Machine Learning: Algorithms and Applications

Philip O. Ogunbona

Advanced Multimedia Research Lab
University of Wollongong

Artificial Neural Networks and Deep Learning: An Introduction (II)

Outline

CNN - Convolutional Neural Network

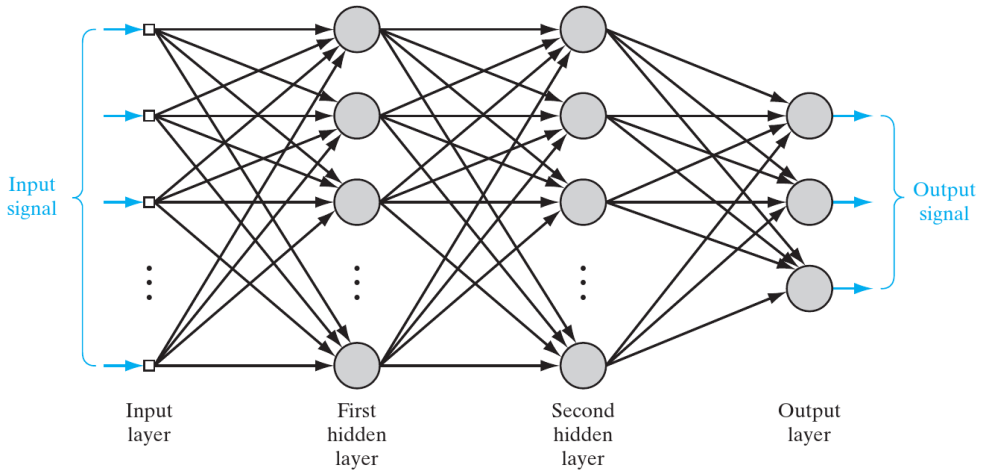

Figure 1: Fully connected MLP (Haykin 2009)

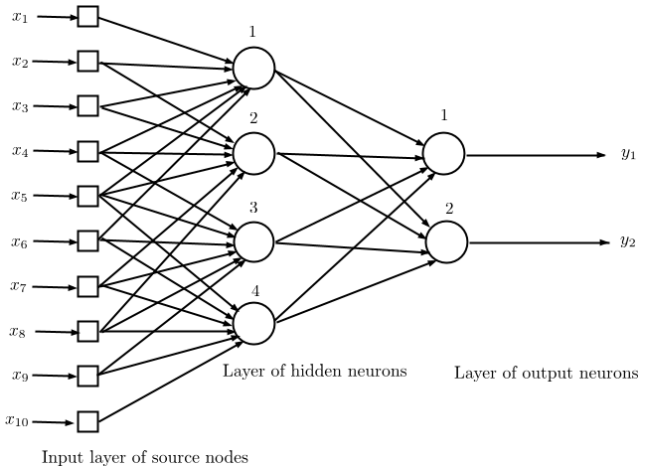

Figure 2: Simple CNN architecture (Haykin 2009)

- Input-output relationship for a layer can be described by matrix multiplication

$$\boldsymbol{v_i} = \boldsymbol{W}\boldsymbol{v_{i-1}} \tag{1}$$

Note the use of vector (bold) notation in the multiplication

- Input-output relationship for hidden layer is described by convolution (weight matrix will have to be Toeplitz)

$$v_j = \sum_{i=1}^{6} \omega_i x_{i+j-1}; \quad j = 1, 2, 3, 4 \tag{2}$$

$\{\omega_i\}_{i=1}^{6}$ constitute the same set of weights shared by all four hidden neurons

CNN - Convolutional Neural Network

- CNNs are a special class of multilayer perceptron that are well suited to processing data with grid-like topology.
- Examples include
  - Time-series data : 1-D grid taking samples at regular intervals
  - Image data : 2-D grid of pixels
- CNNs are networks that use convolution in place of general matrix multiplication in at least one of the layers
- CNN is a way of building prior information into neural network design (see Figure 2)
  1. Network architecture restriction - use local connection a.k.a receptive fields
  2. Constrain the choice of synaptic weight - use weight sharing
- CNNs are sometimes designed to recognize 2-D shapes with high degree of invariance to translation, scaling, skewing and other distortions

CNN - Convolutional Neural Network

- More generally structural constrain in CNN aim to achieve:
  - Feature extraction: Each neuron takes its synaptic inputs from a local receptive filed of neurons forcing it to extract local features
  - Feature mapping: Each computational layer has multiple feature maps; they form planes in which neurons are forced to share same set of synaptic weights; beneficial effects
    - shift invariance - achieved through convolution followed by activation function
    - reduction in the number of free parameters - achieved through weight sharing
  - Subsampling (a.k.a pooling) - Convolutional layer followed by a computational layer performing local averaging and subsampling
    - feature map resolution reduction
    - reduction of sensitivity of feature map output to shifts and other distortions

Typical contemporary CNN architecture

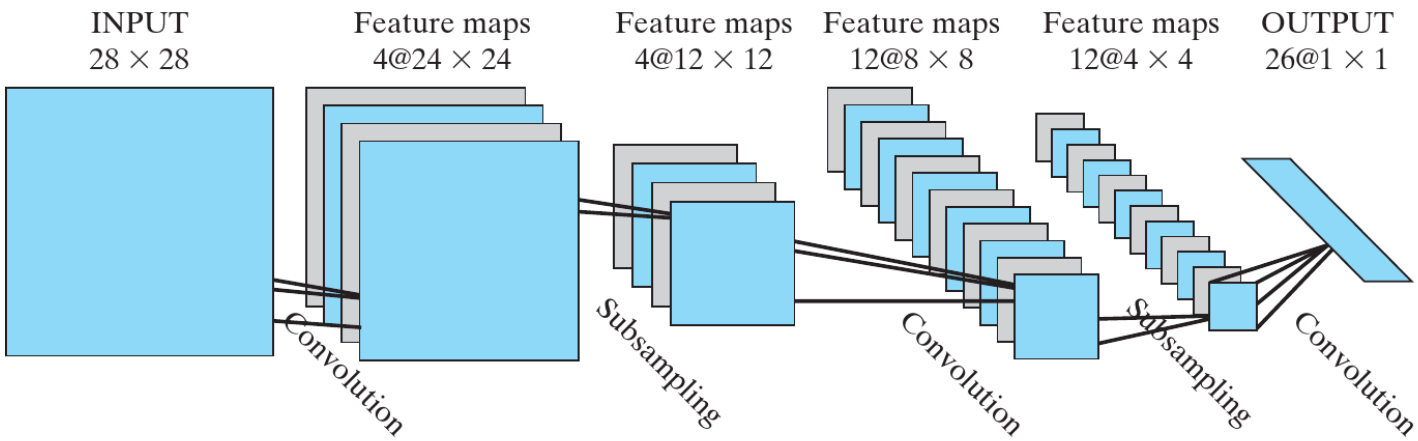

Figure 3: CNN architecture designed for hand-written character recognition (Haykin 2009)

1. Four (4) hidden layers and an output layer
2. First hidden layer - convolution; four feature maps of 24×24 neurons; each neuron assigned a receptive field of size 5×5
3. Second hidden layer - subsampling and local averaging; 4 feature maps of 12×12; each neuron has receptive field size 2×2
4. Third hidden layer - convolution; 12 feature maps of 8×8 neurons;
5. Fourth layer - subsampling and local averaging; 12 feature maps of 4×4 neurons
6. Final layer convolution; 26 neurons, each assigned to one of possible 26 characters; each neuron assigned to receptive field 4×4

What is convolution? A deeper insight

- Given two functions $x(n)$ and $\omega(n)$ the convolution of the two functions is written as

$$s(n) = \sum_k x(k)\omega(n-k) = \sum_k x(n-k)\omega(k) \tag{3}$$

  Equation 3 says, flip the function $\omega(n)$ relative to $x(n)$ and slide it across the function $x(n)$, each time compute the product of overlapping samples; note commutative property of convolution

- For the purpose of neural networks, $x(n)$ is the input; $\omega(n)$ is the kernel and $s(n)$ is the feature map
- In the two-dimensional case we have

$$s(n,m) = \sum_{k,l} x(k,l)\omega(n-k,m-l) = \sum_{k,l} \omega(k,l)x(n-k,m-l) \tag{4}$$

  Summation is over the non-zero values of the kernel; usually defined to be finite in spatial extent

- Convolution as defined in Equation 4 is rarely used in machine learning; rather use cross-correlation (but referred to as convolution); See Figure 4

$$s(n,m) = \sum_{k,l} x(k,l)\omega(n+k,m+l) \tag{5}$$

Convolution (without flipping) operation in 2-D case

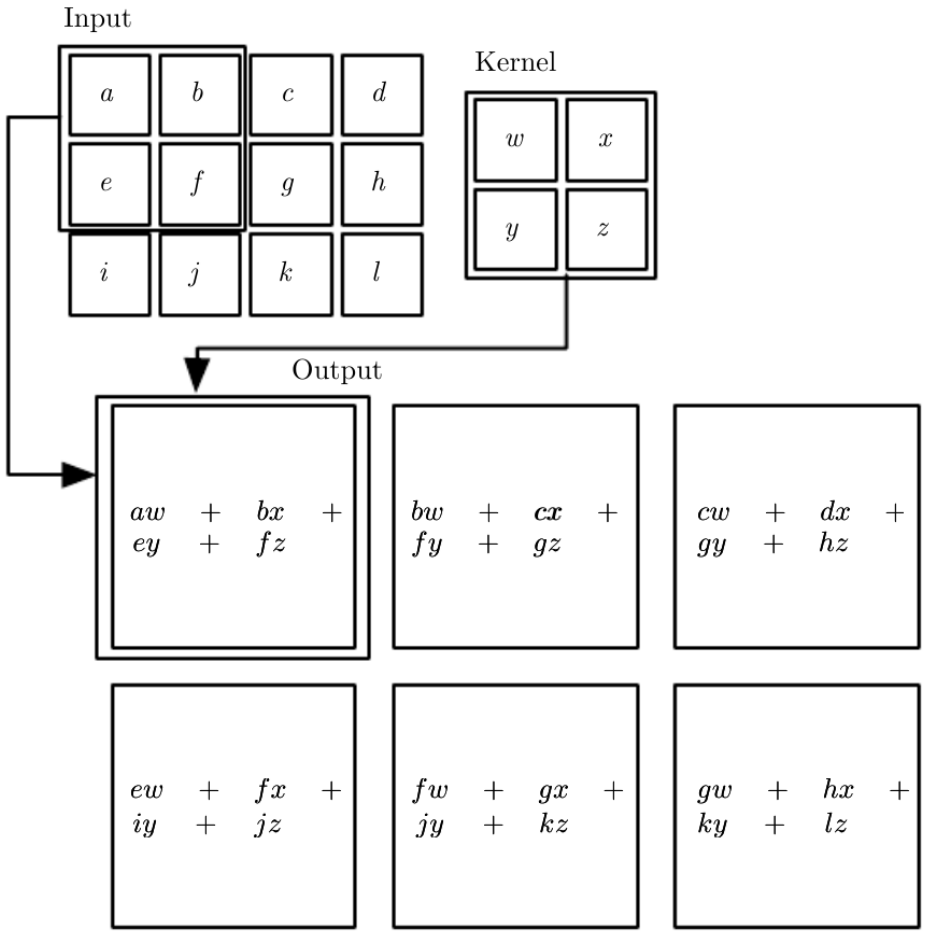

Figure 4: 2-D convolution operation (Goodfellow et al. 2016)

Convolution benefits revisited

- Convolution introduces three ideas beneficial for machine learning systems:
  1. Sparse interactions
  2. parameter sharing
  3. Equivariant representation

Sparse interactions

- Sparse weights leads to needing fewer parameters to store and improving statistical efficiency
- In deep CNN deeper layers indirectly interact with a larger portion of the input allowing network to efficiently describe complicated interactions

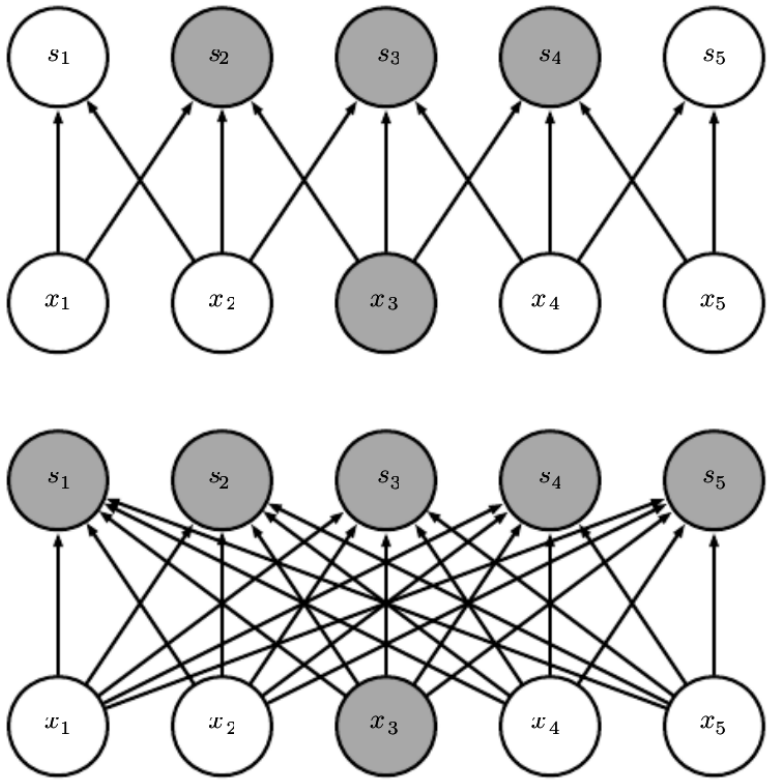

Figure 5: Sparse connectivity viewed from input (note how many outputs affected by one input) (Goodfellow et al. 2016)

Sparse interactions

- Sparse weights leads to needing fewer parameters to store and improving statistical efficiency
- In deep CNN deeper layers indirectly interact with a larger portion of the input allowing network to efficiently describe complicated interactions

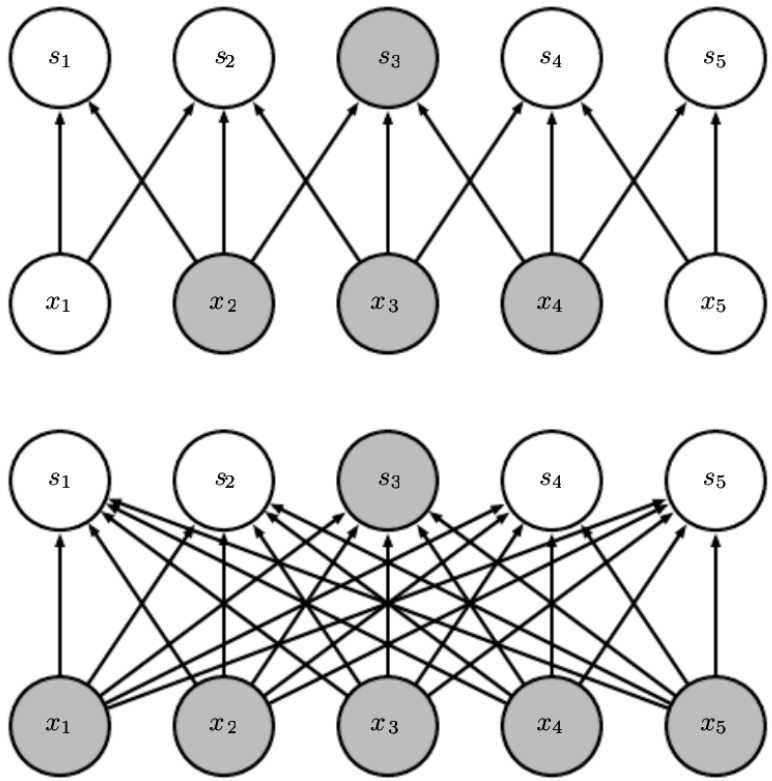

Figure 5: Sparse connectivity viewed from output (note how many inputs affect one output) (Goodfellow et al. 2016)

Parameter sharing

- Rather than learn separate set of parameters for every location we learn only one set
- Since kernel is usually much less than input data size, convolution is more efficient than dense matrix multiplication

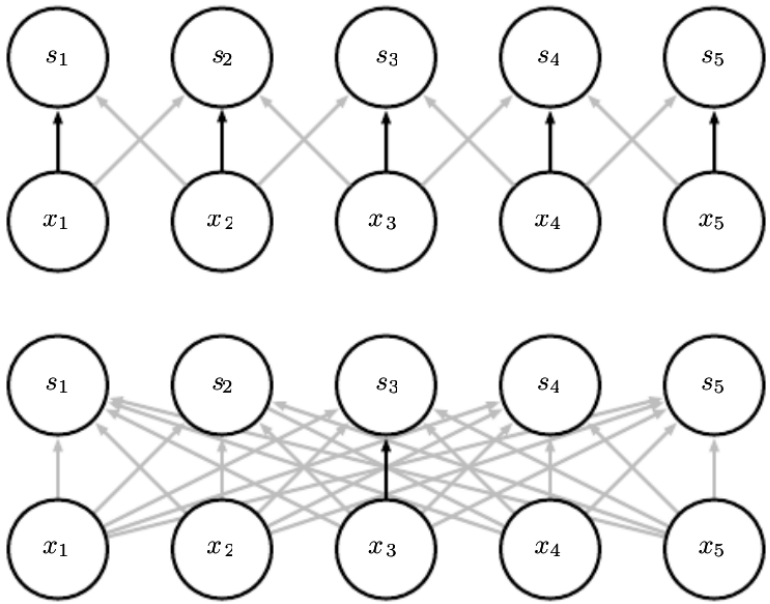

Figure 6: Parameter sharing; (top) central element in 3-element kernel is used multiple times; (bottom) Shown central element of weight matrix is used once in the fully connected architecture (Goodfellow et al. 2016)

Equivariance

- Convolution as considered here leads to a form of parameter sharing that generates a property called equivariance to translation. In signal processing this is the same as linear time invariance.
- Equivariant means that if the input changes, the output changes in the same way.
- For example, let g map an image I to another image I'; $I' = g(I)$ $I'(x, y) = I(x - 1, y)$; applying convolution after the mapping is the same as applying the convolution and then transforming the result.
- In a data stream if we delay a feature in time (or spatially) the same representation appears in the output later (by amount of the delay)

Pooling

- A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs.
- Examples:
  1. Max pooling operation reports the maximum output within a rectangular neighbourhood (E.g. see Figure 7)
  2. L^2-norm of a rectangular neighbourhood
  3. Weighted average based on distance from central pixel
- Pooling helps to make the representation approximately invariant to small translations of input (e.g. see Figure 8)
- Use of pooling can be interpreted as using a strong prior that the function being learned (in the layer) must be invariant to small translation
- Pooling over over spatial regions leads to translation invariance
- Pooling over separately parametrized convolutions leads to learning transformations that are invariant (e.g. see Figure 9)

Pooling

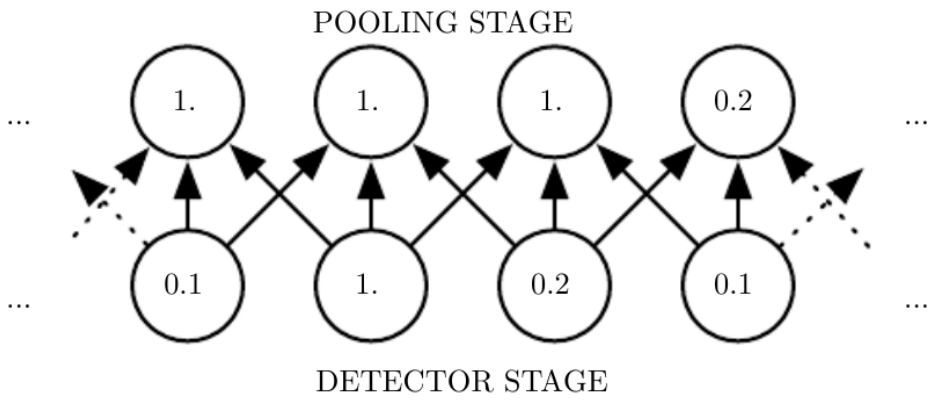

Figure 7: Max pooling of width 3 and stride 1 (Goodfellow et al. 2016)

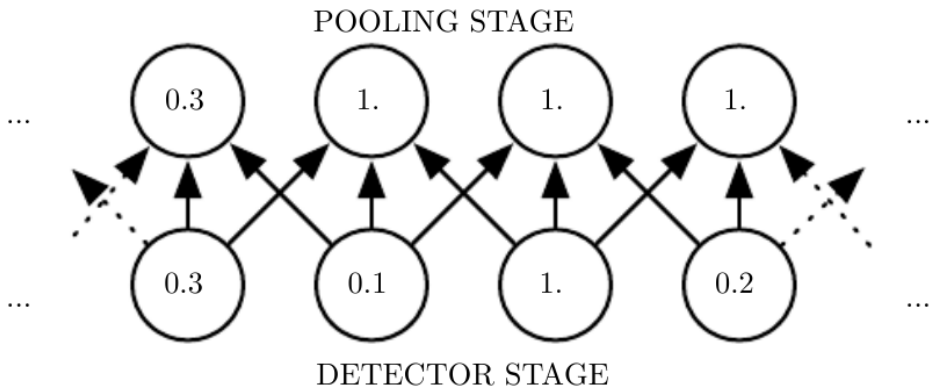

Figure 8: Max pooling of width 3 and stride 1 and input shifted to the right (to show invariance to translation) (Goodfellow et al. 2016)

Pooling

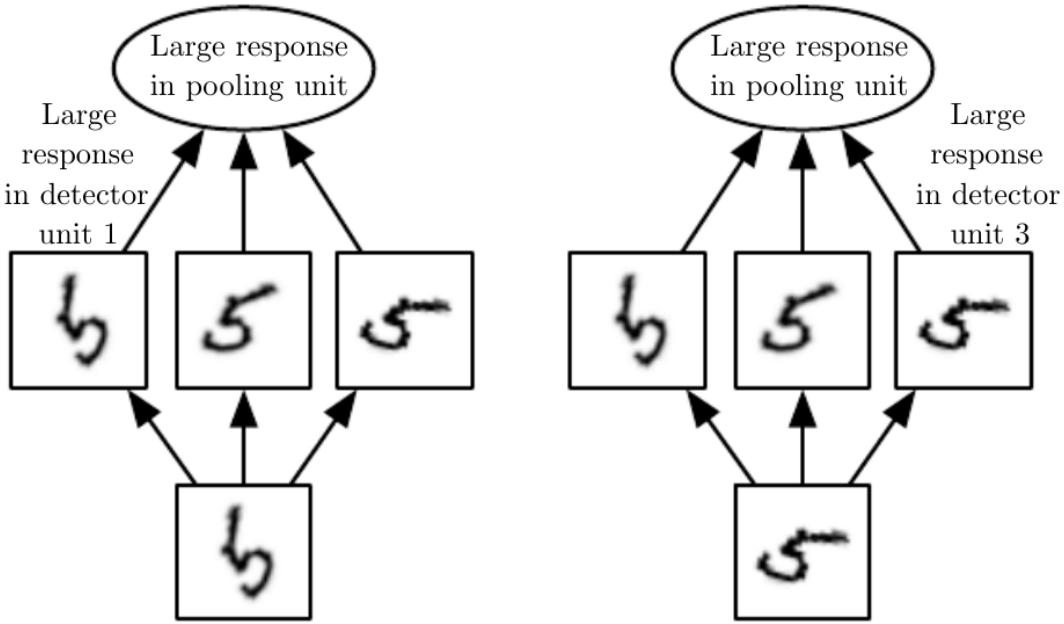

Figure 9: Pooling over multiple features that are learned with separate parameters can learn to be invariant to transformations of the input. (Goodfellow et al. 2016)

Pooling

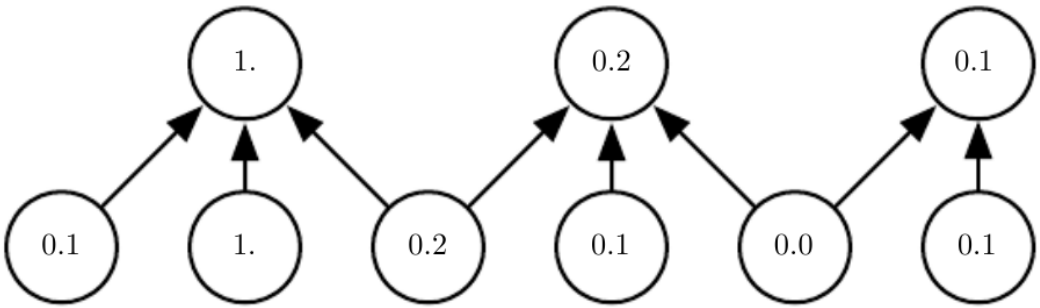

Figure 10: Pooling with downsampling: max-pooling with a pool width of three and a stride between pools of two. (Goodfellow et al. 2016)

Popular Convolutional NN Architectures

LeNet-5 Architecture

- Created by Yann LeCunn (1998)

Layer	Type	Maps	Size	Kernel	Stride	Activation
Out	Fully Connected	-	10	-	-	RBF
F6	Fully Connected	-	84	-	-	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Poling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	-	-	-

Table 1: LeNet-5 architecture (Géron 2017)

Popular Convolutional NN Architectures

LeNet-5 Architecture

- Tested on the MNIST dataset (padding required - 32×32)
- Pooling layer - special average pooling $((\mu \times c_i) + b_i)$; c_i and b_i are learnable parameters; μ is the average of the nodes in the activaton field of the filter
- Encouraged to find and read original paper by LeCunn

Popular Convolutional NN Architectures

AlexNet

- Created by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton
- Won 2012 ImageNet ImageNet Large Scale Visual Recognition Competition (ILSVRC)
- Similar to LeNet but larger and deeper
- Note that it stacked Conv layers atop each other in layers C5, C6, C7, rather than “Conv-Pool” configuration (see Table 2)

Popular Convolutional NN Architectures

AlexNet

Layer	Type	Maps	Size	Kernel	Stride	Padding	Activation
Out	Fully Connected	-	1,000	-	-	-	Softmax
F9	Fully Connected	-	4096	-	-	-	ReLU
F8	Fully Connected	-	4096	-	-	-	ReLU
C7	Convolution	256	13×13	3×3	1	SAME	ReLU
C6	Convolution	384	13×13	3×3	1	SAME	ReLU
C5	Convolution	384	13×13	3×3	1	SAME	ReLU
S4	Max Pooling	256	13×13	3×3	2	VALID	-
C3	Convolution	256	27×27	5×5	1	SAME	ReLU
S2	Max Pooling	96	27×27	3×3	2	VALID	-
C1	Convolution	96	55×55	11×11	4	SAME	ReLU
In	Input	3(RGB)	224×224	-	-	-	-

Table 2: AlexNet Architecture (Géron 2017)

Popular Convolutional NN Architectures

AlexNet

- Two regularization schemes adopted to reduce overfitting
  - Dropout (during training) applied in layers F8 and F9
  - Data augmentation: random shift offsets applied to training images; flipping horizontally and varying lighting conditions
- Competitive normalization - Local Response Normalization - applied after ReLU step of layers C1 and C3;
  - Strongly activating neurons inhibit neurons in similar location but in neighbouring feature maps;
  - Encourages maps to specialize and immprove generalization

Popular Convolutional NN Architectures

GoogLeNet

- Developed (2014) by Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich
- Won the 2014 ImageNet Large Scale Visual Recognition Competition(ILSVRC) (top-5 error rate less than 7%)
- Performance attributed to deeper architecture than previous CNNs
- Introduced the use of Inception Module
- Used fewer parameters than previous architectures (10 times fewer parameters than AlexNet; ~ 6 million instead of 60 million)

Popular Convolutional NN Architectures

GoogLeNet

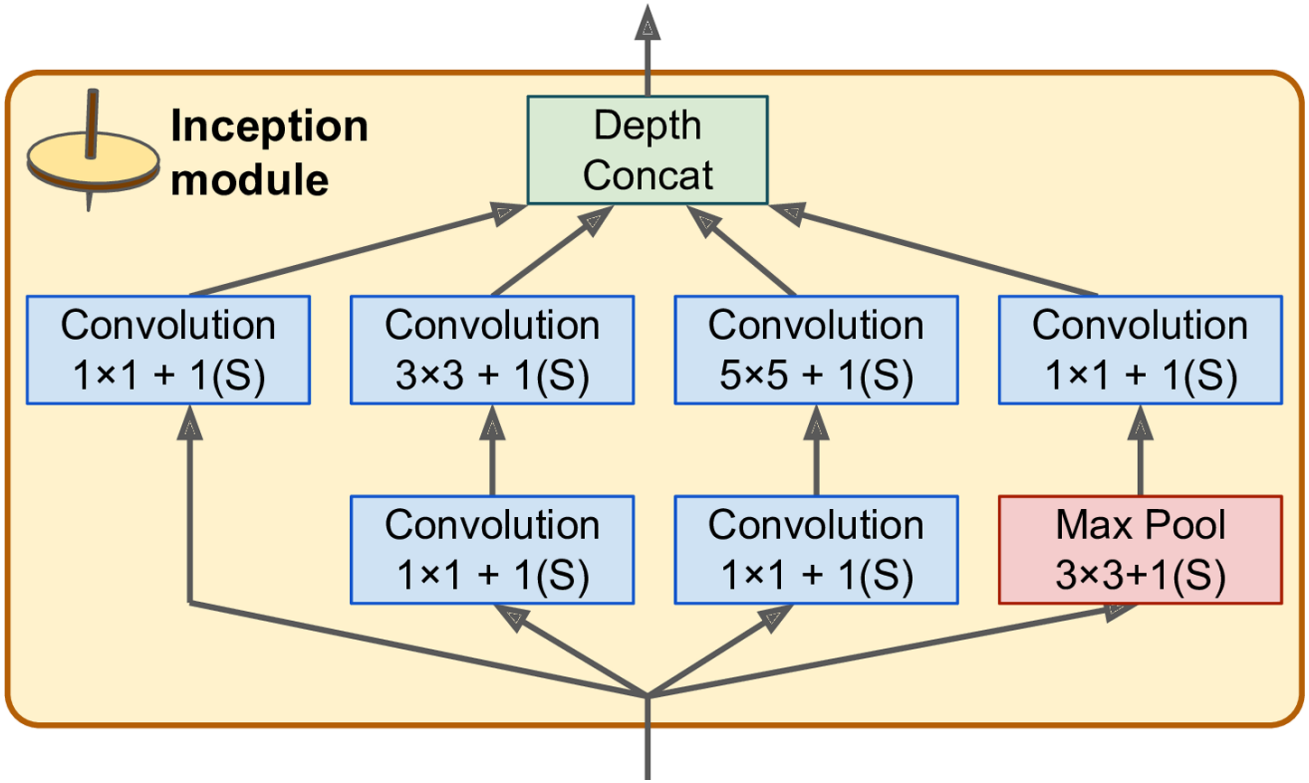

Figure 11: Inception module of GoogLeNet (Géron 2017)

Popular Convolutional NN Architectures

GoogLeNet

- In Figure 11, notation $3 \times 3 + 1(\text{S})$ implies 3×3 convolution with stride of 1 and "SAME" padding
- "SAME" padding means: add zero padding to input data if required to obtain number of output neurons equal to ROUND(# Input Neurons/Stride)
- All outputs of convolution layer have same height, width and depth and can be concatenated along depth axis
- 1×1 convolution is a way of collapsing the feaure maps

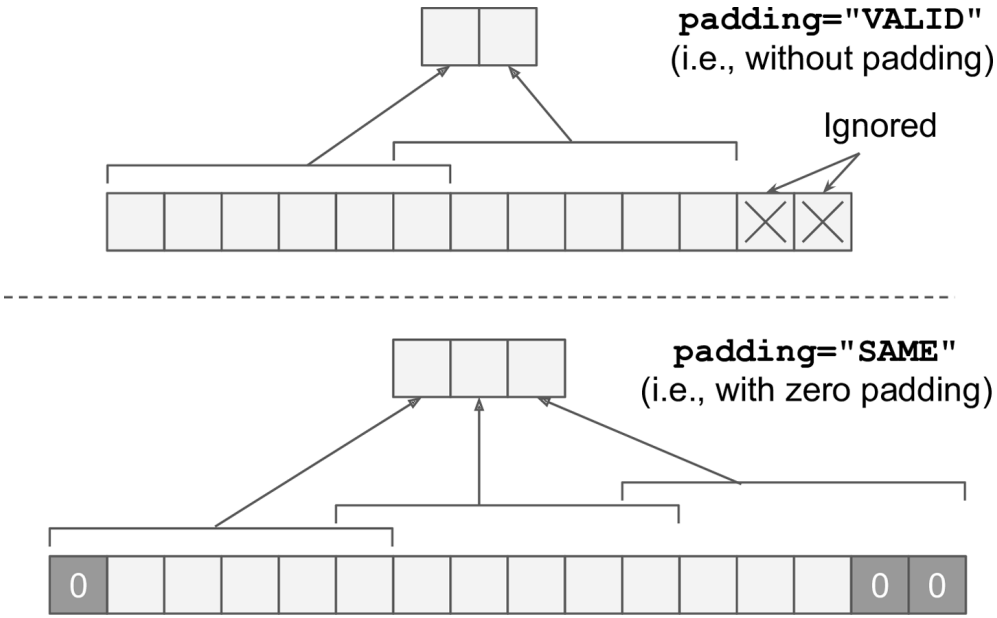

Figure 12: Padding options: input width=13; width=6; stride=5 (Géron 2017)

Popular Convolutional NN Architectures

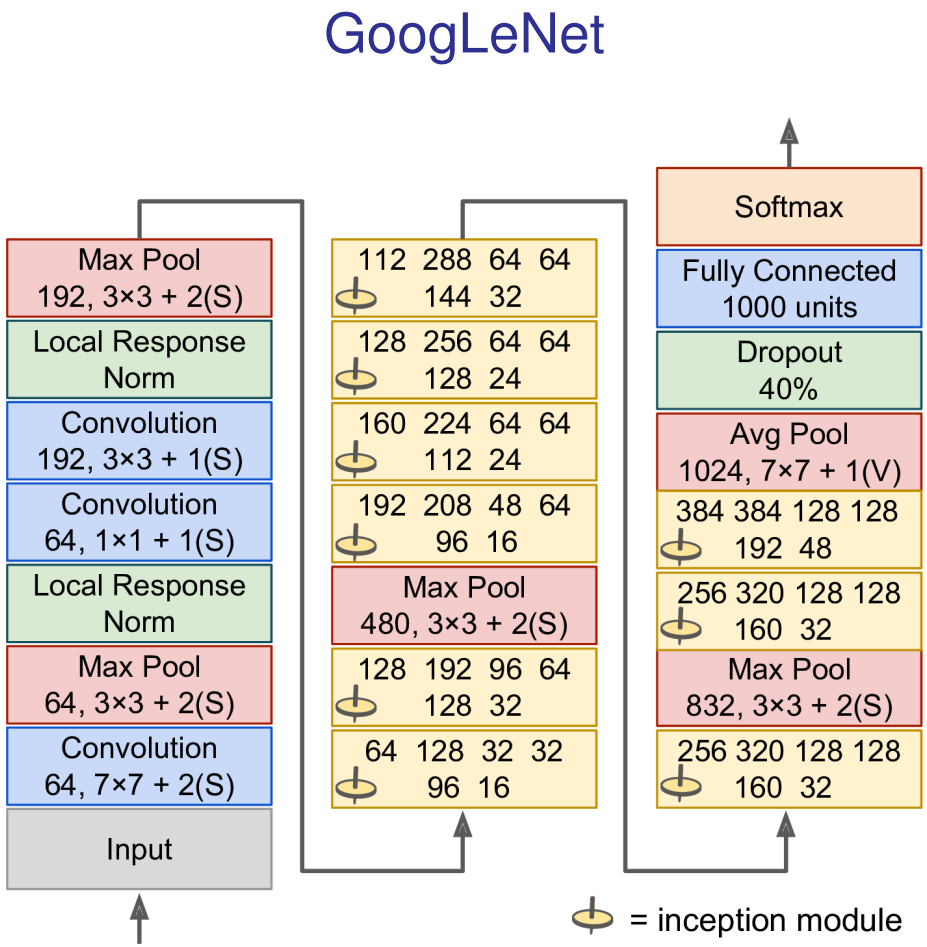

Figure 13: GoogLeNet Architecture (Géron 2017)

- Please refer to the GoogLeNet paper by Szegedy et al. (2014) for detailed description and the book by Auréien Géron (Géron 2017, pp. 373-376)

Popular Convolutional NN Architectures

ResNet

- Developed by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun (Microsoft Research) (Kaiming He 2015); Won 2015 ILSVRC
- Motivated by: *"increased network depth leads to saturated accuracy and then rapid degradation. ... degradation is not caused by overfitting, and adding more layers ... leads to higher training error"* - paraphrased from Kaiming He (2015)
- Degradation problem addressed by "Residual Learning Framework"
- Stacked layers fit a residual mapping ($\mathcal{F}(x)$) rather the desired underlying mapping, $\mathcal{H}(x)$:

$$\mathcal{F}(x) := \mathcal{H}(x) - x$$

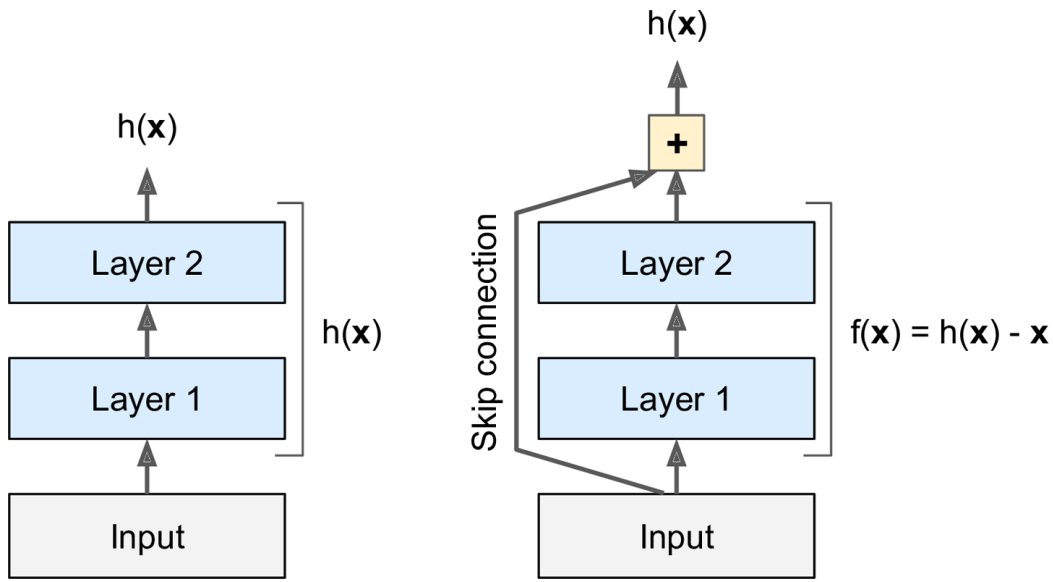

Figure 14: Residual learning framework(Géron 2017)

Popular Convolutional NN Architectures

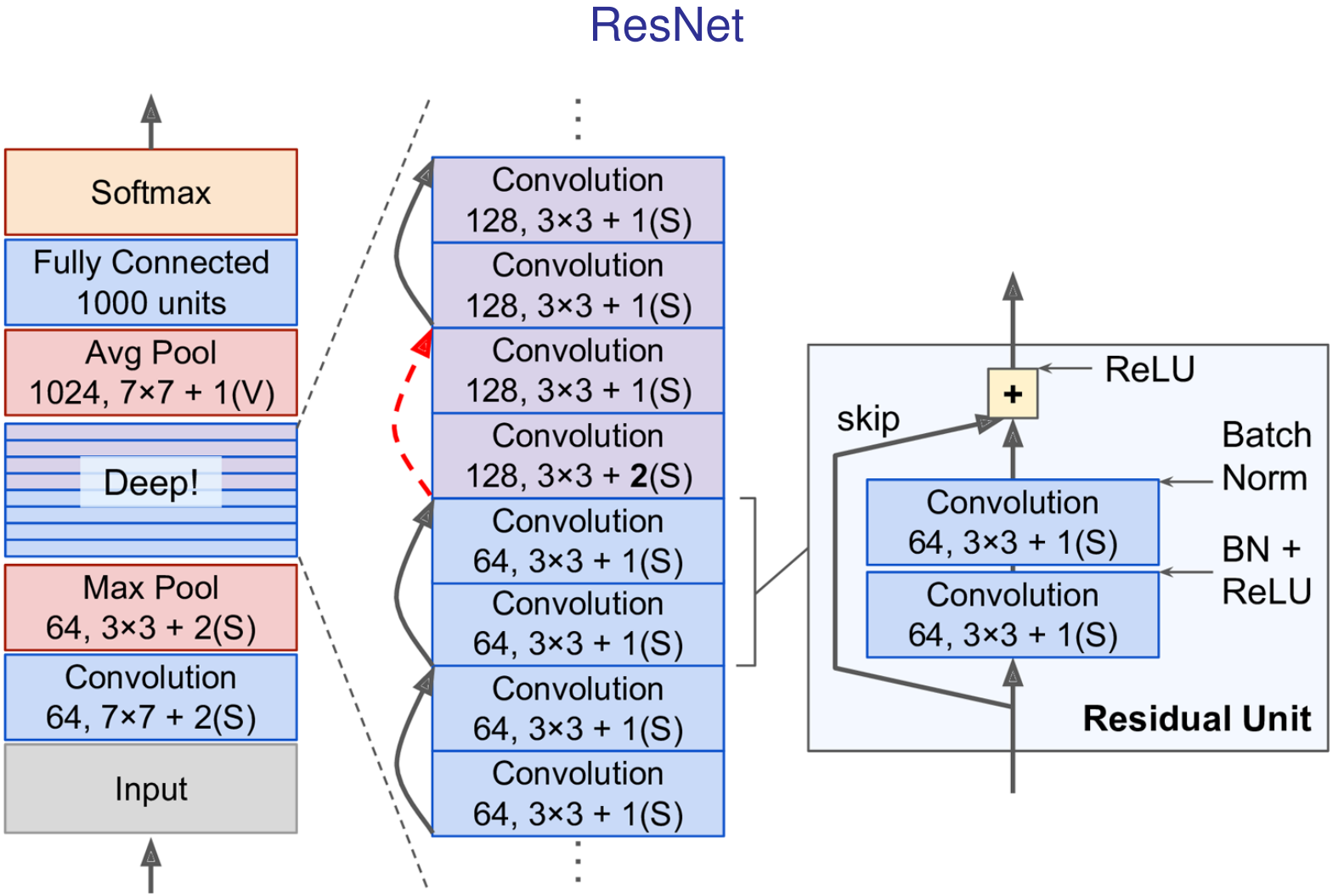

Figure 15: ResNet Architecture

Bibliography

Géron, A. (2017), *Hands-on Machine Learning with Scikit-Learn and TensorFlow*, O'Reilly Media, Inc., CA, USA.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press.

Haykin, S. (2009), *Neural Networks and Learning Machines*, Third edn, Pearson Education.

Kaiming He, Xiangyu Zhang, S. R. J. S. (2015), Deep residual learning for image recognition, Technical Report arXiv:1512.03385 [cs.CV], ArXiV.
URL: *https://arxiv.org/pdf/1512.03385.pdf*

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2014), Going deeper with convolutions, Technical Report arXiv:1409.4842 [cs.CV], ArXiV.
URL: *https://arxiv.org/pdf/1409.4842.pdf*