Machine Learning

2024 (ML-2024)
Lecture 7. Convolutional neural networks

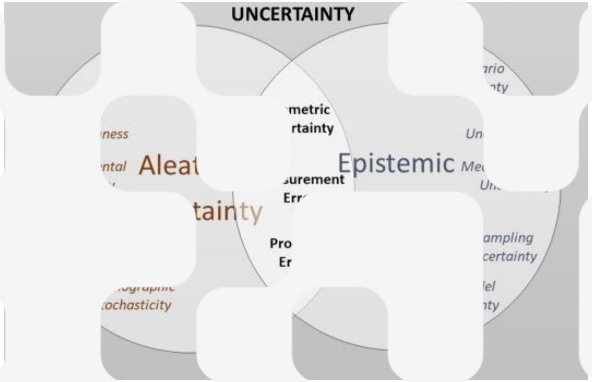
by Alexei Valerievich Kornaev, Dr. habil. in Eng. Sc.,
Researcher at the RC for AI, Assoc. Prof. of the Robotics and CV Master's Program, Innopolis University
Researcher at the RC for AI, National RC for Oncology n.a. NN Blohin
Professor at the Dept. of Mechatronics, Mechanics, and Robotics, Orel State University

Agenda

I. RECAP ON ANNs

II. CONVOLUTIONAL NEURAL NETWORKS (CNNs)

III. RESIDUAL NEURAL NETWORKS (ResNets)



All models are wrong, but ~~some~~ models that know when they are wrong, are useful
/George Box + unknown researcher from the Google AI Brain Team/

Frequentist vs Bayesian frameworks

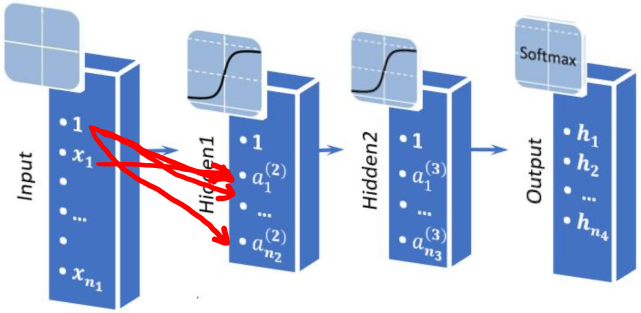
	Frequentist	Bayesian
Randomness	Objective indefiniteness	Subjective ignorance
Inference	Random and Deterministic	Everything is random
Estimates	Maximal likelihood	Bayes theorem
Applicability	$n \gg size(\boldsymbol{\theta})$	$\forall n$

Deep Bayes School

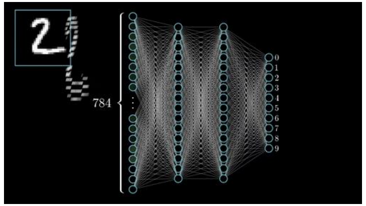
Recap: feed-forward (fully-connected, multi-layer perceptron) neural networks intuition

$$L\left(\boldsymbol{\Theta}^{(k)}\right) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n_l}\left(y_j^{(i)}\ln\left(h_j^{(i)}\right)\right) + \frac{\lambda}{2m}\sum_{k=1}^{l-1}\sum_{i=1}^{n_k}\sum_{j=1}^{n_{k+1}}\left(\theta_{ij}^{(k)}\right)^2 \Rightarrow \min.$$

$\boldsymbol{x} = \left(x_1 \dots x_{n_1}\right); \rightarrow$



$$\rightarrow \boldsymbol{h} = \begin{pmatrix} 0.01 \\ 0.003 \\ 0.27 \\ \dots \\ 0.012 \end{pmatrix} \leftrightarrow \boldsymbol{y} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \dots \\ 0 \end{pmatrix}.$$



Algorithm:

1. Initialize weights $\boldsymbol{\Theta}^{(k)}$ randomly.
2. Calculate $\nabla L = \left[\partial L / \partial \theta_{ij}^{(k)}\right]$ with backpropagation.
3. Update weights $\boldsymbol{\Theta}^{(k)}$: $\theta_{ij}^{(k)H} = \theta_{ij}^{(k)C} - \alpha \frac{\partial L}{\partial \theta_{ij}^{(k)}}$.
4. Repeat pp. 2-3 until $L^H - L^C < \delta$ or #iter $> N_{max}$.
5. Save the best model (with min. validation loss): $\boldsymbol{\Theta}^{(k)}$.

$$\Theta^{(k)} = \left(\begin{pmatrix} \theta_{01}^{(k)} & \theta_{02}^{(k)} & & \theta_{0n_2}^{(k)} \\ \theta_{11}^{(k)} & \theta_{12}^{(k)} & \dots & \theta_{1n_3}^{(k)} \\ \dots & \dots & & \dots \\ \theta_{n_k1}^{(k)} & \theta_{n_k2}^{(k)} & & \theta_{n_kn_{k+1}}^{(k)} \end{pmatrix}\right).$$

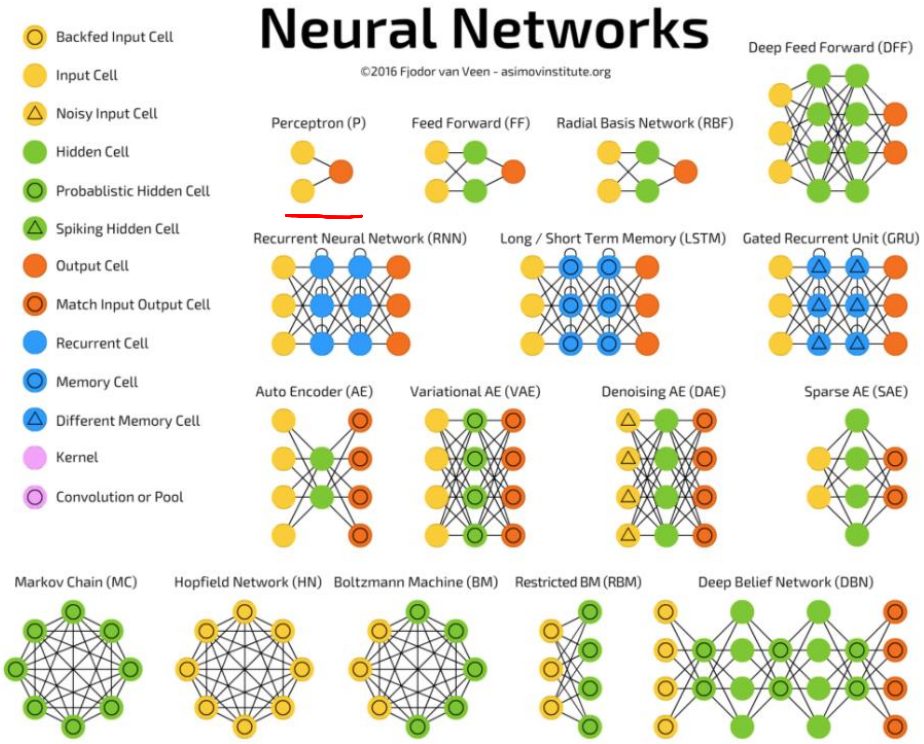
$\Theta_{ij}^{(k)}$

(k) – # слоя ИНС

j – # нейр. "k+1" слоя

i – # нейрона k-го слоя

Recap: some of the ANN architectures



MLP

[Almost complete chart of NNs (2016)]

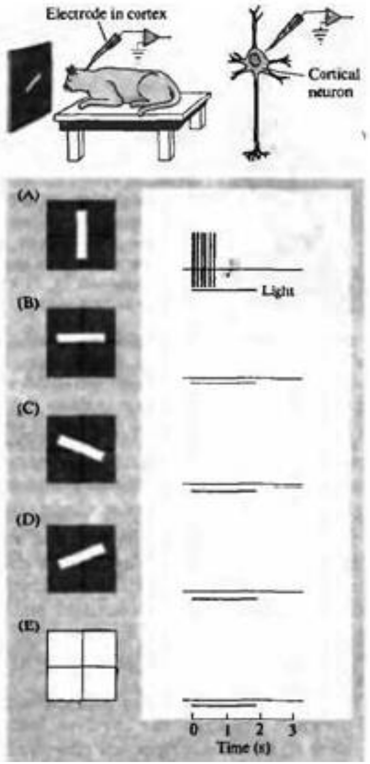
Why fully-connected networks are not good for image processing?

1. **Parameters explosion.** Fully-connected layers require a large number of parameters, especially when dealing with high-resolution images. For example, a 224x224 RGB image has 150,528 features (224 * 224 * 3). If the first hidden layer has 1000 neurons, the weight matrix would have 150,528 * 1000 = 150,528,000 parameters.
2. **Spatial Structure Ignorance.** Fully-connected layers treat each input feature independently, ignoring the spatial relationships between pixels in an image. In images, neighboring pixels are often correlated and contain important information about edges, textures, and shapes.
3. **Invariance to Translation.** Fully-connected layers are not inherently invariant to translation (i.e., shifting the image). This means that the network would need to learn separate representations for objects in different positions, which is inefficient.

Physiology of cats

«Мы как раз вставляли слайд на стекле в виде тёмного пятна в разъём офтальмоскопа, когда внезапно, через аудиомонитор, клетка зарядила как пулемёт. Спустя некоторое время, после небольшой паники, мы выяснили, что же случилось. Конечно, сигнал не имел никакого отношения к тёмному пятну. Во время того, как мы вставляли слайд на стекле, его край отбрасывал на сетчатку слабую, но чёткую тень, в виде прямой тёмной линии на светлом фоне. Это было именно то, чего хотела клетка, и, более того, она хотела, чтобы эта линия имела строго определённую ориентацию».

/Д. Хьюбел Фрагмент нобелевской речи, 1981 г./



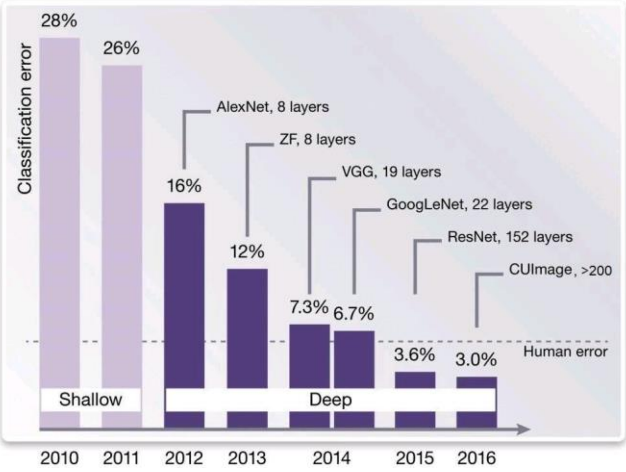
Ответы клетки зрительной коры кошки на предъявление полосок света [Дж.Г. Николлс и др. От нейрона к мозгу, 2003]

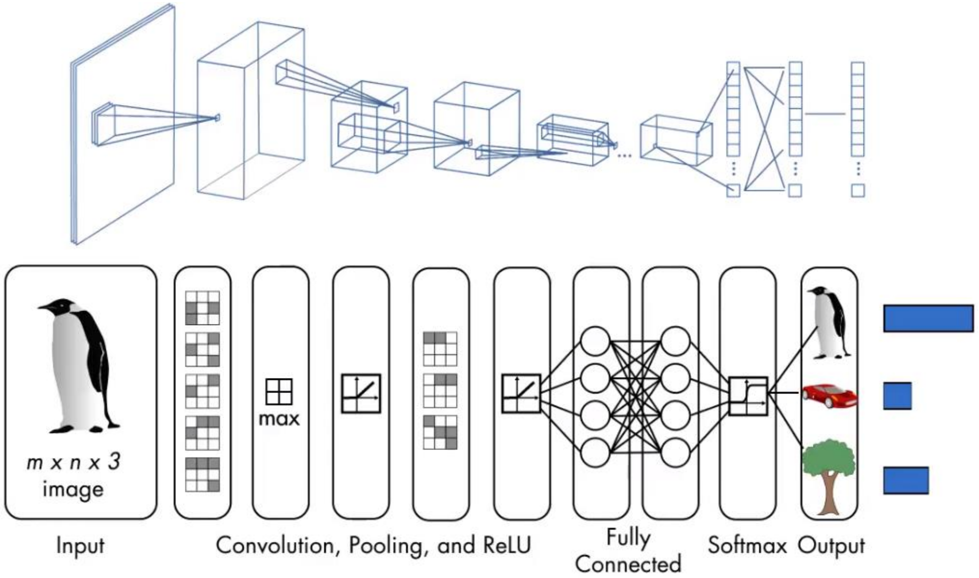
[Эксперимент с котом Хьюбела и Визеля, YouTube]

[Как видят кошки]

Convolutional neural networks (CNNs)

CNN utilizes the features of the visual cortex, where simple cells are activated by simple features (such as lines), and complex cells by combinations of activations of simple cells. The CNN is associated with the mathematical operation of convolution for reducing matrix sizes.

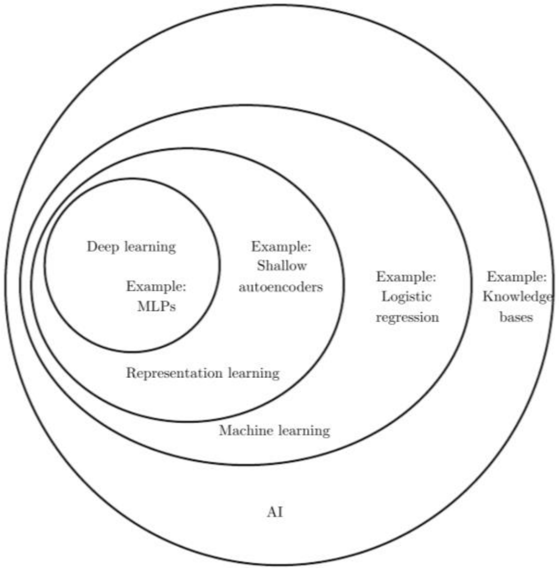


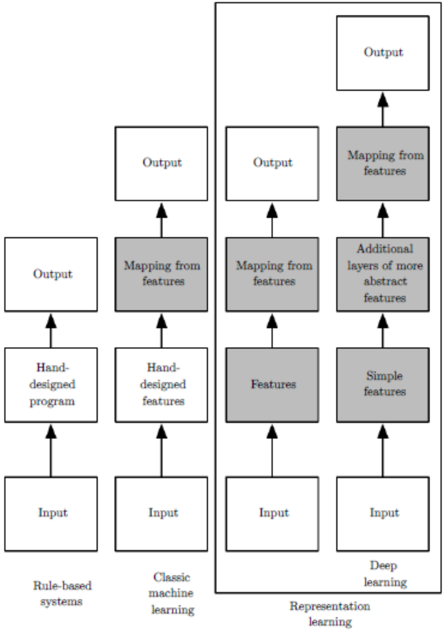


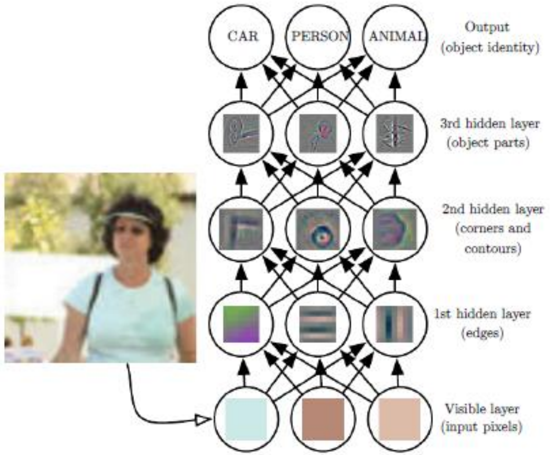
[Deep Learning using MATLAB: AlexNet arhitecture]

Convolutional neural networks (CNNs)

CNN utilizes the features of the visual cortex, where simple cells are activated by simple features (such as lines), and complex cells by combinations of activations of simple cells. The CNN is associated with the mathematical operation of convolution for reducing matrix sizes.





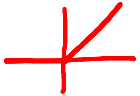


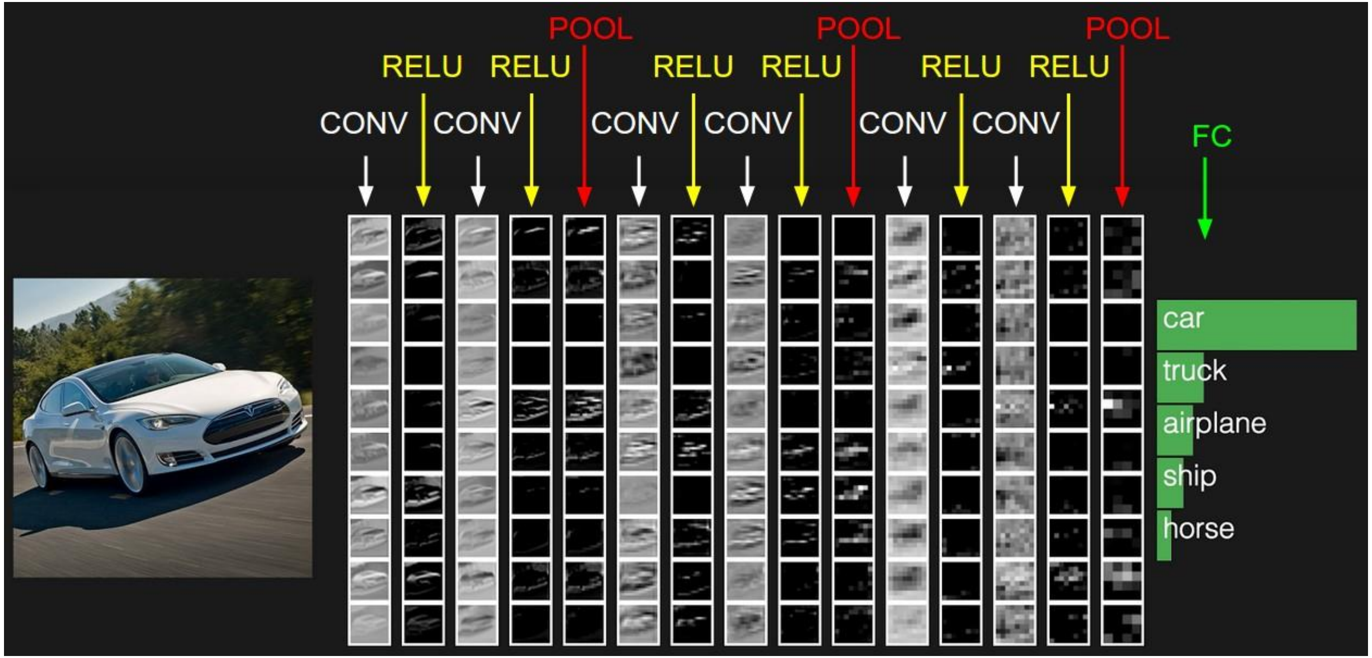
Deep Learning by I.Goodfellow

Convolutional neural networks (CNNs)

CNN utilizes the features of the visual cortex, where simple cells are activated by simple features (such as lines), and complex cells by combinations of activations of simple cells. The CNN is associated with the mathematical operation of convolution for reducing matrix sizes.

ReLU

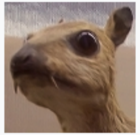
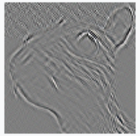
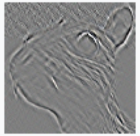
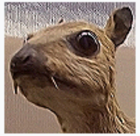




[CS231n: Deep Learning for Computer Vision. Stanford, 2024](https://cs231n.stanford.edu)

Parts of a CNN: kernels

CNN utilizes the features of the visual cortex, where simple cells are activated by simple features (such as lines), and complex cells by combinations of activations of simple cells. The CNN is associated with the mathematical operation of convolution for reducing matrix sizes.

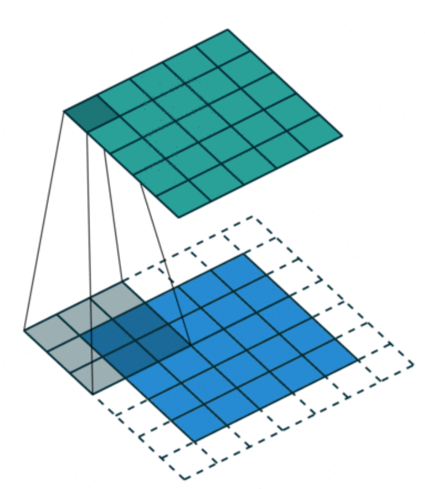
Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Kernel (image processing), Wikipedia

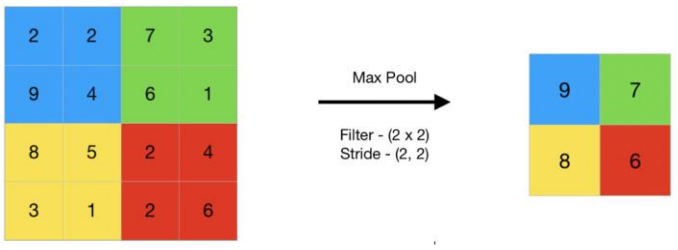
Parts of a CNN: padding, pooling, striding

CNN utilizes the features of the visual cortex, where simple cells are activated by simple features (such as lines), and complex cells by combinations of activations of simple cells. The CNN is associated with the mathematical operation of convolution for reducing matrix sizes.

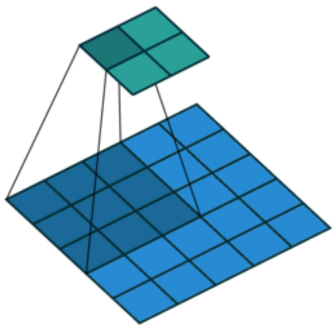
Дополнение / Padding



Группирование / Pooling



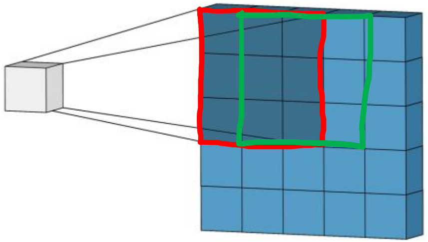
Шагание / Striding



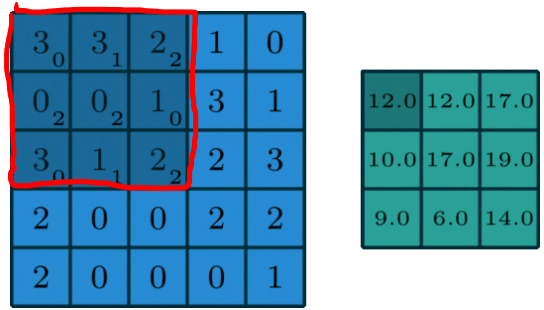
CNNs by Neurohive.io

Parts of a CNN: convolution (one filter image)

CNN utilizes the features of the visual cortex, where simple cells are activated by simple features (such as lines), and complex cells by combinations of activations of simple cells. The CNN is associated with the mathematical operation of convolution for reducing matrix sizes.



$$X = \begin{pmatrix} 3 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 3 & 1 \\ 3 & 1 & 2 & 2 & 3 \\ 2 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow A^{(1)} = \begin{pmatrix} 3 & 3 & 2 & 0 & 0 & 1 & 3 & 1 & 2 \\ 3 & 2 & 1 & 0 & 1 & 3 & 1 & 2 & 2 \\ 2 & 1 & 0 & 1 & 3 & 1 & 2 & 2 & 3 \\ 0 & 0 & 1 & 3 & 1 & 2 & 2 & 0 & 0 \\ 0 & 1 & 3 & 1 & 2 & 2 & 0 & 0 & 2 \\ 1 & 3 & 1 & 2 & 2 & 3 & 0 & 2 & 2 \\ 3 & 1 & 2 & 2 & 0 & 0 & 2 & 0 & 0 \\ 1 & 2 & 2 & 0 & 0 & 2 & 0 & 0 & 0 \\ 2 & 2 & 3 & 0 & 2 & 2 & 0 & 0 & 1 \end{pmatrix};$$

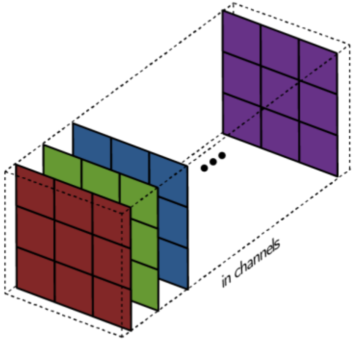


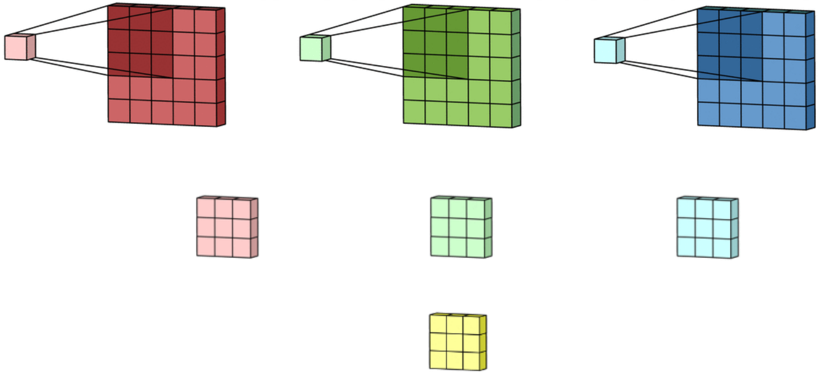
$$\Theta^{(1)} = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix} \rightarrow \Theta^{(1)} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 2 \\ 2 \\ 0 \\ 0 \\ 1 \\ 2 \end{pmatrix}; \; \underbrace{Z^{(2)} = A^{(1)}\Theta^{(1)}}_{\text{Convolution}} = \begin{pmatrix} 12 \\ 12 \\ 17 \\ 10 \\ 17 \\ 19 \\ 9 \\ 6 \\ 14 \end{pmatrix} \rightarrow Z^{(2)} = \begin{pmatrix} 12 & 12 & 17 \\ 10 & 17 & 19 \\ 9 & 6 & 14 \end{pmatrix}.$$

CNNs by Neurohive.io

Parts of a CNN: convolution (3 filters image)

CNN utilizes the features of the visual cortex, where simple cells are activated by simple features (such as lines), and complex cells by combinations of activations of simple cells. The CNN is associated with the mathematical operation of convolution for reducing matrix sizes.





[CNNs by Neurohive.io]

Parts of a CNN: convolution (3 filters image, 2 kernels)

Example: given a color image of size [5 5 3], convolution with 2 kernels of size [3 3 3], padding [1], stride [2 2].

The result of convolving an image of size [q q] with a kernel of size [k k], with padding (p) and stride [s s], is a matrix of size [r r]:

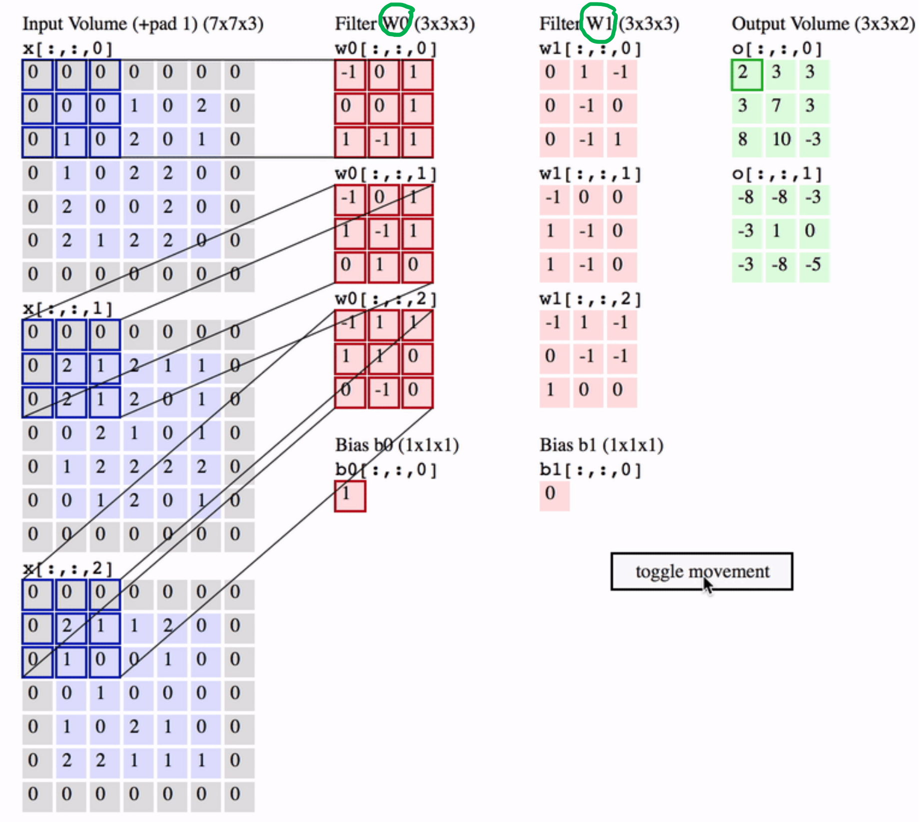
$$r = \frac{q-k+2p}{s} + 1 = \frac{5-3+2}{2} + 1 = 3.$$

# of convolutions: $r^2 = 9$.

$X, [5\ 5\ 3] \to X, [7\ 7\ 3] \to A^{(1)},\ size = [9\ 27];$

$\Theta^{(1)}, [3\ 3\ 3\ 2] \to \Theta^{(1)},\qquad size = [27\ 2];$

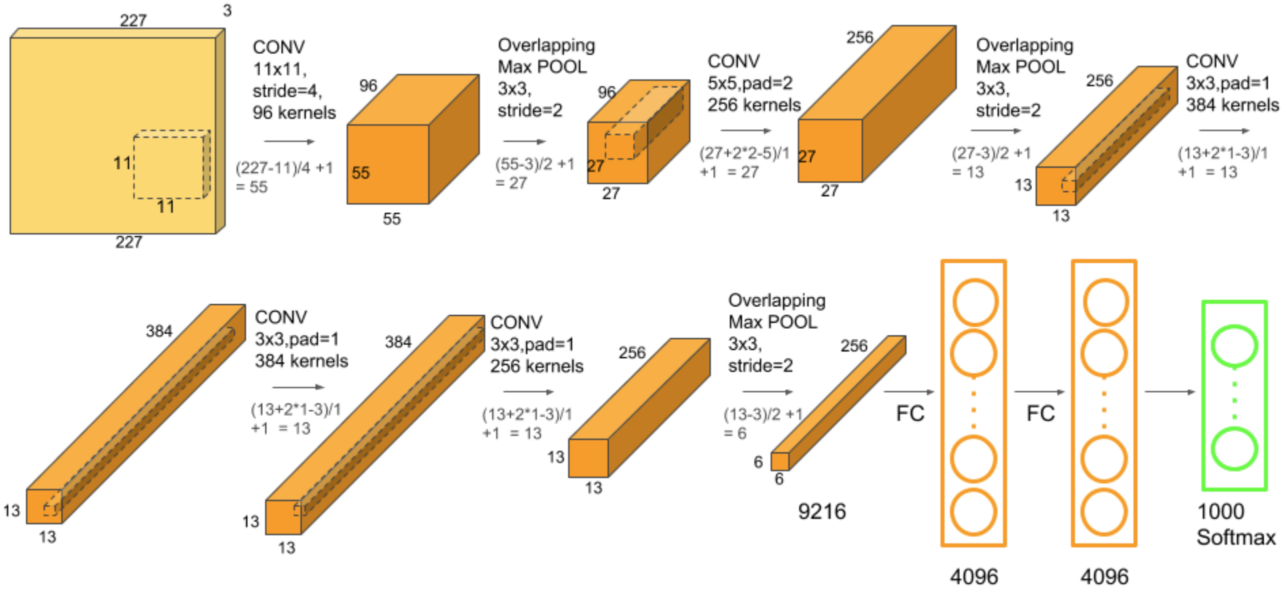
$Z^{(2)} = A^{(1)}\Theta^{(1)}, [9\ 2] \to Z^{(2)} = Z^{(2)} + \Theta_0^{(1)}, [9\ 2] \to Z^{(2)},\ \ size = [3\ 3\ 2].$



[CS231n: Deep Learning for Computer Vision. Stanford, 2024]()

Parts of a CNN: collecting all the parts

Example: AlexNet (designed by Alex Krizhevsky) – a deep convolutional neural network for recognizing 1000 classes, recognized as the best in 2012 in the ImageNet Large Scale Visual Recognition Challenge.

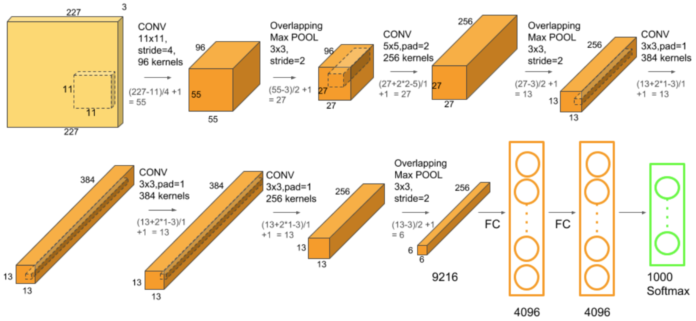


[AlexNet architecture]

Parts of a CNN: collecting all the parts

$$L\left(\boldsymbol{\Theta}^{(k)}\right) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n_l}\left(y_j^{(i)}\ln\left(h_j^{(i)}\right)\right) + \frac{\lambda}{2m}\sum_{k=1}^{l-1}\sum_{i=1}^{n_k}\sum_{j=1}^{n_{k+1}}\left(\theta_{ij}^{(k)}\right)^2 \Rightarrow \min.$$

Consider a model $\boldsymbol{f} = \left[\boldsymbol{x}^{(i)}, \boldsymbol{\Theta}\right]$ parameterized with weights $\boldsymbol{\Theta}$ that maps each i-th input sample $\boldsymbol{x}^{(i)}$ into the output $\boldsymbol{z}^{(i)}$ which then transforms into the hypothesis $\boldsymbol{h}^{(i)}$ that should be close to the label $\boldsymbol{y}^{(i)}$.



Algorithm:

1. Initialize weights $\boldsymbol{\Theta}^{(k)}$ randomly.
2. Calculate $\nabla L = \left[\partial L / \partial \theta_{ij}^{(k)}\right]$ with backpropagation.
3. Update weights $\boldsymbol{\Theta}^{(k)}$: ${\theta_{ij}^{(k)}}^H = {\theta_{ij}^{(k)}}^C - \alpha \frac{\partial L}{\partial \theta_{ij}^{(k)}}$.
4. Repeat pp. 2-3 until $L^H - L^C < \delta$ or #iter > N_{max}.
5. Save the best model (with min. validation loss): $\boldsymbol{\Theta}^{(k)}$.

[AlexNet architecture]

Use case: 1D CNN is a powerful tool in signal processing

$$z_i = w_{i-1}x_{i-1} + w_i x_i + w_{i+1}x_{i+1}.$$

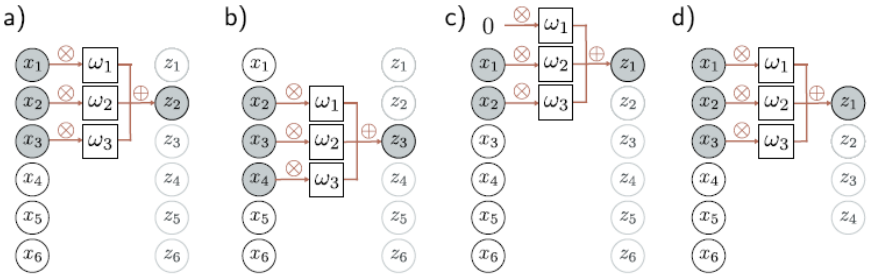
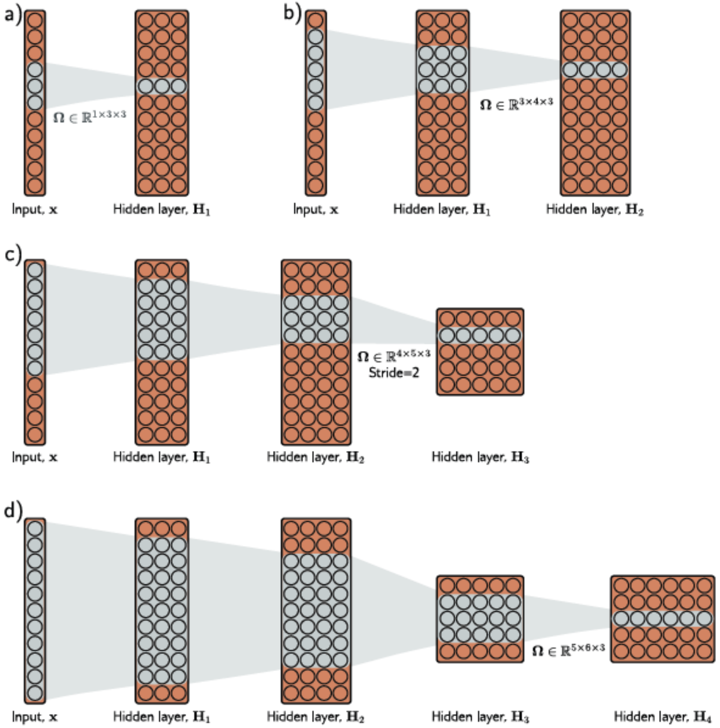


Figure 10.2 1D convolution with kernel size three. Each output z_i is a weighted sum of the nearest three inputs x_{i-1}, x_i, and x_{i+1}, where the weights are $\omega = [\omega_1, \omega_2, \omega_3]$. a) Output z_2 is computed as $z_2 = \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$. b) Output z_3 is computed as $z_3 = \omega_1 x_2 + \omega_2 x_3 + \omega_3 x_4$. c) At position z_1, the kernel extends beyond the first input x_1. This can be handled by zero-padding, in which we assume values outside the input are zero. The final output is treated similarly. d) Alternatively, we could only compute outputs where the kernel fits within the input range ("valid" convolution); now, the output will be smaller than the input.



[S. J. Prince. Understanding Deep Learning. MIT Press, 2023.]

Tune in next time: what to do if your CNN model is overfitted?

Self-study and self-test questions

1. Why does ReLU work so well?
2. How to make convolution preserve the image size?
3. How many trainable parameters in a kernel of size [3,3,2]?
4. Is it possible to avoid using fully-connected layers in CNNs?
5. How does 3D CNNs work?
6. How does hyperspectral image processing work?

Thank you for your attention!

a.kornaev@innopolis.ru, @avkornaev