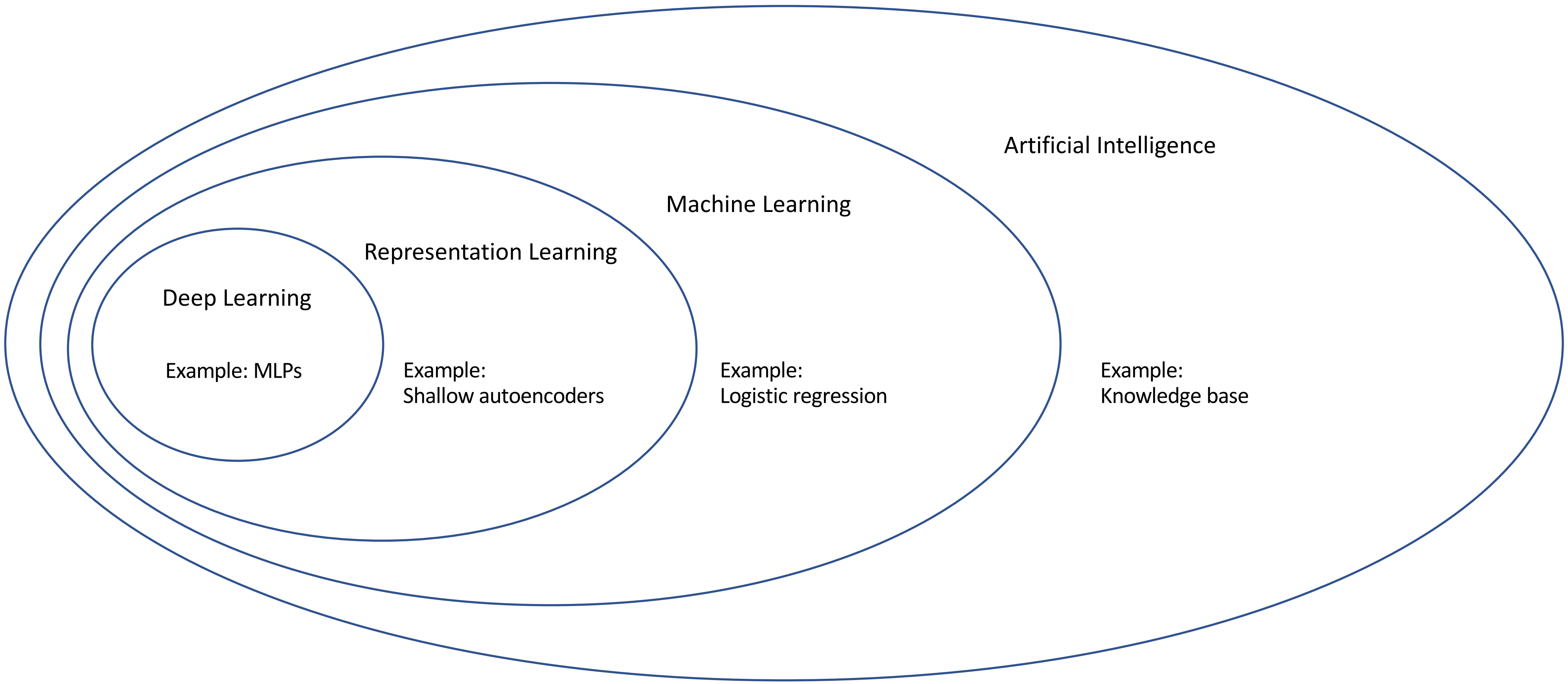
Deep Learning Evolution

Elementary Principles and Architectures

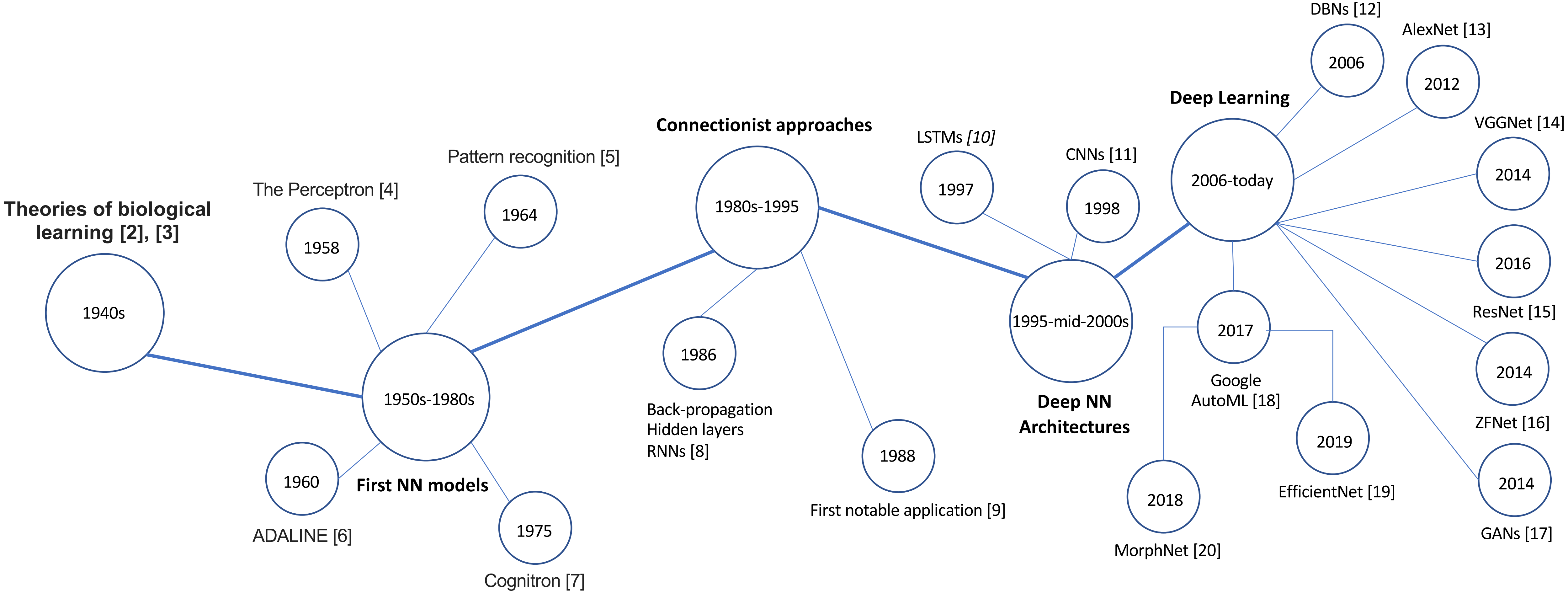
Evgeny Mitichkin (enote, Germany)

Introduction: AI, ML an DL



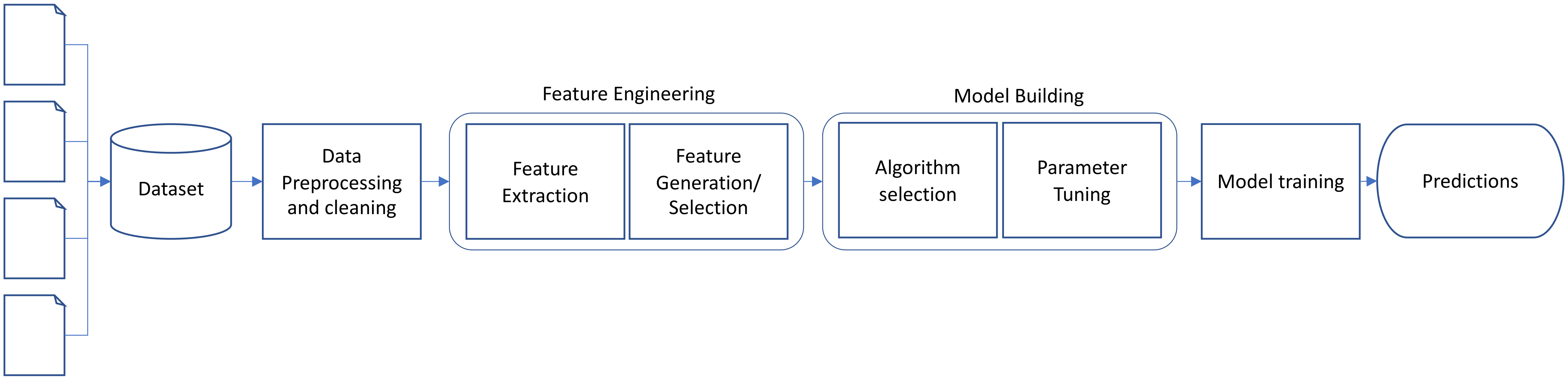
Source: Deep Learning [1]

Introduction: History of Deep Learning



Based on: Deep Learning [1]

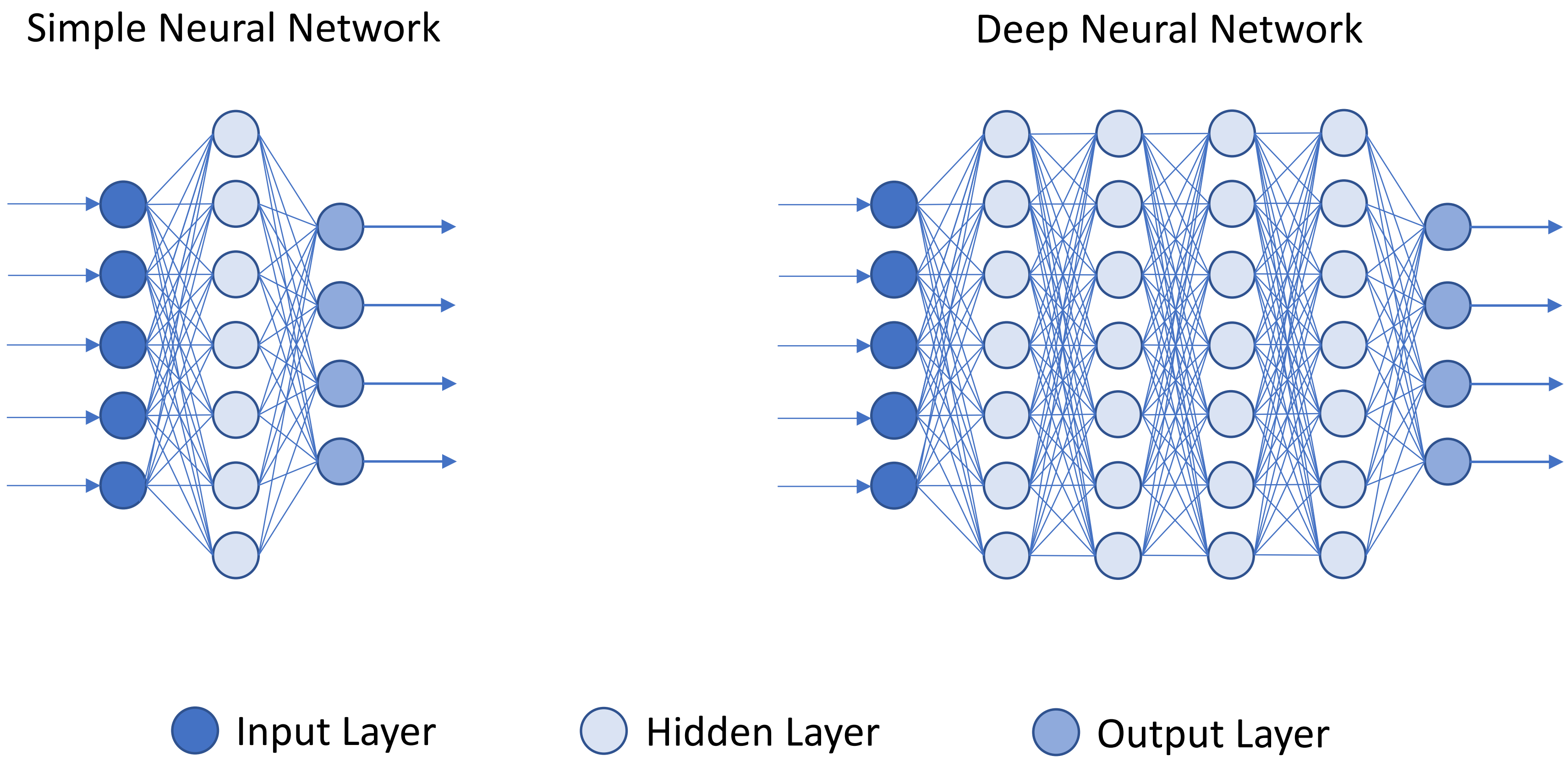
Introduction: Machine Learning - Concept



Based on:

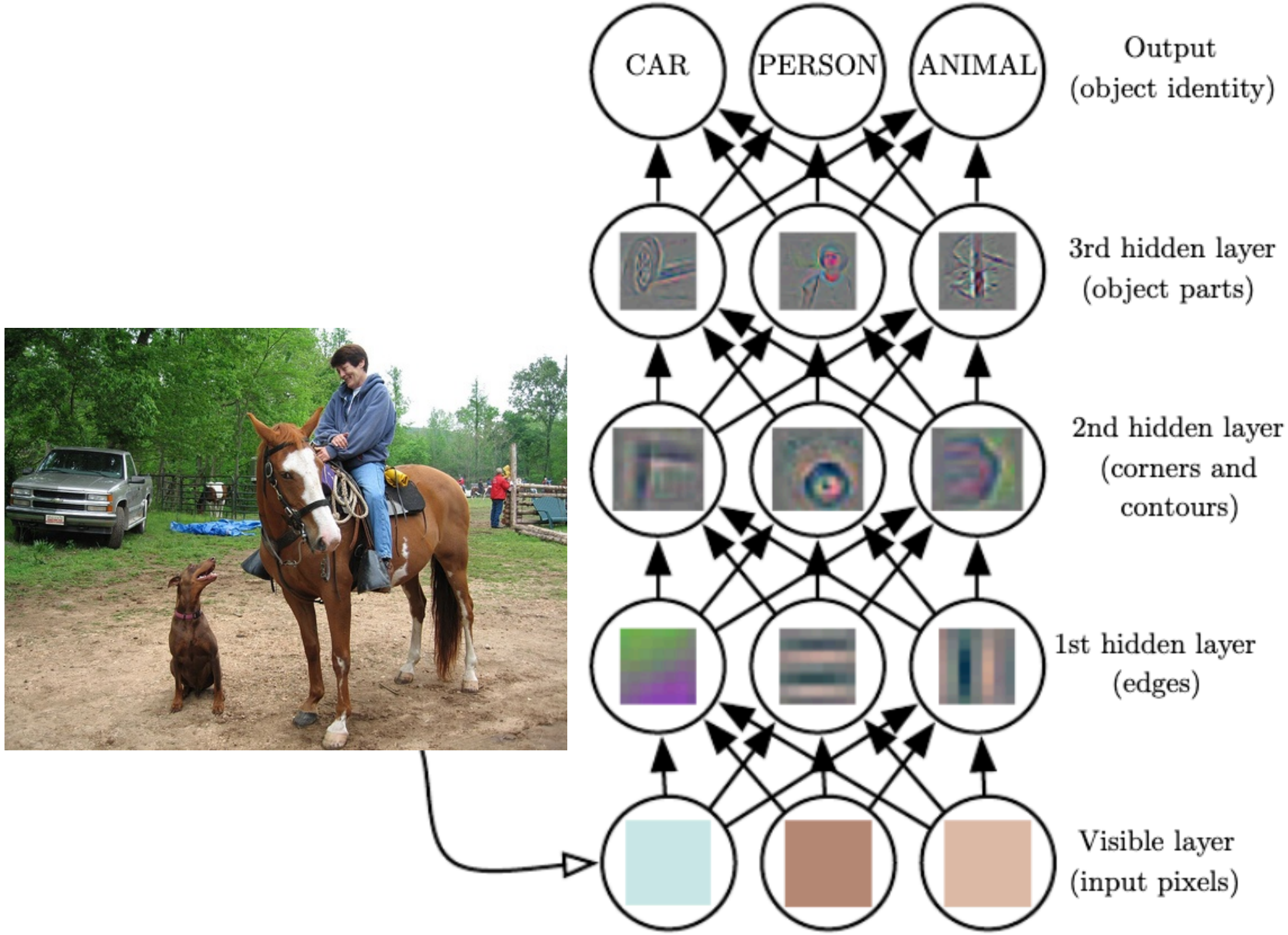
- Benchmark and Survey of Automated Machine Learning Frameworks [22]
- Automated Machine Learning: State-of-The-Art and Open Challenges [23]

Introduction: Deep Learning - Concept



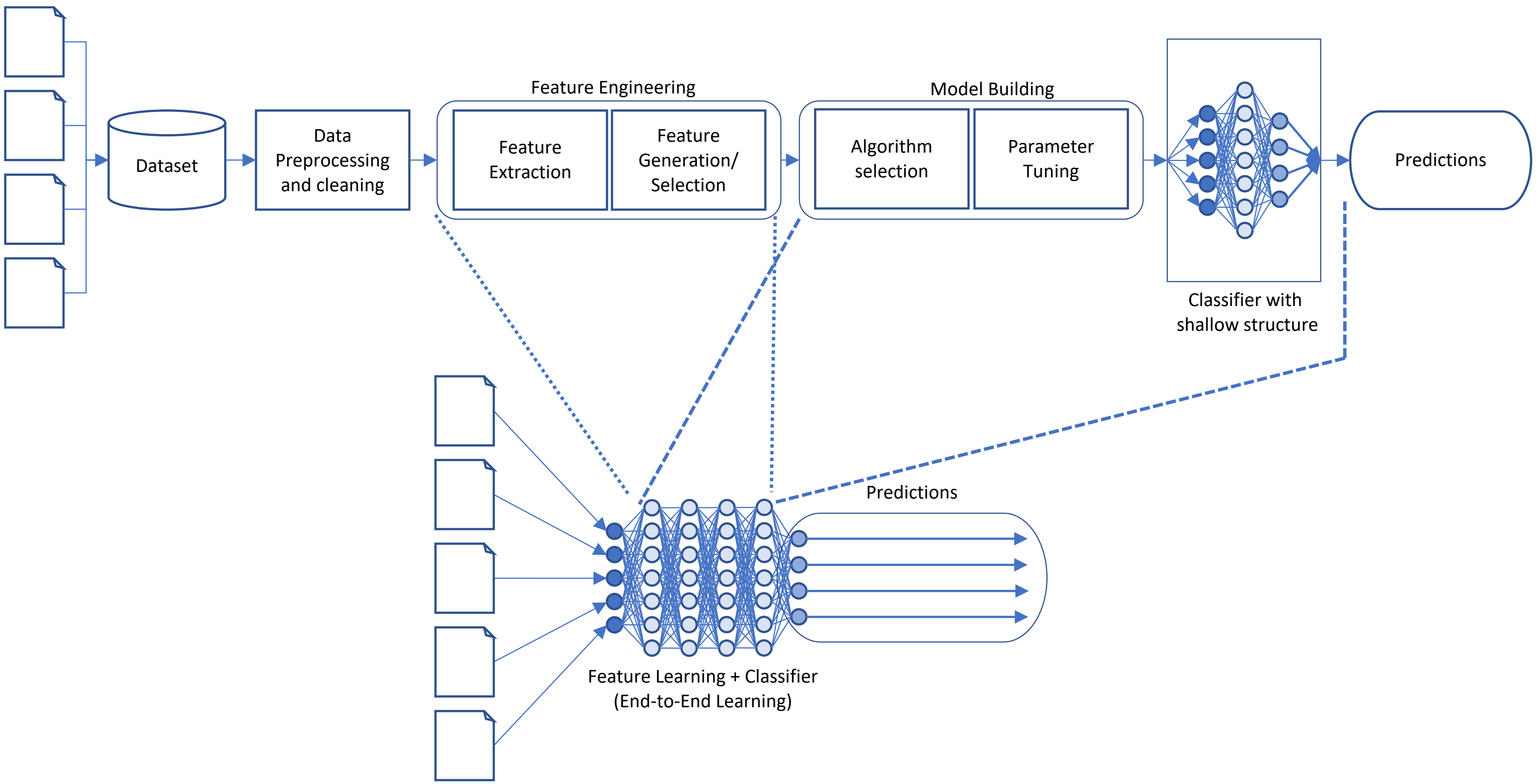
Based on: [27]

Introduction: Deep Learning - Concept



Sources: [1, 34]

Introduction: ML vs DL - Differences



Based on: [32]

Introduction: ML vs DL - Differences

- Principal differences in approaches
- Often different initial requirements and issues occurring on the way

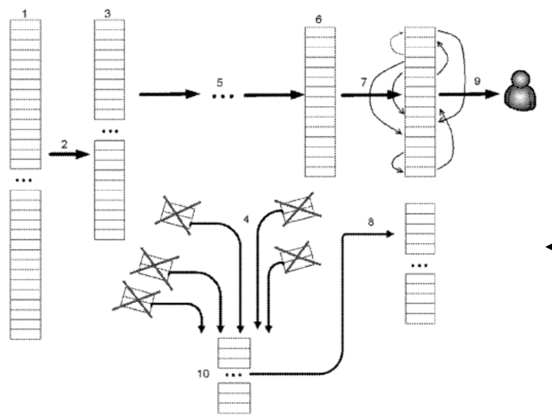
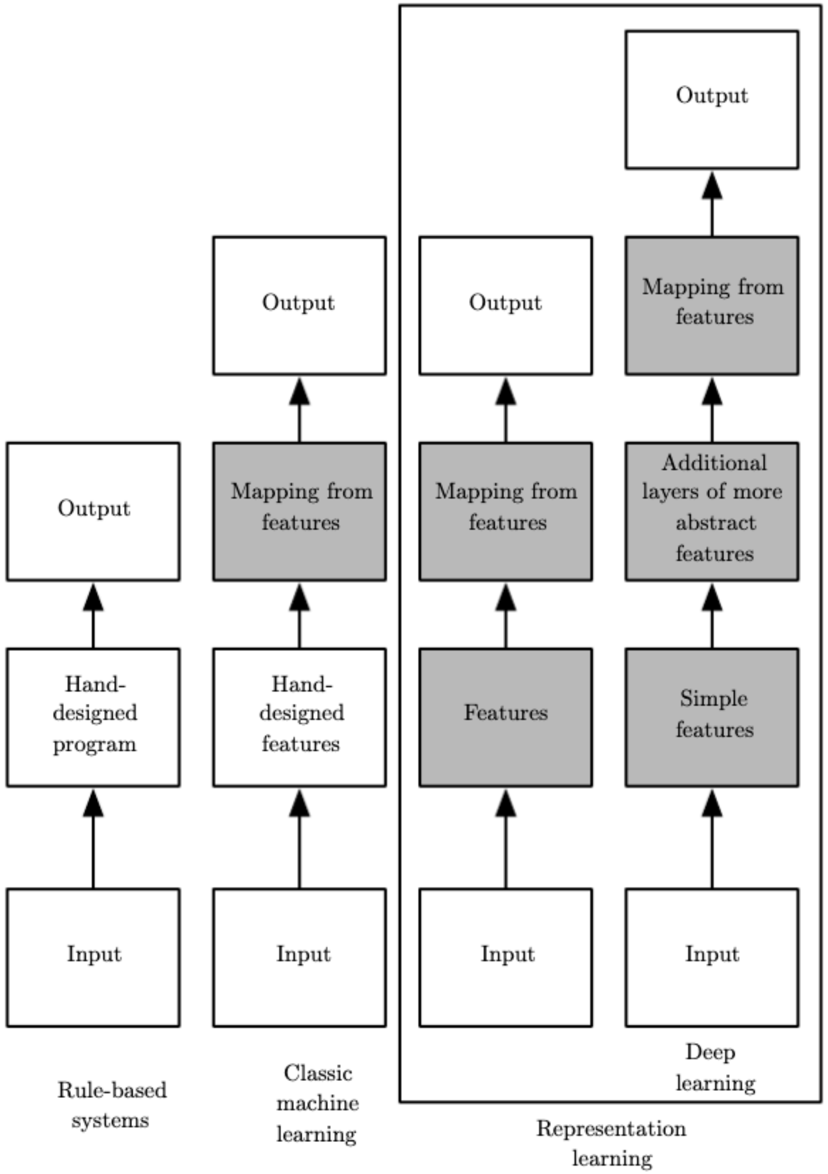


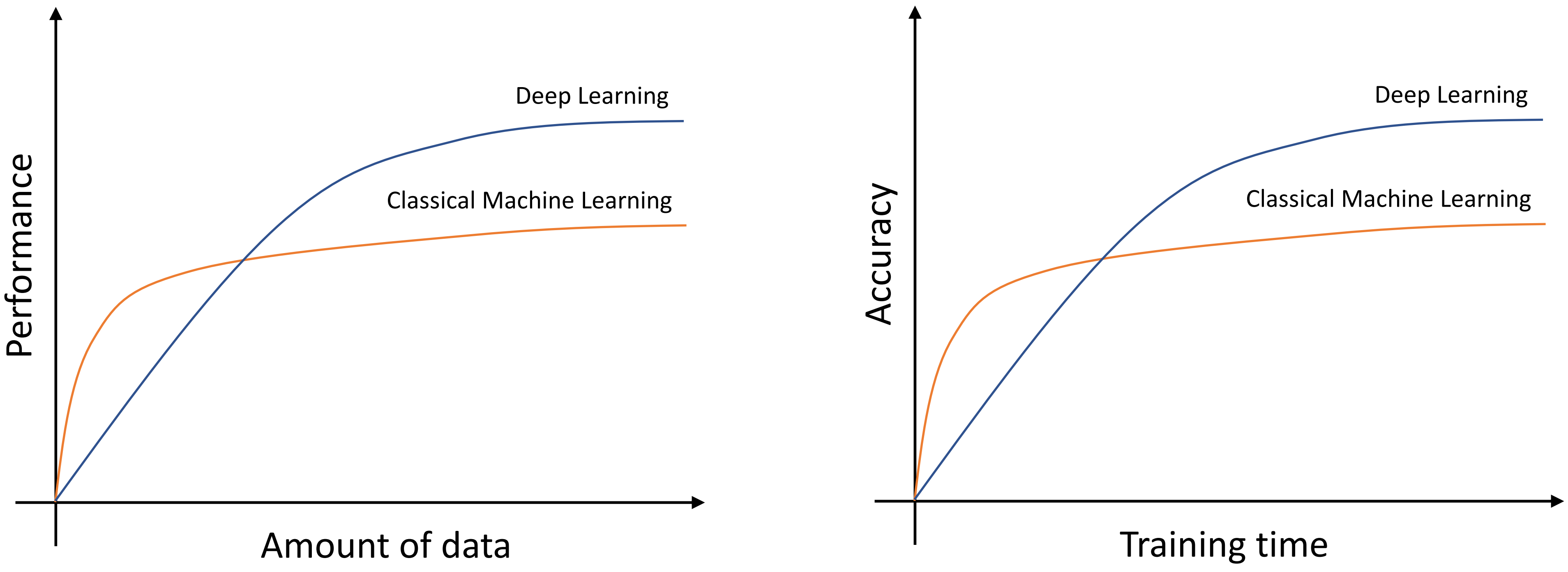
FIG 1

Source: [35]



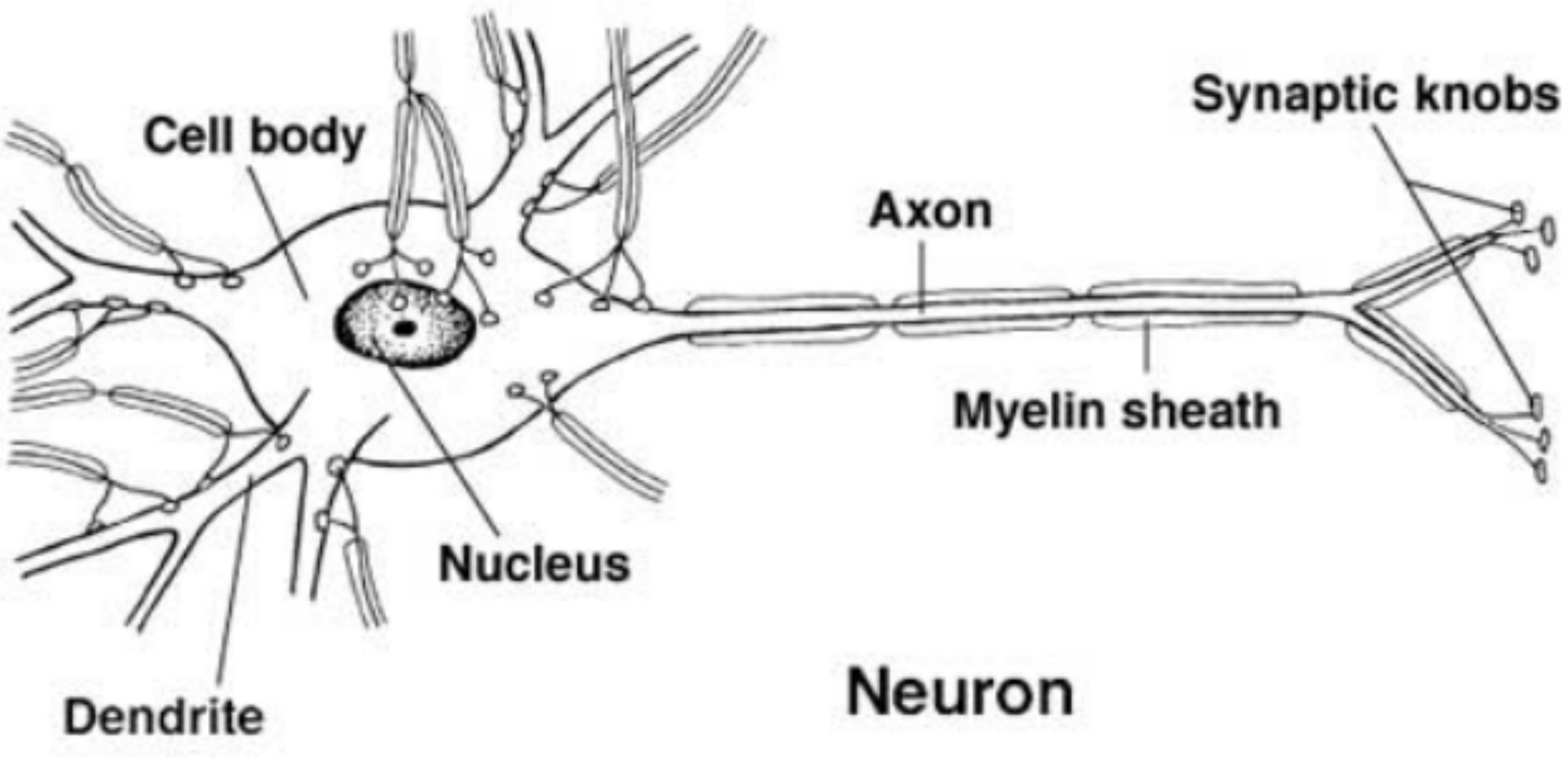
Source: [1]

Introduction: ML vs DL - Differences



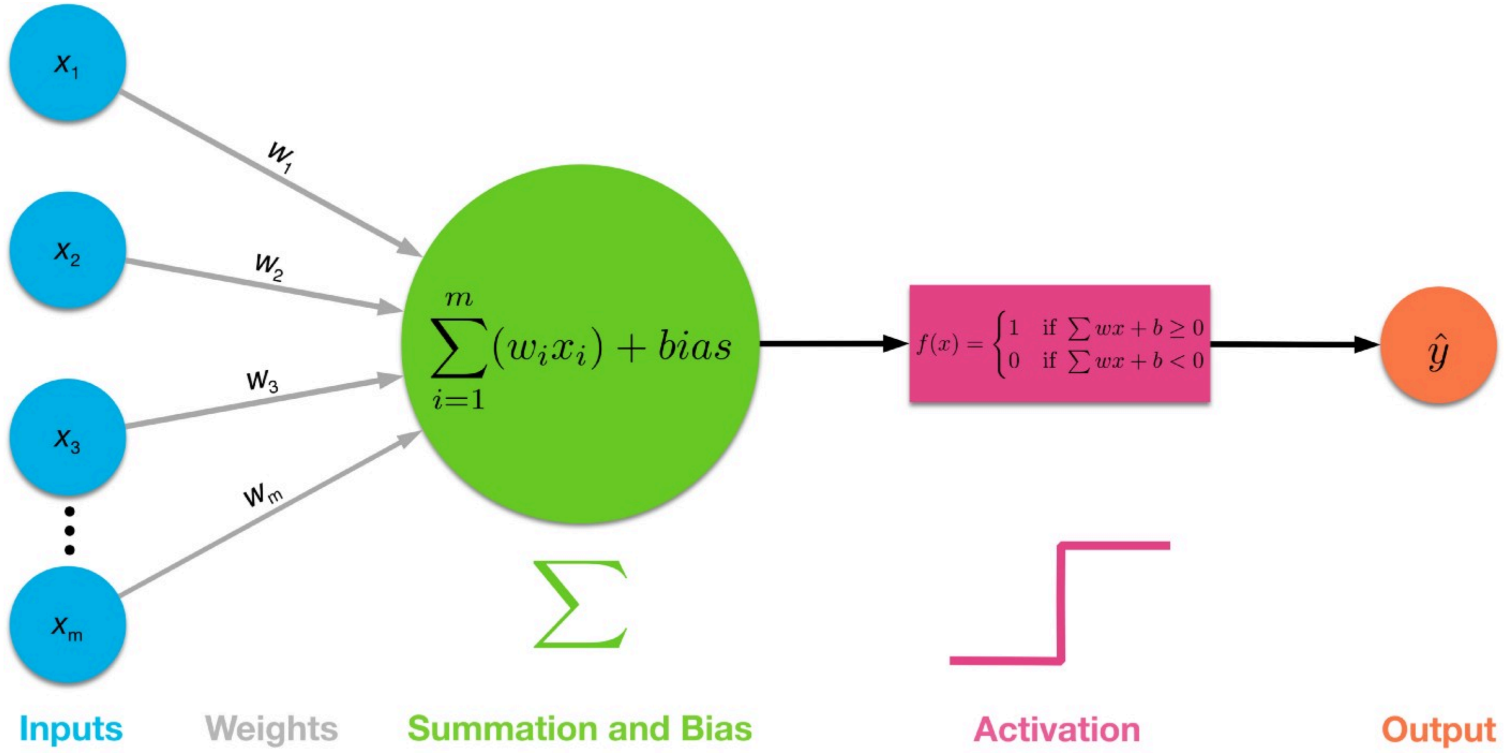
Source: [31]

Introduction: ML vs DL – Recap



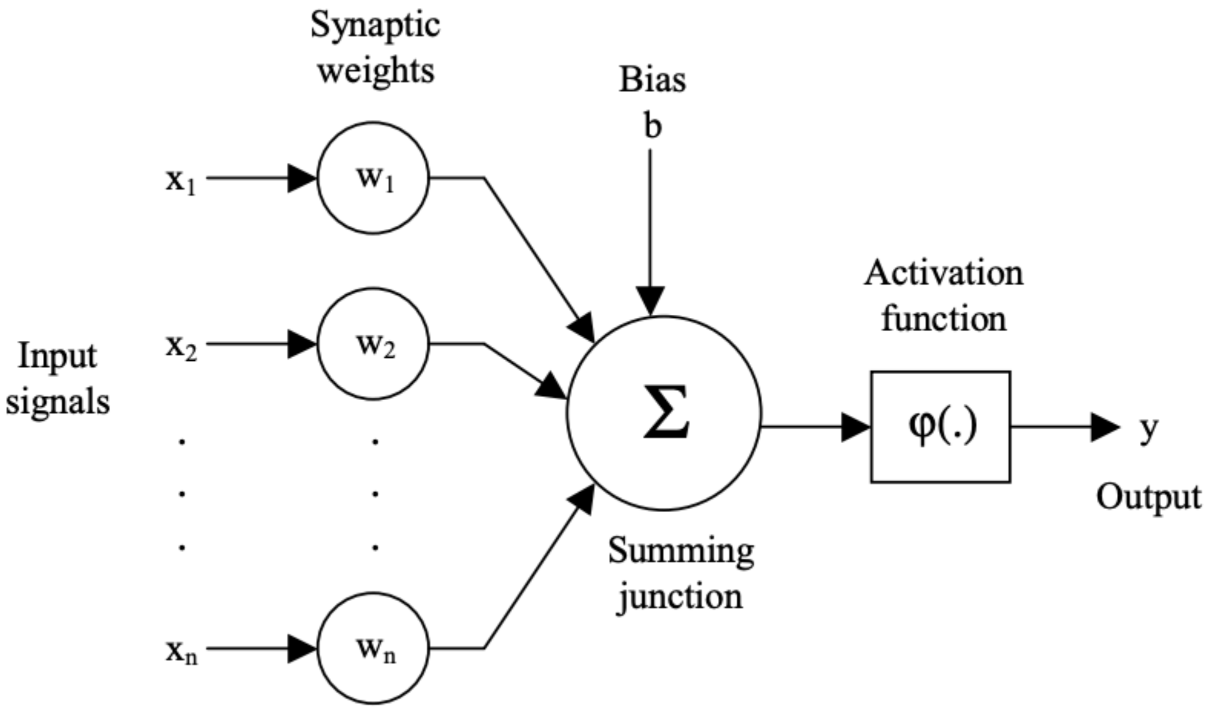
Source: [50]

Deep Learning: Architectures - Recap



Source: [51]

Deep Learning: Architectures - Recap



$$y = \varphi(\sum_{j=1}^{n} w_j x_j + b)$$

- $x_1,\ldots x_n$ - input signals
- $w_1,\ldots,w_n$ - synaptic weights
- φ - activation function to limit the amplitude of the neuron output
- b – external *bias*

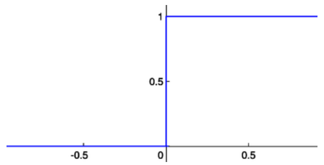
Source: [50]

Deep Learning: Architectures - Recap

Activation functions

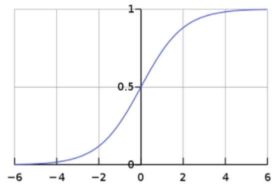
Binary step function

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \ge 0 \end{cases}$$



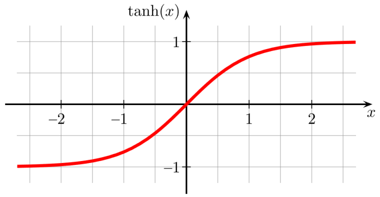
Sigmoid

$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$$



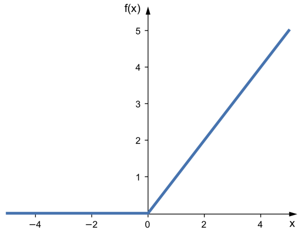
Hyperbolic tangent (Tahn)

$$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$



Rectified Linear Unit (ReLU) And its variants

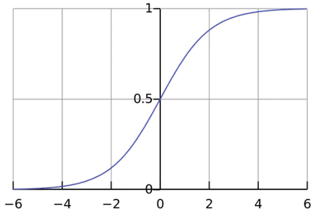
$$f(x) = \begin{cases} 0 & \text{for } x \le 0 \\ x & \text{for } x > 0 \end{cases}$$



Softmax

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

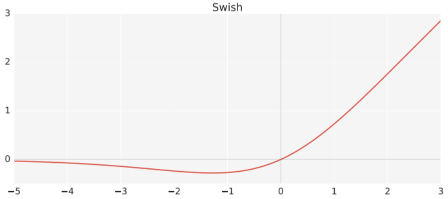
- Values are always in the range [0,1]
- All values add up to 1



Self-Gated activation function [42]

$$f(x) = x \cdot \sigma(x)$$

where $\sigma(x) = (1 + \exp(-x))^{-1}$ is the sigmoid function



Deep Learning: Architectures - Recap

Loss (cost) functions:

- MSE
- Cross-entropy
- Cosine similarity

Layers:

- Dense (Fully connected)
- Softmax
- Convolutional

Optimization algorithm + Backpropagation

- Gradient descent (SGD)
- Momentum method
- RMSProp (Hinton)
- Adagrad [56]

Introduction: ML vs DL – Practical Example

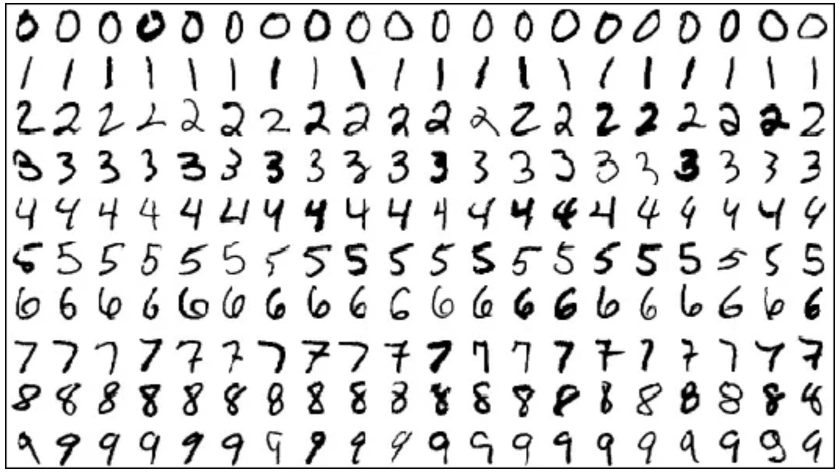
- Problem: handwritten digits recognition – USPS dataset [41]

Classifier	Train set	Test err	Reference
Nearest-neighbor	$USPS^+$	5.9%	(Simard et al., 1993)
LeNet1	$USPS^+$	5.0%	(LeCun et al., 1989)
Optimal margin classifier	USPS	4.6%	(Boser et al., 1992)
SVM	USPS	4.0%	(Schölkopf et al., 1995)
Linear Hyperplane on KPCA features	USPS	4.0%	(Schölkopf et al., 1998b)
Local learning	$USPS^+$	3.3%	(Bottou and Vapnik, 1992)
Virtual SVM	USPS	3.2%	(Schölkopf et al., 1996)
Virtual SVM, local kernel	USPS	3.0%	(Schölkopf, 1997)
Boosted neural nets	$USPS^+$	2.6%	(Drucker et al., 1993)
Tangent distance	$USPS^+$	2.6%	(Simard et al., 1993)
Human error rate	—	2.5%	(Bromley and Säckinger, 1991)

Source: [40]

Introduction: ML vs DL – Practical Example

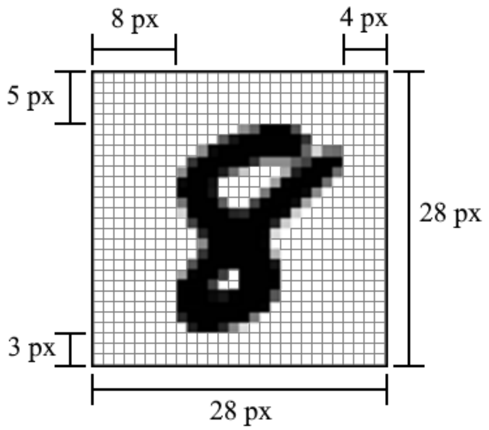
- Problem: handwritten digits recognition – MNIST database [29]



- One of the best k-NN results: accuracy 97.73% [28]
- SVMs: accuracy between 98.6% and 99.44% [29]
- Deep Learning: accuracy 99.84% [30]

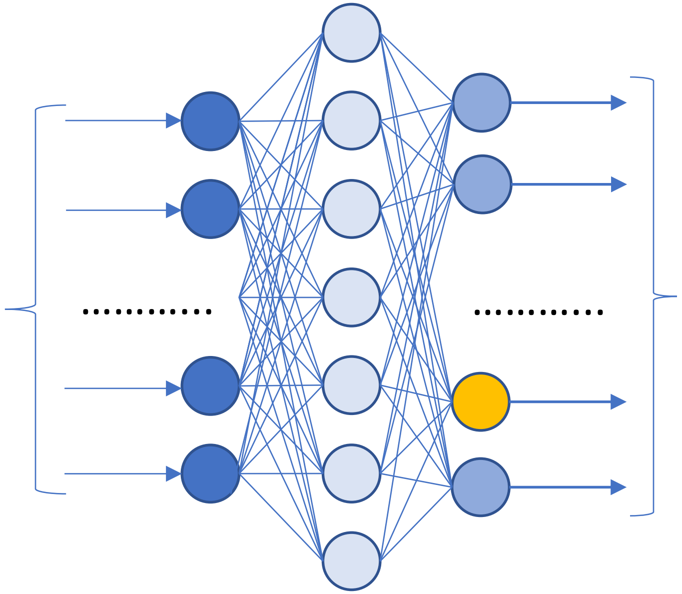
Introduction: MNIST – DL Classification

- Each image contains 28x28 = 784 pixels
- 0 – white, 255 - black



784 neurons as input

Hidden layer (n = 16 neurons)



10 neurons as output

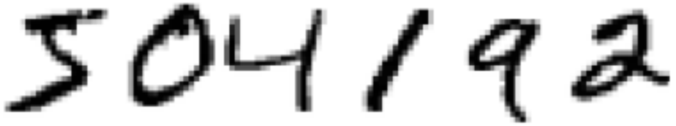
y(x) = (0, 0, 0, 0, 0, 0, 0, 0, 1, 0) T

3D representation available: https://www.youtube.com/watch?v=3JQ3hYko51Y

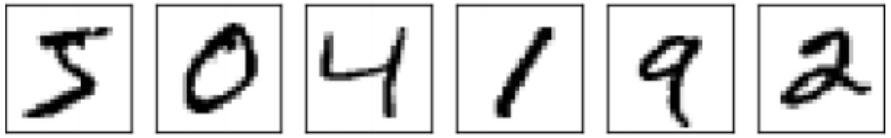
Sources: [43, 44]

Introduction: MNIST – Deep Learning Approach

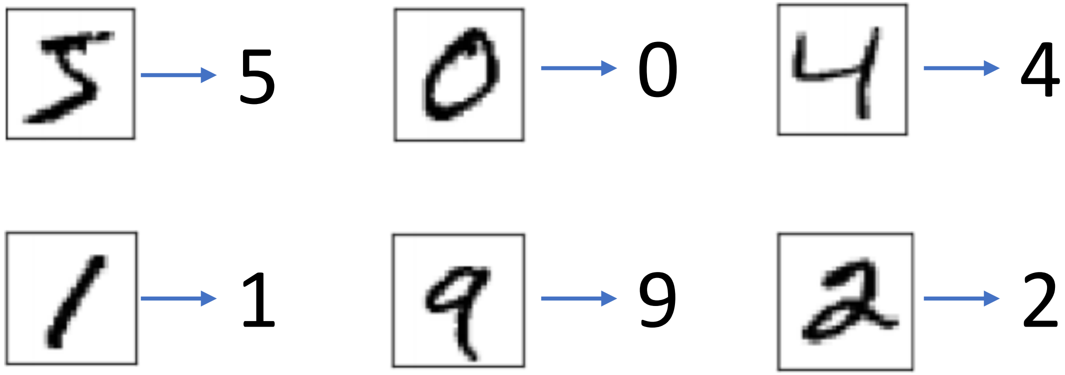
Input:



Segmentation 28x28 greyscale:



Classification:



Source: [43]

Introduction: MNIST – Idea of multiple layers

Extracting local features and combining them
to form higher order features

Forcing hidden units to combine only local sources of
information.

Distinctive features can appear in multiple locations.
Approximate position must be preserved to allow the
next levels to detect higher order, more complex features

Source: [9]

Deep Learning: Universal Approximation Theorem (Cybenko)

"An arbitrary continuous function, defined on [0,1] can be arbitrary well uniformly approximated by a multilayer feed-forward neural network with one hidden layer (that contains only finite number of neurons) using neurons with arbitrary activation functions in the hidden layer and a linear neuron in the output layer."

Sources: [52], [50]

"A feedforward network with a single layer is sufficient to represent any function, but the layer may be infeasibly large and may fail to learn and generalize correctly"

I. Goodfellow [1]

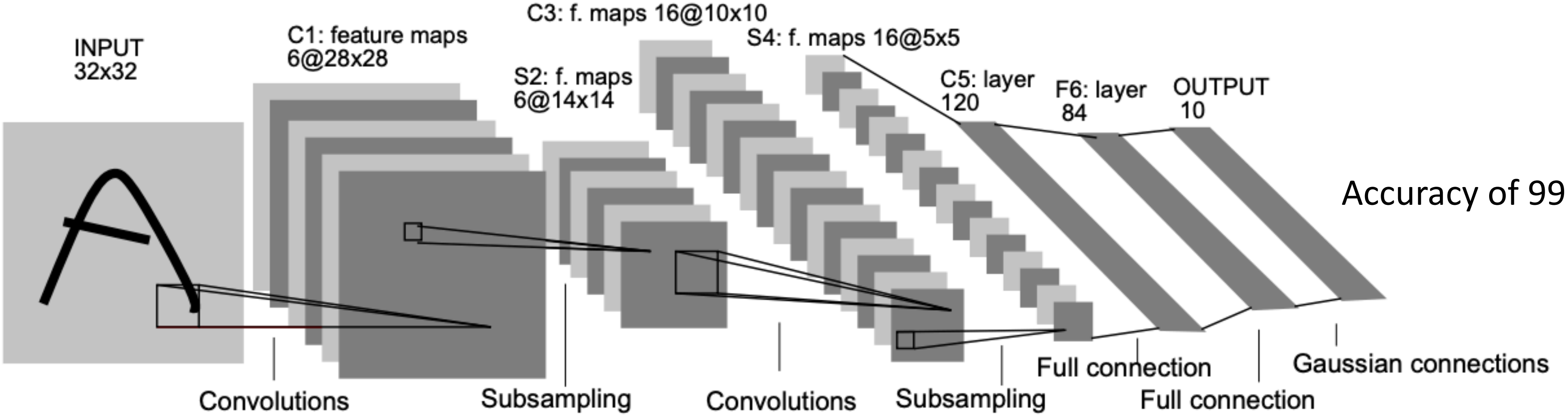
More details, assumption and proofs can be found in [12], [52], [53], [54], [55], [1]

Deep Learning: Using multiple hidden layers

Num Hidden Layers	Result
none	Only capable of representing linear separable functions or decisions.
1	Can approximate any function that contains a continuous mapping from one finite space to another.
2	Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.
>2	Additional layers can learn complex representations (sort of automatic feature engineering) for layer layers.

Source: [50]

Deep Learning: Architectures – CNN – LeNet-5

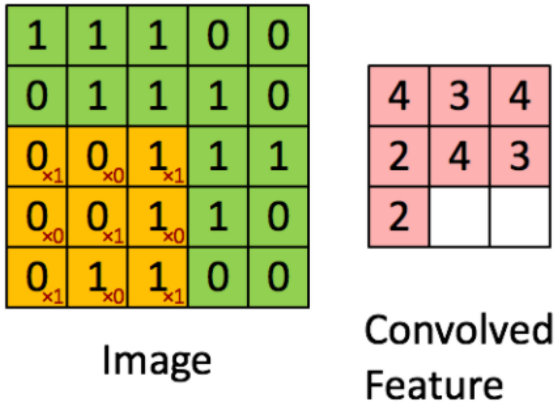


Accuracy of 99.2% on MNIST *[11]*

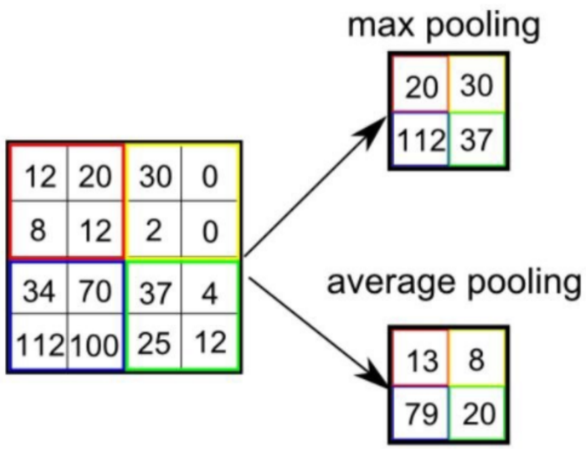
Architecture of LeNet-5 applied to digits recognition problem. Source: [24]

Deep Learning: Architectures – CNN – LeNet-5

Convolution



Pooling



Fully Connected

- Learning non-linear combinations of the high-level features as represented by the output of the convolutional layer
- Approximating a continuous non-linear function

Gaussian connections or Softmax

- Normalizing predictions
- Defining a loss

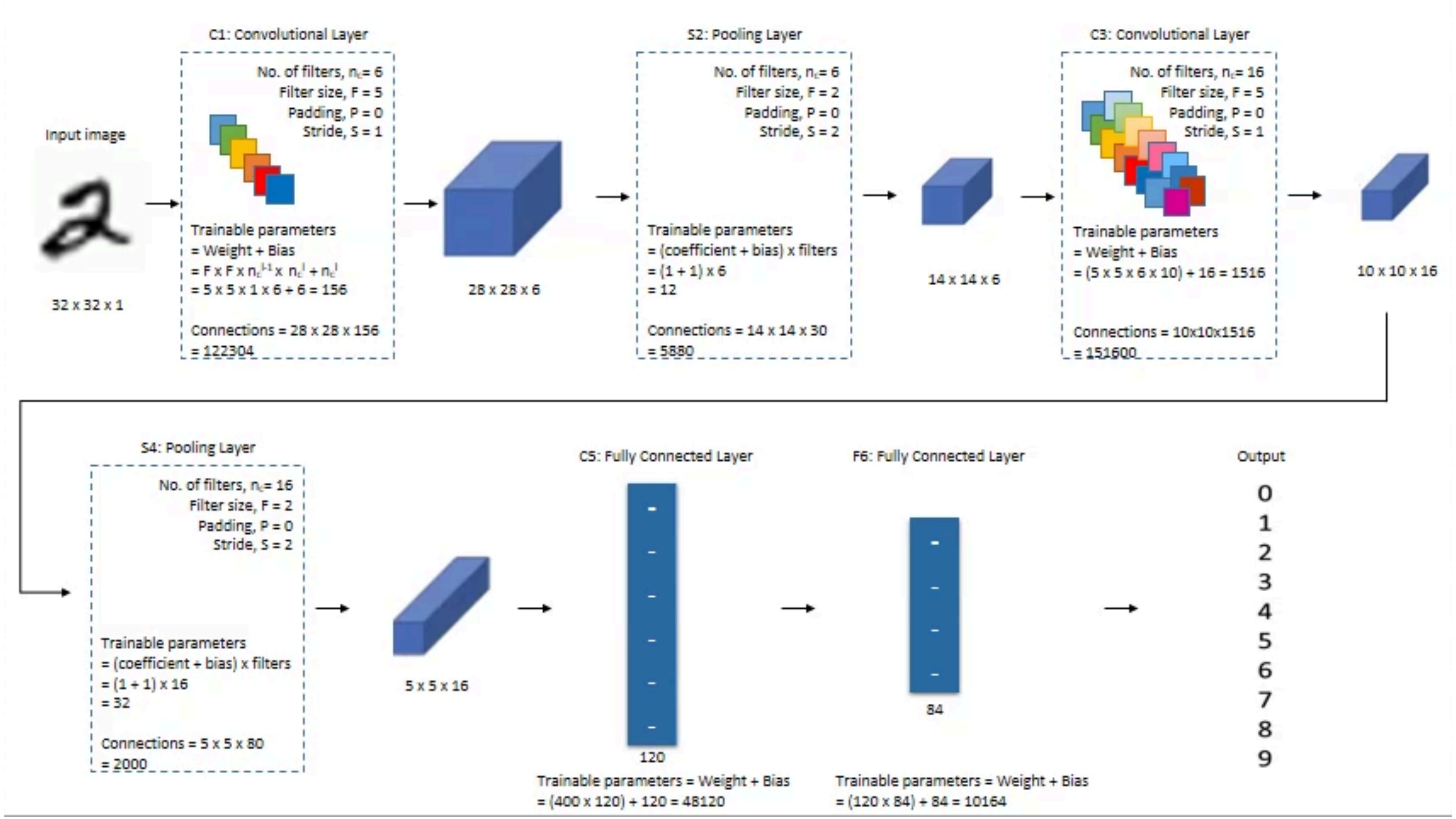
Source: [57]

Deep Learning: Architectures – CNN – LeNet-5

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

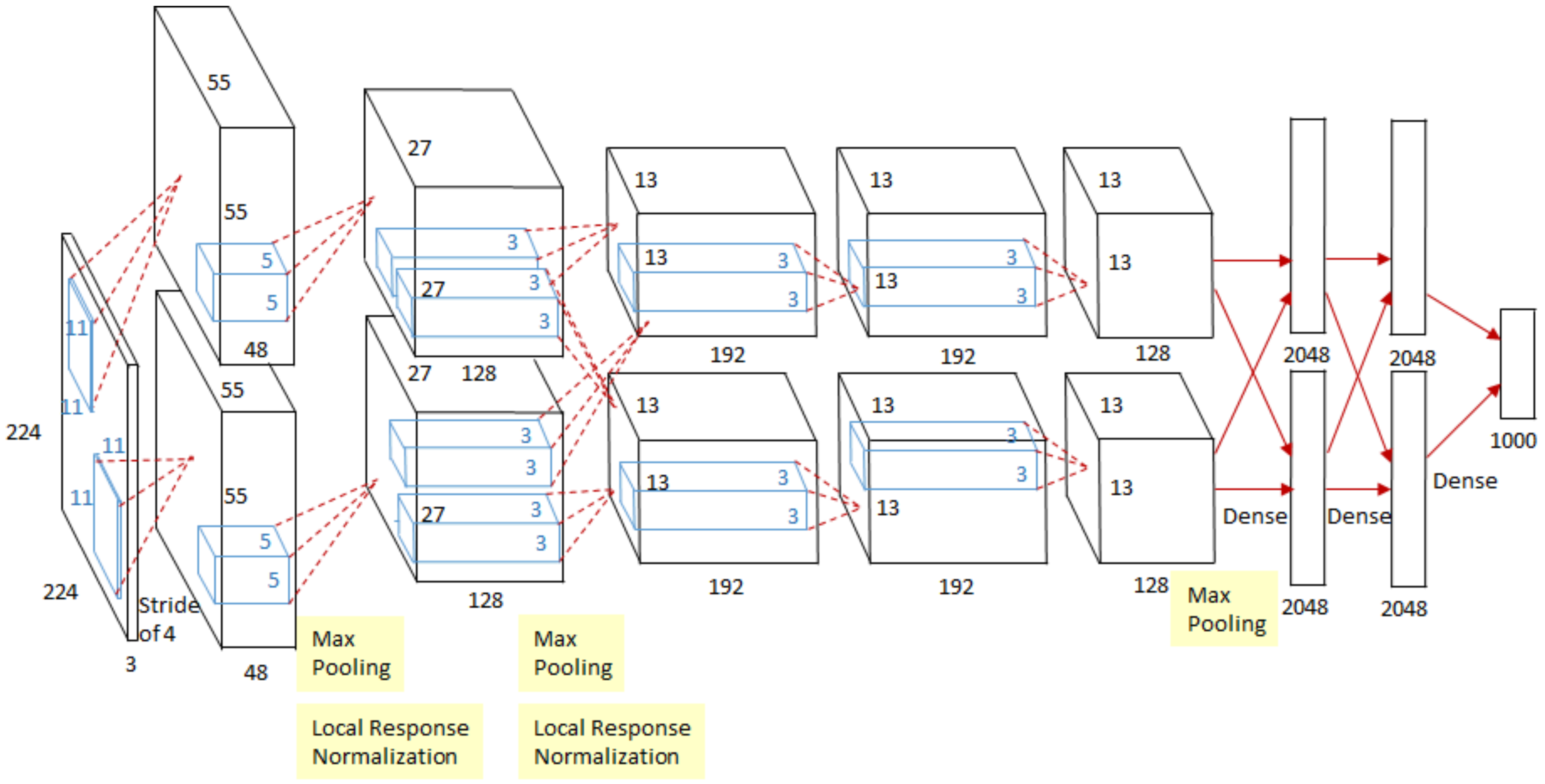
LeNet-5 Summarized Architecture. Source: [45]

Deep Learning: Architectures – CNN – LeNet-5



LeNet-5 Summarized Architecture. Source: [45]

Deep Learning: Architectures – CNN - AlexNet



AlexNet Architecture. Source: [13]

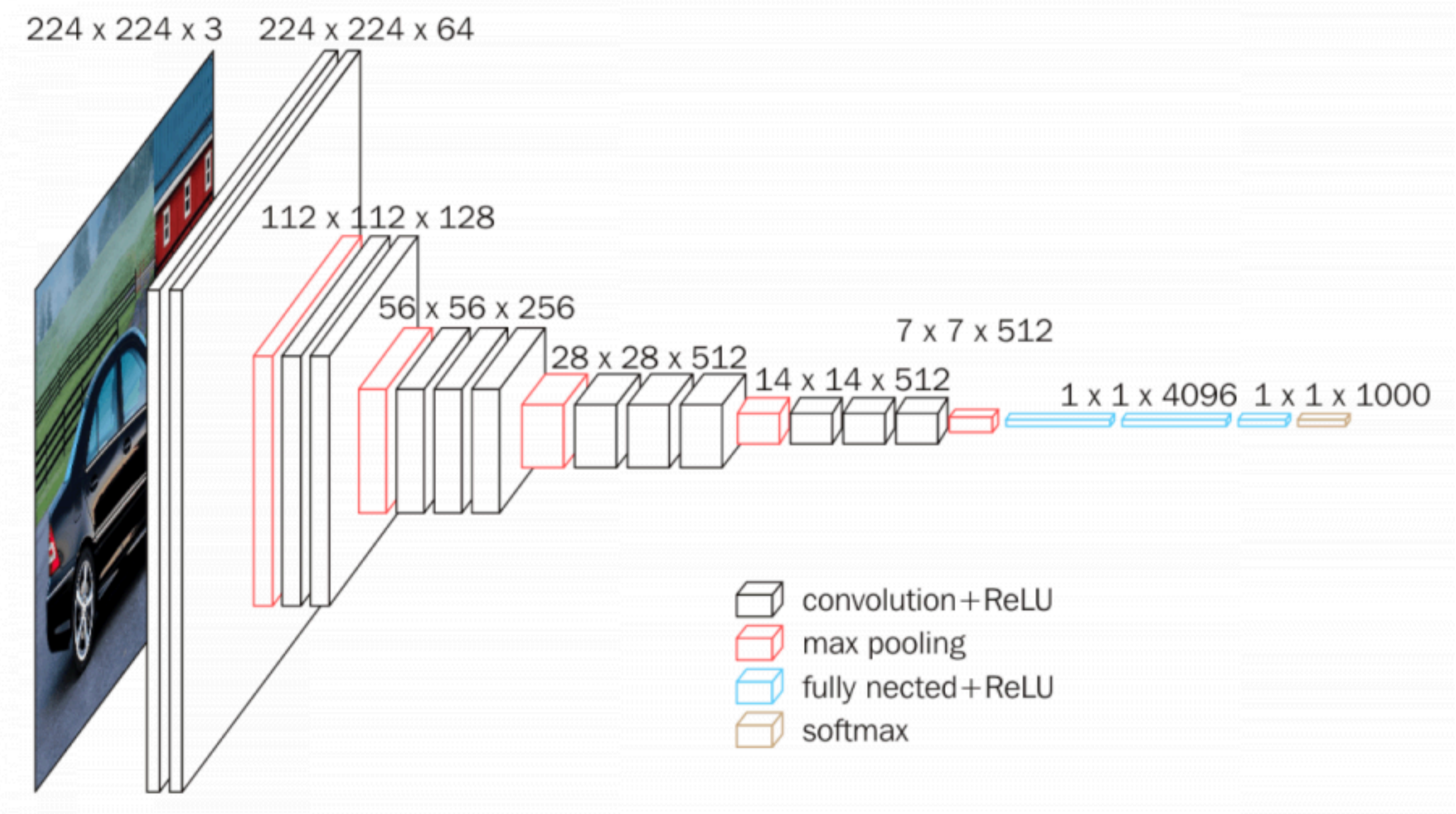
Deep Learning: Architectures – CNN - AlexNet

Some facts:

- 62.3 million parameters
- 1.1 billion computation units in a forward pass
- Uses ReLU instead of Tanh
  - Fixes vanishing gradient
  - 6-times training speed boost
  - Same accuracy
- Uses Dropout
- Overlap pooling to reduce the size of network

AlexNet Architecture. Source: [13]

Deep Learning: Architectures – CNN - VGGNet



VGGNet Architecture. Source: [14, 46]

Deep Learning: Architectures – R-CNN

- R-CNN – regions with CNN Features [47]
  - Solves the variable out length problem for object detection
  - Extremely time-demanding
  - Very slow inference
  - Fixed selective search algorithm
- Fast-R-CNN [15]:
  - MUCH faster inference (up to 20 times compared to R-CNN)
- Faster R-CNN
  - Further dramatic speed improvement (up to 250 times compared to R-CNN)
- Cascade R-CNN [48]

Deep Learning: Architectures

- RNN
  - Generating Image Descriptions with Multimodal Recurrent Neural Network [49]
- LSTM:
  - The foundations laid by Hochreiter and Schmidhuber [10]
- Boltzmann machine
- Autoencoders
- GANs [17]
- DBNs [12]
- AutoML [18], [19], [20]

Deep Learning Tasks

Notable practical applications

- Object Detection
- Semantic Segmentation
- Data Classification
- Data Generation

A comprehensive collection of state-of-the-art drafts/papers:

https://paperswithcode.com/sota

Deep Learning Today

Notable practical applications

- Autonomous driving
- Machine translation (NLP)
- Healthcare applications
- Fraud detection
- Search and recommender systems
- Various computer vision and audio-related challenges

Deep Learning in Healthcare

- Microscopy Analysis
- Object detection for medical images
- Tissue segmentation
- MRI segmentation
- Image registration and Medical Image Synthesis
- Drug Design

Sources: [36, 37]

Deep Learning in Fraud Detection

- Traffic monitoring
- Credit card fraud detection
- Cyber-Network Intrusion Detection
- IoT behavior tracking
- Traffic analysis and fraud detection in telecom

Sources: [38, 39]

References

1. Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. MIT press.
2. Hebb, D.O., 1949. The organization of behavior: a neuropsychological theory. J. Wiley; Chapman & Hall.
3. McCulloch, W.S. and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), pp.115-133.
4. Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), p.386.
5. Aizerman, M. A. and Braverman, E. M. and Lev I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control, 25:821–837, 1964..
6. Widrow, B., 1960. Adaptive" adaline" Neuron Using Chemical" memistors.".
7. Fukushima, K., 1975. Cognitron: A self-organizing multilayered neural network. Biological cybernetics, 20(3-4), pp.121-136.
8. Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. Learning representations by back-propagating errors. nature, 323(6088), pp.533-536.
9. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), pp.541-551.

References

10. Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

11. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

12. Hinton, G.E., Osindero, S. and Teh, Y.W., 2006. A fast learning algorithm for deep belief nets. Neural computation, 18(7), pp.1527-1554.

13. Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

14. Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

15. He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

16. Zeiler, M.D. and Fergus, R., 2014, September. Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

17. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

References

18. Zoph, B. and Le, Q.V., 2016. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.

19. Tan, M. and Le, Q.V., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946.

20. Gordon, A., Eban, E., Nachum, O., Chen, B., Wu, H., Yang, T.J. and Choi, E., 2018. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1586-1595).

22. Zöller, M.A. and Huber, M.F., Benchmark and Survey of Automated Machine Learning Frameworks.

23. Elshawi, R., Maher, M. and Sakr, S., 2019. Automated machine learning: State-of-the-art and open challenges. arXiv preprint arXiv:1906.02287.

24. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

References

27. https://mc.ai/mnist-vs-mnist-how-i-was-able-to-speed-up-my-deep-learning/

28. Grover, D. and Toghi, B., 2019, April. MNIST Dataset Classification Utilizing k-NN Classifier with Modified Sliding-window Metric. In Science and Information Conference (pp. 583-591). Springer, Cham.

29. http://yann.lecun.com/exdb/mnist/

30. Byerly, A., Kalganova, T. and Dear, I., 2020. A Branching and Merging Convolutional Network with Homogeneous Filter Capsules. arXiv preprint arXiv:2001.09136.

31. Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M.S., Hasan, M., Van Essen, B.C., Awwal, A.A. and Asari, V.K., 2019. A state-of-the-art survey on deep learning theory and architectures. Electronics, 8(3), p.292.

32. O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G.V., Krpalkova, L., Riordan, D. and Walsh, J., 2019, April. Deep learning vs. traditional computer vision. In Science and Information Conference (pp. 128-144). Springer, Cham.

33. Wang, Y.E., Wei, G.Y. and Brooks, D., 2019. Benchmarking TPU, GPU, and CPU platforms for deep learning. arXiv preprint arXiv:1907.10701.

34. OpenCV issue: https://github.com/opencv/opencv/issues/11697

35. Aleskerov, F.T., Mitichkin, E.O., Chistyakov, V.V., Shvydun, S.V. and Iakuba, V.I., National Research University Higher School Of Economics (hse), 2019. Method for selecting valid variants in search and recommendation systems (variants). U.S. Patent 10,275,418.

References

36. Shen, D., Wu, G. and Suk, H.I., 2017. Deep learning in medical image analysis. Annual review of biomedical engineering, 19, pp.221-248.

37. List of startups using AI for drug design: https://blog.benchsci.com/startups-using-artificial-intelligence-in-drug-discovery

38. Pumsirirat, A. and Yan, L., 2018. Credit card fraud detection using deep learning based on auto-encoder and restricted boltzmann machine. International Journal of advanced computer science and applications, 9(1), pp.18-25.

39. Chalapathy, R. and Chawla, S., 2019. Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.

40. Decoste, D. and Schölkopf, B., 2002. Training invariant support vector machines. Machine learning, 46(1-3), pp.161-190.

41. USPS dataset: https://cs.nyu.edu/~roweis/data/usps_all.mat

42. Ramachandran, P., Zoph, B. and Le, Q.V., 2017. Swish: a self-gated activation function. arXiv preprint arXiv:1710.05941, 7.

43. Nielsen, M.A., 2015. *Neural networks and deep learning* (Vol. 2018). San Francisco, CA, USA:: Determination press.

44. Byerly, A., Kalganova, T. and Dear, I., 2020. A Branching and Merging Convolutional Network with Homogeneous Filter Capsules. *arXiv preprint arXiv:2001.09136*.

45. LeNet-5 – A Classic CNN Architecture (Visual explanation of LeNet-5): https://engmrk.com/lenet-5-a-classic-cnn-architecture/

References

46. VGG16 – Convolutional Network for Classification and Detection: https://neurohive.io/en/popular-networks/vgg16/

47. Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

48. Cai, Z. and Vasconcelos, N., 2018. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6154-6162).

49. Karpathy, A. and Fei-Fei, L., 2015. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3128-3137).

50. Csáji, B.C., 2001. Approximation with artificial neural networks. Faculty of Sciences, Etvs Lornd University, Hungary, 24(48), p.7.

51. Single Layer Perceptron visual representation: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

52. G. Cybenko. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4):303–314, 1989.

53. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366, 1989.

54. K. Funahashi. On the approximate realization of continuous mappings by neural networks. Neural networks, 2(3):183–192, 1989.

References

55. V. Kurková. Kolmogorov’s theorem and multilayer neural networks. Neural networks, 5(3): 501–506, 1992.

56. Adagrad: Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, 12, 2121–2159. Retrieved from http://jmlr.org/papers/v12/duchi11a.html

57. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

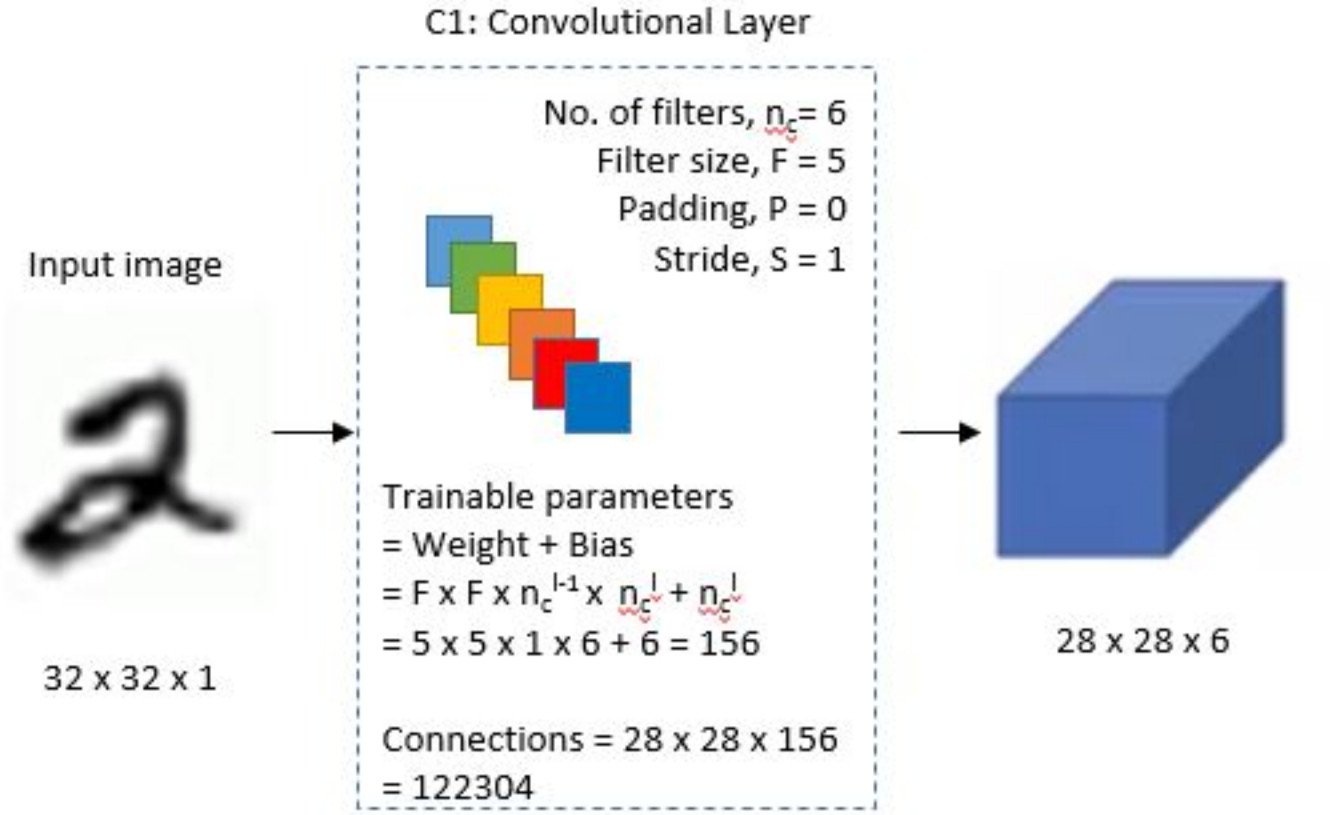
Appendix 1: Deep Learning Hardware

Platform	*Unit*	*Version*	*Mem Type*	*Mem (GB)*	*Mem Bdw (GB/s)*	Peak *FLOPS*
CPU	1 VM	Skylake	DDR4	120	16.6	2T SP†
GPU (DGX-1)	1 Pkg	V100 (SXM2)	HBM2	16	900	125T
TPU	1 Board (8 cores)	v2	HBM	8	2400	180T
TPUv3	8 cores	v3	HBM	16	3600*	420T

† Single precision: 2 FMA × 32 SP × 16 cores × 2G frequency = 2 SP TFLOPS
* Estimated based on empirical results (Section 4.5).

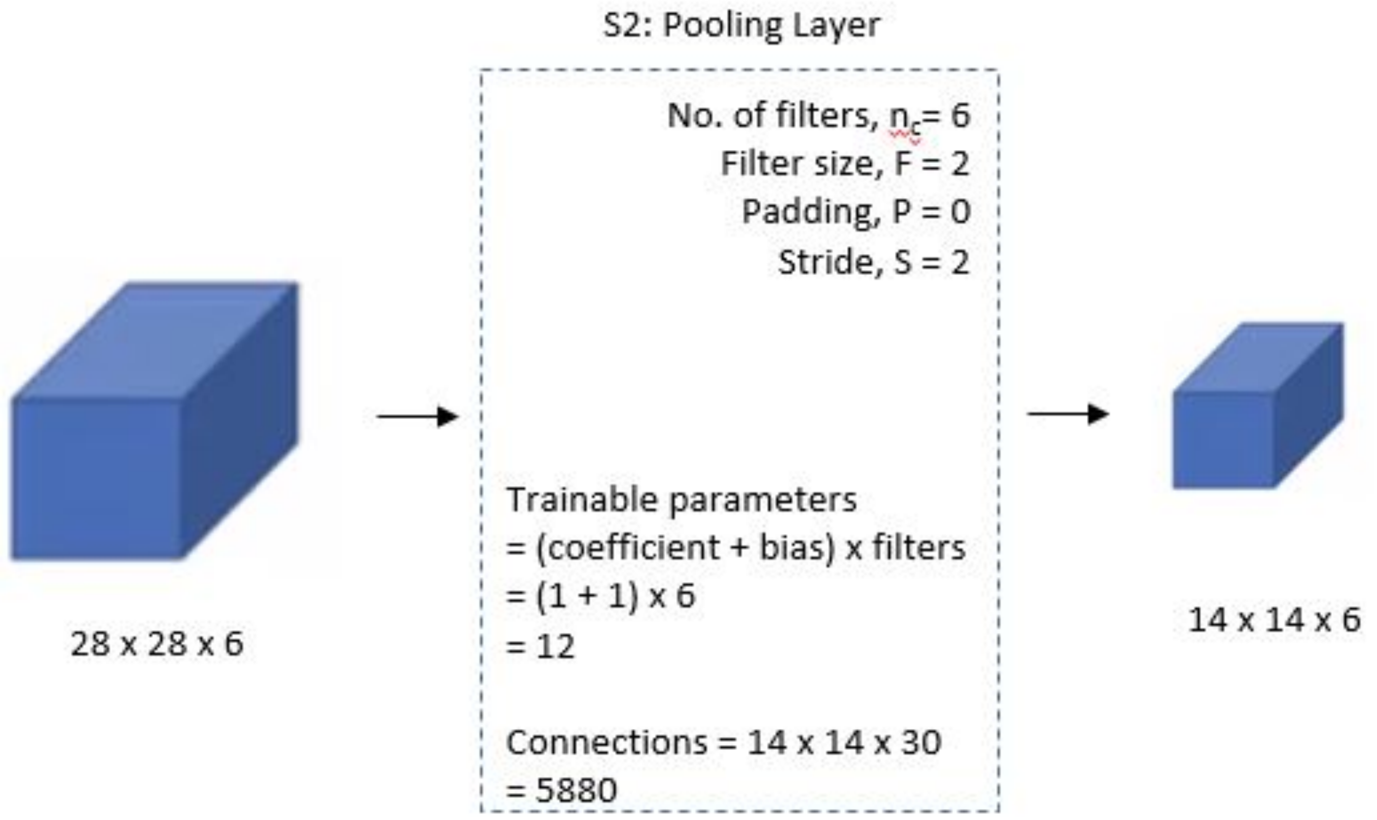
Source: [33]

Deep Learning: Architectures – CNN – LeNet-5



LeNet-5 C1 Convolutional Layer. Source: [45]

Deep Learning: Architectures – CNN – LeNet-5



LeNet-5 S2 Average Pooling Layer. Source: [45]

Deep Learning: Architectures – CNN – LeNet-5

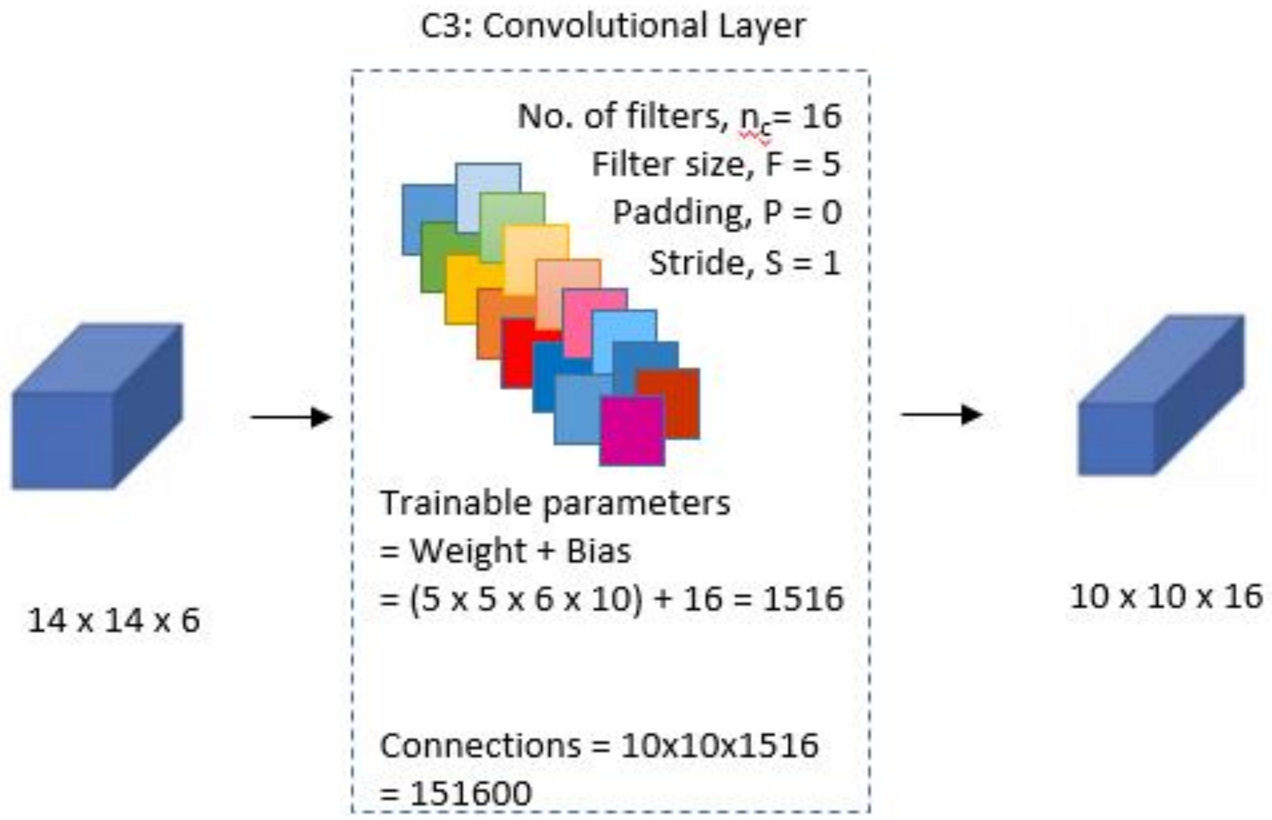
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

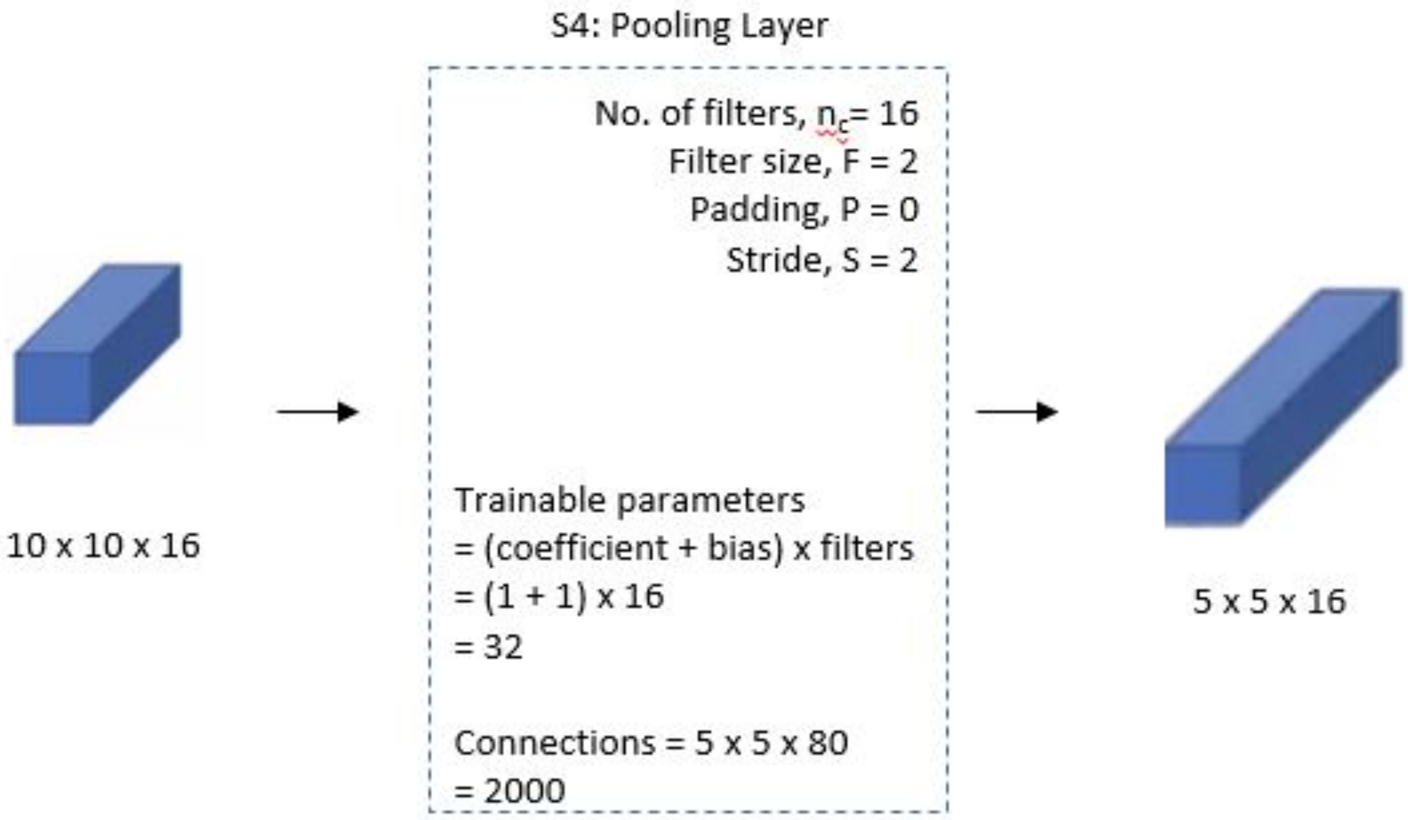
LeNet-5 S2 – C3 feature mapping. Source: [45]

Deep Learning: Architectures – CNN – LeNet-5



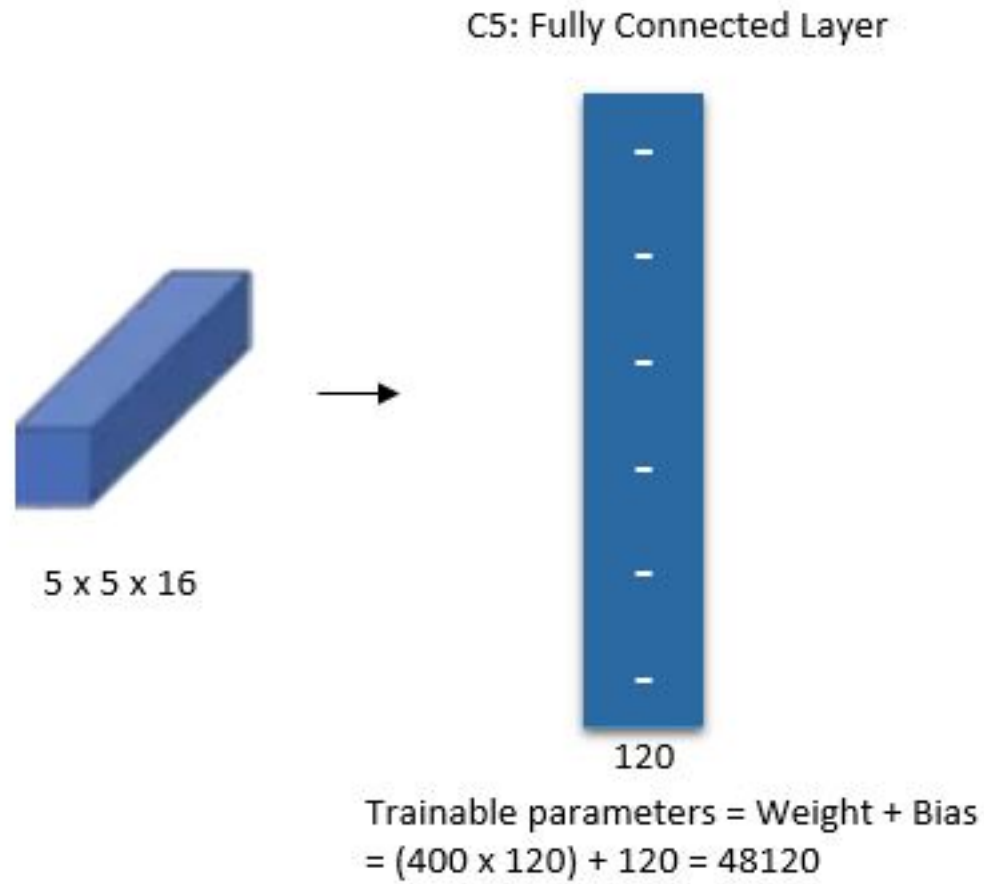
LeNet-5 C3 Convolutional Layer. Source: [45]

Deep Learning: Architectures – CNN – LeNet-5



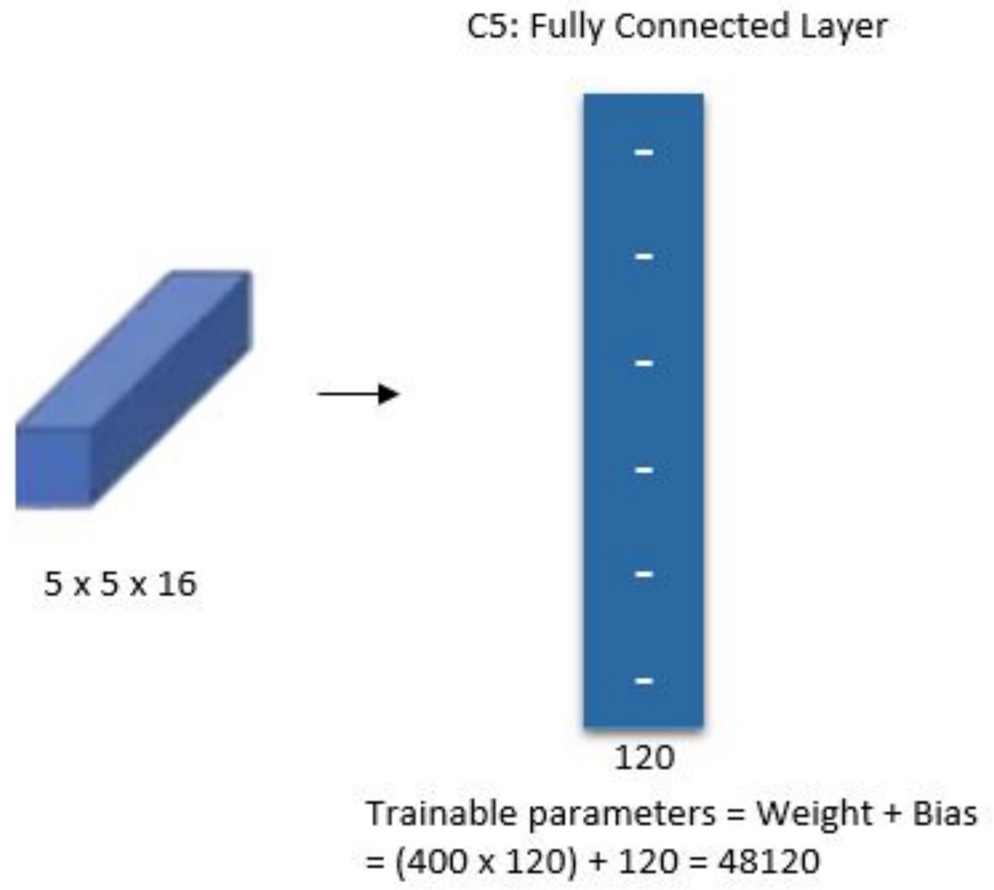
LeNet-5 S4 Average Pooling Layer. Source: [45]

Deep Learning: Architectures – CNN – LeNet-5



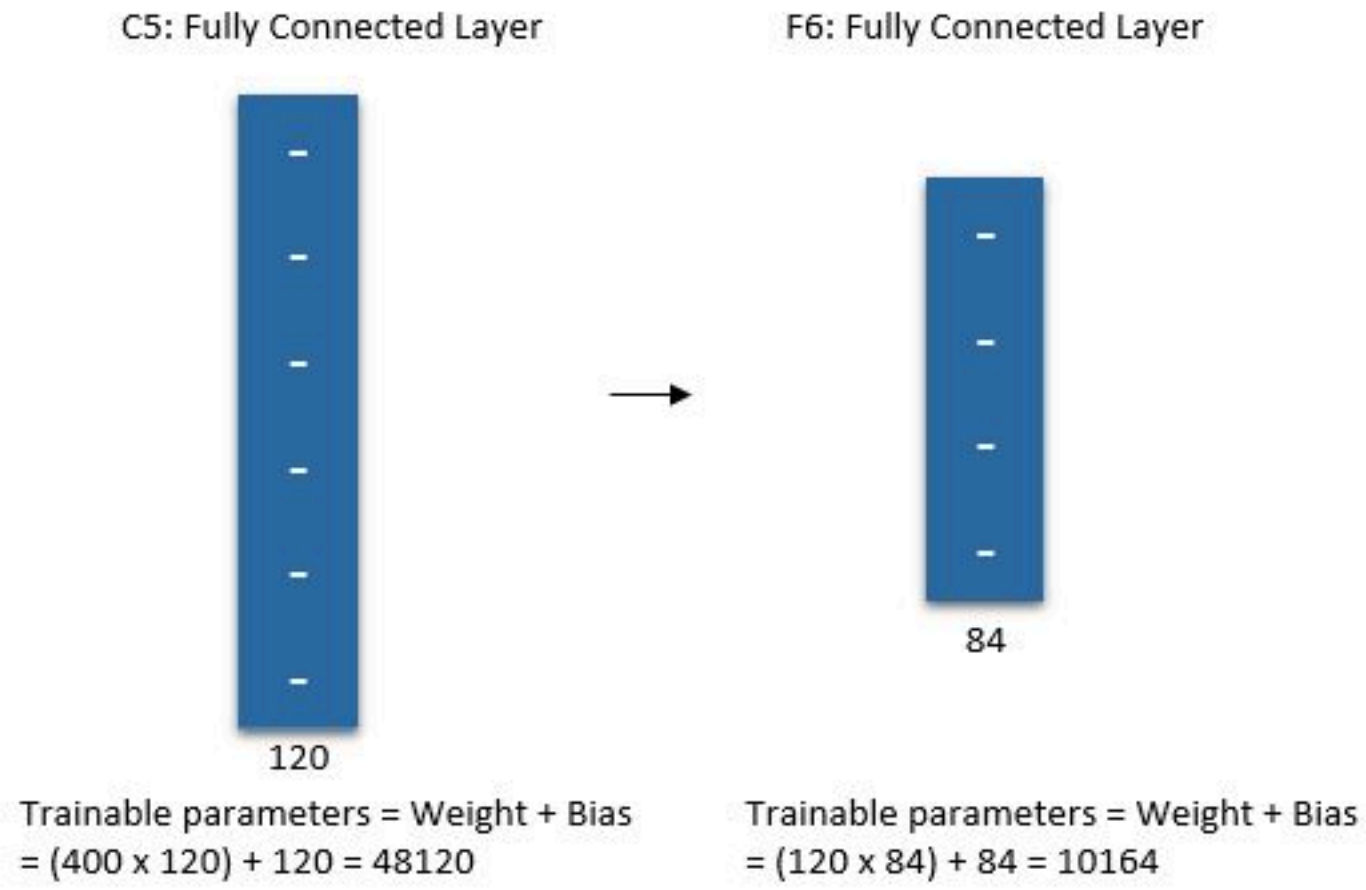
LeNet-5 C5 Fully Connected Layer. Source: [45]

Deep Learning: Architectures – CNN – LeNet-5



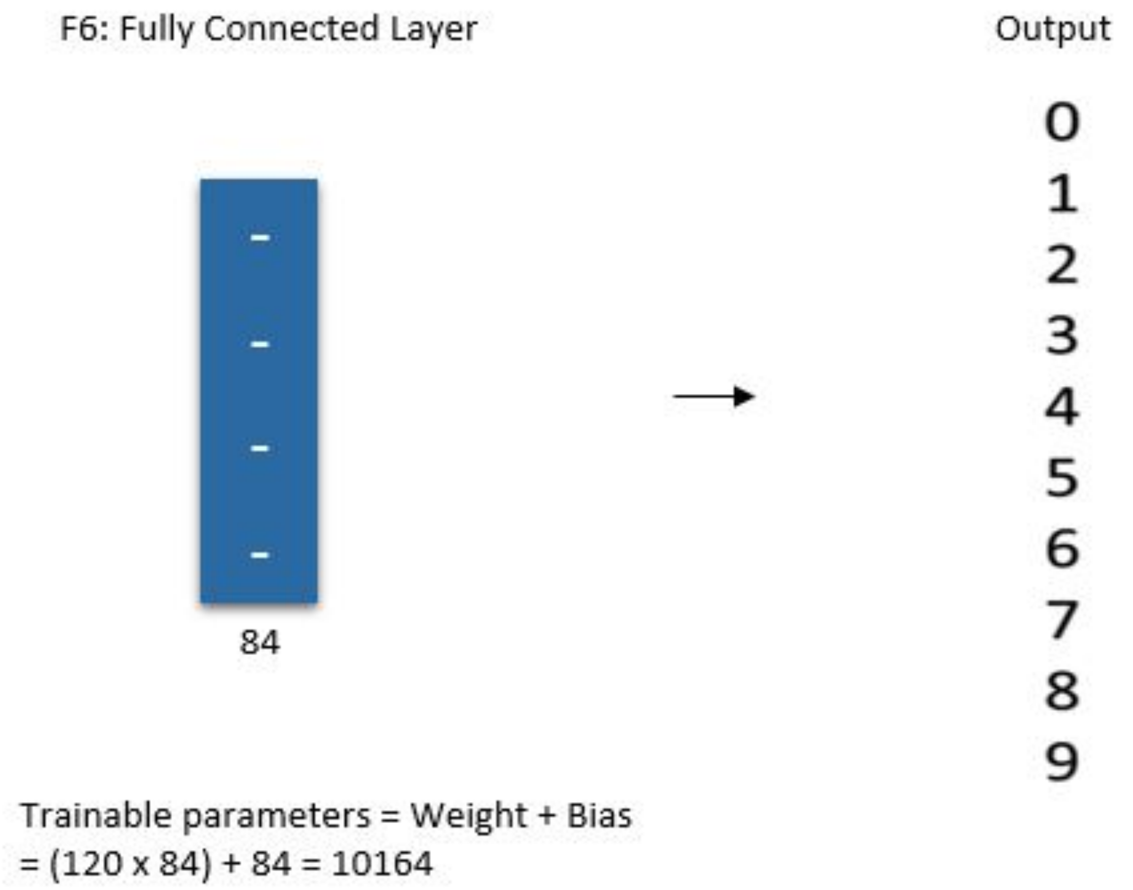
LeNet-5 C5 Fully Connected Layer. Source: [45]

Deep Learning: Architectures – CNN – LeNet-5



LeNet-5 F6 Fully Connected Layer. Source: [45]

Deep Learning: Architectures – CNN – LeNet-5



LeNet-5 Fully Connected Output Layer. Source: [45]