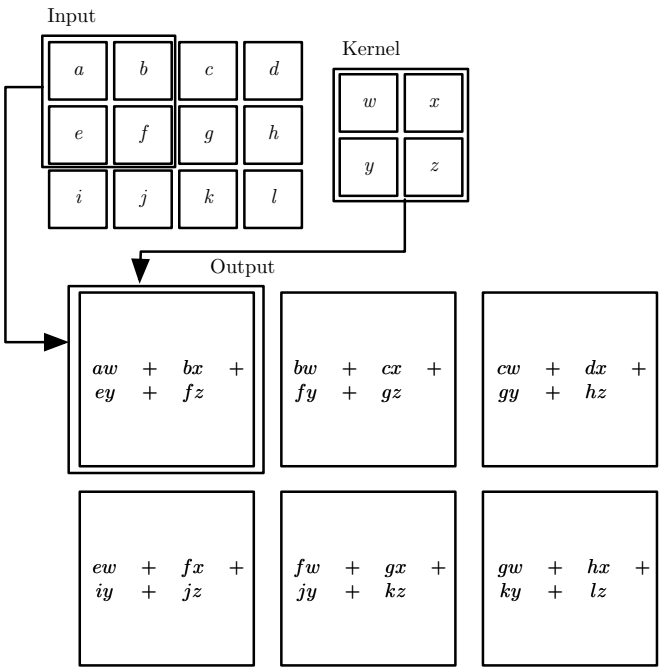
Convolutional Neural Networks

- Three principles:
  - local receptive fields
  - weight sharing
  - subsampling
- Multiple layers of Convolution, Activation, and Pooling may be used in a CNN
- These layers act as feature extraction, to find useful features from the input
- Generally, a final Fully Connected layer is added via a MLP for classification or regression purposes



Convolution Operation

- A mathematical depiction of the convolution operation on an input image
- A CNN learns the values of the filter (or kernel) on its own during the training process
- It outputs feature maps
- Most of the slides based on
  - http://www.deeplearningbook.org/contents/convnets.html

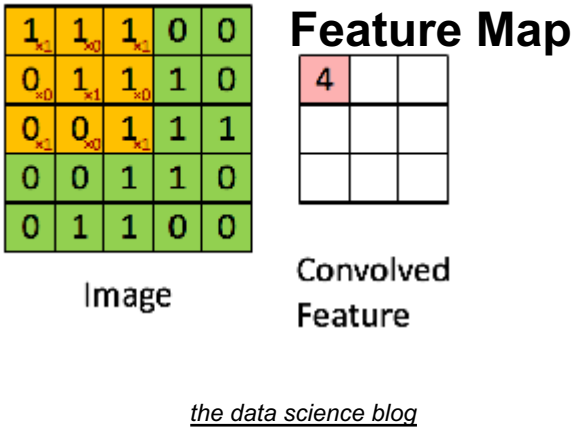


Goodfellow 2016

Example: Convolution on an Image

- Suppose you are given the following binary image, X

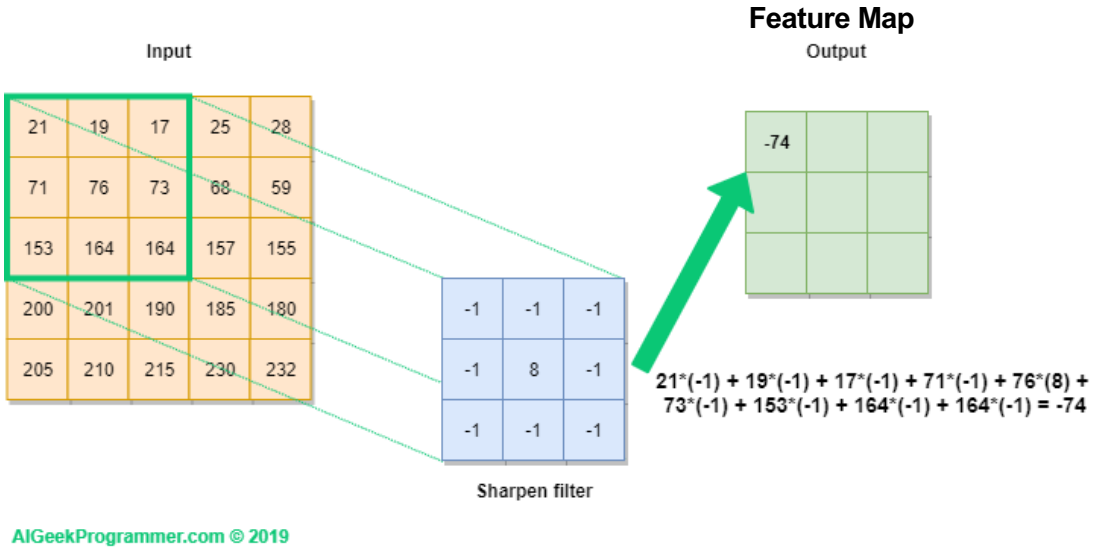
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0



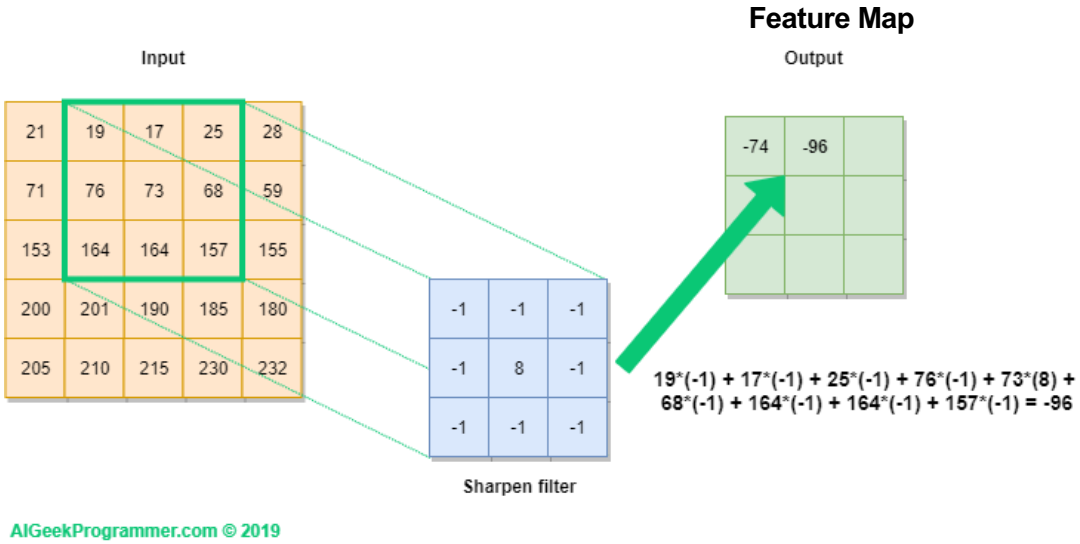
- You want to convolve this image with matrix, W (below). W is called '**filter**' or '**kernel**' or '**feature detector**'

1	0	1
0	1	0
1	0	1

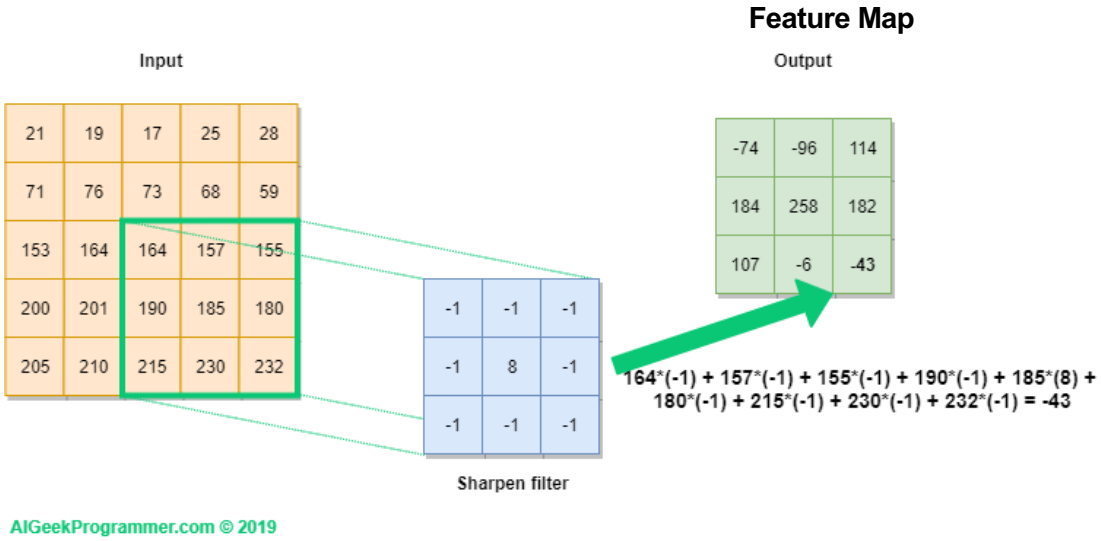
Example: Convolution on an Image



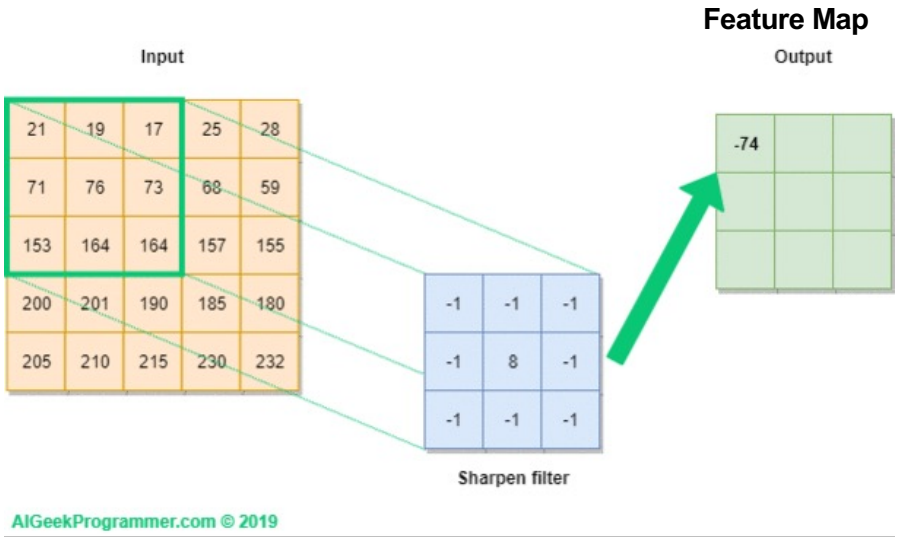
Example: Convolution on an Image



Example: Convolution on an Image



Example: Convolution on an Image



Convolution

- Convolution is a linear mathematical operation on two functions
- Convolution of functions *f(t)* and *g(t)* is:
- 2D: $$f(t) * g(t) = \sum_{a=-\infty}^{\infty} f(a)g(t-a)$$

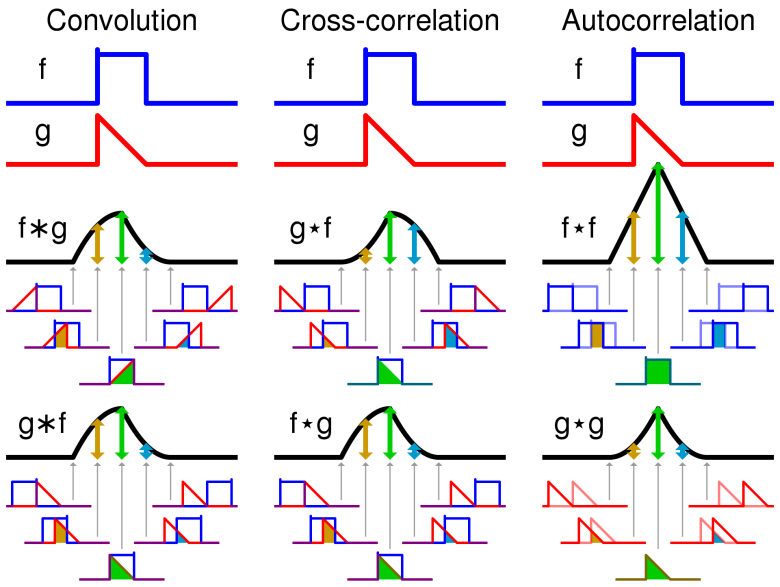
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

- Cross-correlation of functions *f(t)* and *g(t)* is:

$$f(t) \star g(t) = \sum_{a=-\infty}^{\infty} f(a)g(t+a)$$

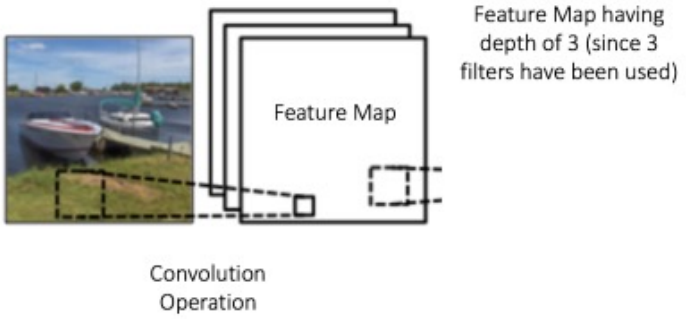
- 2D:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$



Feature Map

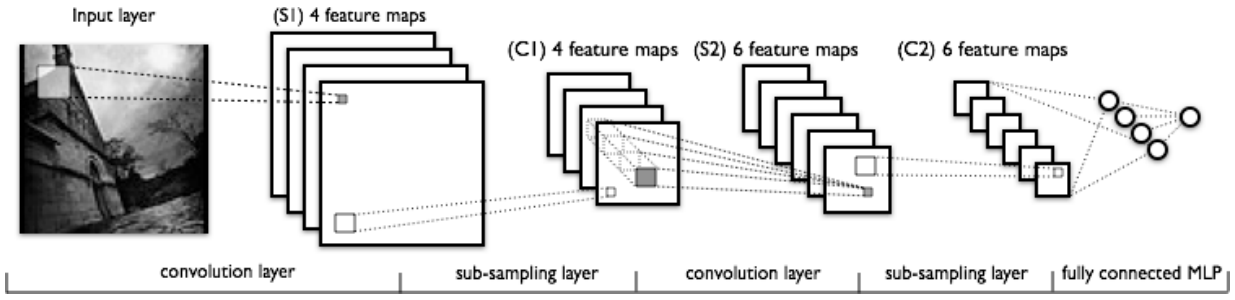
- The size of the Feature Map (resulting image after convolution) is controlled by three parameters
  - Depth: Number of different filters to use for the convolution operation
  - Stride: is the amount by which the kernel is moved by as the kernel is passed over the input
  - Zero-padding: May pad the input with zeros around the border



the data science blog

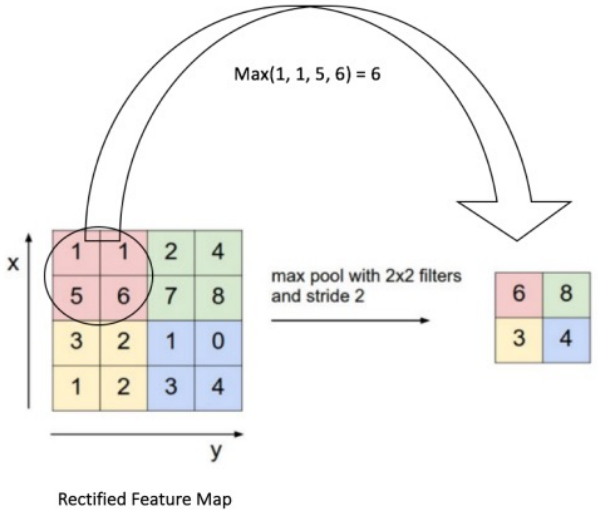
Convolutional Neural Networks

- Three principles:
  - local receptive fields
  - weight sharing
  - subsampling
- Multiple layers of Convolution, Activation, and Pooling may be used in a CNN
- These layers act as feature extraction, to find useful features from the input
- Generally, a final Fully Connected layer is added via a MLP for classification or regression purposes



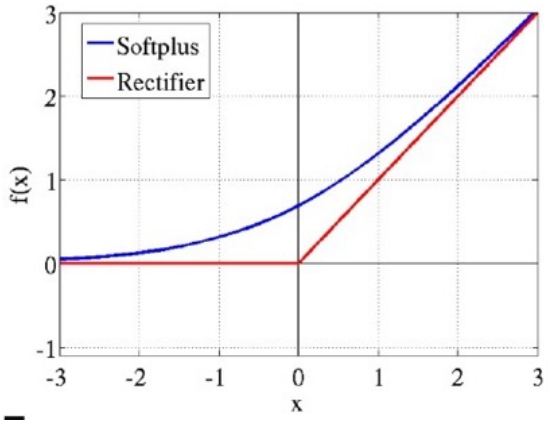
Pooling

- Pooling (aka spatial pooling, subsampling, or downsampling) is used to reduce the dimensionality of the feature map
- Different types of pooling include: Max, Average, Sum, etc.
- A window is defined, and the pooling operation is performed over the elements within that window
- The pooling window slides over the feature map by the stride amount
- It is applied to each feature map



Rectified Linear (ReLU) Activation

- A rectified linear (ReLU) activation operation may be applied after the convolution operation
  - Introduces nonlinearity to the network
- ReLU(x) = max(0, x)
- ReLU is applied to every element (pixel)
  - Negative values are replaced by 0
- Other nonlinear activation functions may be used instead

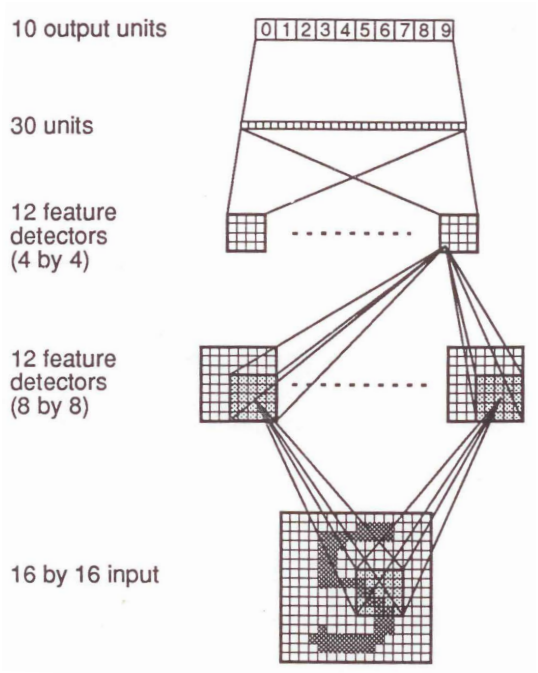


CNN Training

- The Backpropagation algorithm is used to train the parameters of a CNN
- Basic steps:
  - Randomly initialize all filters (or kernels) and weights
  - Propagate the input forward through the layers of the CNN (convolution, activation, pooling, MLP) to get an output(s)
  - Calculate the error between the actual output(s) and the desired output(s)
  - Use backpropagation to calculate the gradients and deltas, and then update the filters and weights accordingly

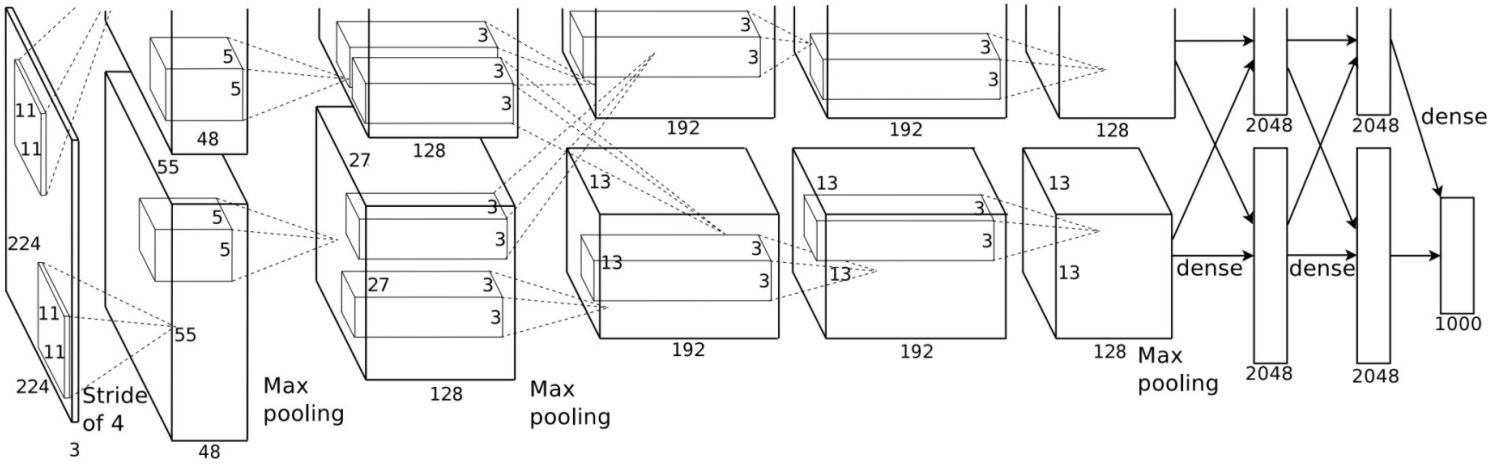
An Early CNN Application

- **Task**: Handwritten Zip code Recognition (1989)
- **Network Description**
  - Input: binary pixels for each digit
  - Output: 10 digits
  - Architecture: 4 layers (16 x 16 – 12x8x8 – 12x4x4 -30 -10)
- **Performance**: Trained on 7300 digits and tested on 2000 new ones
  - Achieved 1% error on the training set and 5% error on the test set
  - If allowing rejection (no decision), 1% error on the test set
  - This task is not easy

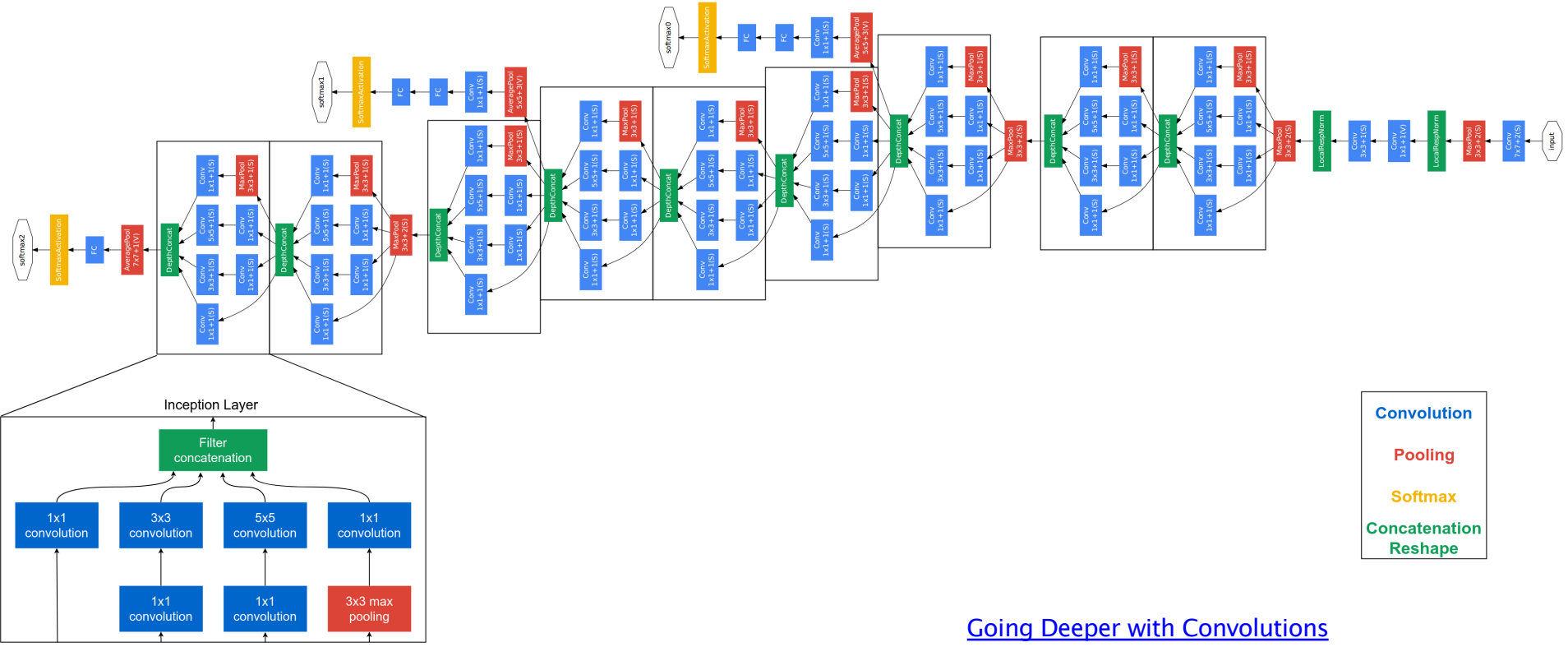


AlexNet

- Krizhevsky, NeurIPS 2012 (Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton)



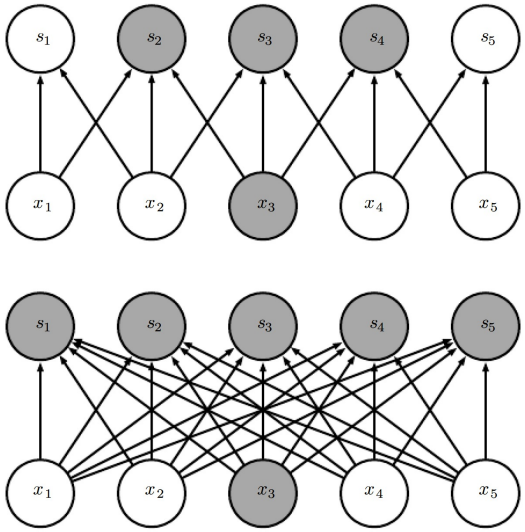
GoogLeNet



[Going Deeper with Convolutions]

Important properties of CNNs

- Sparse interactions (sparse connectivity or sparse weights)
  - Kernel is smaller than the input (top plot)
- Parameter sharing (tied weights)
  - using the same parameter for more than one function in a model
- Equivariance to translation
  - If we move the object in the input, its representation will move the same amount in the output



http://www.deeplearningbook.org/contents/convnets.html

CNN mysteries

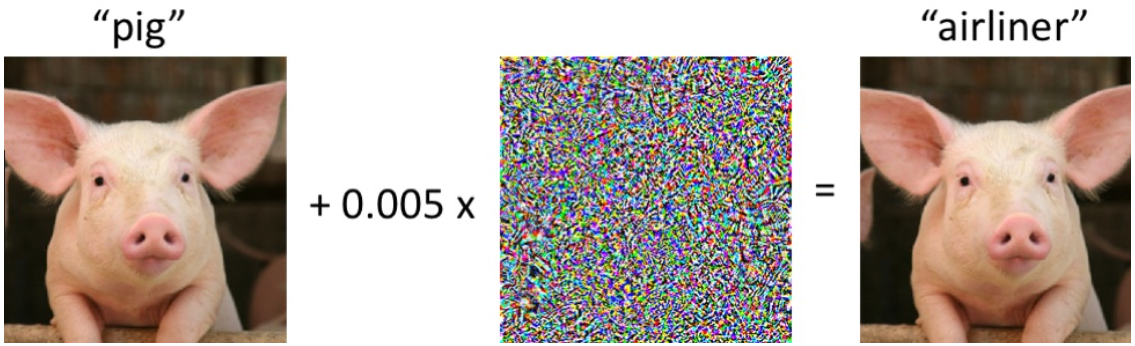
CNNs can sometimes outperform humans. Even after adding extreme noise, all these digits are recognized correctly!

But, only first column of these are recognized correctly. Second column digits are all wrong!

Goodfellow et al

Problems with CNNs

Sample of how an object in a noncanonical orientation and context fools many current object classification systems (Alcorn et al., 2018)



The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence

Visualizing Convnets

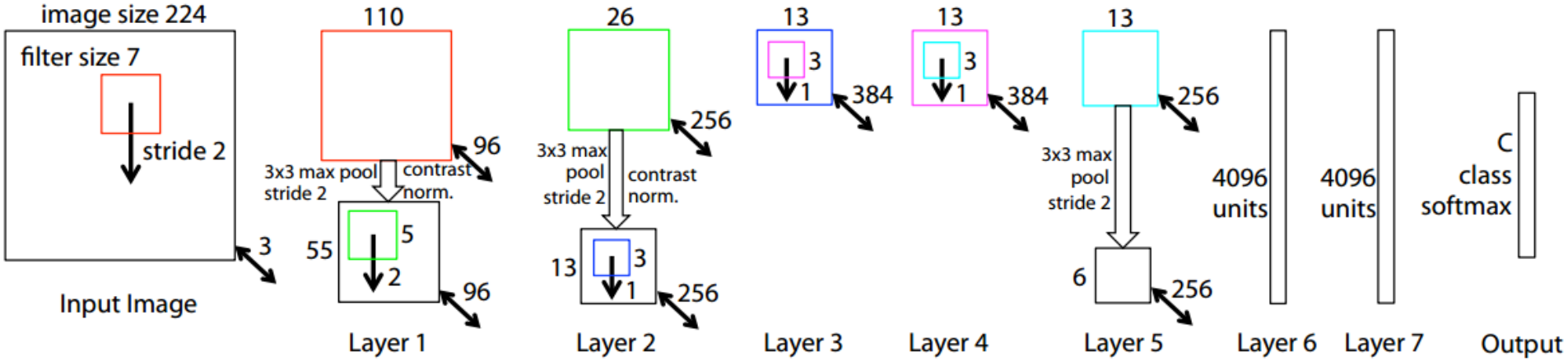
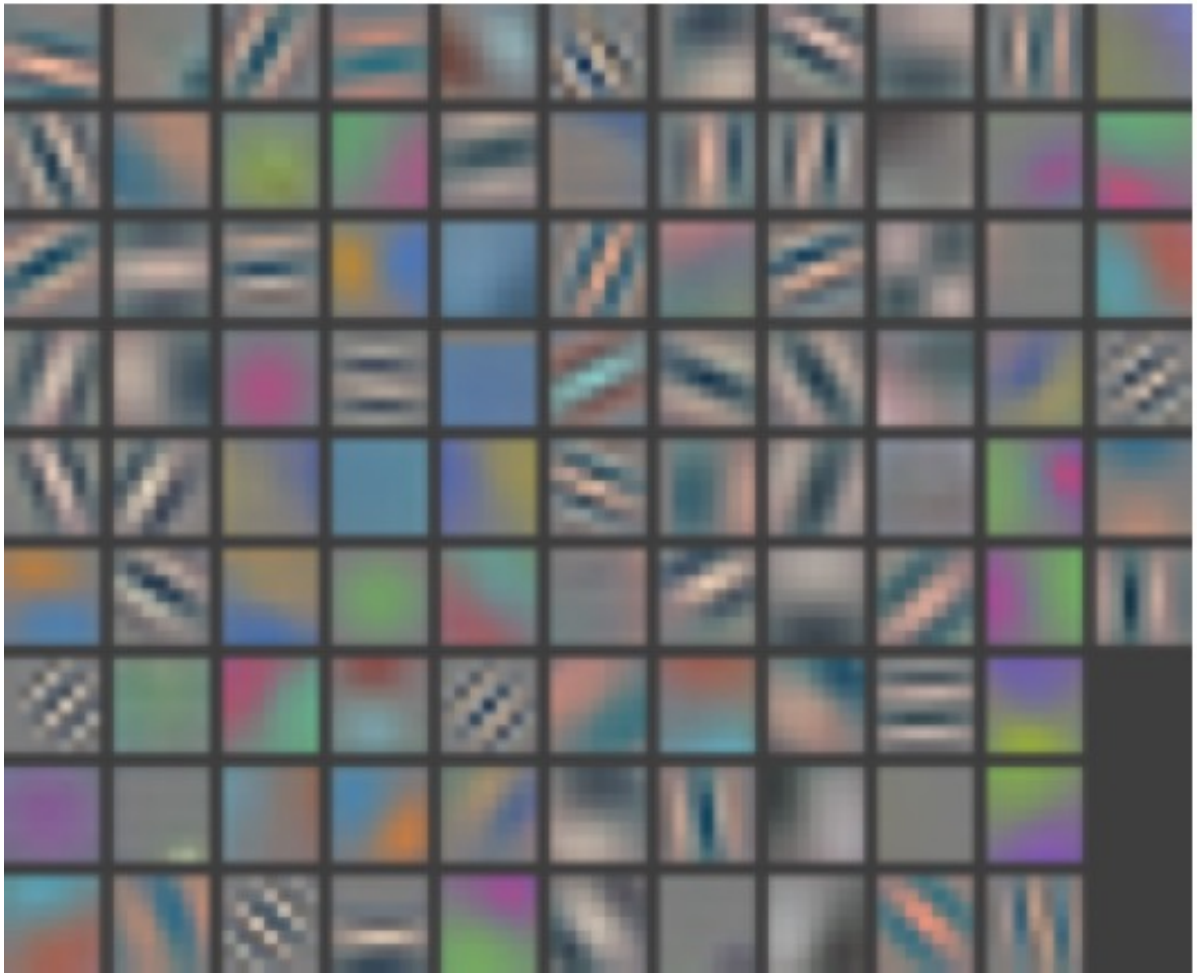


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, arXiv preprint, 2013

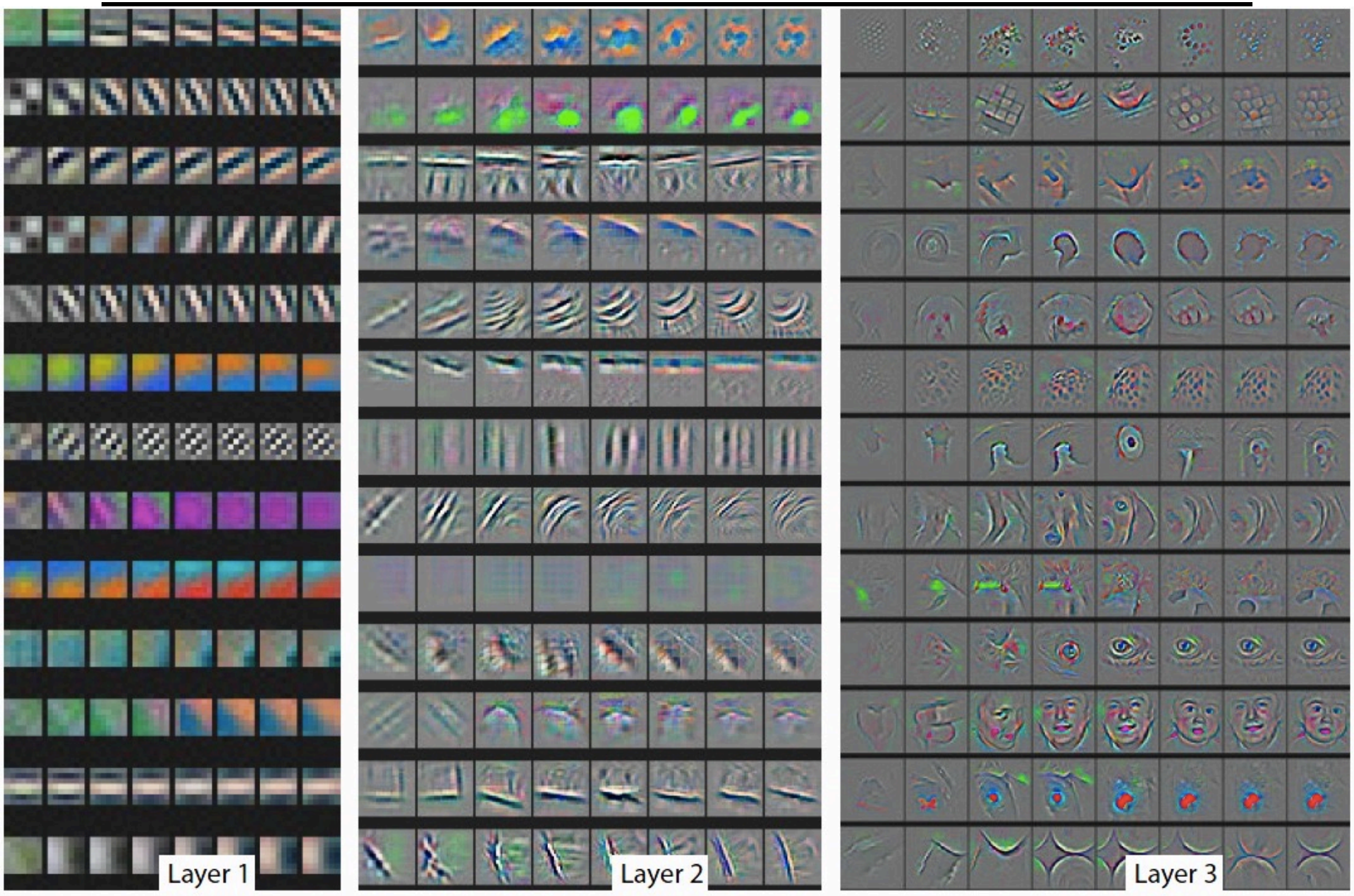
Slide credit: Rob Fergus

Layer 1 Filters

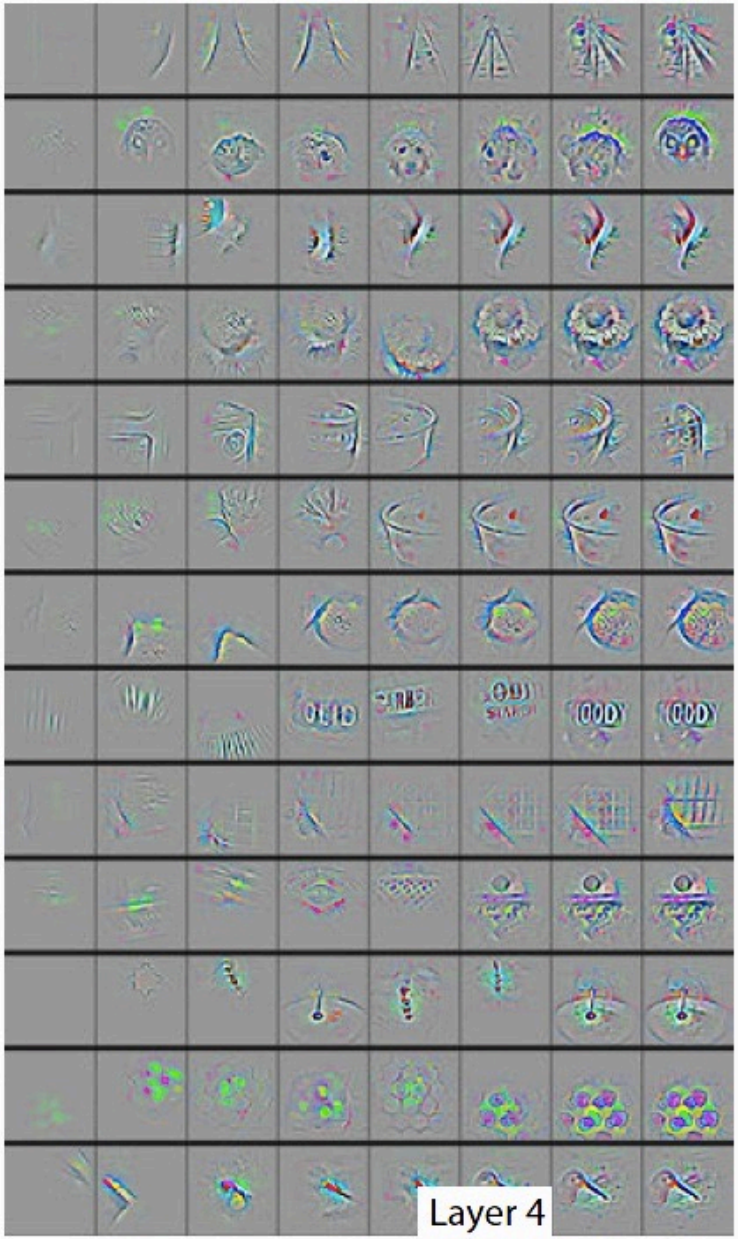


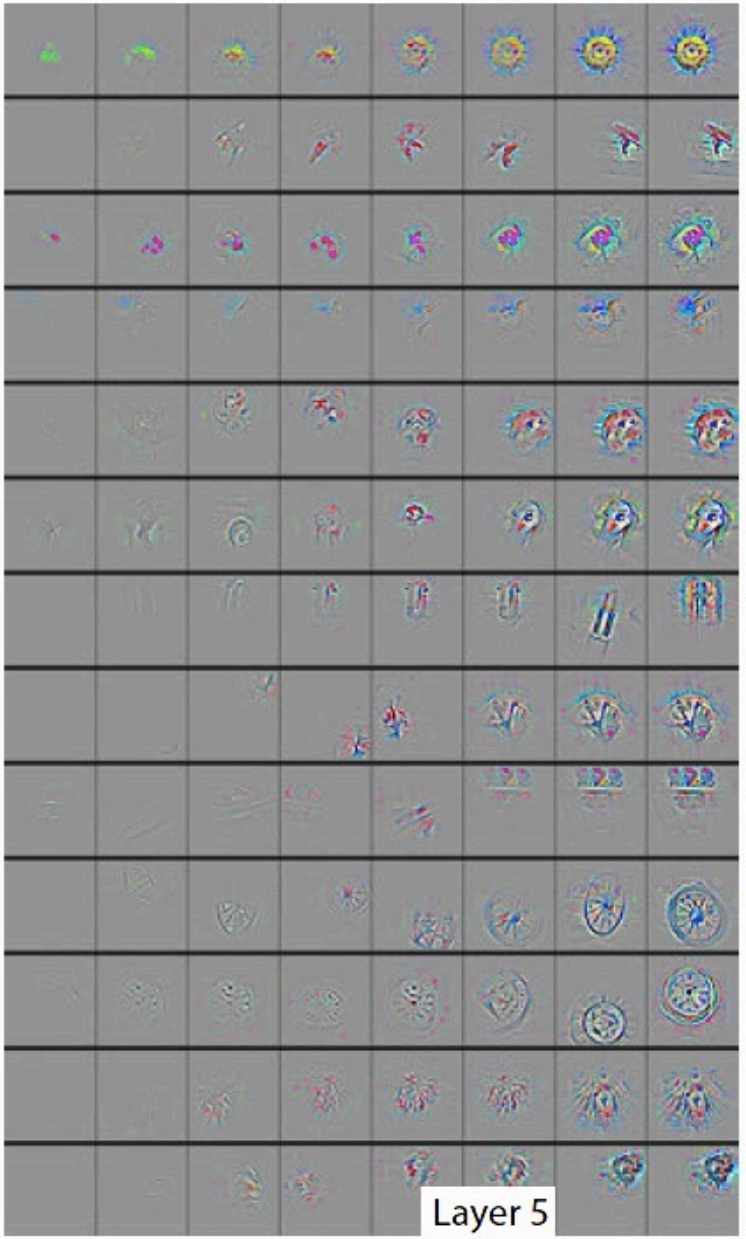
Slide credit: Rob Fergus

Evolution of Features During Training



Evolution of Features During Training





Other optimization and regularization techniqeues

Weight decay

Weight decay:

- Penalizes complexity and prevent overfitting

$$\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

Weight decay

Weight decay

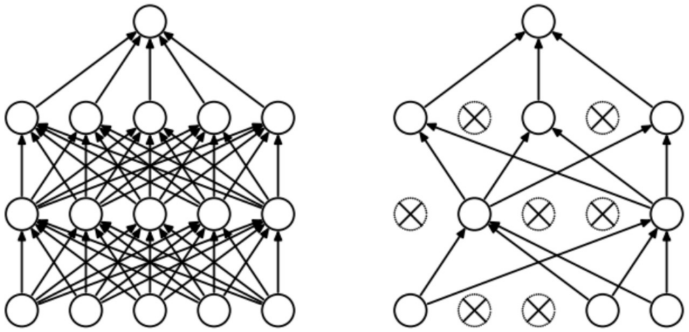
- Penalizes complexity and prevent overfitting

$$\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda_1}{2} \sum_{w \in \mathcal{W}_1} w^2 + \frac{\lambda_2}{2} \sum_{w \in \mathcal{W}_2} w^2$$

Dropout

Dropout

- Dropping out units with (both hidden and visible) in a neural network. There is some probability that a unit will be dropped out (ignored in both forward and backward pass) at each training iteration.



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

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Early stopping

Early stopping

- Stopping the training of a neural network once the model performance stops improving on a hold out validation dataset.

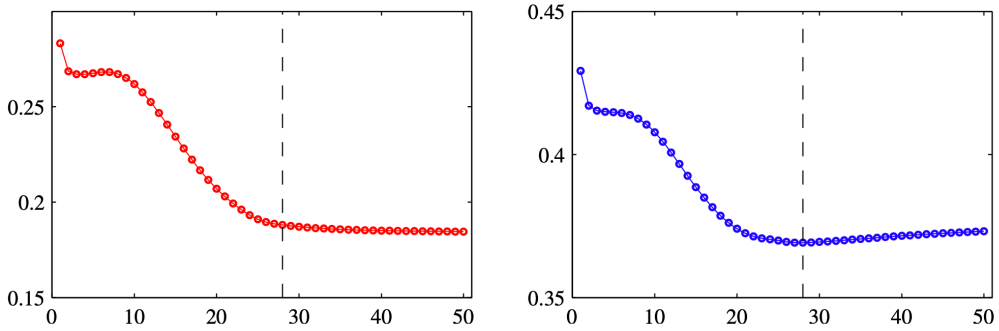


Figure 5.12 An illustration of the behaviour of training set error (left) and validation set error (right) during a typical training session, as a function of the iteration step, for the sinusoidal data set. The goal of achieving the best generalization performance suggests that training should be stopped at the point shown by the vertical dashed lines, corresponding to the minimum of the validation set error.

From Bishop's Machine Learning book, chapter 5 (Neural networks); 5.5 (Regularization)

Change learning rate during learning

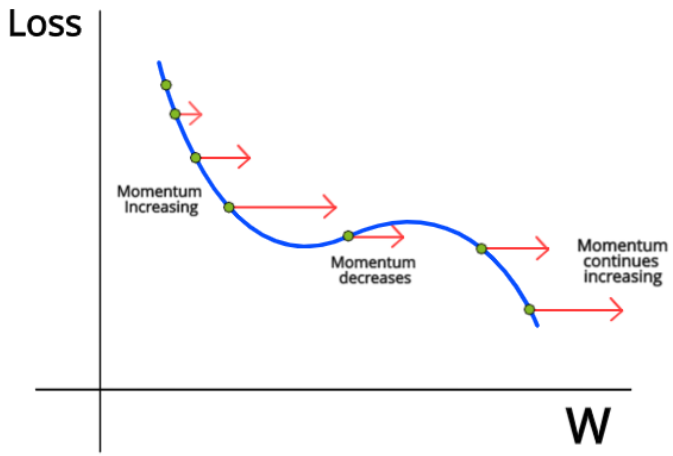
Change learning rate during learning:

- Make η large at the beginning of learning and decrease it later
- Simulated annealing: cleverly adjust learning rate using physics-inspired parameter called temperature

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)})$$

Momentum

Momentum: a way to escape small gradients



$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

weight increment — learning rate — weight gradient

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

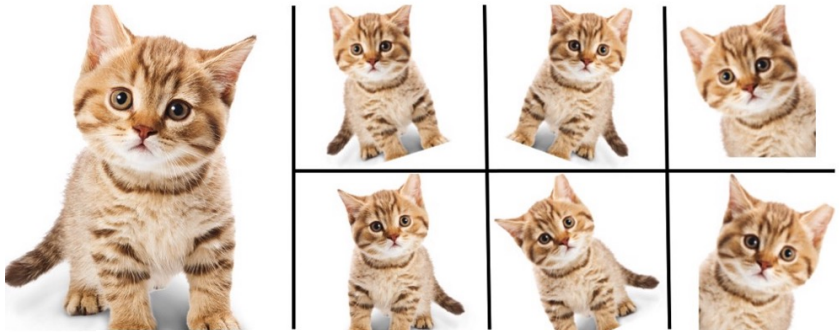
momentum factor — weight increment, previous iteration

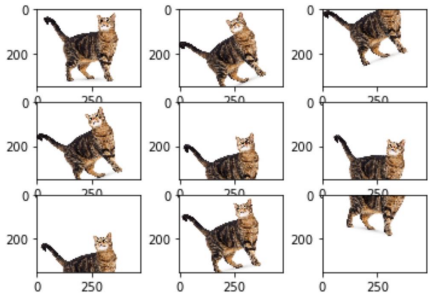
https://deeplearningdemystified.com/article/fdl-4

Dataset augmentation

Dataset augmentation

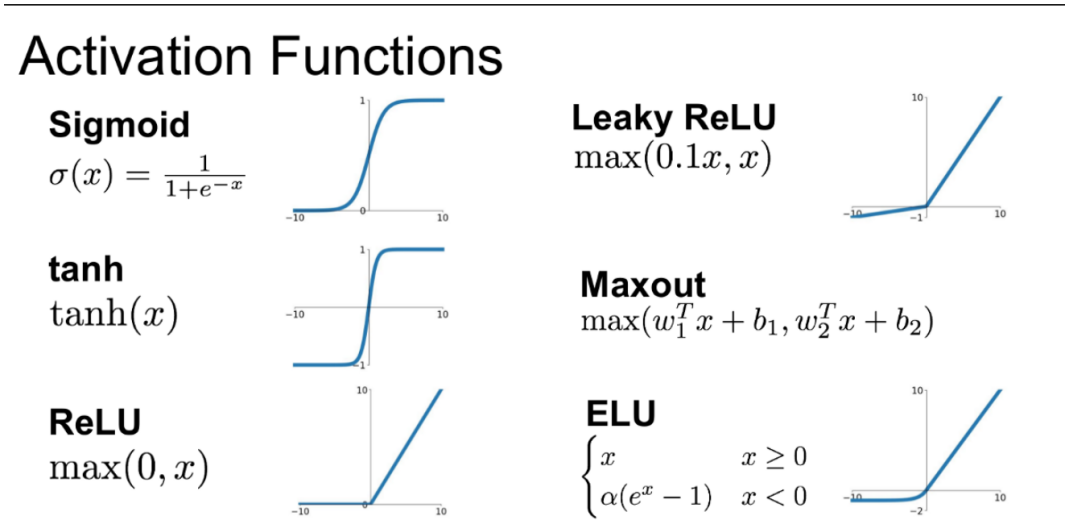
- E.g. adding noise to the input, applying spatial transformations to the input





Activation function selection

In deep (many layers) networks, to avoid small gradient we can use activations functions such as ReLU, which have a large gradient for a wide range of values.



Neural Network References

For additional examples and references, check out the following:

- The Data Science Blog - https://ujjwalkarn.me/blog/
- Deep Learning Book - http://www.deeplearningbook.org
  - See also video lectures on the same page

Video lectures:

- https://www.youtube.com/watch?v=Xogn6veSyxA (Ian Goodfellow)
- https://www.youtube.com/watch?v=H-HVZJ7kGI0&t=1057s (Ava Soleimany)

Practical perspective:

- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition
- See here for the code:
  - https://github.com/ageron/handson-ml2
  - https://github.com/ageron/handson-ml2/blob/master/14_deep_computer_vision_with_cnns.ipynb

Next class

More deep learning (Recurrent Neural Networks, Attention in Neural Networks, Reinforcement Learning)