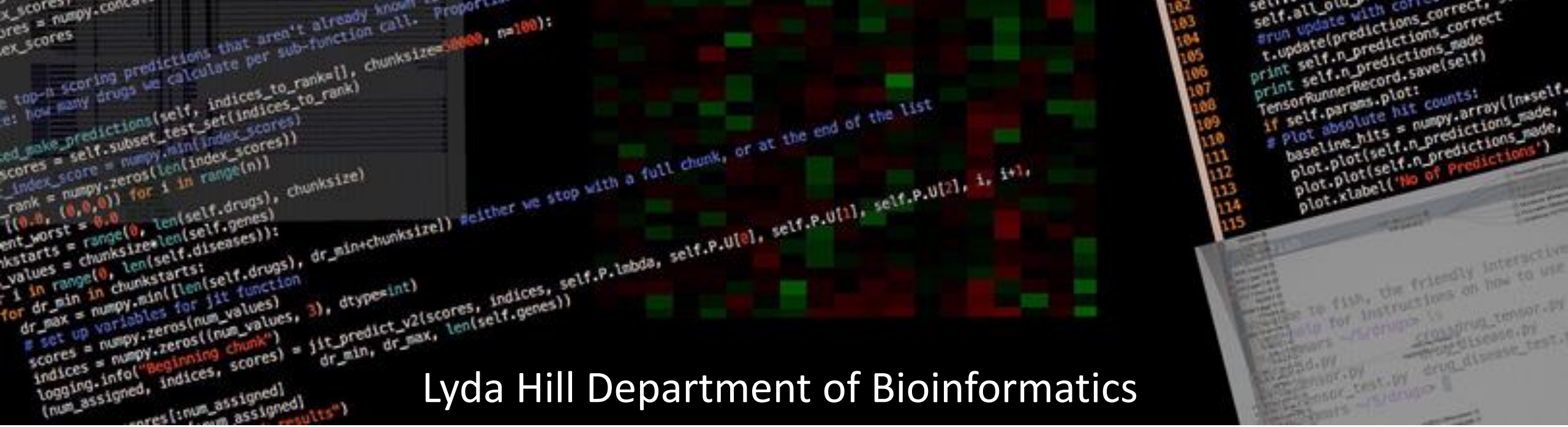


Introduction to convolutional neural networks

Albert Montillo, Ph.D.

Albert.Montillo@UTSouthwestern.edu

Deep Learning for Precision Health Lab
Web: utsouthwestern.edu/labs/montillo

Special thanks

- This talk consists of original slides and slides I've adapted from several sources. Special thanks to slides with content curtesy of those listed below.

  – Dr Andrew Ng, CS Dept, Stanford University

  – Dr Hugo Larochelle, Université de Sherbrooke

  – Dr Aarti Singh, CMU

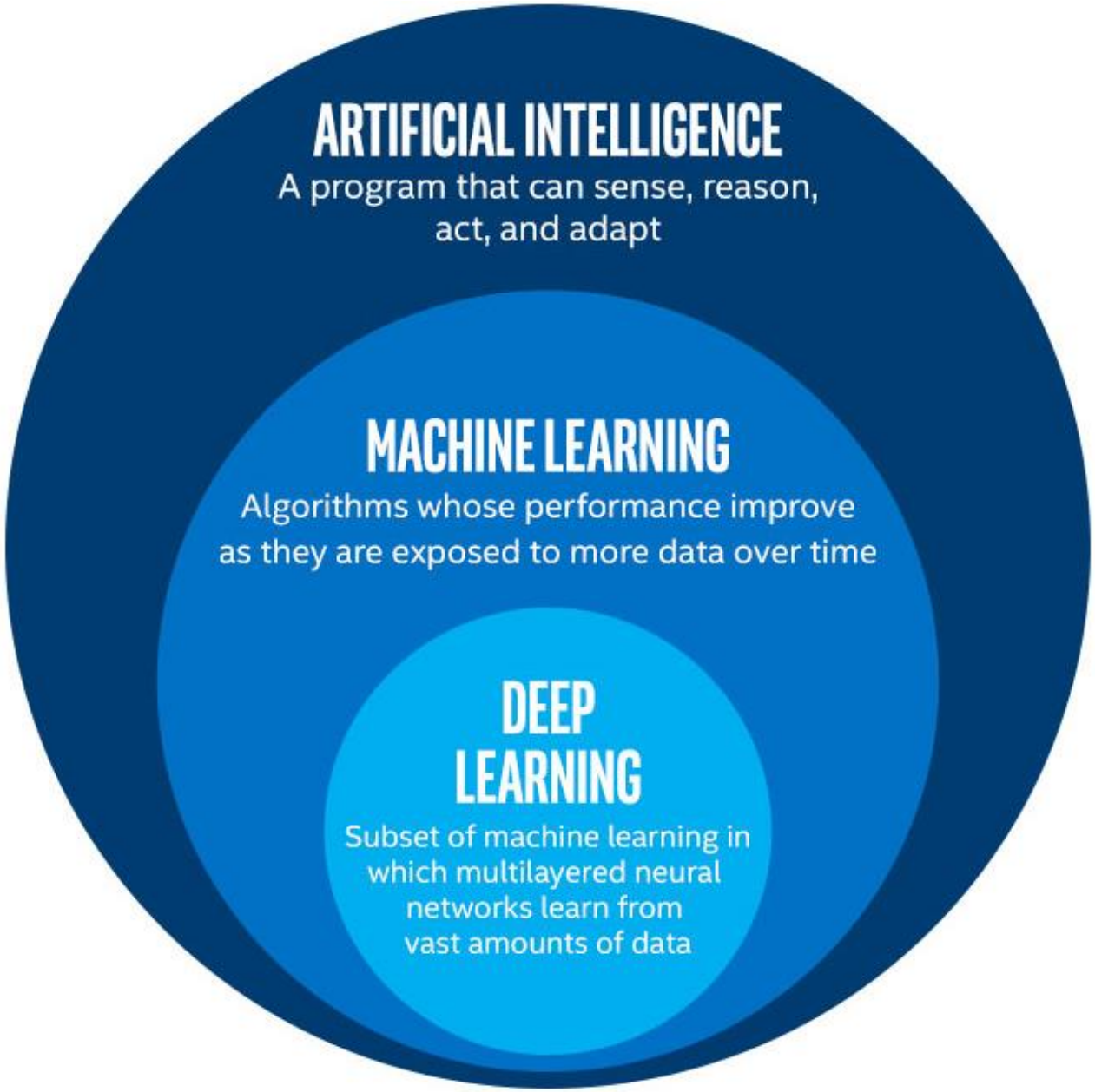
  – Adam Harley, Ryerson University

  – Lily Peng, MD/PhD, Google

  – Brett Kuprel, EE dept, Stanford

  – Weifeng Li, Victor Benjamin, Xiao Liu, and Hsinchun Chen, University of Arizona

What is machine and deep learning? Where do they fit in?

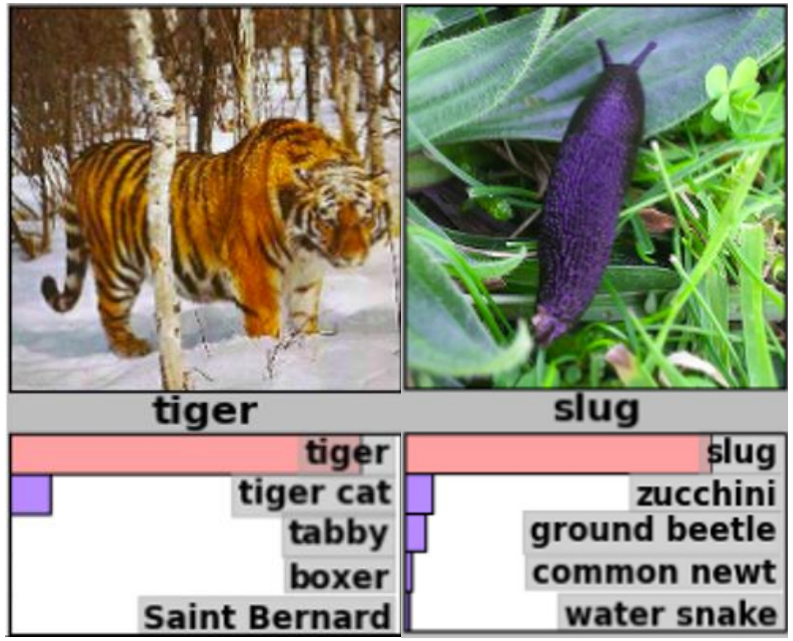


How powerful are CNNs and what are their relevancy for biomedical science and medicine?

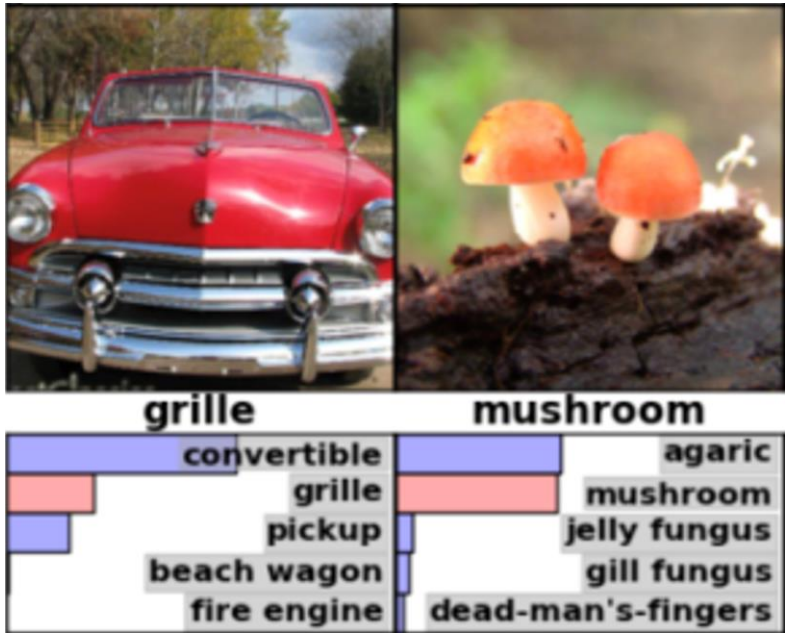
- **Computers can learn tasks previously thought impossible for a computer and they can keep on learning. They do not need to be explicitly told what to do.**

- **Computers can learn to see, everything!**
  - **ImageNet challenge: ~14 million images from the internet where the dominant objects is labeled from 1,000 classes.**
  - **Goal: Pixels → class label**
  - **CNN based solution**

Answer is alg's top pick



Answer is in Alg's top 5

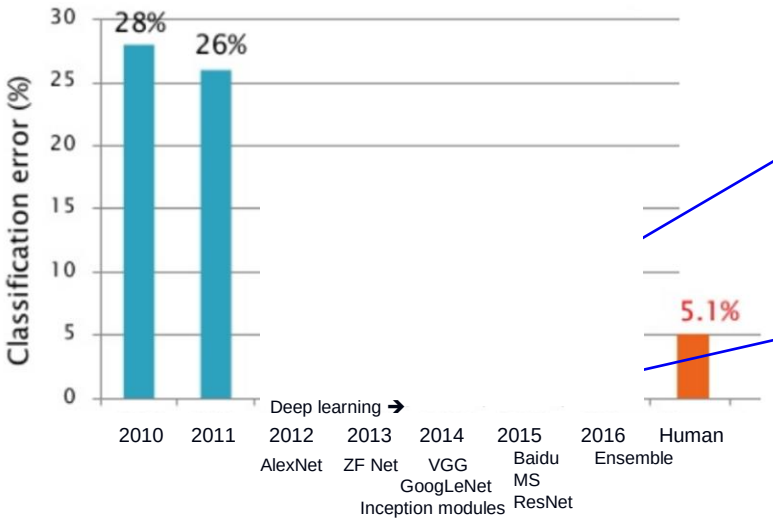


Answer is not in Alg's top5

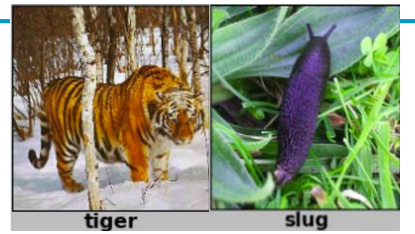
[Alex Krizhevsky et al. NIPS'2012]

How powerful are CNNs and what are their relevancy for biomedical science and medicine?

- **Consider whether computers can learn to interpret images of real world with human-level performance? For long time skepticism…**



Yes, achieved in 2015 (Baidu, MS)



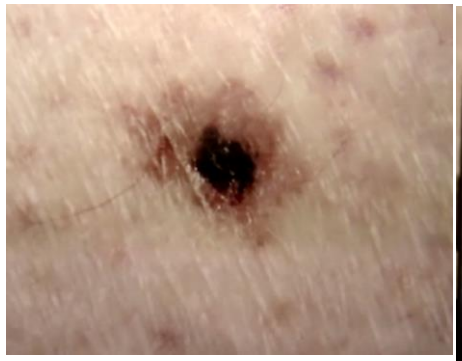
- **Better than human-level performance?** Yes, achieved later in 2015 (3.5% ResNet) and reduced again in 2016

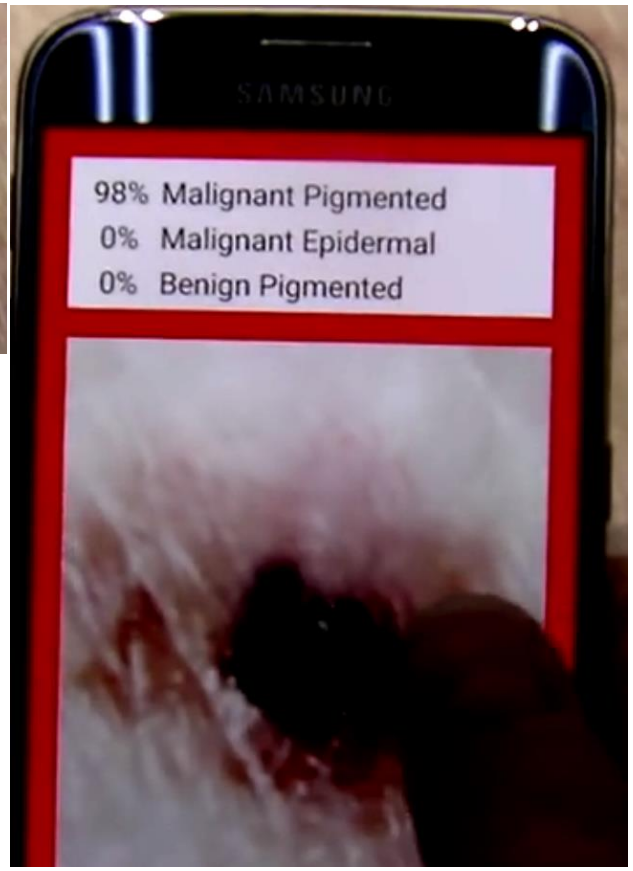
- **Prediction: Computers will be taught to interpret multi-modal volumetric medical imaging data, which will *revolutionize* the fields employing imaging: neuroscience, cell biology, neurology, radiology, pathology, cardiology…**

- **Driven by:**
  - **Continuing exponential growth in compute capacity**
  - **Ever improving algorithms from computer scientists for training the networks.**

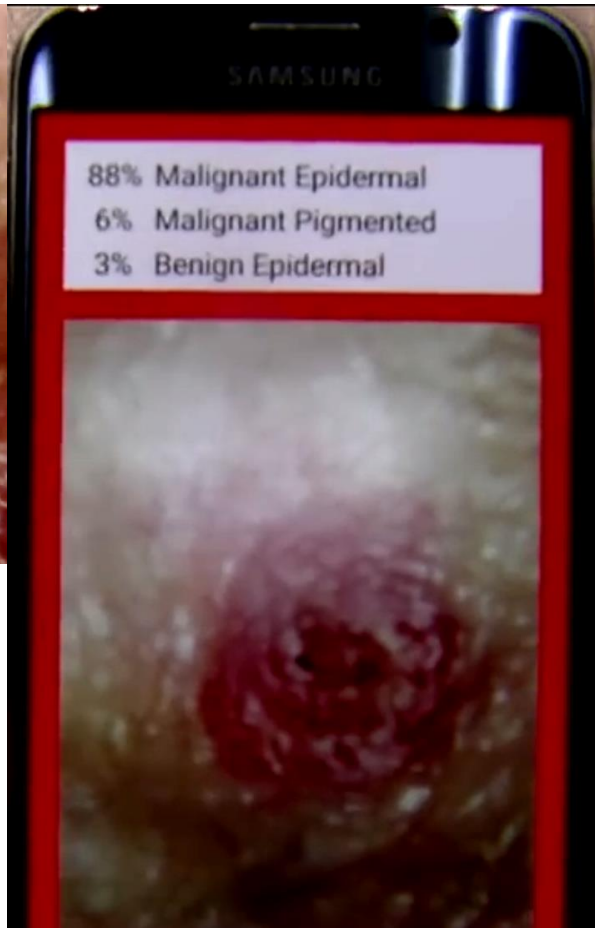
Example 1: Impact of deep learning in medicine

Skin cancer detection, in real time, with just your cell phone





6:50 – 7:15 video

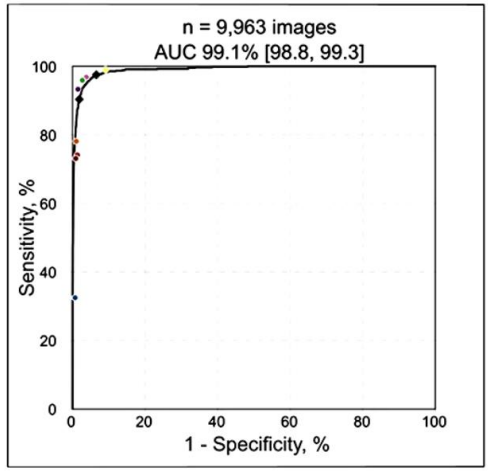


Example 2: impact of deep learning in medicine

automated diabetic retinopathy grading via fundus image classification

JAMA | Original Investigation | INNOVATIONS IN HEALTH CARE DELIVERY

Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs



F-score

0.95	0.91
Algorithm	Ophthalmologist (median)

"The study by Gulshan and colleagues **truly represents the brave new world in medicine.**"

Dr. Andrew Beam, Dr. Isaac Kohane
Harvard Medical School

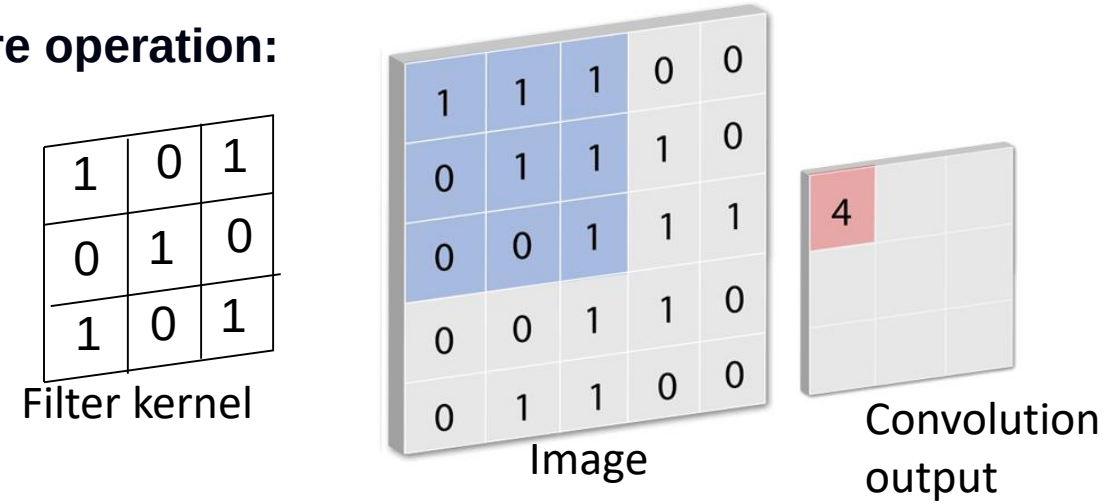
"Google just published this paper in JAMA (impact factor 37) [...] **It actually lives up to the hype.**"

Dr. Luke Oakden-Rayner
University of Adelaide

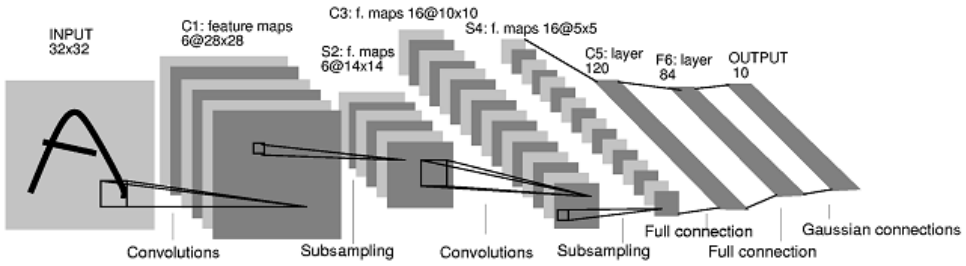
How do CNNs work? (If you remember nothing else)

Neural net is sequence of operations transforming inputs into an output class or prediction. CNN is a special type of neural net suited for image interpretation.

Convolution is a core operation:



CNN is formed by stacking multiple layers together



Many design decisions:

 # layers, # filters/layer, filter sizes,

choice of non-linear activation functions, pooling to downsample

Convolutional Neural Network (CNN)

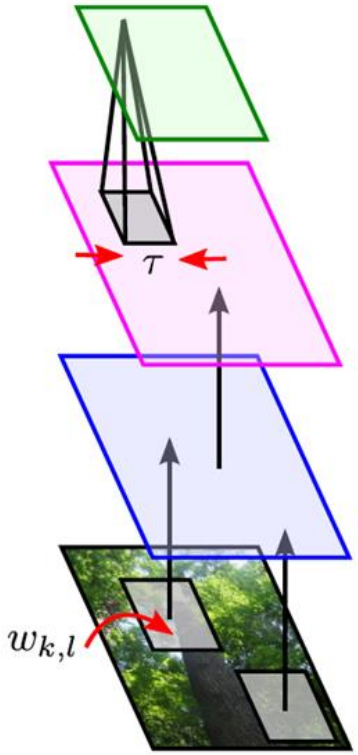
Convolutional Neural Networks are inspired by mammalian visual cortex.

- **The visual cortex contains a complex arrangement of cells, which are sensitive to small sub-regions of the visual field, called a receptive field. These cells act as local filters over the input space and are well-suited to exploit the strong spatially local correlation present in natural images.**

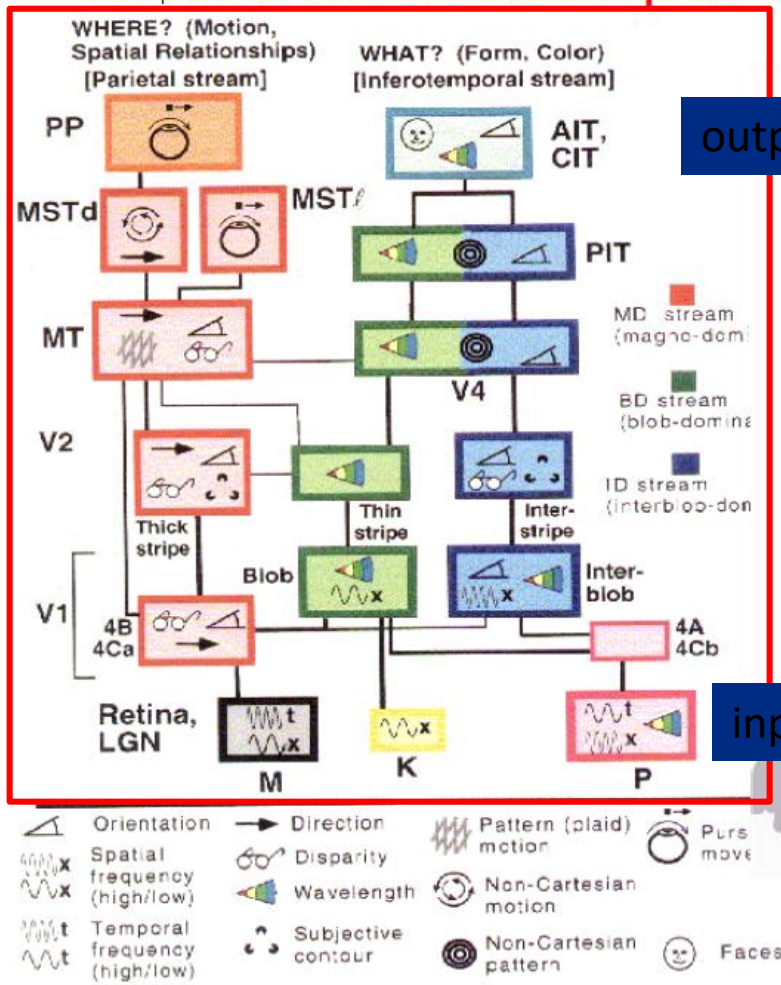
- **Two basic cell types:**
  - **Simple cells respond maximally to specific edge-like patterns within their receptive field.**
  - **Complex cells have larger receptive fields and are locally invariant to the exact position of the pattern.**

The Mammalian Visual Cortex Inspires the CNN

Convolutional Neural Net

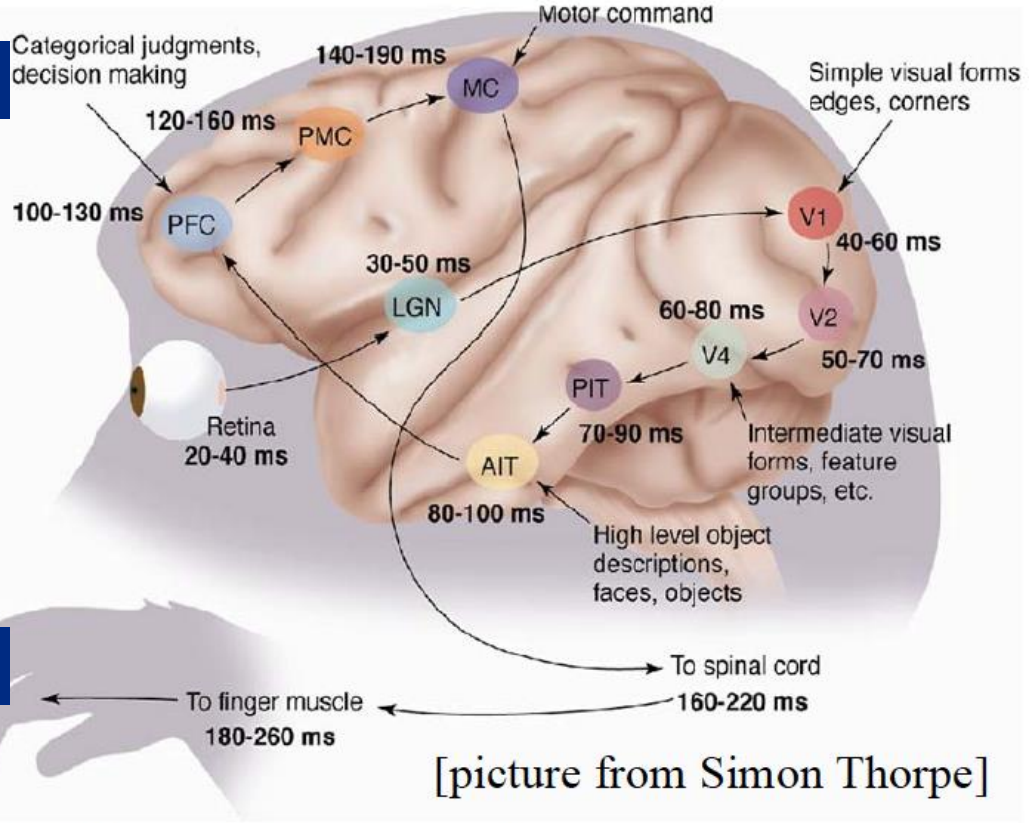


- The ventral (recognition) pathway in the visual cortex has multiple stages
- Retina - LGN - V1 - V2 - V4 - PIT - AIT ....
- Lots of intermediate representations



output

input



[picture from Simon Thorpe]

[Gallant & Van Essen]

Visual object recognition with neural networks

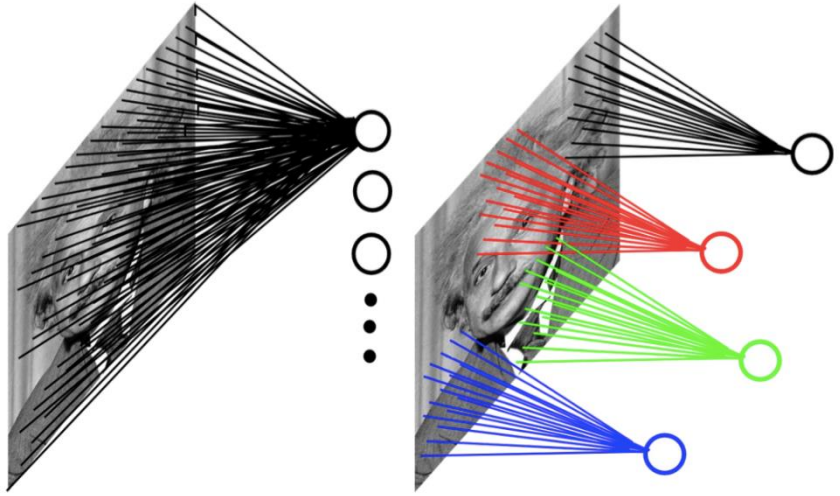
Like ImageNet example discussed above, consider the task of visual object recognition

Given some input image, identify what object it contains

- We can design neural networks that are specifically adapted for such problems
  - must deal with very high-dimensional inputs
    - 150 x 150 pixels = 22500 inputs, or 3 x 22500 if RGB pixels
  - can exploit the 2D topology of pixels (or 3D for video data)
  - can build in invariance to certain variations we can expect
    - translations, illumination, etc.

Motivation for an architectural departure from fully connected DNNs

- **Images are very high dimension. Using a fully-connected neural network would need a large amount of parameters.**
- **Inspired by the neurophysiological experiments conducted by [Hubel & Wiesel 1962], CNNs are a special type of neural network whose hidden units are only connected to local receptive field. The number of parameters needed by CNNs is much smaller.**



Example: 200x200 image

a) Single fully connected layer: 40,000 hidden units => 1.6 billion parameters

b) CNN: single conv layer with 5x5 kernel generating 100 feature maps => 2,500 parameters

The CNN Architectural solution

CNN architecture is an amalgamation of several ideas.

Intuition: Neural network with specialized connectivity structure,

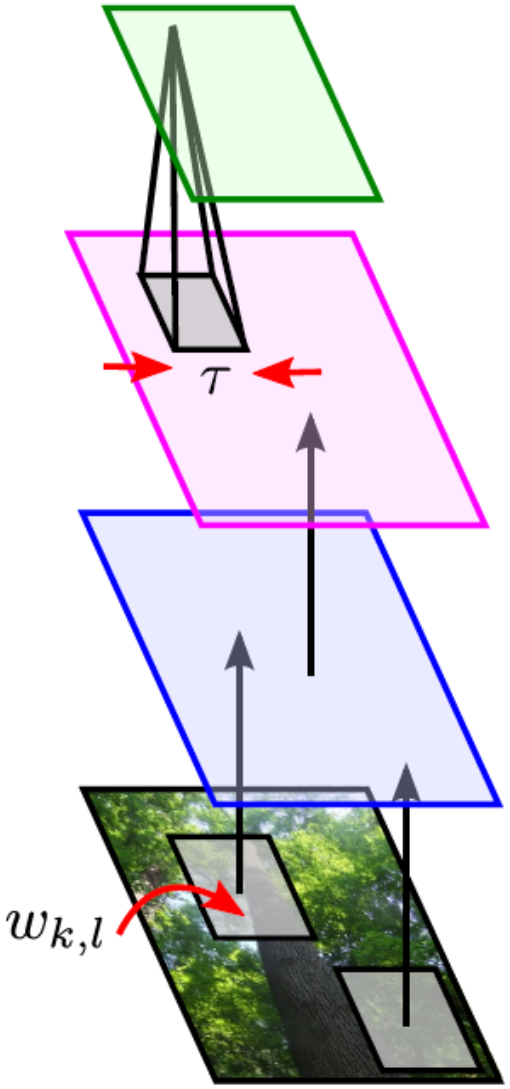
- **Stacking multiple layers of feature extractors**
- **Low-level layers extract local features.**
- **High-level layers extract learn global patterns.**

A CNN consists of a list of layers that transform the input data into an output class/prediction.

There are a few distinct types of layers:

- **Convolutional layer**
- **Non-linear layer**
- **Pooling layer**

Building-blocks for CNN's



$$x_{i,j} = \max_{|k|<\tau,|l|<\tau} y_{i-k,j-l}$$

mean or subsample also used

pooling stage

Feature maps of a larger region are combined.

$$y_{i,j} = f(a_{i,j})$$

e.g. $f(a) = [a]_+$

$f(a) = \text{sigmoid}(a)$

non-linear stage

Feature maps are trained with neurons.

$$a_{i,j} = \sum_{k,l} w_{k,l} z_{i-k,j-l}$$

Shared weights

convolutional stage

Each sub-region yields a feature map, representing its feature.

$z_{i,j}$

Images are segmented into sub-regions.

input image

CNN First idea: Local Connectivity

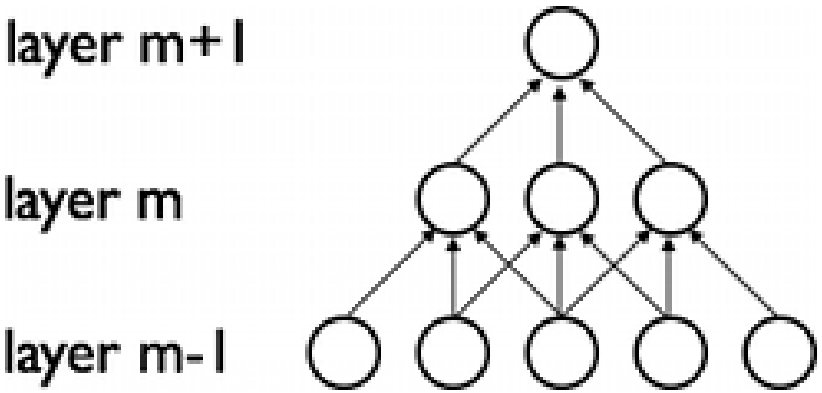
Neurons in layer m are only connected to 3 adjacent neurons in the m-1 layer.

Neurons in layer m+1 have a similar connectivity with the layer below.

Each neuron is unresponsive to variations outside of its *receptive field* with respect to the input.

- **Receptive field: small neuron collections which process portions of the input data**

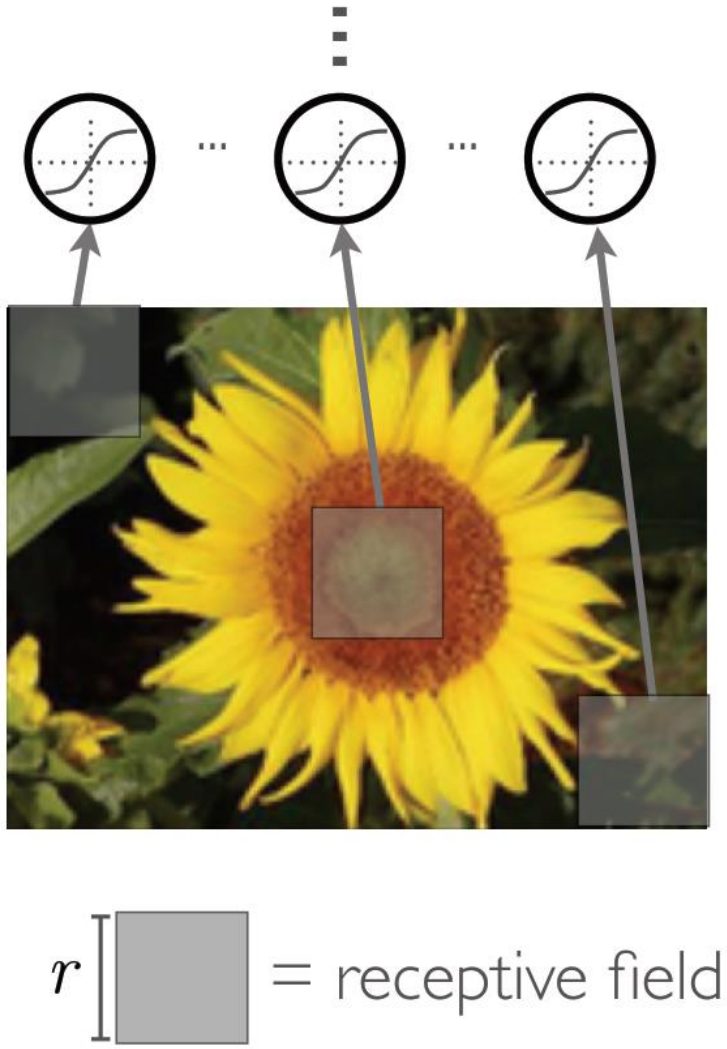
The architecture thus ensures that the learnt feature extractors produce the strongest response to a spatially local input pattern.



CNN First idea: Local Connectivity

Illustration of local connectivity

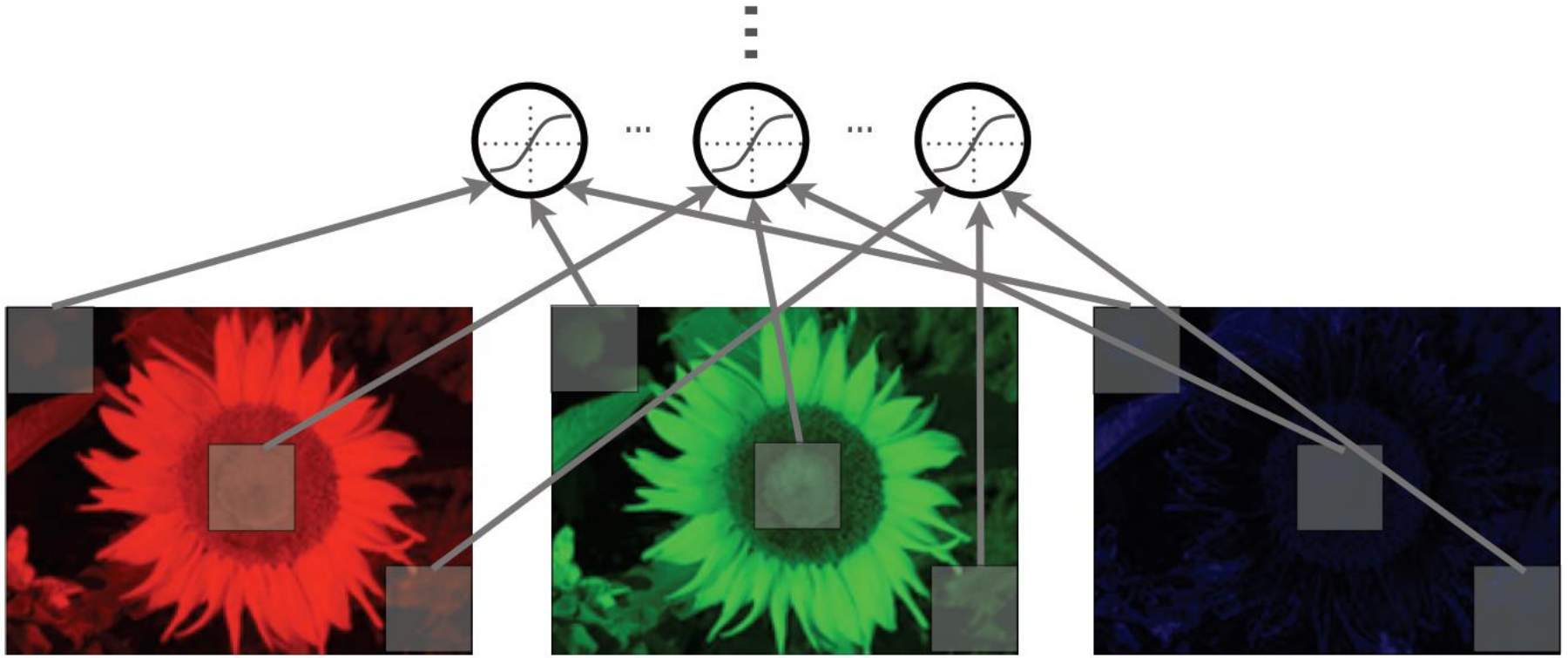
- First idea: use a local connectivity of hidden units
  - each hidden unit is connected only to a subregion (patch) of the input image
  - it is connected to all channels
    - 1 if greyscale image
    - 3 (R, G, B) for color image
- Solves the following problems:
  - fully connected hidden layer would have an unmanageable number of parameters
  - computing the linear activations of the hidden units would be very expensive



CNN First idea: Local Connectivity

Illustration of local connectivity

- Units are connected to all channels:
  - 1 channel if grayscale image, 3 channels (R, G, B) if color image



CNN Second idea: parameter sharing

We show 3 hidden neurons belonging to the same feature map (the layer right above the input layer).

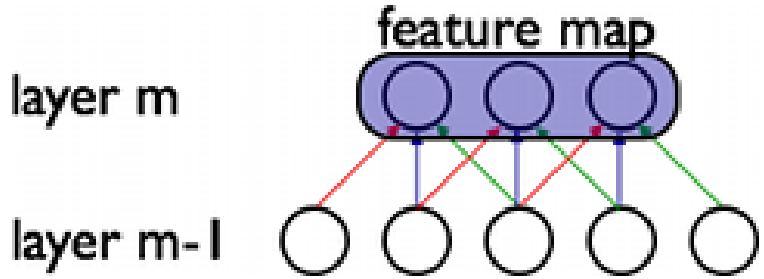
Weights of the same color are shared—constrained to be identical.

Gradient descent can still be used to learn such shared parameters, with only a small change to the original algorithm.

The gradient of a shared weight is simply the sum of the gradients of the parameters being shared.

Replicating neurons in this way allows for features to be detected regardless of their position in the input.

Additionally, weight sharing increases learning efficiency by greatly reducing the number of free parameters being learnt.



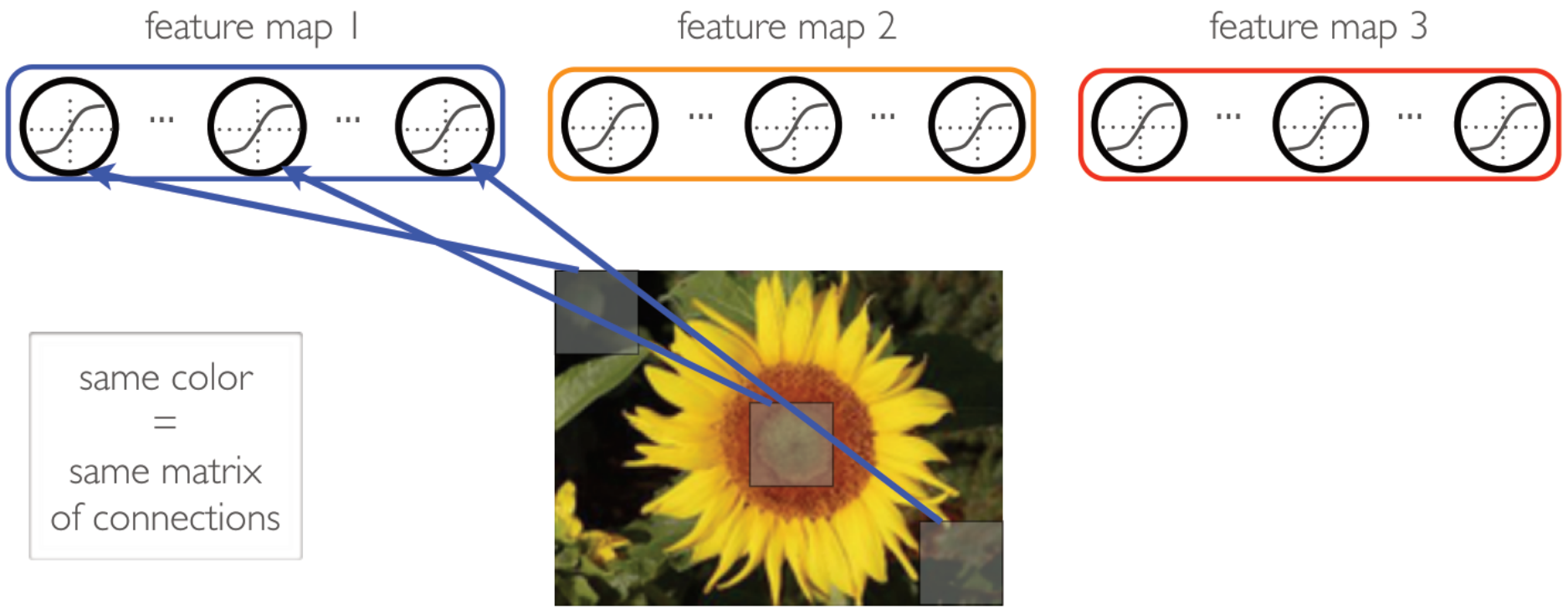
CNN Second idea: parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image

same color
=
same matrix
of connections

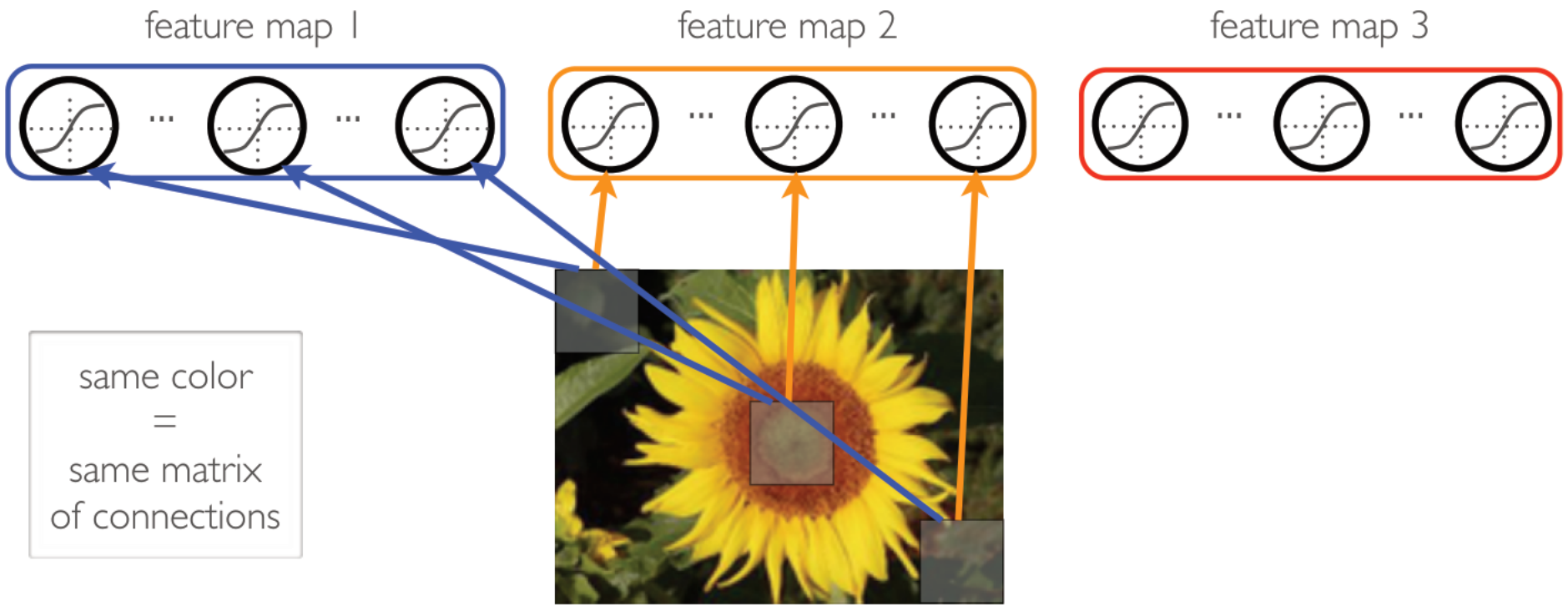
CNN Second idea: parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image



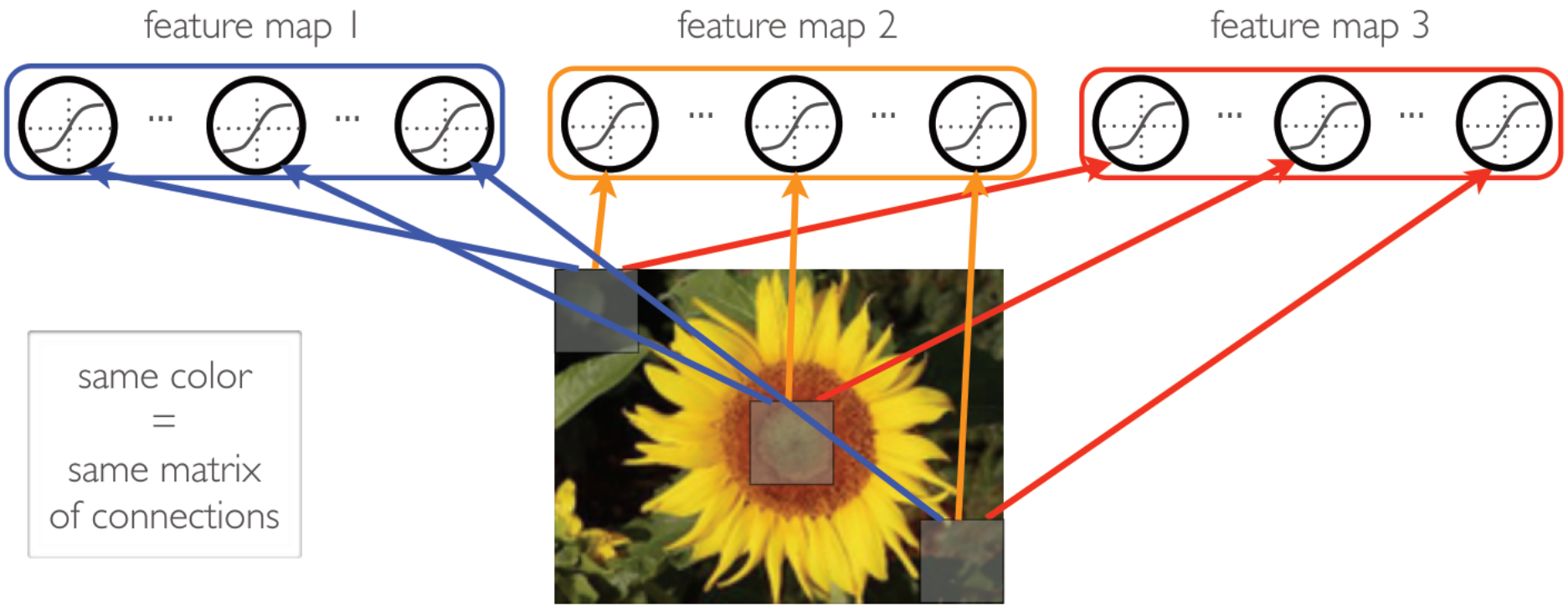
CNN Second idea: parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image



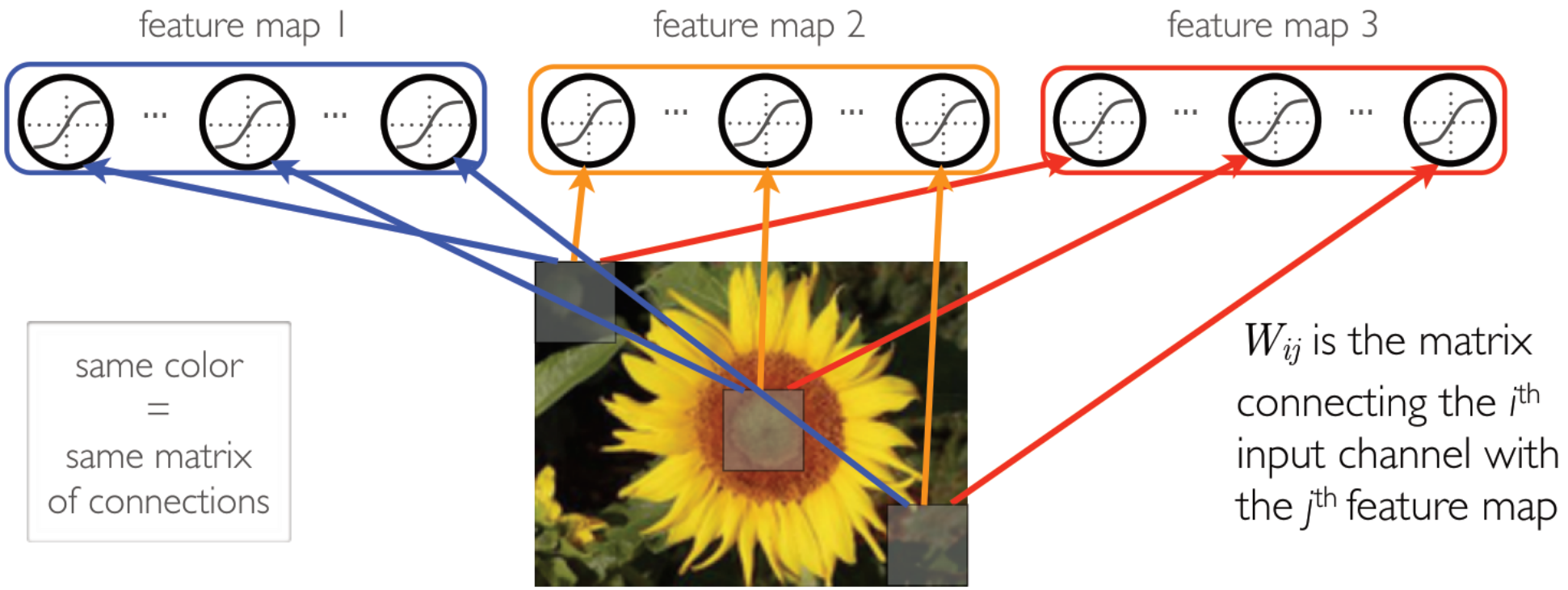
CNN Second idea: parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image



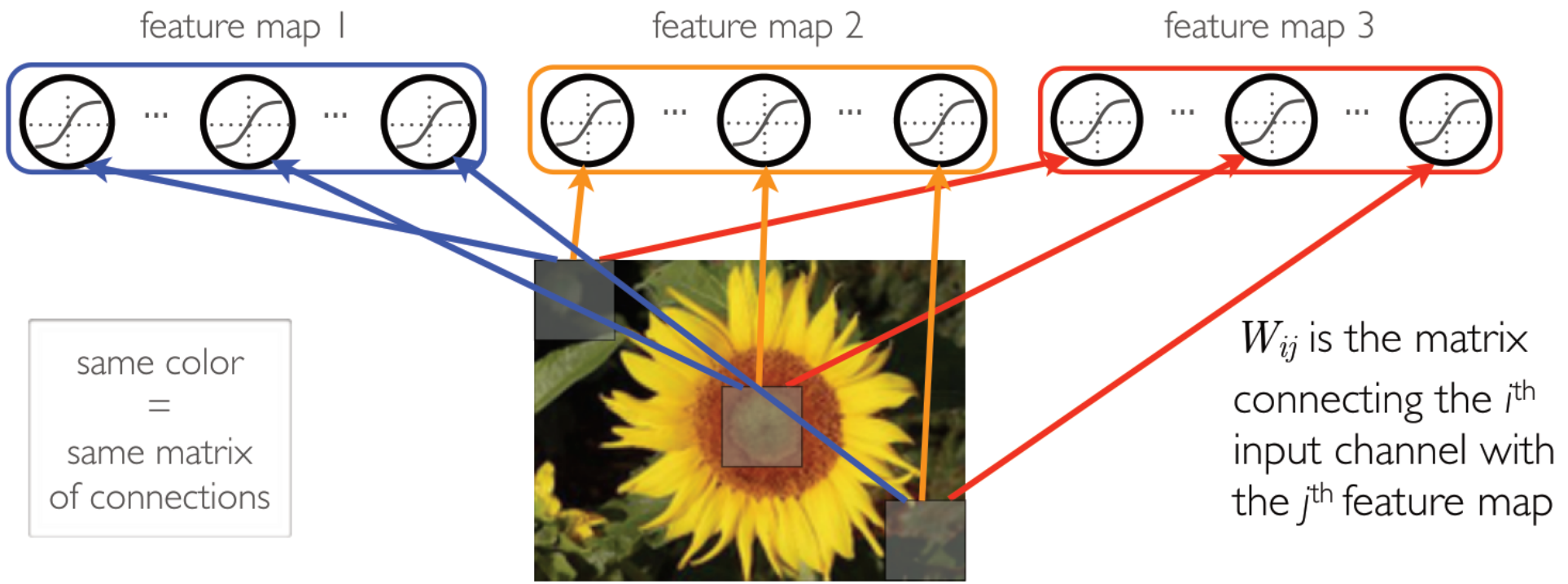
CNN Second idea: parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image



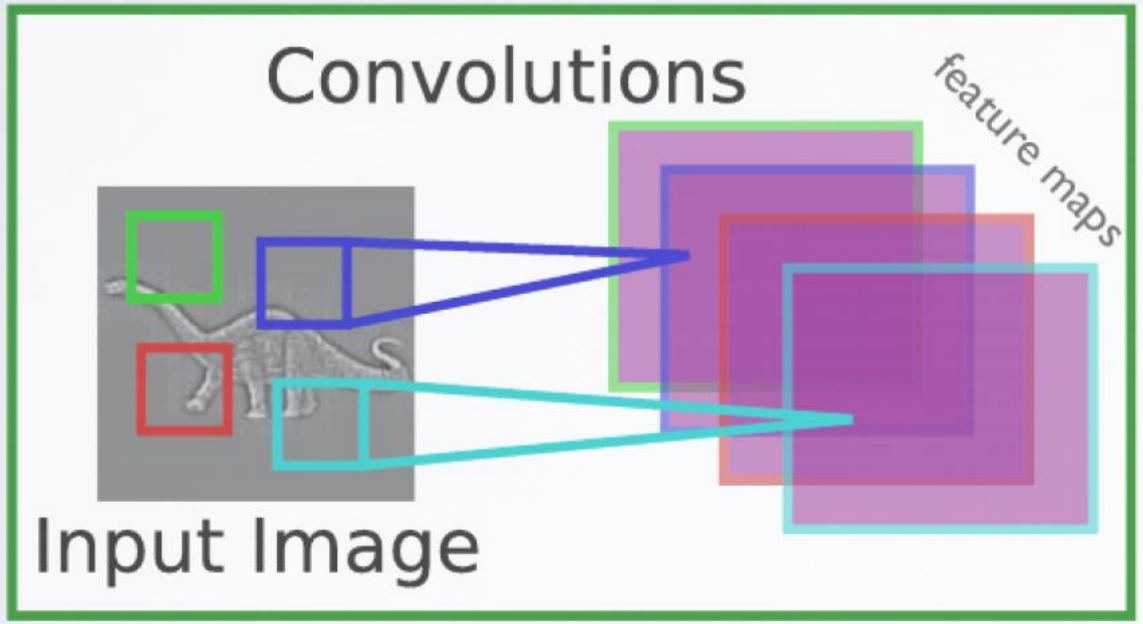
CNN Second idea: parameter sharing

- Solves the following problems:
  - reduces even more the number of parameters
  - will extract the same features at every position (features are "equivariant")



CNN Second idea: parameter sharing

- Each feature map forms a 2D grid of features
  - can be computed with a discrete convolution ($*$) of a kernel matrix k_{ij} which is the hidden weights matrix W_{ij} with its rows and columns flipped



- x_i is the ith channel of input
- k_{ij} is the convolution kernel
- g_j is a learned scaling factor
- y_j is the hidden layer

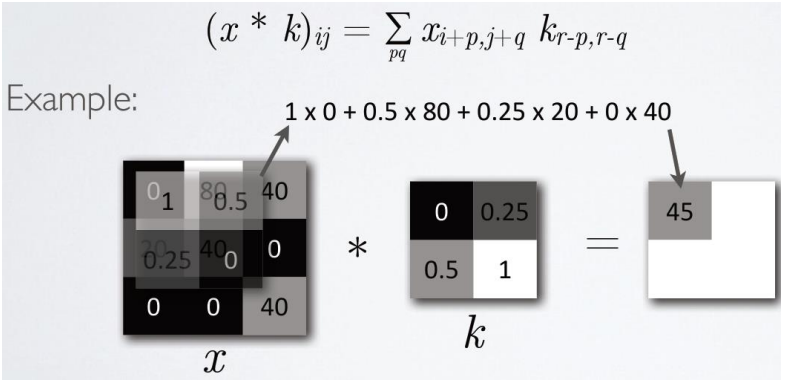
(could have added a bias)

$$y_j = g_j \tanh(\sum_i k_{ij} * x_i)$$

Jarret et al. 2009

CNN Architecture: Convolutional Layer

- The core layer of CNNs
- The convolutional layer consists of a set of filters.
  - Each filter covers a spatially small portion of the input data.
- Each filter is convolved (sliding window style) across the dimensions of the input data, producing a multidimensional feature map.
  - As we convolve the filter, we are computing the dot product between the parameters of the filter and the input.



- Intuition: the network will learn through supervised training, the filters that activate when they see some specific type of feature at some spatial position in the input.
- The key architectural characteristics of the convolutional layer is local connectivity and shared weights.

- Convolutional layer consists of a set of filters each is a template pattern of interest to be learned to look for in the data.

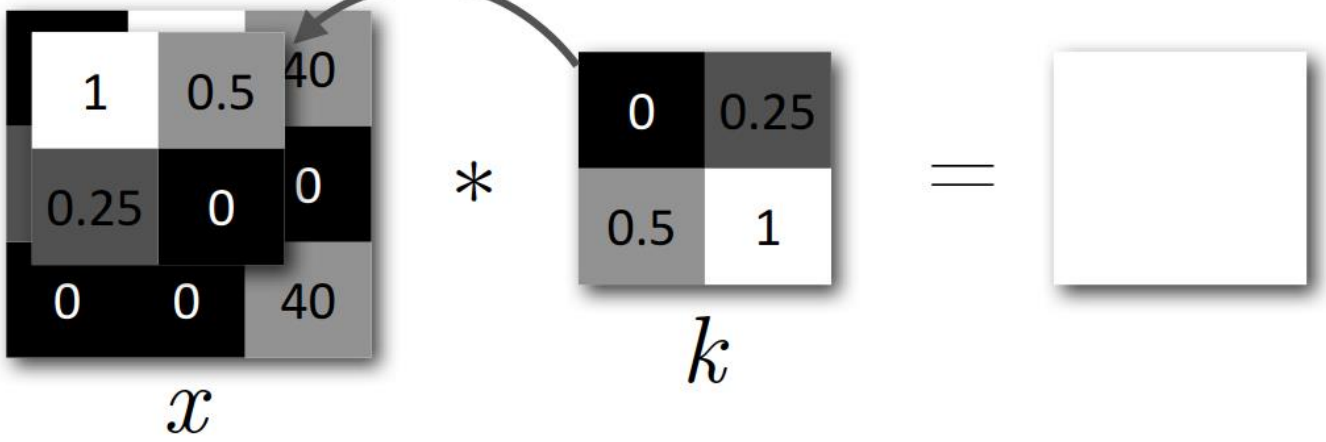
CNN Architecture: Convolutional Layer

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q}\ k_{r-p,r-q}$$

- Example:

$\tilde{k} = k$ with rows and columns flipped

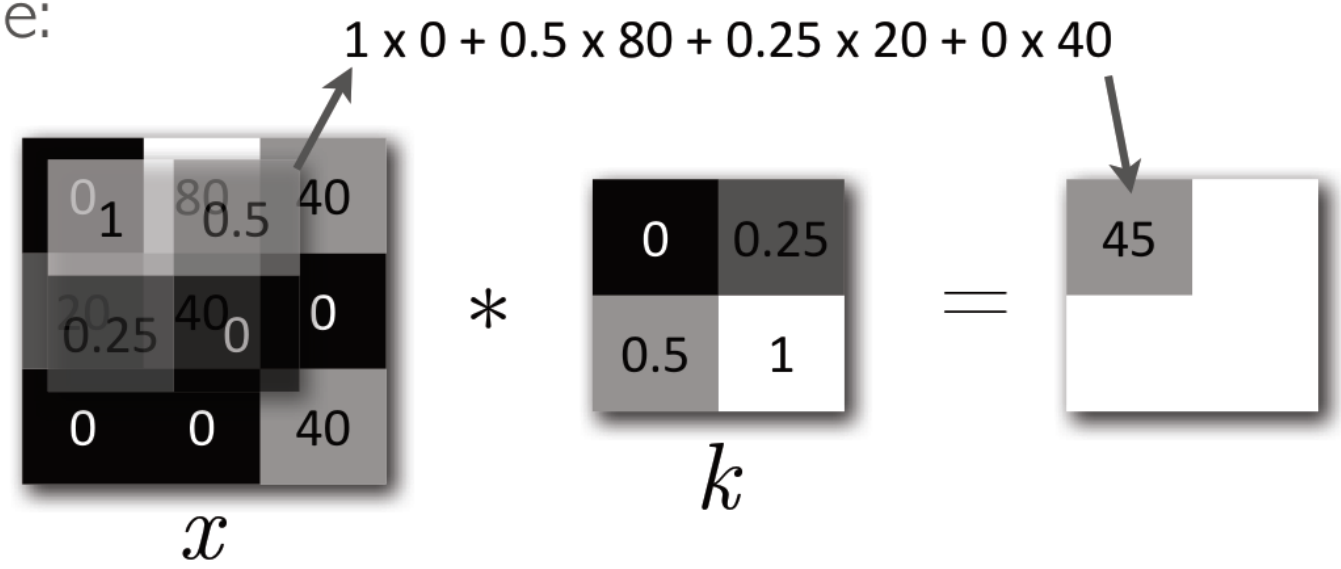


CNN Architecture: Convolutional Layer

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:

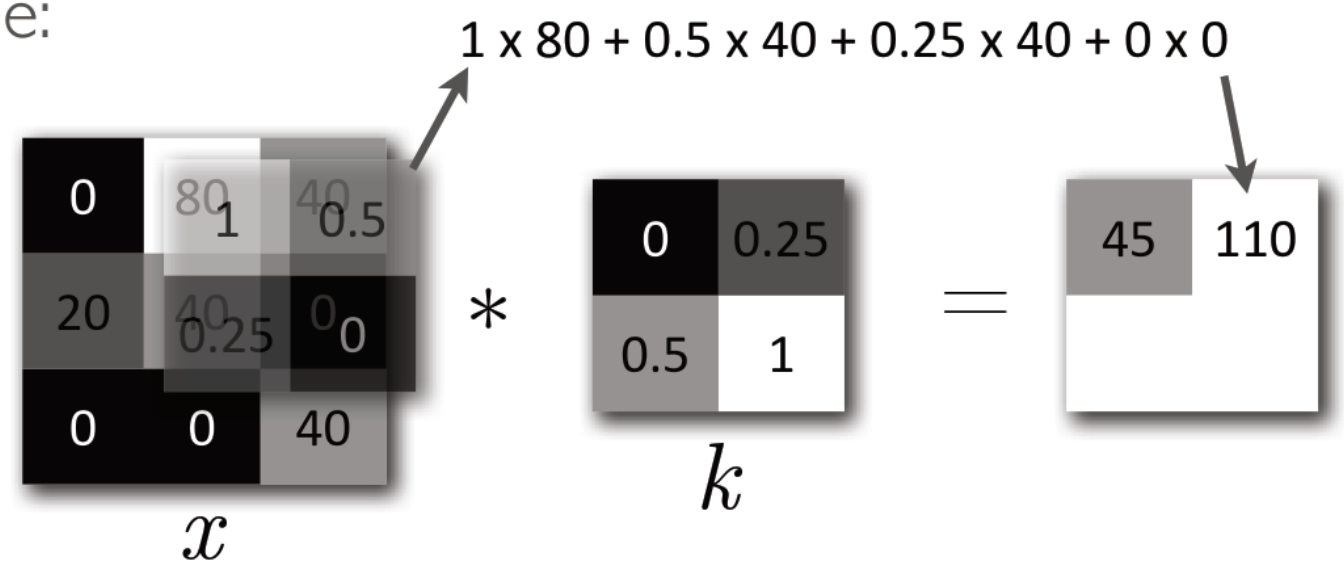


CNN Architecture: Convolutional Layer

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:

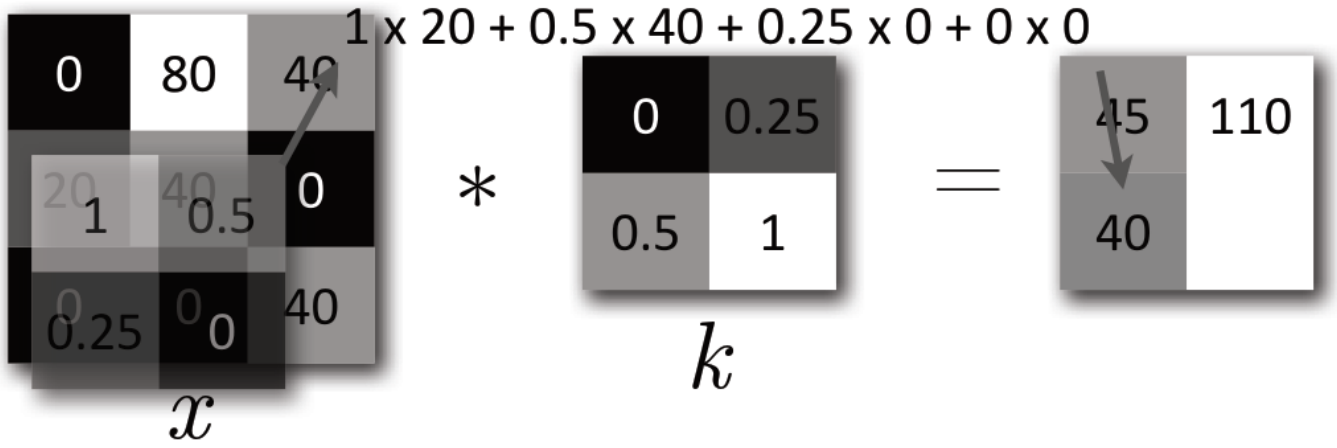


CNN Architecture: Convolutional Layer

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:

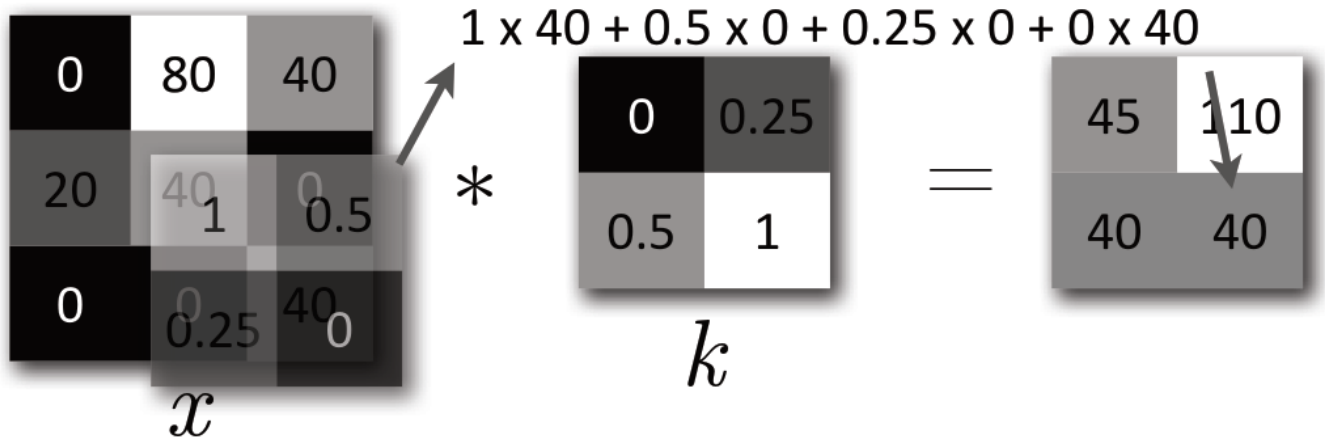


CNN Architecture: Convolutional Layer

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:



CNN Architecture: Non-linear Layer

Intuition: Non-linearly transforming the convolution output increases the type of relationships recoverable between input and the output.

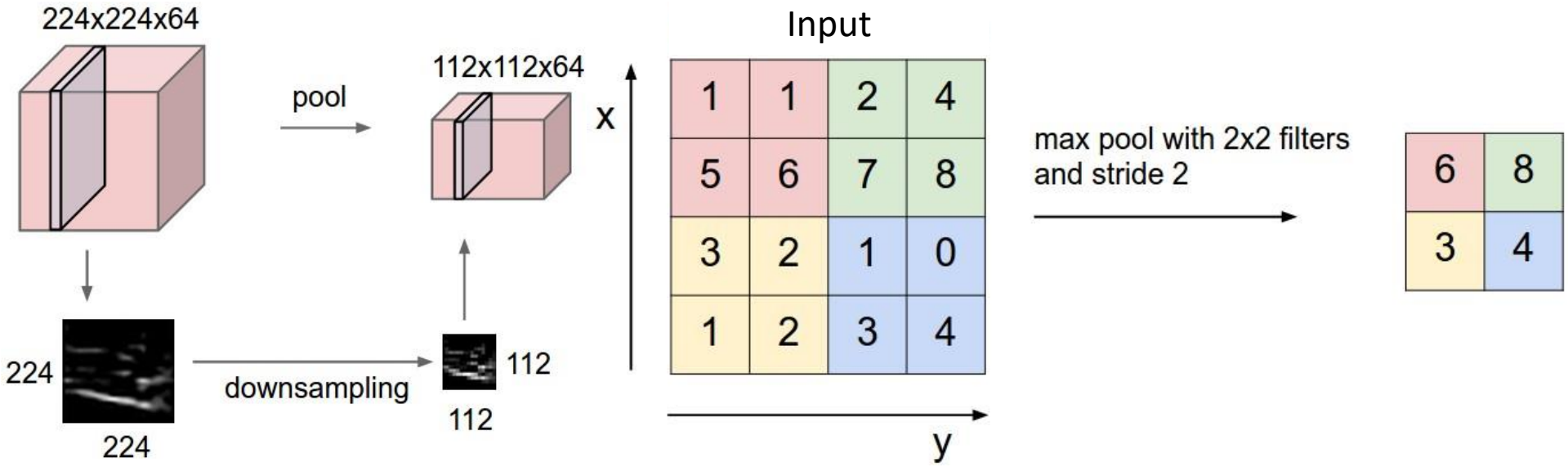
Example nonlinear activation functions including:

- $f(x) = \max(0, x)$
- $f(x) = \tanh x$
- $f(x) = |\tanh x|$
- $f(x) = (1 + e^{-x})^{-1}$

CNN Third idea: Pooling and subsampling layer

Intuition: to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting

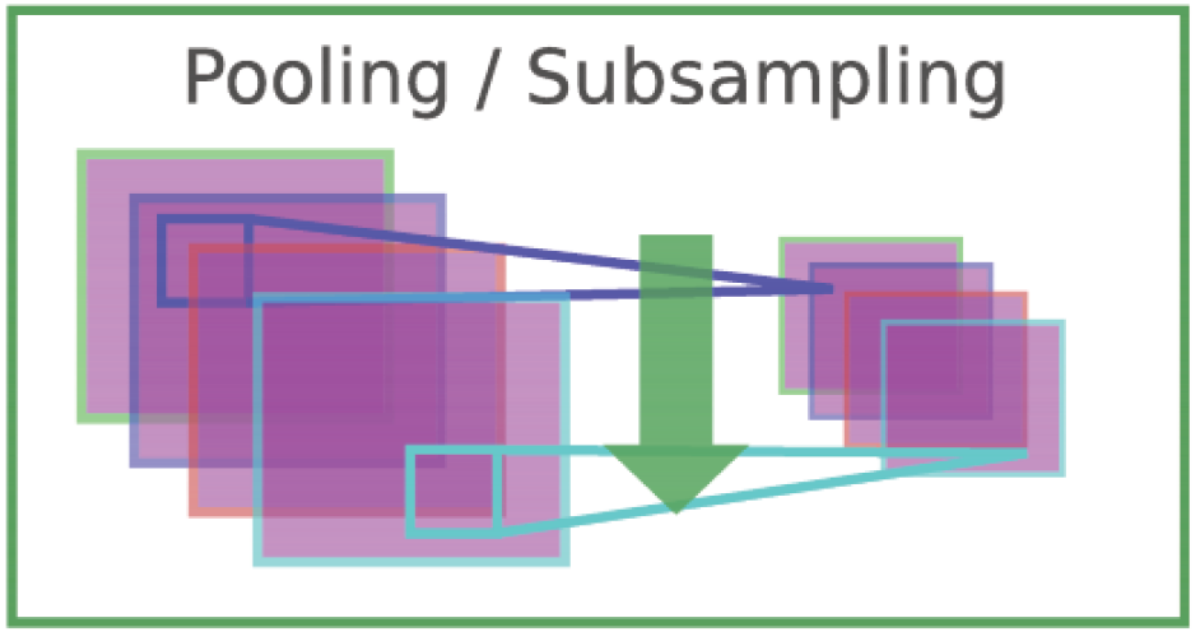
Pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value of the features in that region.



CNN Architecture: Pooling and subsampling layer

Pool hidden units in the same neighborhood: illustration with multiple feature maps

- pooling is performed in non-overlapping neighborhoods (subsampling)



Jarret et al. 2009

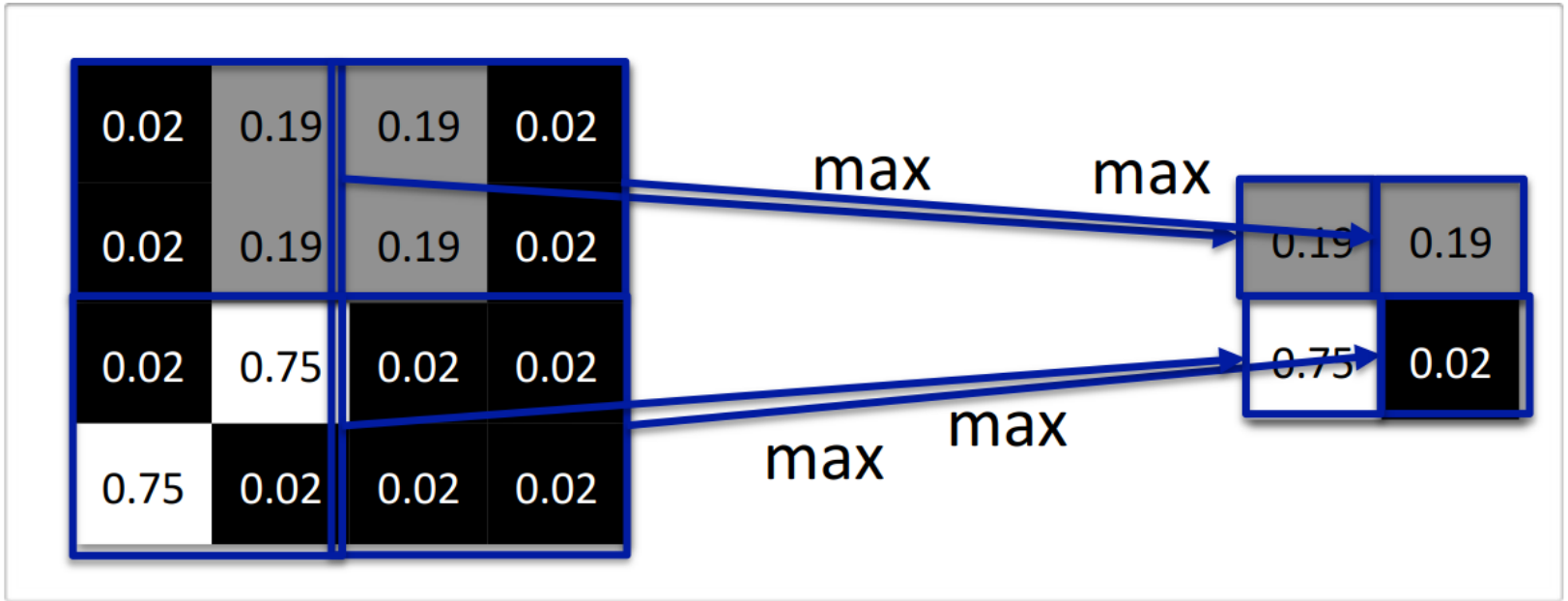
- $x_{i,j,k}$ is value of the i[th] feature map at position j,k
- p is vertical index in local neighborhood
- q is horizontal index in local neighborhood
- y_{ijk} is pooled and subsampled layer

$$y_{ijk} = \max_{p,q} x_{i,j+p,k+q}$$

CNN Architecture: Pooling and subsampling layer

Pool hidden units in the same neighborhood

- Illustration of pooling/subsampling operation

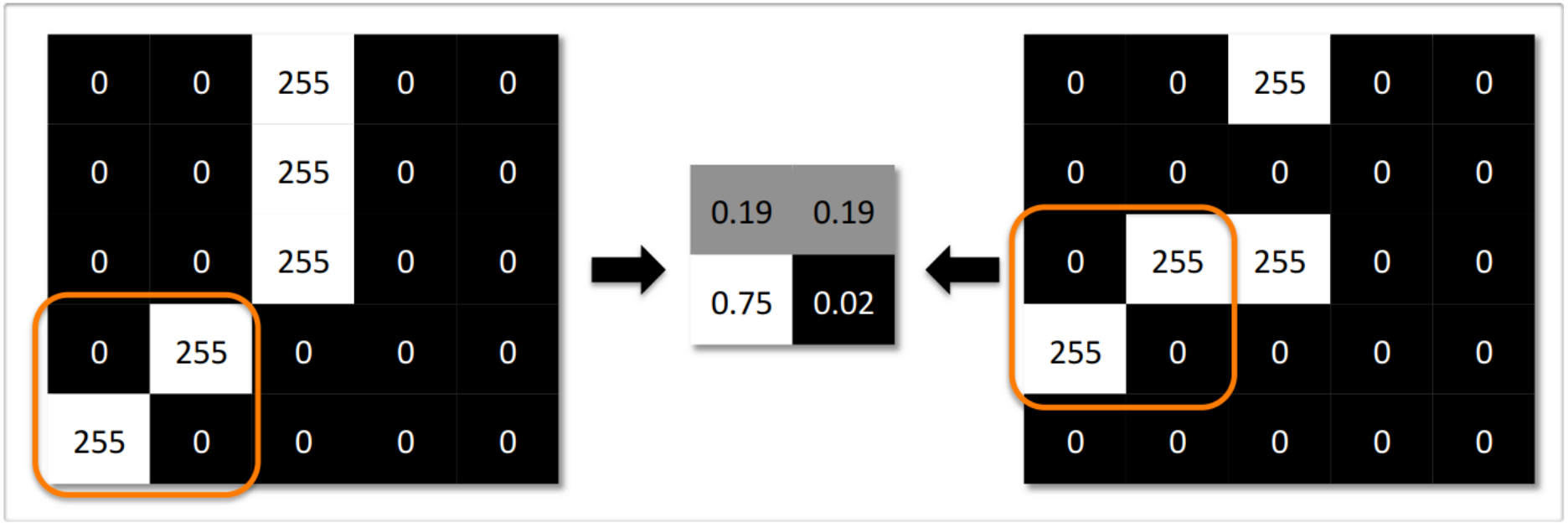


- Solves the following problems:
  - introduces invariance to local translations
  - reduces the number of hidden units in hidden layer

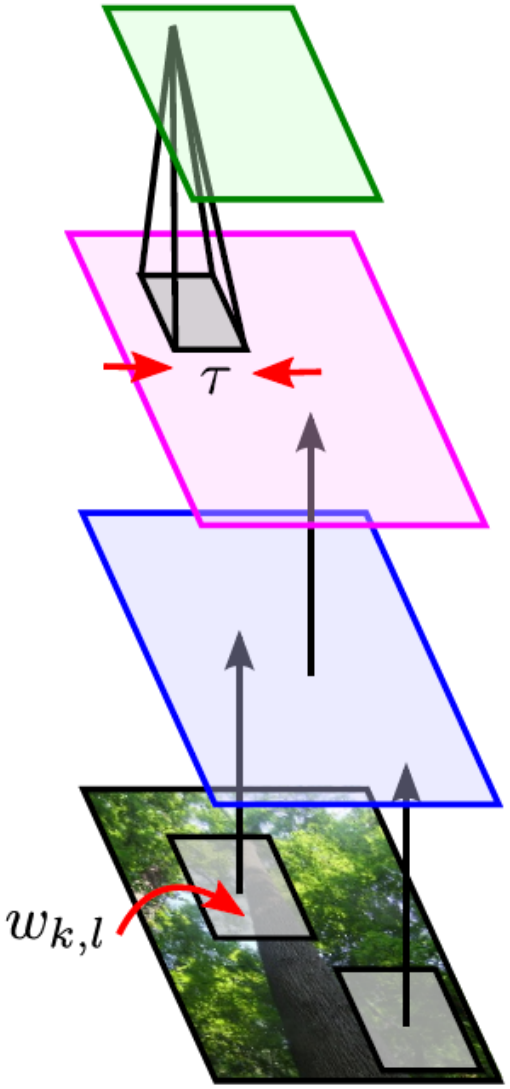
CNN Architecture: Pooling and subsampling layer

Illustration of local translation invariance

Max operator allows to yield the same feature map even if there are some translations in the input. Smaller shifts in the image do not require additional features to be learned.



Building-blocks for CNN's



$$x_{i,j} = \max_{|k|<\tau,|l|<\tau} y_{i-k,j-l}$$

mean or subsample also used

pooling stage

Feature maps of a larger region are combined.

$$y_{i,j} = f(a_{i,j})$$

e.g. $f(a) = [a]_+$

$f(a) = \text{sigmoid}(a)$

non-linear stage

Feature maps are trained with neurons.

$$a_{i,j} = \sum_{k,l} w_{k,l} z_{i-k,j-l}$$

Shared weights

convolutional stage

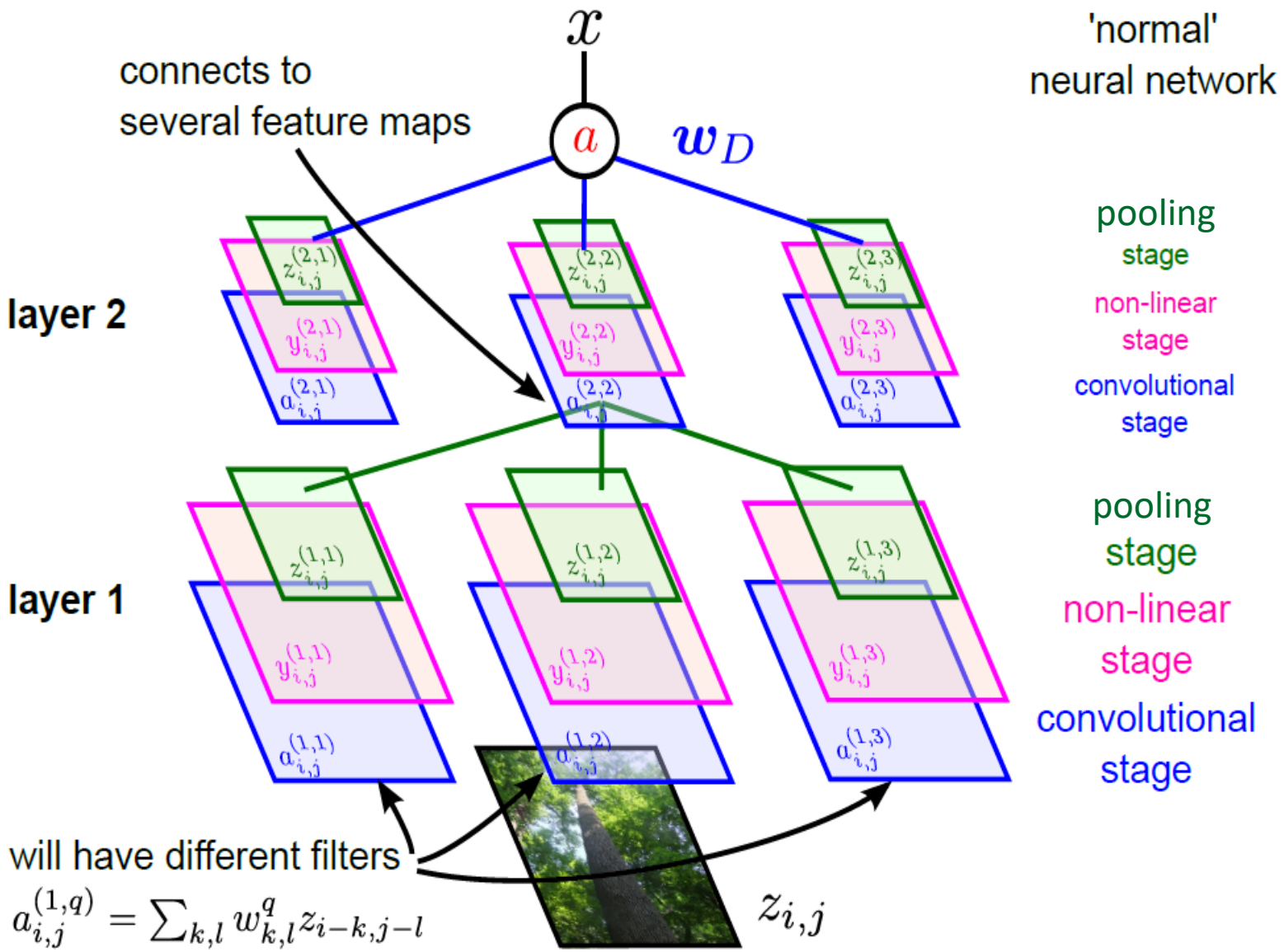
Each sub-region yields a feature map, representing its feature.

$z_{i,j}$

Images are segmented into sub-regions.

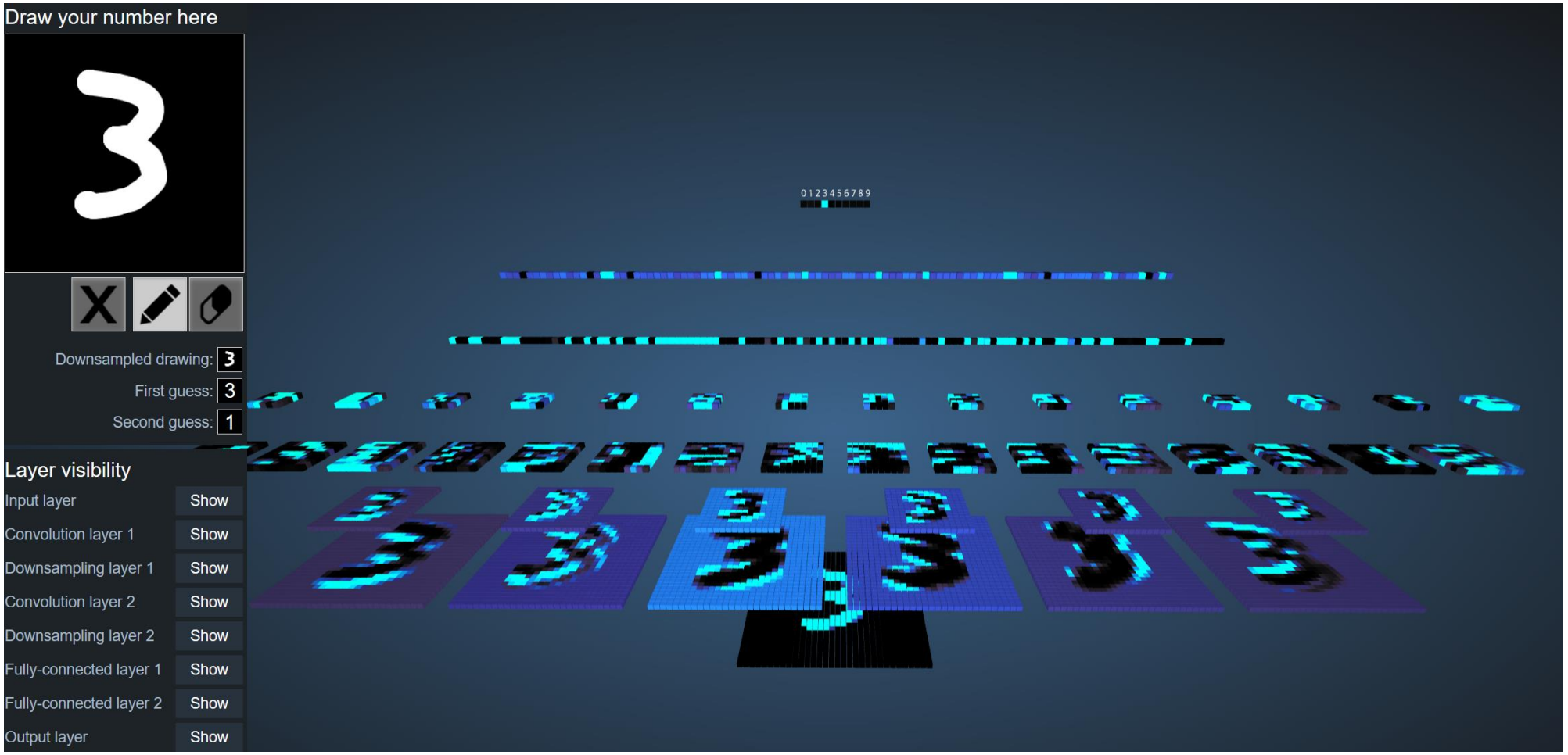
input image

Full CNN


x
connects to
several feature maps
a
w_D
'normal'
neural network
pooling
stage
non-linear
stage
convolutional
stage
layer 2
layer 1
pooling
stage
non-linear
stage
convolutional
stage
will have different filters
$a_{i,j}^{(1,q)} = \sum_{k,l} w_{k,l}^{q} z_{i-k,j-l}$
$z_{i,j}$

Innovative visualizations of what the network as learned

digit recognition CNN
http://scs.ryerson.ca/~aharley/vis/conv/



Questions

Albert.Montillo@UTSouthwestern.edu