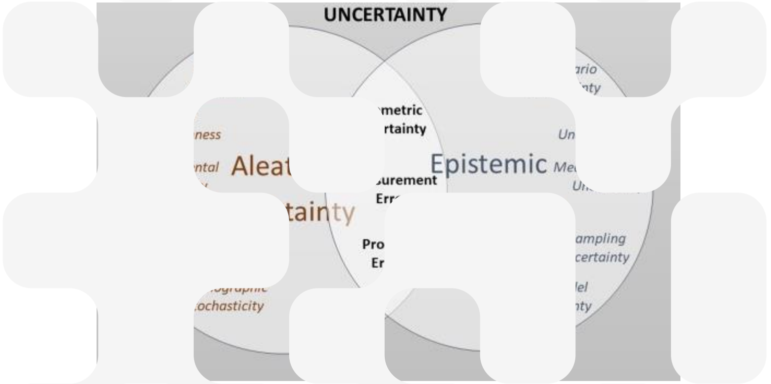
Machine Learning

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

ML-2024
by Alexei Kornaev

Agenda



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	$n \gg size(\boldsymbol{\theta})$	$\forall n$

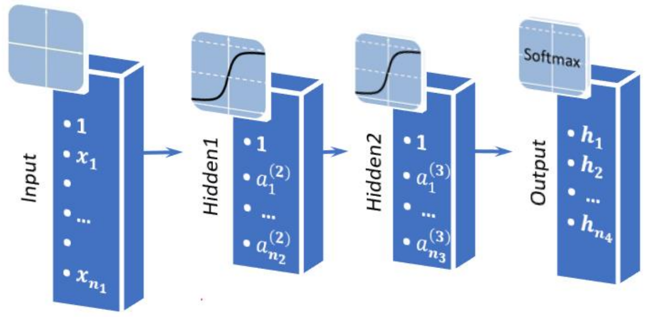
Deep Bayes School

ML-2024
by Alexei Kornaev

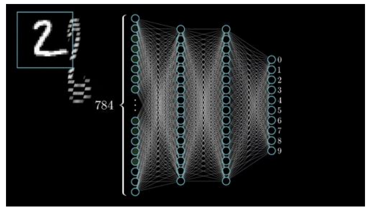
Recap: feed-forward (fully-connected, multi-layer perceptron) neural networks intuition

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

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



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



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): $\boldsymbol{\Theta}^{(k)}$.

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

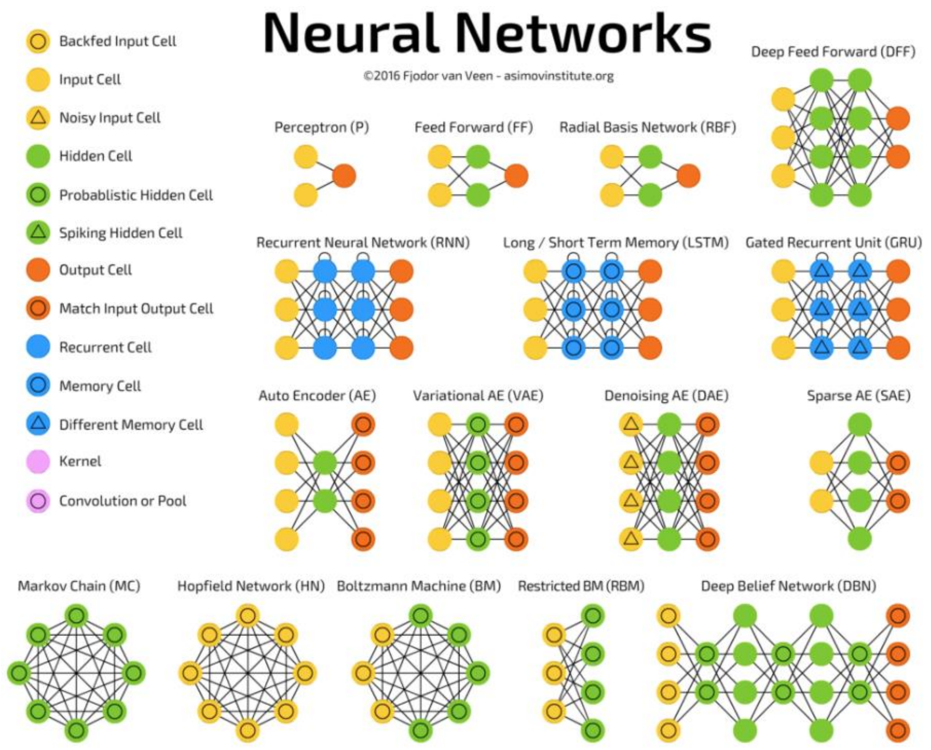
$\Theta_{ij}^{(k)}$

# слоя ИНС

# нейр. "k+1" слоя

# нейрона $k^{\text{го}}$ слоя

Recap: some of the ANN architectures



Almost complete chart of NNs (2016)

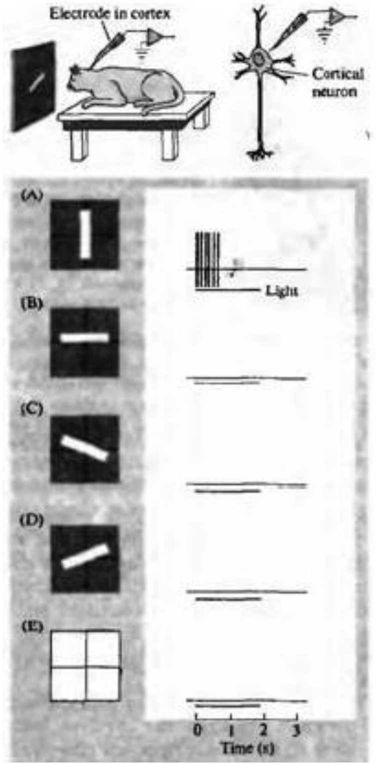
Why fully-connected networks are not good for image processing?

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./



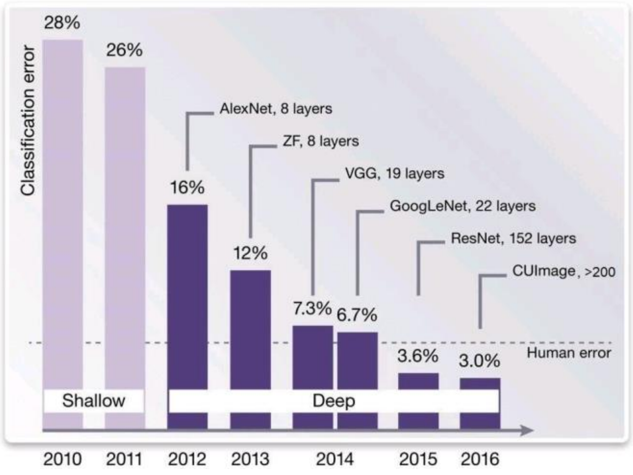
Ответы клетки зрительной коры кошки на предъявление полосок света [Дж.Г. Николлс и др. От нейрона к мозгу, 2003]

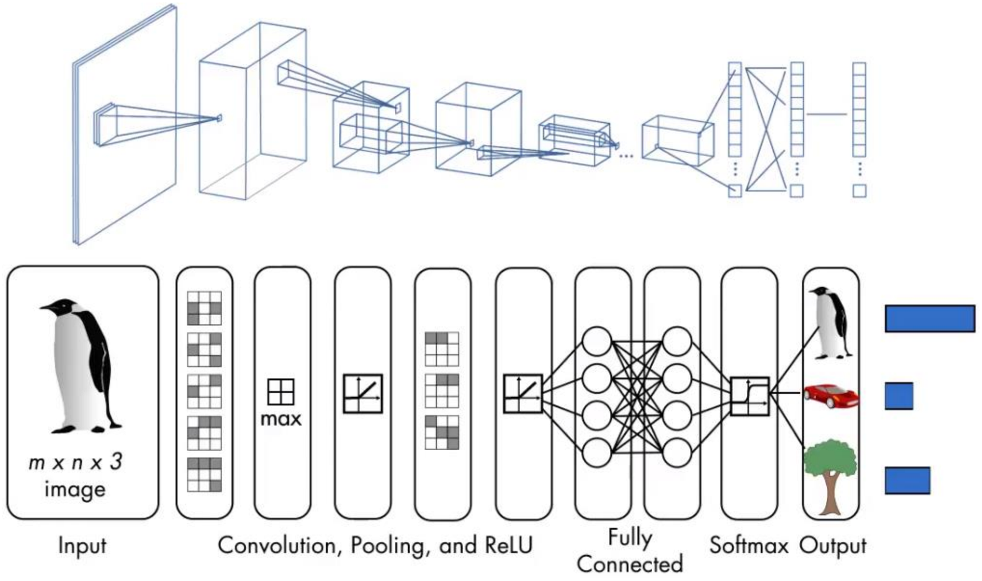
Эксперимент с котом Хьюбела и Визеля, YouTube

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.

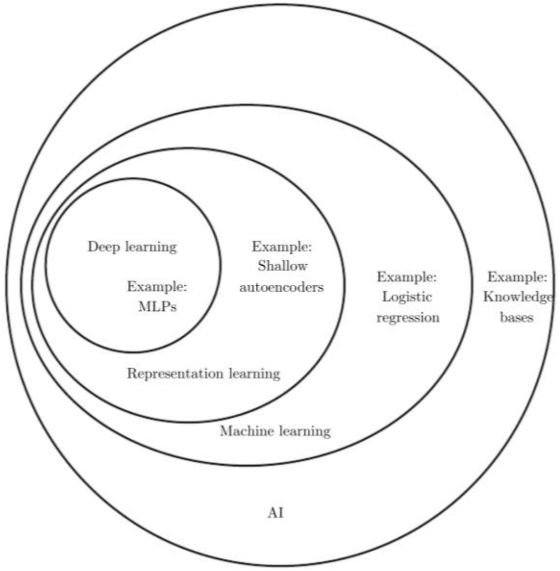


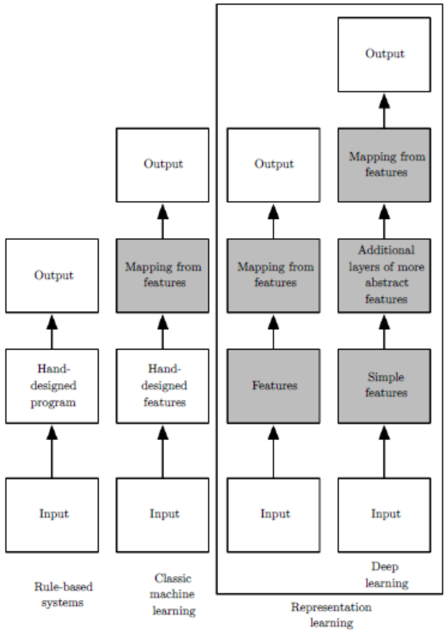


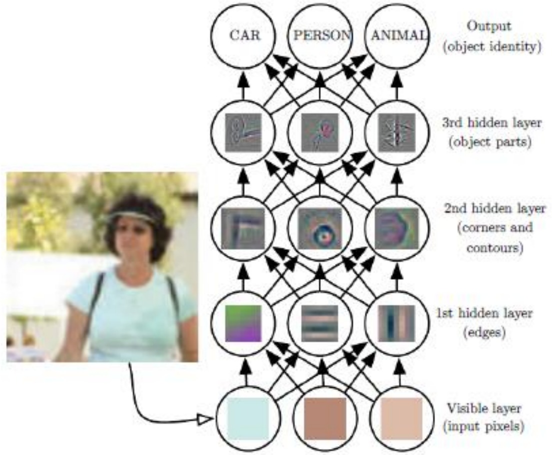
[Deep Learning using MATLAB: AlexNet arhitecture]()

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.



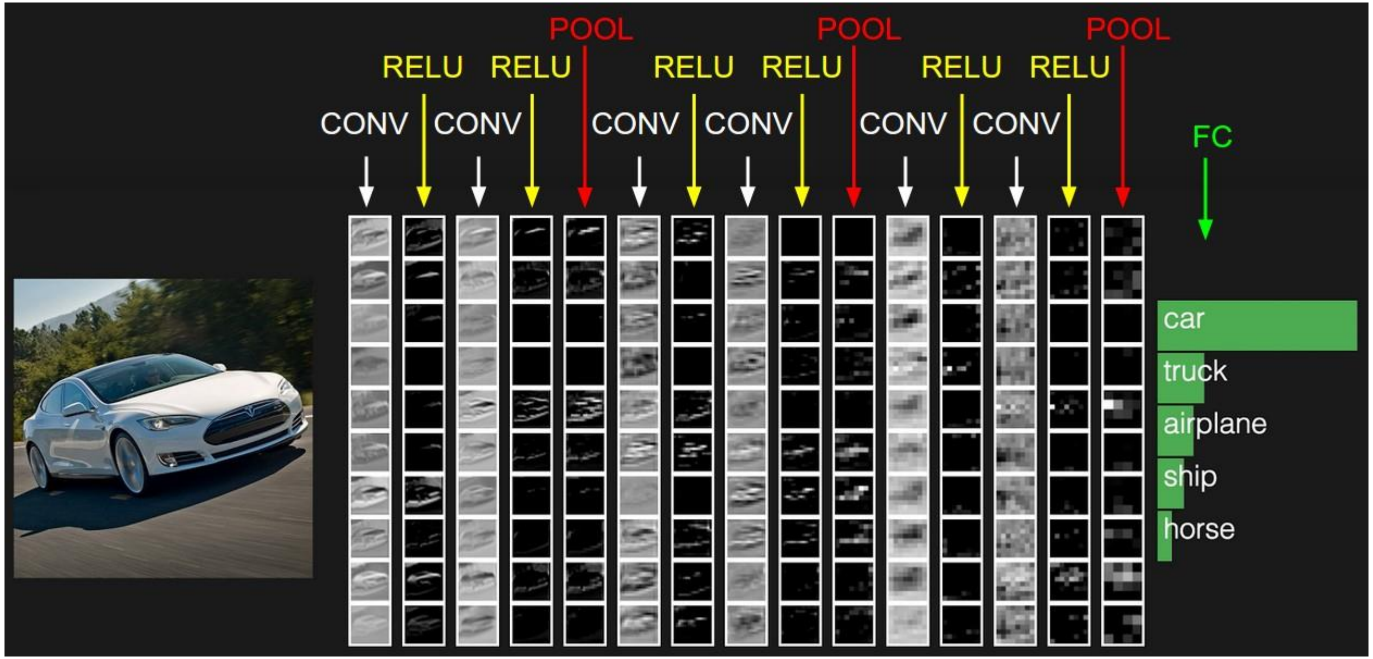



CAR
PERSON
ANIMAL
Output (object identity)
3rd hidden layer (object parts)
2nd hidden layer (corners and contours)
1st hidden layer (edges)
Visible layer (input pixels)

Deep Learning by I.Goodfellow

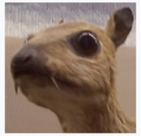
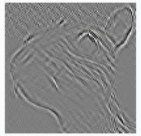
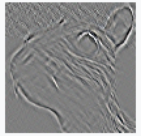
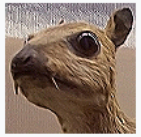
Convolutional neural networks (CNNs)

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.



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

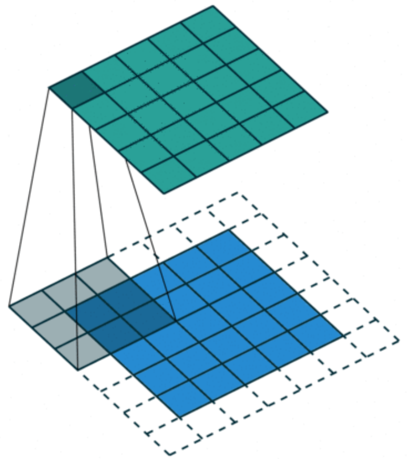
Parts of a CNN: padding, pooling, striding

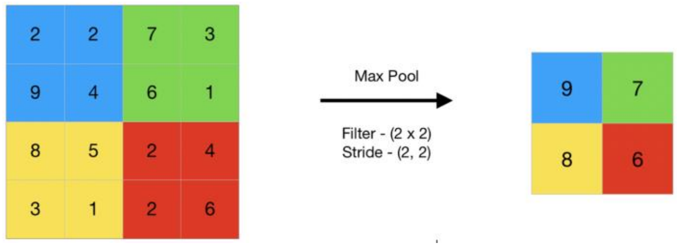
CNN utilizes the features of the visual cortex, where simple cells are activated by simple features (such as lines), and complex cells by combinations of activations of simple cells. The CNN is associated with the mathematical operation of convolution for reducing matrix sizes.

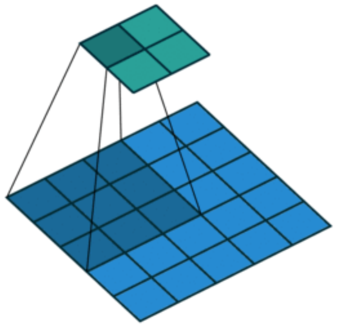
Дополнение / Padding

Группирование / Pooling

Шагание / Striding



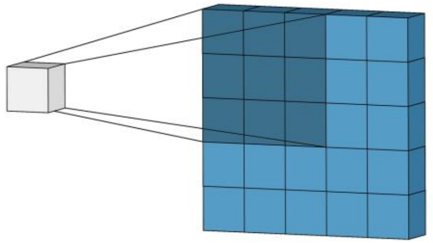




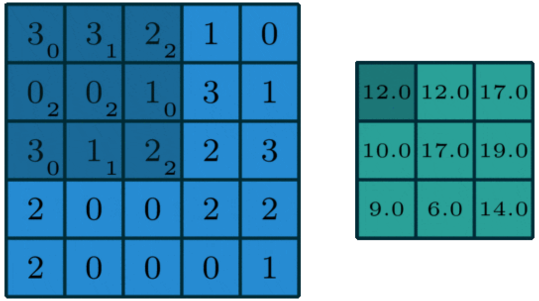
CNNs by Neurohive.io

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.



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

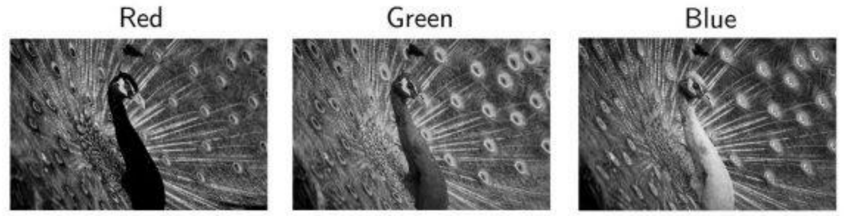


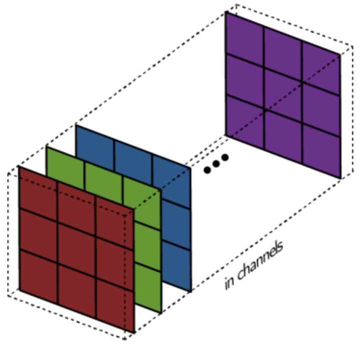
$$\Theta^{(1)} = \begin{pmatrix} 0&1&2 \\ 2&2&0 \\ 0&1&2 \end{pmatrix} \rightarrow \Theta^{(1)} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 2 \\ 2 \\ 0 \\ 0 \\ 1 \\ 2 \end{pmatrix}; \quad Z^{(2)} = A^{(1)}\Theta^{(1)} = \begin{pmatrix} 12 \\ 12 \\ 17 \\ 10 \\ 17 \\ 19 \\ 9 \\ 6 \\ 14 \end{pmatrix} \rightarrow Z^{(2)}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

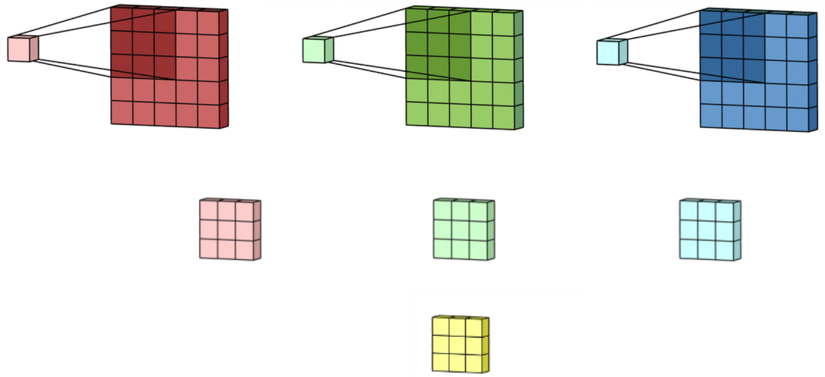
[CNNs by Neurohive.io]

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]:

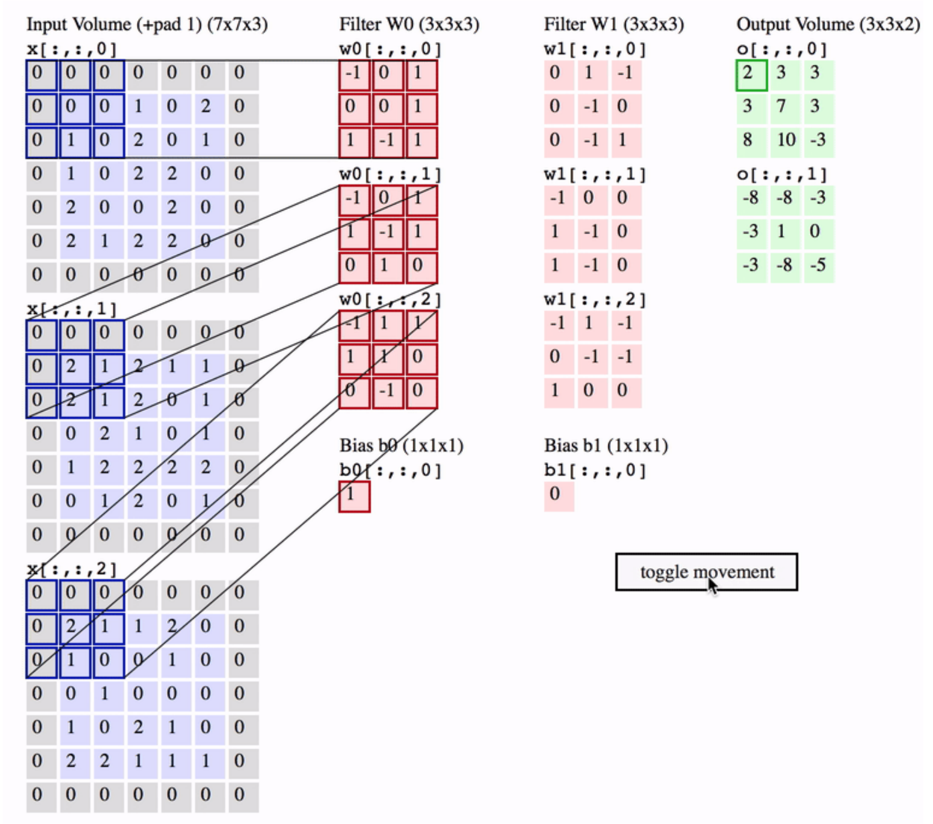
$$r = \frac{q-k+2p}{s} + 1 = \frac{5-3+2}{2} + 1 = 3.$$

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

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

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

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



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

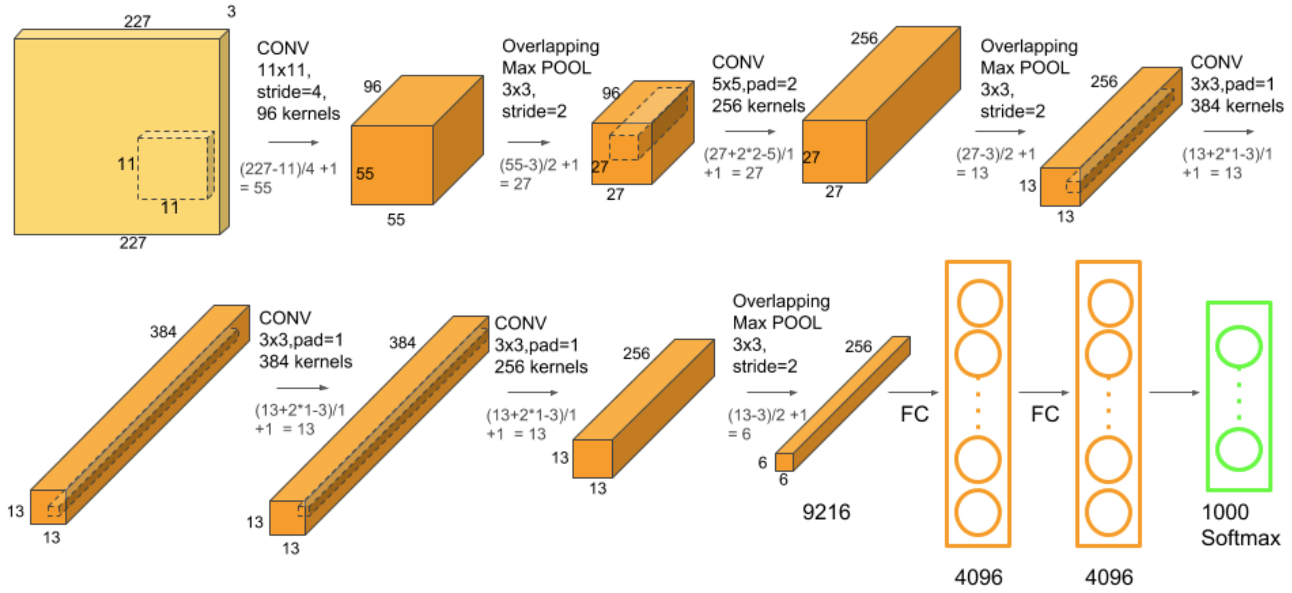
o[:,:,1]

-8	-8	-3
-3	1	0
-3	-8	-5

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.

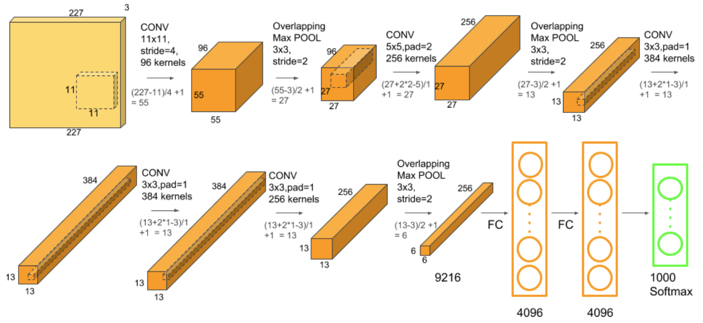


[AlexNet architecture]

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 $\boldsymbol{y}^{(i)}$.


227
3
11
11
227
CONV 11x11, stride=4, 96 kernels
(227-11)/4 +1 = 55
96
55
55
Overlapping Max POOL 3x3, stride=2
(55-3)/2 +1 = 27
96
27
27
CONV 5x5,pad=2 256 kernels
(27+2*2-5)/1 +1 = 27
256
27
27
Overlapping Max POOL 3x3, stride=2
(27-3)/2 +1 = 13
256
13
13
CONV 3x3,pad=1 384 kernels
(13+2*1-3)/1 +1 = 13
384
13
13
CONV 3x3,pad=1 384 kernels
(13+2*1-3)/1 +1 = 13
384
13
13
CONV 3x3,pad=1 256 kernels
(13+2*1-3)/1 +1 = 13
256
13
13
Overlapping Max POOL 3x3, stride=2
(13-3)/2 +1 = 6
256
6
6
9216
FC
4096
FC
4096
1000 Softmax

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

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

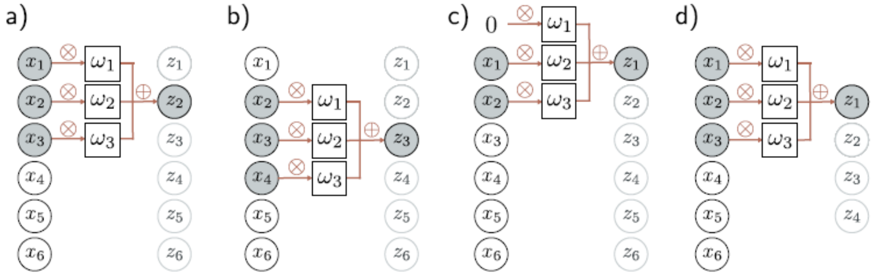
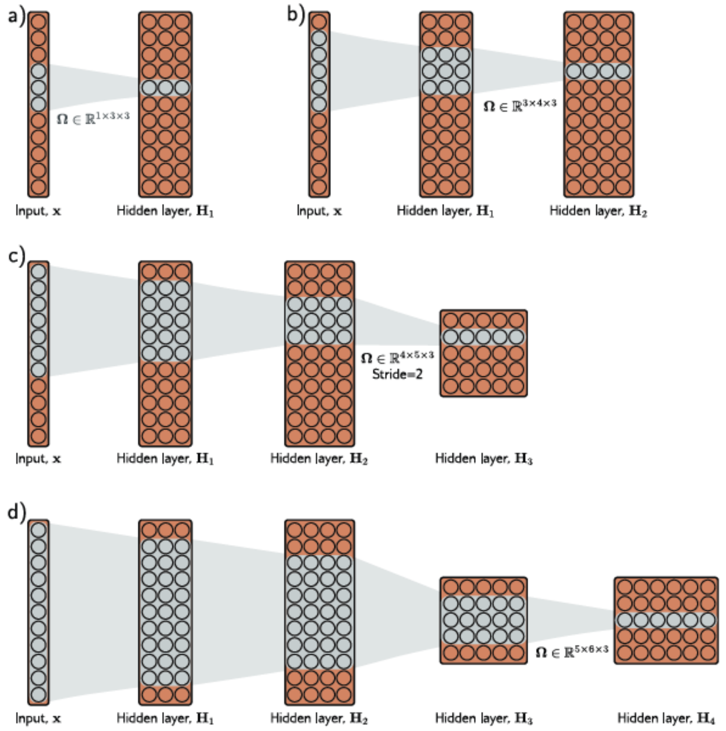
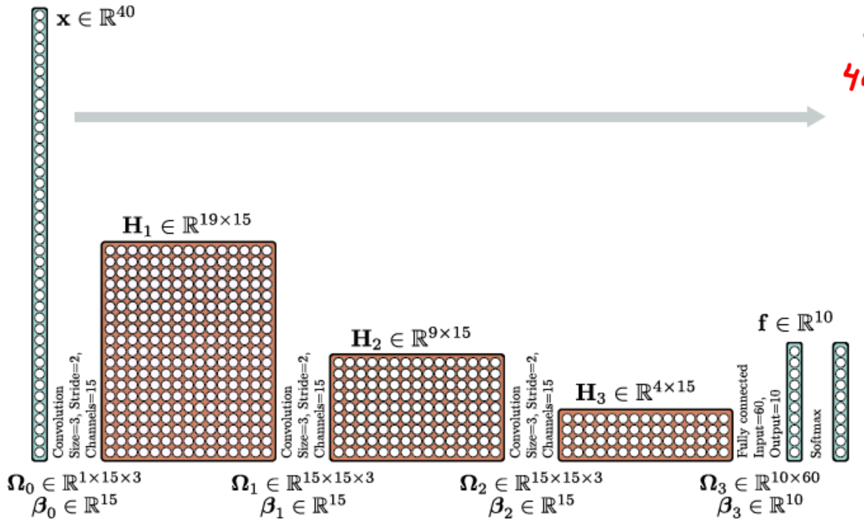

a)
b)
c)
d)
0
x_1 x_2 x_3 x_4 x_5 x_6
ω_1 ω_2 ω_3
z_1 z_2 z_3 z_4 z_5 z_6

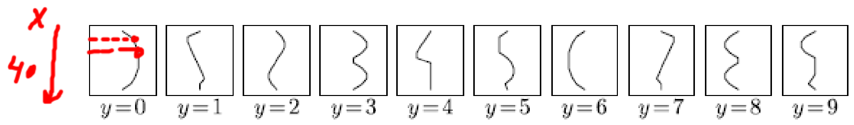
Figure 10.2 1D convolution with kernel size three. Each output z_i is a weighted sum of the nearest three inputs x_{i-1}, x_i, and x_{i+1}, where the weights are $\omega = [\omega_1, \omega_2, \omega_3]$. a) Output z_2 is computed as $z_2 = \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$. b) Output z_3 is 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.


a)
$\Omega \in \mathbb{R}^{1\times3\times3}$
Input, $\mathbf{x}$
Hidden layer, $\mathbf{H}_1$
b)
$\Omega \in \mathbb{R}^{3\times4\times3}$
Hidden layer, $\mathbf{H}_2$
c)
$\Omega \in \mathbb{R}^{4\times5\times3}$
Stride=2
Hidden layer, $\mathbf{H}_3$
d)
$\Omega \in \mathbb{R}^{5\times6\times3}$
Hidden layer, $\mathbf{H}_4$

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

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


$\mathbf{x} \in \mathbb{R}^{40}$
$\mathbf{H}_1 \in \mathbb{R}^{19\times 15}$
$\mathbf{H}_2 \in \mathbb{R}^{9\times 15}$
$\mathbf{H}_3 \in \mathbb{R}^{4\times 15}$
$\mathbf{f} \in \mathbb{R}^{10}$
Convolution Size=3, Stride=2, Channels=15
Convolution Size=3, Stride=2, Channels=15
Convolution Size=3, Stride=2, Channels=15
Fully connected Input=60, Output=10
Softmax
$\Omega_0 \in \mathbb{R}^{1\times 15\times 3}$
$\beta_0 \in \mathbb{R}^{15}$
$\Omega_1 \in \mathbb{R}^{15\times 15\times 3}$
$\beta_1 \in \mathbb{R}^{15}$
$\Omega_2 \in \mathbb{R}^{15\times 15\times 3}$
$\beta_2 \in \mathbb{R}^{15}$
$\Omega_3 \in \mathbb{R}^{10\times 60}$
$\beta_3 \in \mathbb{R}^{10}$



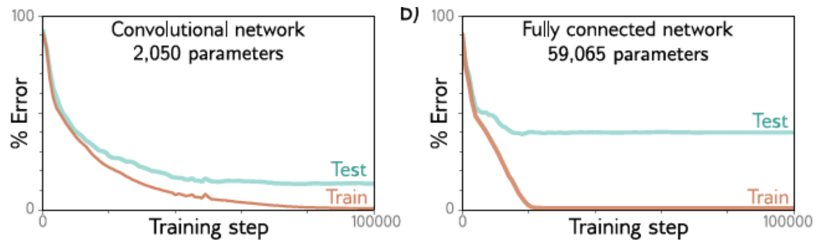


Figure 10.7 Convolutional network for classifying MNIST-1D data (see figure 8.1). The MNIST-1D input has dimension $D_i = 40$. The first convolutional layer has fifteen channels, kernel size three, stride two, and only retains "valid" positions to make a hidden layer with nineteen positions and fifteen channels. The following two convolutional layers have the same settings, gradually reducing the representation size at each subsequent hidden layer. Finally, a fully connected layer takes all sixty hidden units from the third hidden layer. It outputs ten activations that are subsequently passed through a softmax layer to produce the ten class probabilities.

Figure 10.8 MNIST-1D results. a) The convolutional network from figure 10.7 eventually fits the training data perfectly and has ~17% test error. b) A fully connected network with the same number of hidden layers and the number of hidden units in each learns the training data faster but fails to generalize well with ~40% test error. The latter model can reproduce the convolutional model but fails to do so. The convolutional structure restricts the possible mappings to those that process every position similarly, and this restriction improves performance.

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

What if we add some more layers in a CNN?

$$\begin{tikzcd} x \arrow[r] & \boxed{f_1} \arrow[r] & \boxed{f_2} \arrow[r] & \boxed{f_3} \arrow[r] & \boxed{f_4} \arrow[r] & h \end{tikzcd}$$

Sequential processing intuition. Standard neural networks pass the output of each layer directly into the next layer

$$h = f_4\Big(f_3\big(f_2(f_1(x))\big)\Big),$$
$$\frac{\partial h}{\partial f_1} = \frac{\partial f_4}{\partial f_3}\frac{\partial f_3}{\partial f_2}\frac{\partial f_2}{\partial f_1}.$$

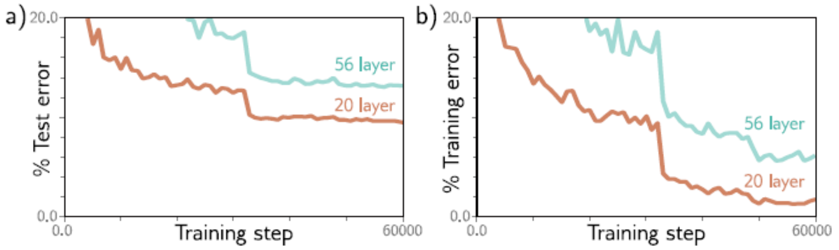
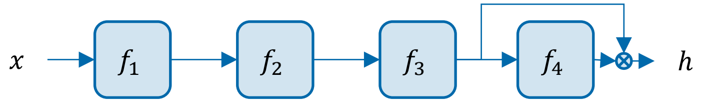


Figure 11.2 Decrease in performance when adding more convolutional layers. a) A 20-layer convolutional network outperforms a 56-layer neural network for image classification on the test set of the CIFAR-10 dataset (Krizhevsky & Hinton, 2009). b) This is also true for the training set, which suggests that the problem relates to training the original network rather than a failure to generalize to new data. Adapted from He et al. (2016a).

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

ML-2024
by Alexei Kornaev

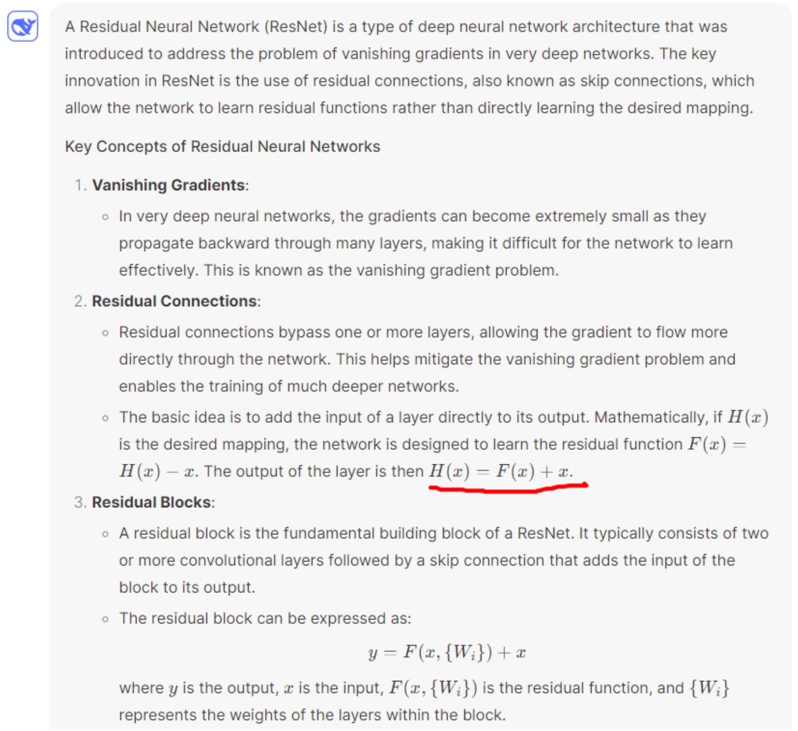
Residual neural networks are the CNNs of an upper level



Sequential processing intuition. Standard neural networks pass the output of each layer directly into the next layer

$$h = f_4\left(f_3\left(f_2(f_1(x))\right)\right) + f_3\left(f_2(f_1(x))\right),$$
$$\frac{\partial h}{\partial f_1} = \frac{\partial f_4}{\partial f_3}\frac{\partial f_3}{\partial f_2}\frac{\partial f_2}{\partial f_1} + \frac{\partial f_3}{\partial f_2}\frac{\partial f_2}{\partial f_1}.$$

Generated by DeepSeek-V2.5



A Residual Neural Network (ResNet) is a type of deep neural network architecture that was introduced to address the problem of vanishing gradients in very deep networks. The key innovation in ResNet is the use of residual connections, also known as skip connections, which allow the network to learn residual functions rather than directly learning the desired mapping.

Key Concepts of Residual Neural Networks

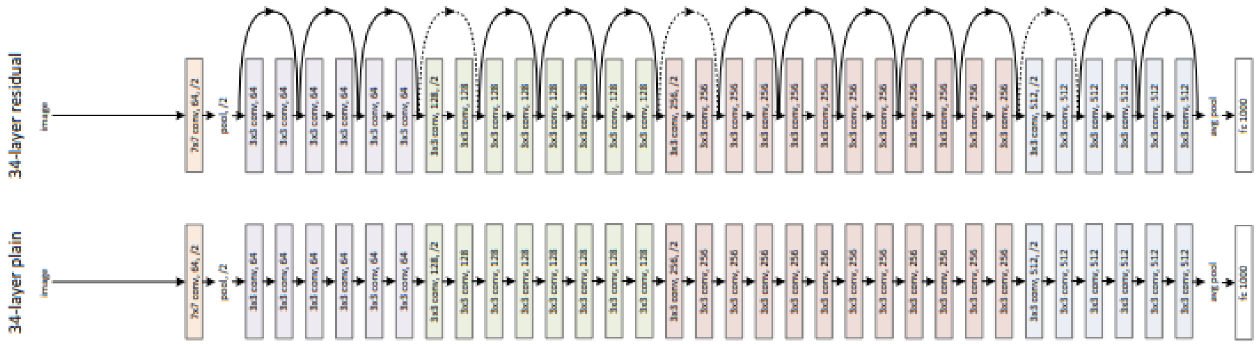
1. **Vanishing Gradients**:
   - In very deep neural networks, the gradients can become extremely small as they propagate backward through many layers, making it difficult for the network to learn effectively. This is known as the vanishing gradient problem.
2. **Residual Connections**:
   - Residual connections bypass one or more layers, allowing the gradient to flow more directly through the network. This helps mitigate the vanishing gradient problem and enables the training of much deeper networks.
   - The basic idea is to add the input of a layer directly to its output. Mathematically, if $H(x)$ is the desired mapping, the network is designed to learn the residual function $F(x) = H(x) - x$. The output of the layer is then $\underline{H(x) = F(x) + x}$.
3. **Residual Blocks**:
   - A residual block is the fundamental building block of a ResNet. It typically consists of two or more convolutional layers followed by a skip connection that adds the input of the block to its output.
   - The residual block can be expressed as:

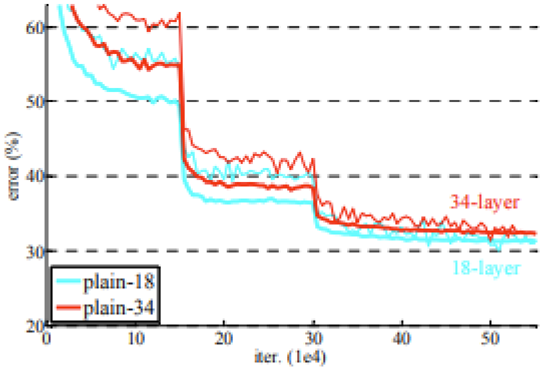
     $$y = F(x, \{W_i\}) + x$$

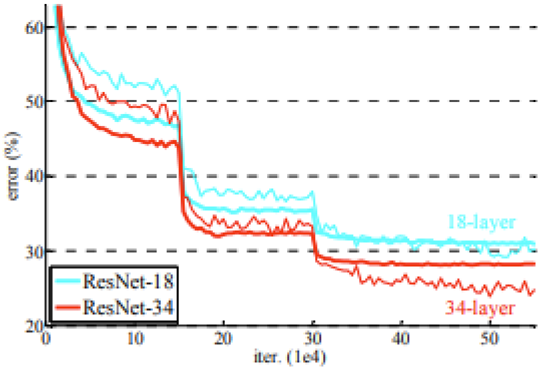
     where y is the output, x is the input, $F(x, \{W_i\})$ is the residual function, and $\{W_i\}$ represents the weights of the layers within the block.

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

Residual neural networks are the CNNs of an upper level







[Deep Residual Learning for Image Recognition, CVPR, 2015]

Residual neural networks are the CNNs of an upper level

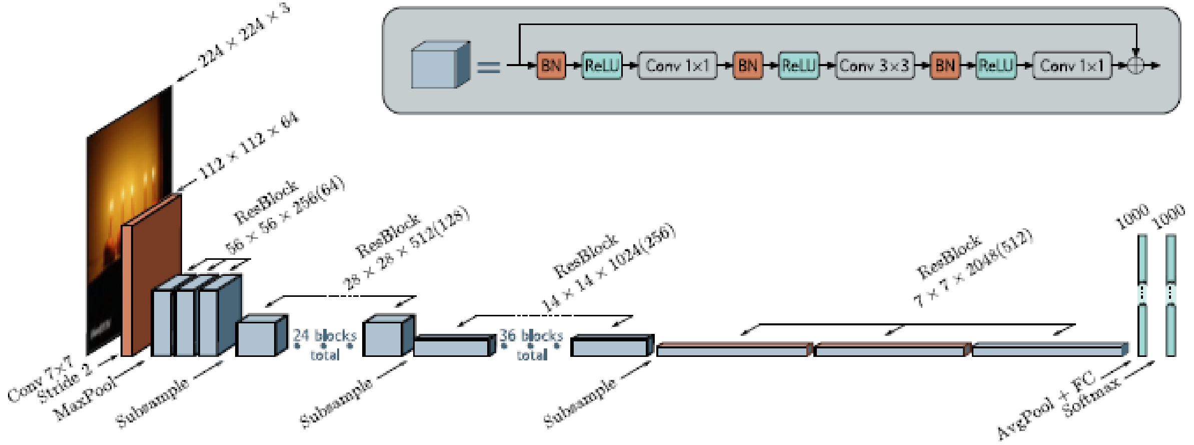


Figure 11.8 ResNet-200 model. A standard 7×7 convolutional layer with stride two is applied, followed by a MaxPool operation. A series of bottleneck residual blocks follow (number in brackets is channels after first 1×1 convolution), with periodic downsampling and accompanying increases in the number of channels. The network concludes with average pooling across all spatial positions and a fully connected layer that maps to pre-softmax activations.

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

Residual neural networks are the CNNs of an upper level

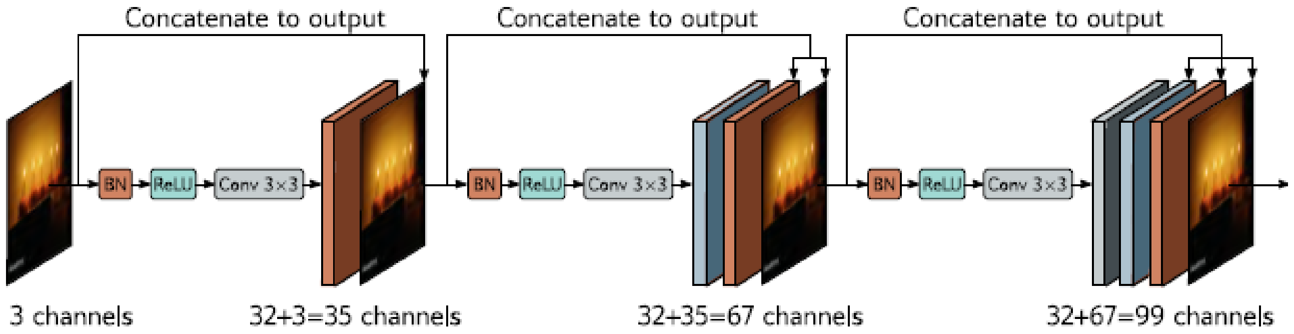
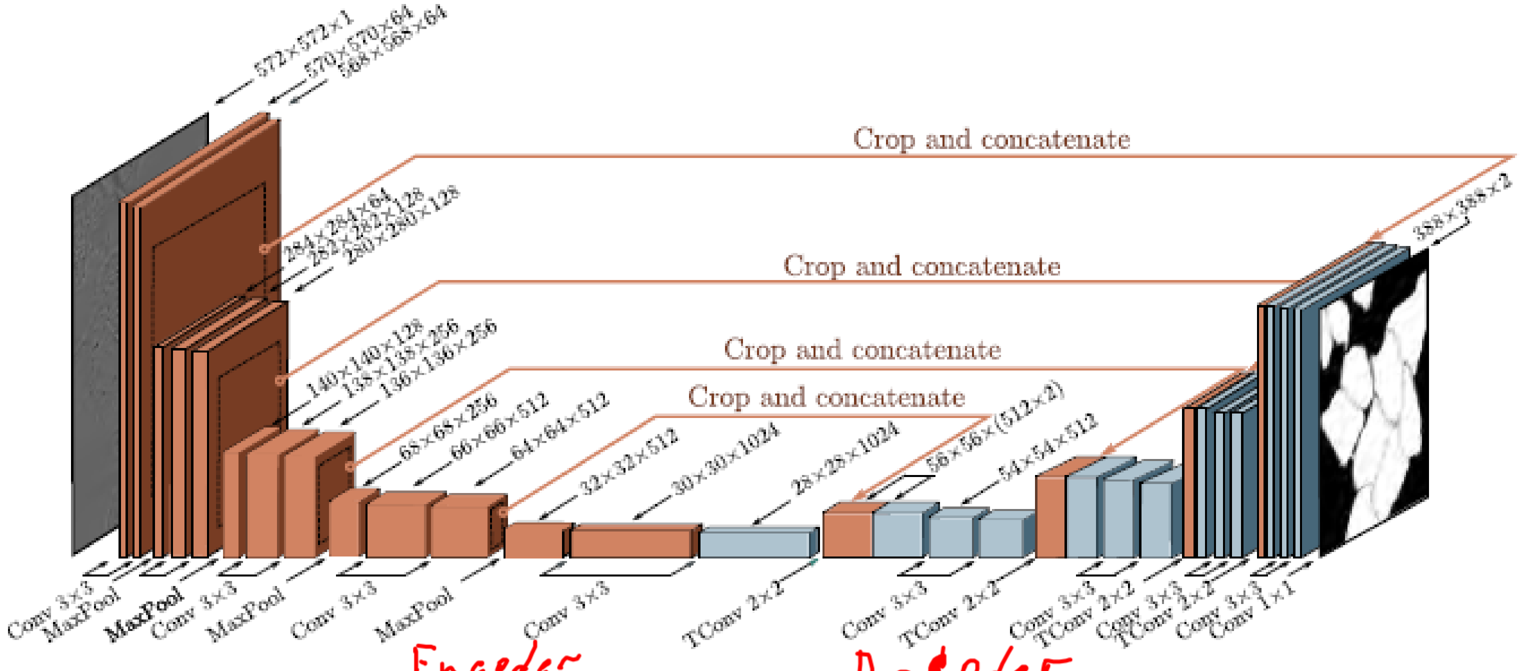


Figure 11.9 DenseNet. This architecture uses residual connections to concatenate the outputs of earlier layers to later ones. Here, the three-channel input image is processed to form a 32-channel representation. The input image is concatenated to this to give a total of 35 channels. This combined representation is processed to create another 32-channel representation, and both earlier representations are concatenated to this to create a total of 67 channels and so on.

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

Residual neural networks are the CNNs of an upper level



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

Why residual connections help?

- allow deeper networks to be trained;
- reduces shattered gradients at the start of training;
- smooth the loss surface near the minimum;
- work like an ensemble of shorter networks.

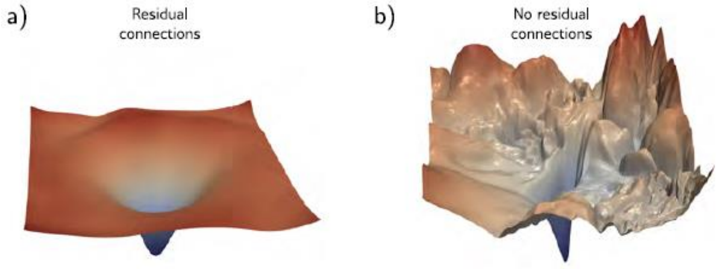


Figure 11.13 Visualizing neural network loss surfaces. Each plot shows the loss surface in two random directions in parameter space around the minimum found by SGD for an image classification task on the CIFAR-10 dataset. These directions are normalized to facilitate side-by-side comparison. a) Residual net with 56 layers. b) Results from the same network without skip connections. The surface is smoother with the skip connections. This facilitates learning and makes the final network performance more robust to minor errors in the parameters, so it will likely generalize better. Adapted from Li et al. (2018b).

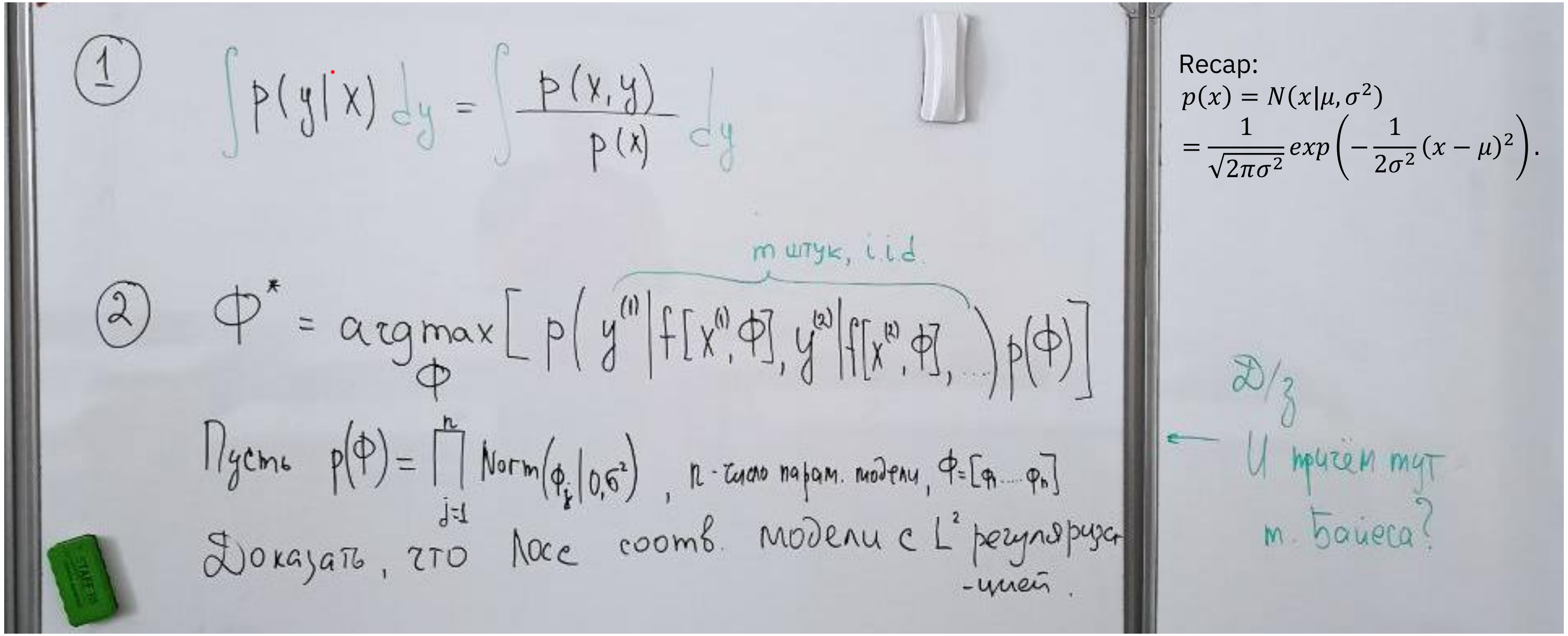
S. J. Prince. Understanding Deep Learning. MIT Press, 2023.

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



1. $$\int p(y|x)\,dy = \int \frac{p(x,y)}{p(x)}\,dy$$

2. $$\Phi^* = \underset{\Phi}{\arg\max}\Big[\, p\big(\overbrace{y^{(1)}|f[x^{(1)},\Phi],\ y^{(2)}|f[x^{(2)},\Phi], \ldots}^{m \text{ штук, i.i.d}}\big)\, p(\Phi) \Big]$$

Пусть $p(\Phi) = \prod_{j=1}^{n} Norm(\phi_j|0,\sigma^2)$, n - число парам. модели, $\Phi = [\phi_1 \ldots \phi_n]$

Доказать, что Loss соотв. модели с L^2 регуляриза-цией.

Д/з

← И причём тут т. Байеса?

Recap:
$$p(x) = N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right).$$