بینایی کامپیوتر
فصل دوم
شبکه‌های عصبی کانولوشنی (پیچشی)
مفاهیم پایه
CONVOLUTIONAL NEURAL NETWORK(CNN)

محمدجواد فدائی اسلام

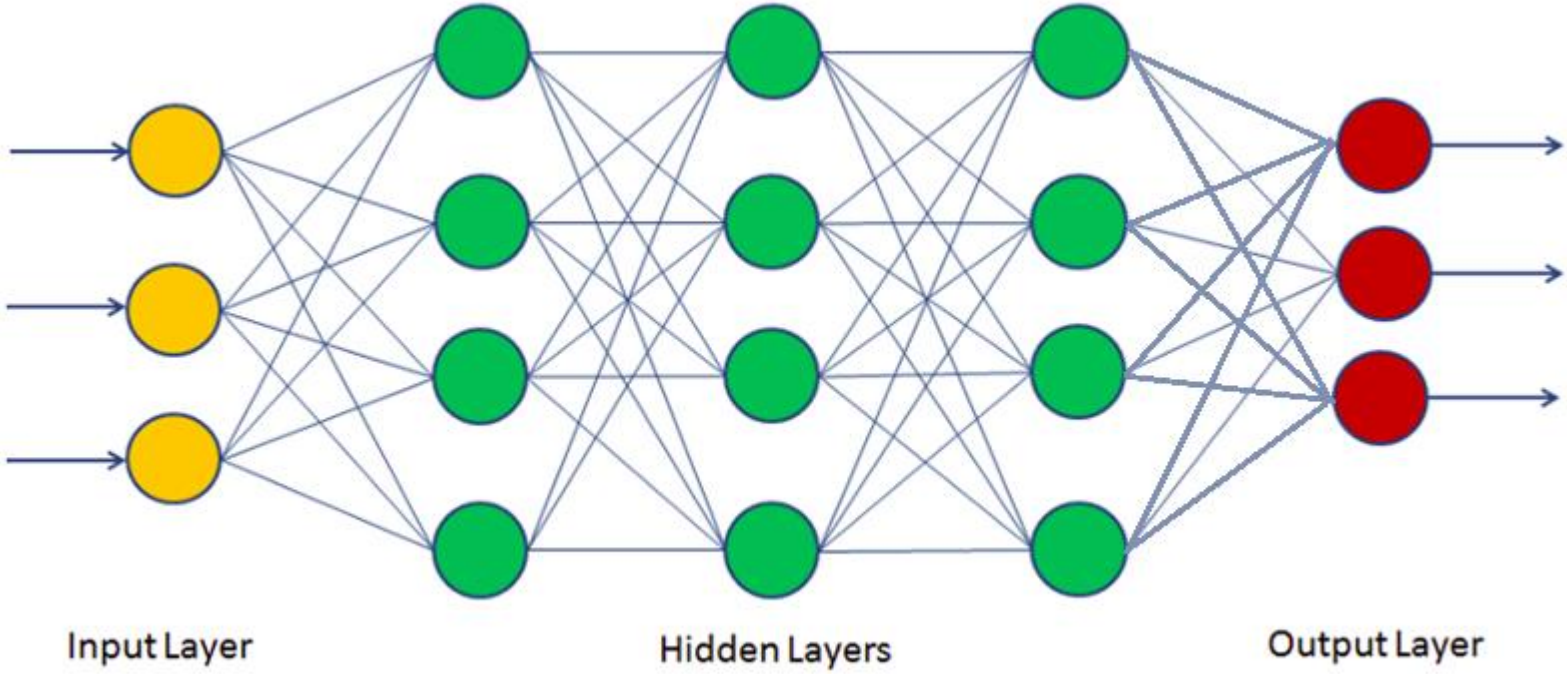
ویرایش بهمن ۱۴۰۳

یادگیری عمیق (تعریف)
DEEP LEARNING (DEFINITION)

- تا قبل از ظهور AlexNet، یا به عبارتی یادگیری عمیق در سال ۲۰۱۲، بیشتر تکنیک‌های یادگیری ماشین و پردازش سیگنال از معماری‌های کم‌عمق بهره‌برداری می‌کردند. این معماری‌ها معمولاً حاوی حداکثر یک یا دو لایه مخفی بودند[۳].

- معماری که چندین لایه پنهان را پشت سرهم قرار دهد، یادگیری عمیق نامیده می‌شود [۲].

شبکه پرسپترون چندلایه
شبکه کاملا متصل با معماری کم عمق



ضعف شبکه‌های عصبی کاملا متصل

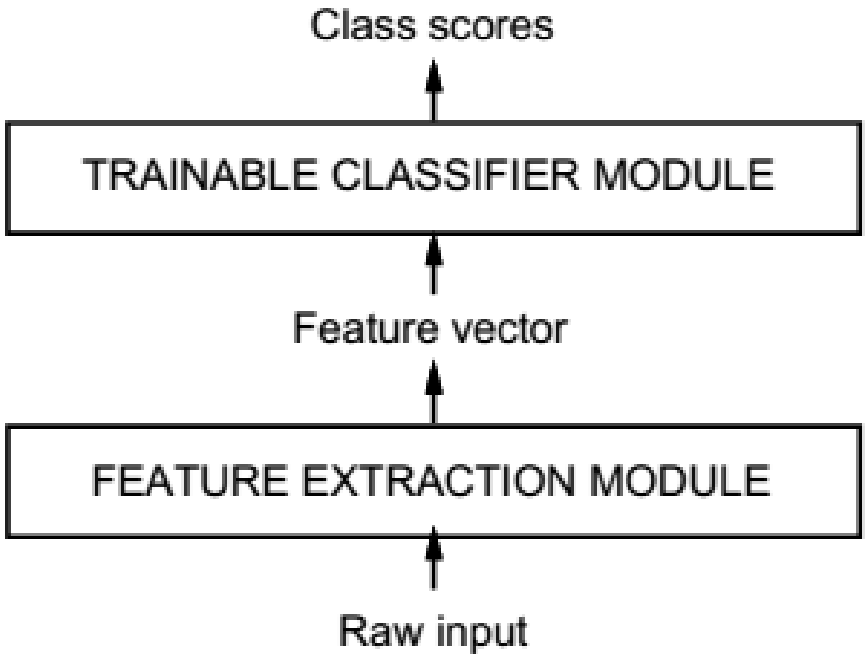
- تعداد زیادی وزن در شبکه عصبی کاملا متصل وجود دارد، هر نرون به تمام نرون‌های لایه قبل متصل است.

- با افزودن لایه‌های مخفی، عمق شبکه افزایش می‌یابد؛ الگوریتم پس‌انتشار خطا که روشی برای آموزش MLP است؛ بر اساس اطلاعات گرادیان محلی عمل می‌نماید و در این شرایط، اغلب در ماکزیمم های محلی ضعیف به دام می‌افتد.

- پیچیدگی زمانی این روش در صورت افزایش عمق، بسیار زیاد می‌شود [۲،۳].

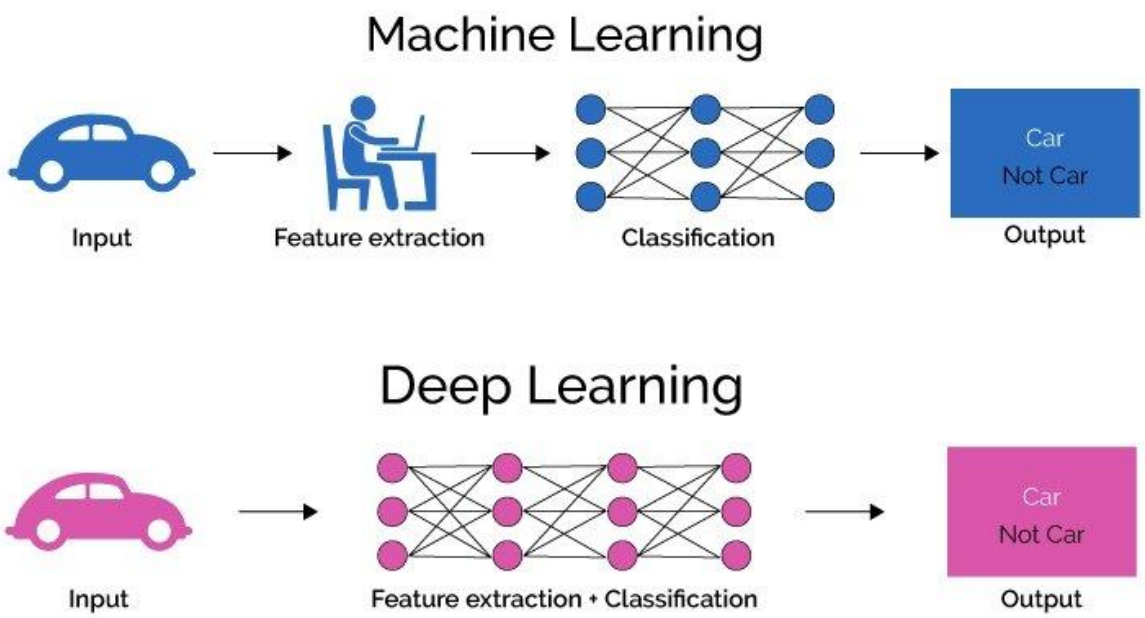
راه حل کلاسیک برای غلبه بر ضعف شبکه‌های عصبی کاملا متصل

- برای کاهش پیچیدگی زمانی، در روش‌های مرسوم، تشخیص الگو با دو ماژول انجام می‌شود: یک استخراج ویژگی ثابت و یک طبقه‌بندی قابل آموزش [1].

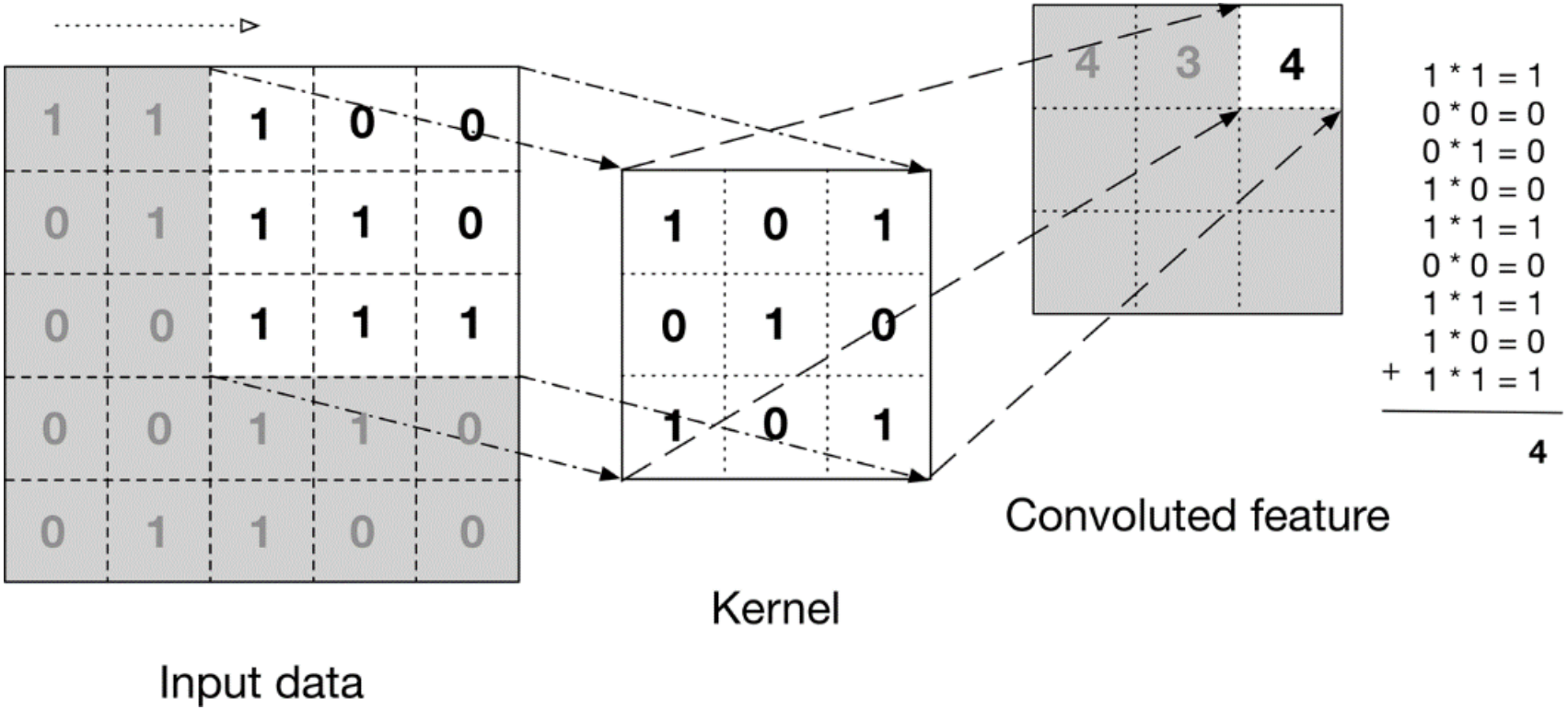


ایده شبکه‌های عصبی کانولوشنی (پیچشی)
CONVOLUTIONAL NEURAL NETWORK

- شبکه‌های عصبی کانولوشنی (پیچشی) یک روش یادگیری عمیق باناظر هستند.
- در سال ۱۹۸۹، آقای LeCun و همکارانش در مقاله خود ادعا نمودند که سیستم‌های تشخیص الگوی بهتری می‌توان با تکیه بیشتر بر یادگیری خودکار و کمتر بر روی اکتشافات طراحی شده با دست ساخت [۱]. به عبارت ساده‌تر ویژگی هم به صورت خودکار استخراج شود.
- در معماری آنها ورودی شبکه عصبی کانولوشنی، تصویر (خام) است.



کانولوشن


1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
1 0 1
0 1 0
1 0 1
4 3 4
1 * 1 = 1
0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1
0 * 0 = 0
1 * 1 = 1
1 * 0 = 0
+ 1 * 1 = 1
4
Convoluted feature
Kernel
Input data

مثال
کرنل (فیلتر) سوبل برای کشف لبه
EDGE DETECTION USING SOBEL FILTER

-1	0	+1
-2	0	+2
-1	0	+1

کشف لبه های عمودی

3	1	4	2
3	2	5	1
2	3	4	5
3	4	5	6

	7		

Example "receptive field"

$$(-1*3)+(1*4)+(-2*3)+(2*5)+(-1*2)+(1*4) = 7$$

+1	+2	+1
0	0	0
-1	-2	-1

کشف لبه‌های افقی

3	1	4	2
3	2	5	1
2	3	4	5
3	4	5	6

	-3		

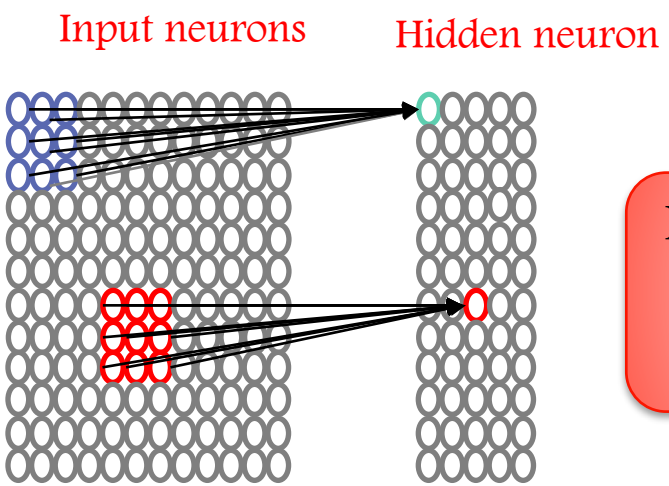
Example "receptive field"

$$(1*3)+(2*1)+(1*4)+(-1*2)+(-2*3)+(-1*4) = -3$$

مثال
کرنل (فیلتر) سوبل برای کشف لبه

معماری شبکه‌های کانولوشنی
CNN ARCHITECTURE

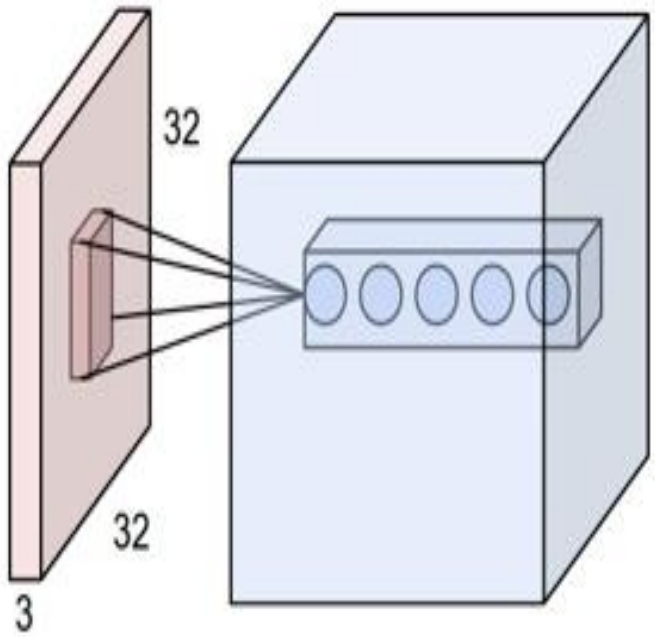
- از سه نوع لایه تشکیل شده‌اند:
- لایه کانولوشن
- لایه ادغام، لایه نمونه‌برداری (Pooling layer, Subsampling layer)
- لایه کاملا متصل (Fully-connected layer)
- برخلاف شبکه‌های عصبی مرسوم، نورون‌های درون لایه‌های کانولوشن و ادغام فقط به ناحیه کوچکی از لایه قبل از آن متصل می‌شوند [۲]؛ این ناحیه receptive filed نام دارد.



Every hidden layer neuron has a local receptive field of region 3×3 pixels

Convolutional Layer --Depth

- هر (کرنل) فیلتر کانولوشن یک لایه ایجاد می‌کند که به تعداد آنها عمق لایه کانولوشن می‌گویند.



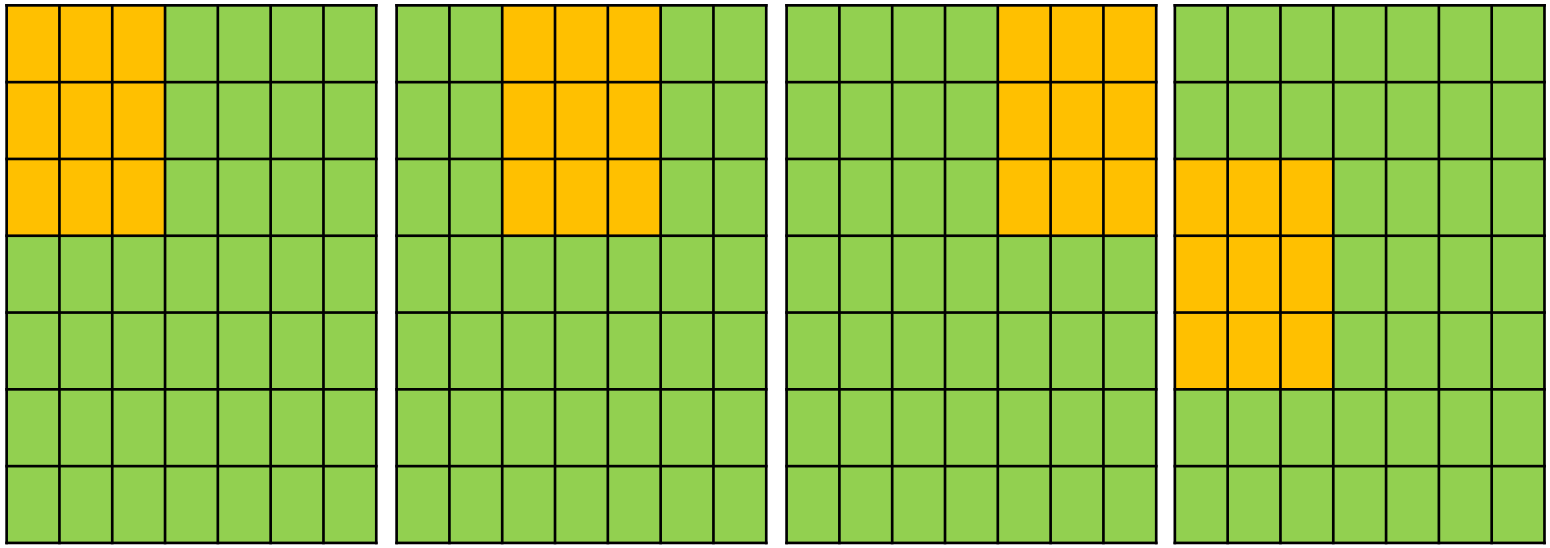
Depth = 5

گام حرکت
STRIDE

ما باید گامی را که با آن فیلتر را می‌لغزانیم مشخص کنیم.

وقتی گام ۱ باشد، فیلتر را هر بار یک پیکسل حرکت می‌دهیم.

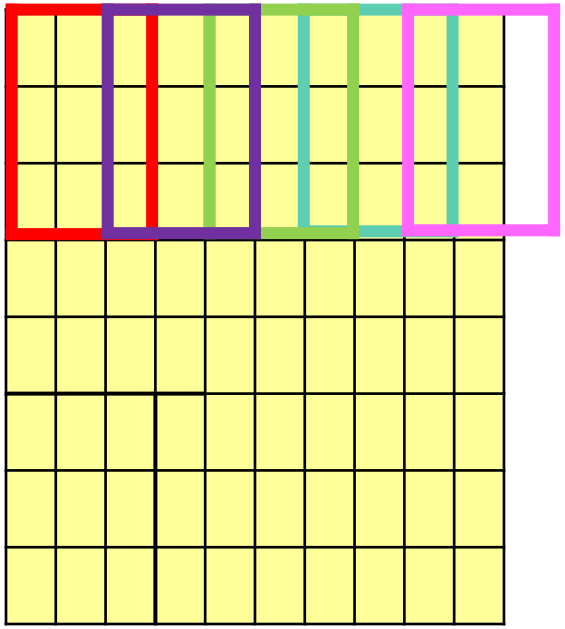
هنگامی که گام ۲ باشد، فیلتر ۲ پیکسل پرش می‌کند.



Stride = 2

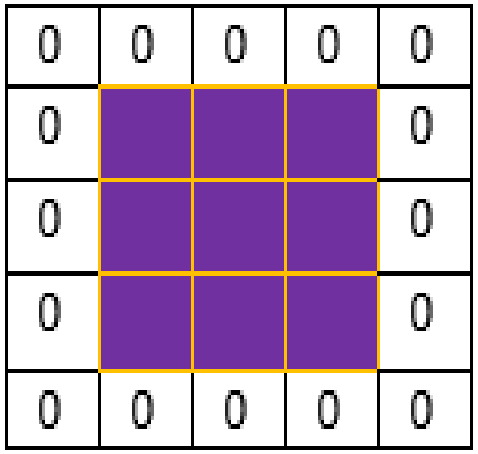
محدودیت روی گام حرکت

When the input has size W=10,
zero-padding P=0,
filter size F=3,
then it would be impossible to use stride S=2.
$(W-F+2P)/S+1=(10-3+0)/2+1=4.5$



لایه‌گذاری صفر
ZERO-PADDING

- گاهی اوقات مناسب است که اطراف ورودی را با صفر پوشش دهیم.
- این تکنیک به ما امکان می دهد اندازه خروجی را کنترل کنیم.
- ما از آن برای حفظ اندازه استفاده می کنیم تا عرض و ارتفاع ورودی و خروجی، یکسان باشد.

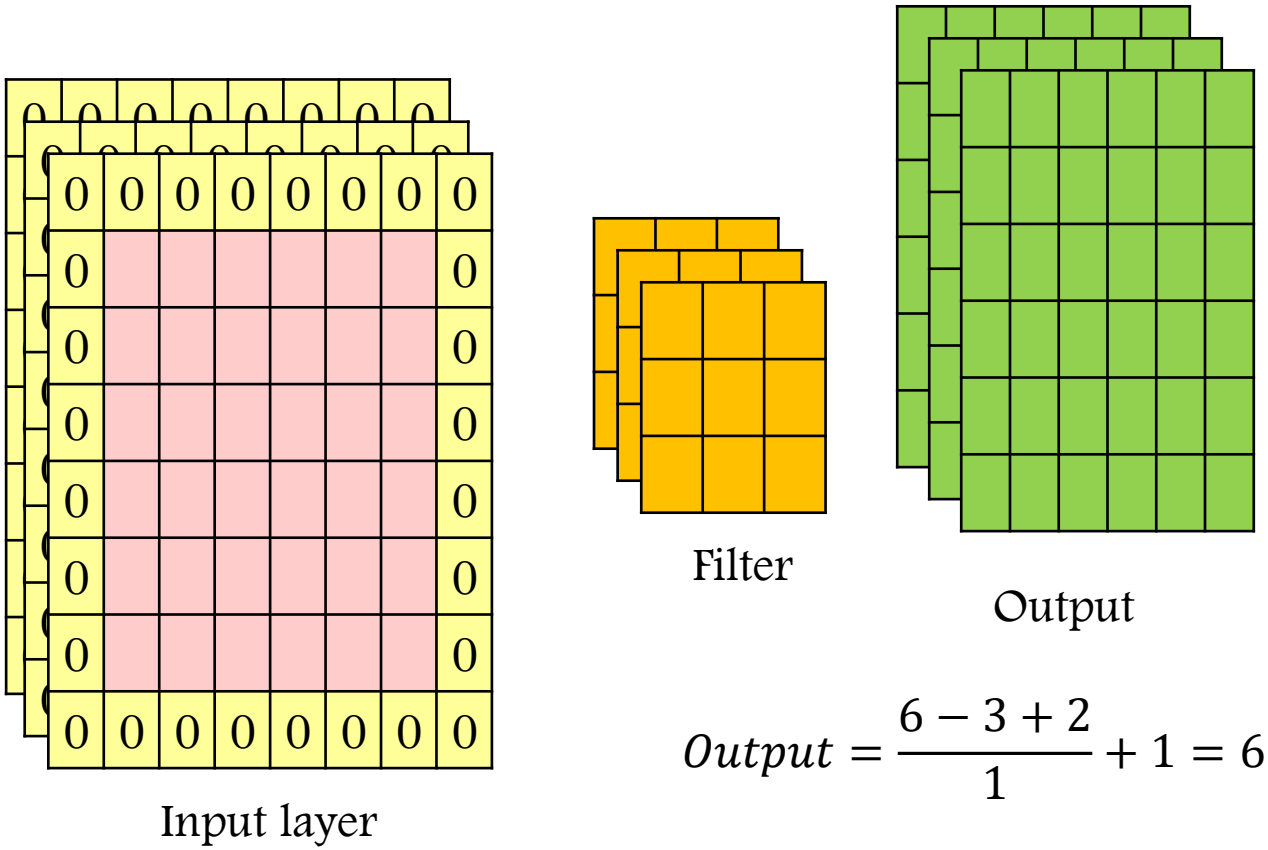


محاسبه اندازه خروجی

$$\frac{(w - F + 2P)}{S} + 1$$

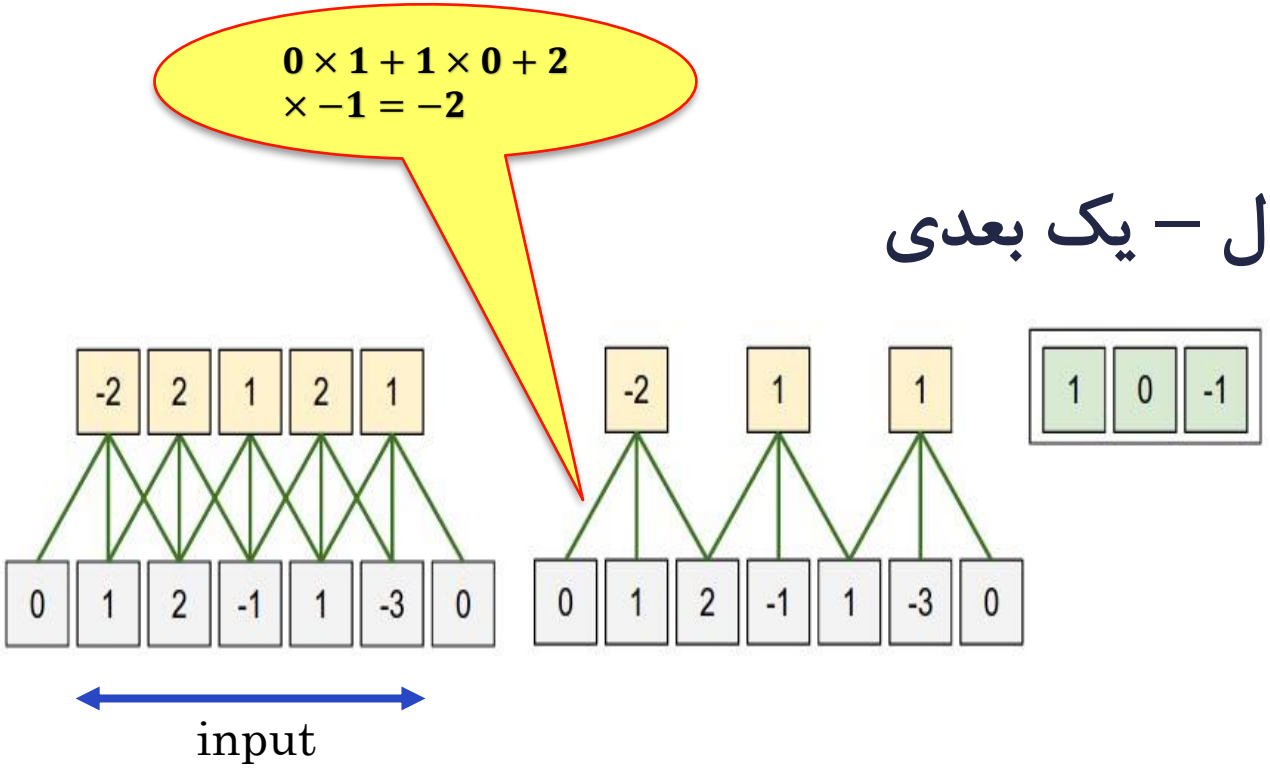
- W: the input volume size
- F: the receptive field size of the Conv Layer neurons
- S: the stride with which they are applied
- P: the amount of zero padding

مثال



$$Output = \frac{6 - 3 + 2}{1} + 1 = 6$$

مثال – یک بعدی



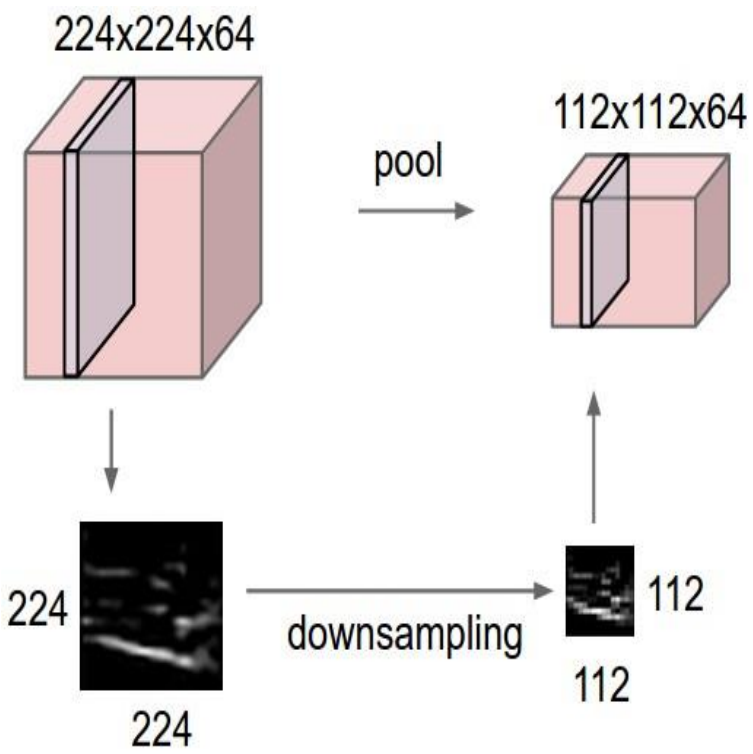
Output size (Left): (5 - 3 +2)/1+1=5.

Output size (Right): (5 - 3 + 2)/2+1 = 3

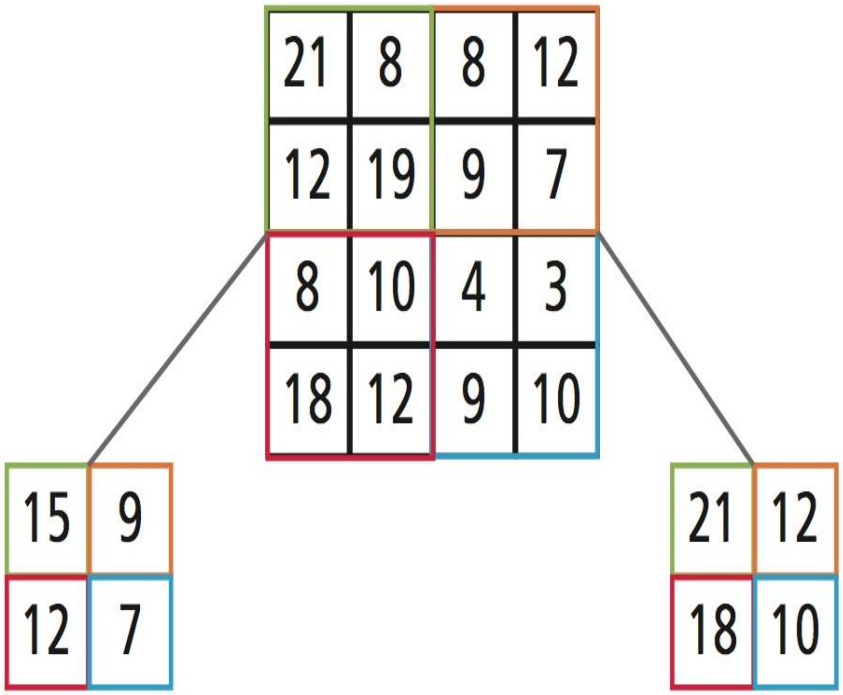
لایه ادغام

POOLING LAYER/SUBSAMPLING LAYERS

- برای کاهش اندازه فضایی، پارامترها و محاسبات لایه ادغام بین دو لایه کانولوشن قرار می‌دهند.
- این روش، ویژگی‌ها را در برابر نویز و اعوجاج قوی می‌کند.
- لایه ادغام برای هر عمق به طور مستقل عمل می‌کند.
- عمق ۶۴ در ورودی و خروجی یکسان است.



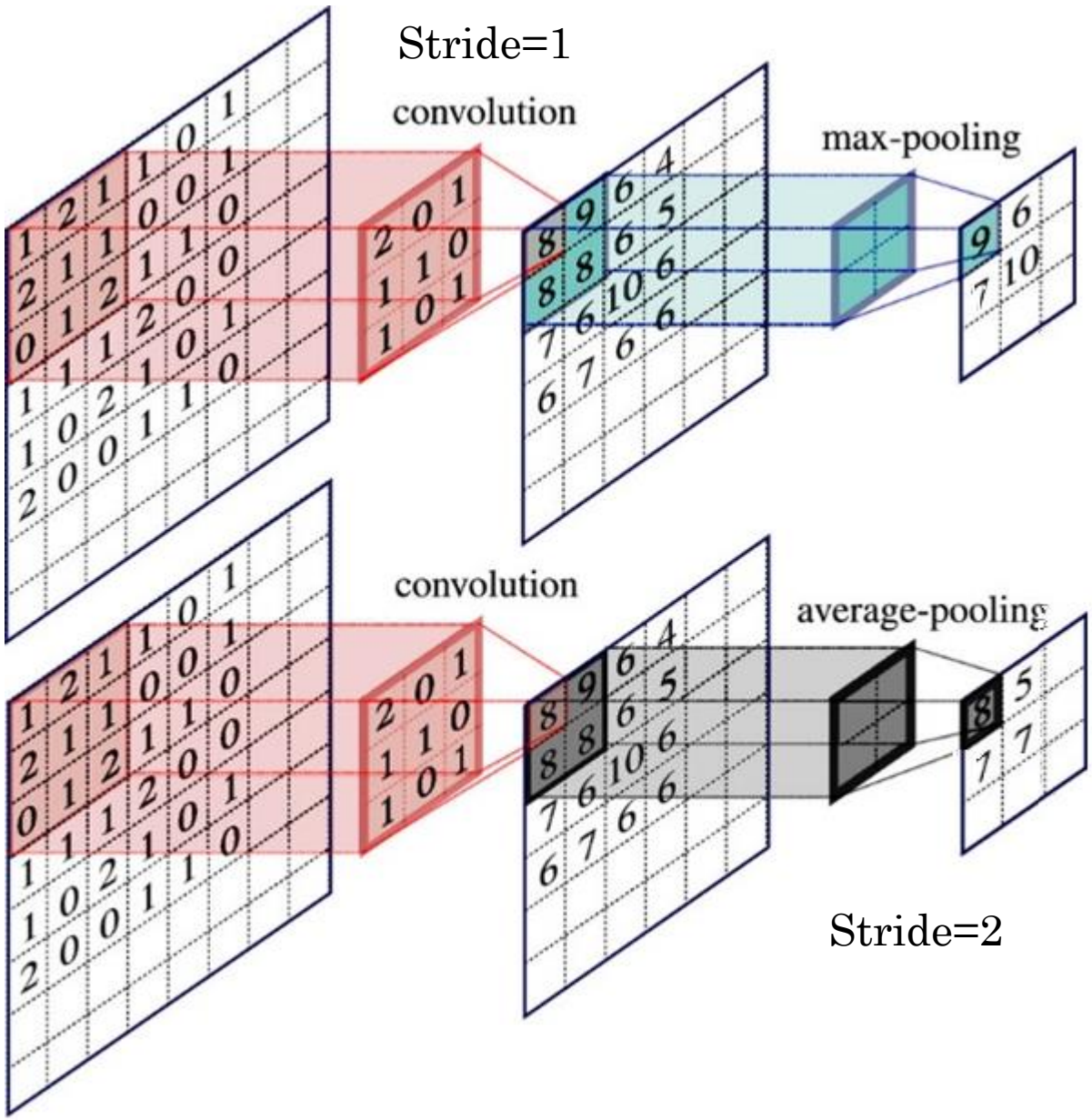
انواع لایه‌های ادغام



Average pooling

Max pooling

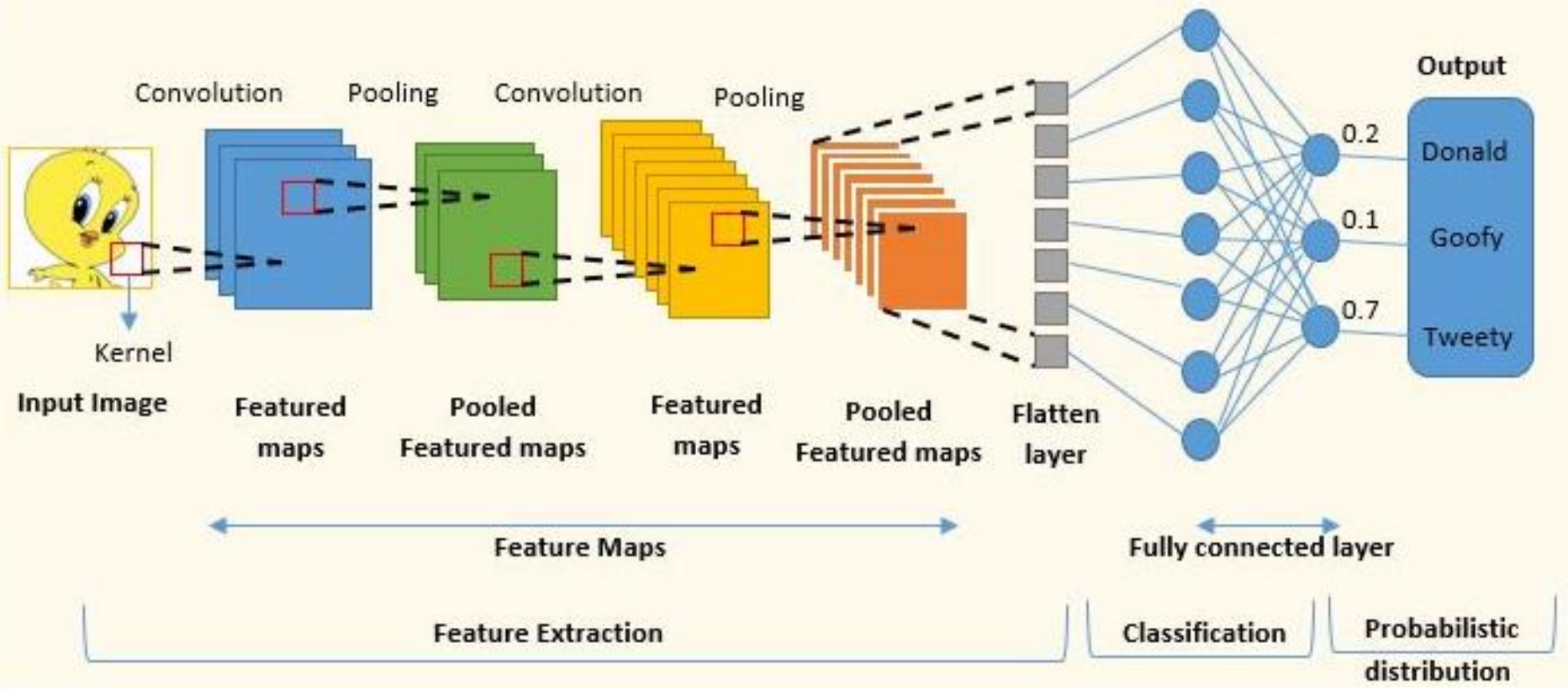
مثالی از لایه کانولوشن و ادغام



لایه تمام متصل
FULLY-CONNECTED LAYER

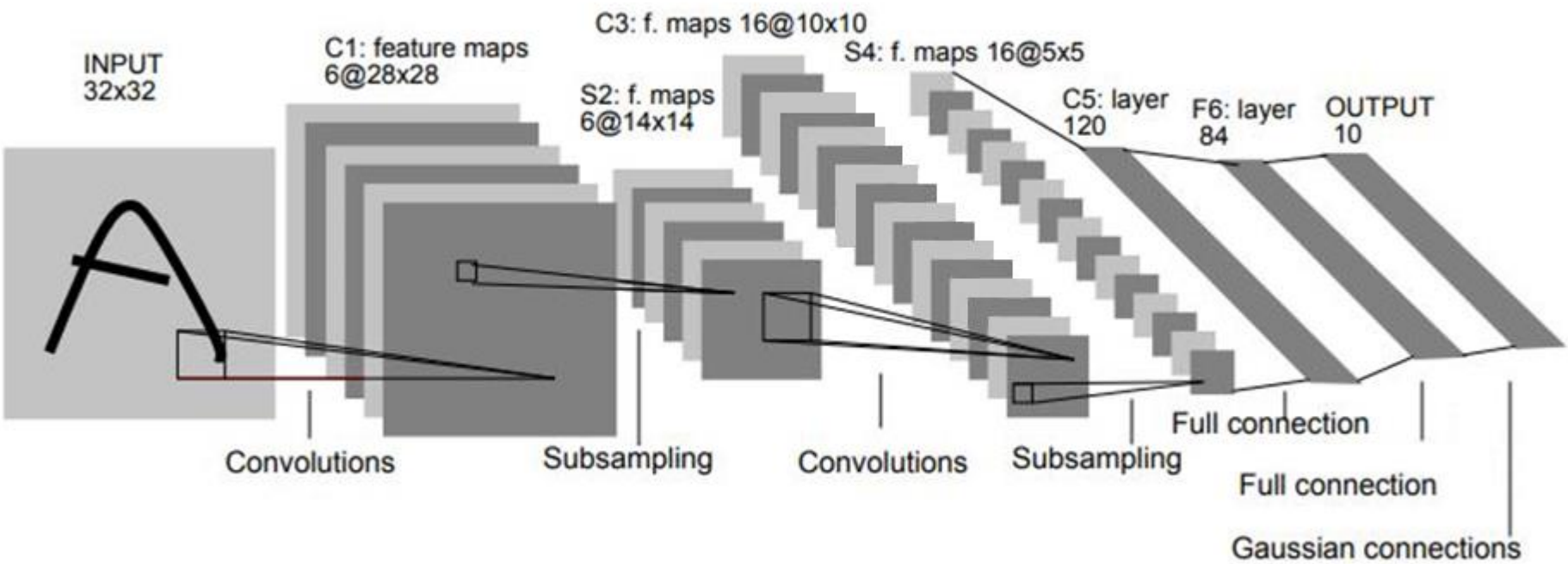
- آنها اغلب به عنوان لایه نهایی معماری کانولوشن استفاده می‌شوند.
- هر نرون در این لایه به تمام نرون‌های لایه قبل متصل است.
- لایه کانولوشن را می‌توان به عنوان یک لایه استخراج ویژگی در نظر گرفت. هر نرون در این لایه، در بر دارنده یک ویژگی است که لایه تمام متصل از جمع وزن‌دار آنها برای کلاس‌بندی بهره می‌برد.

لایه تمام متصل
FULLY-CONNECTED LAYER

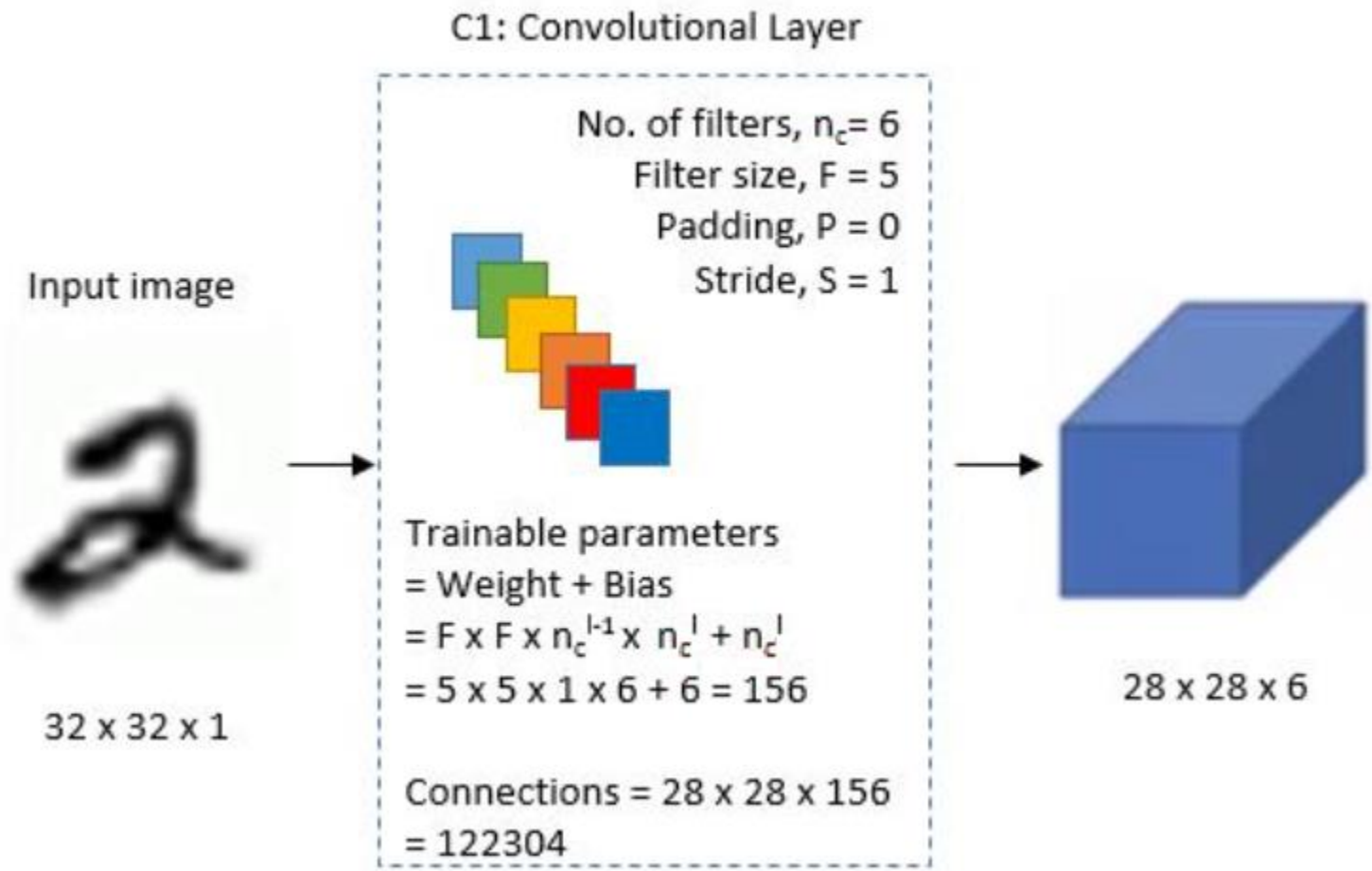


LeNet-5 – A Classic CNN Architecture

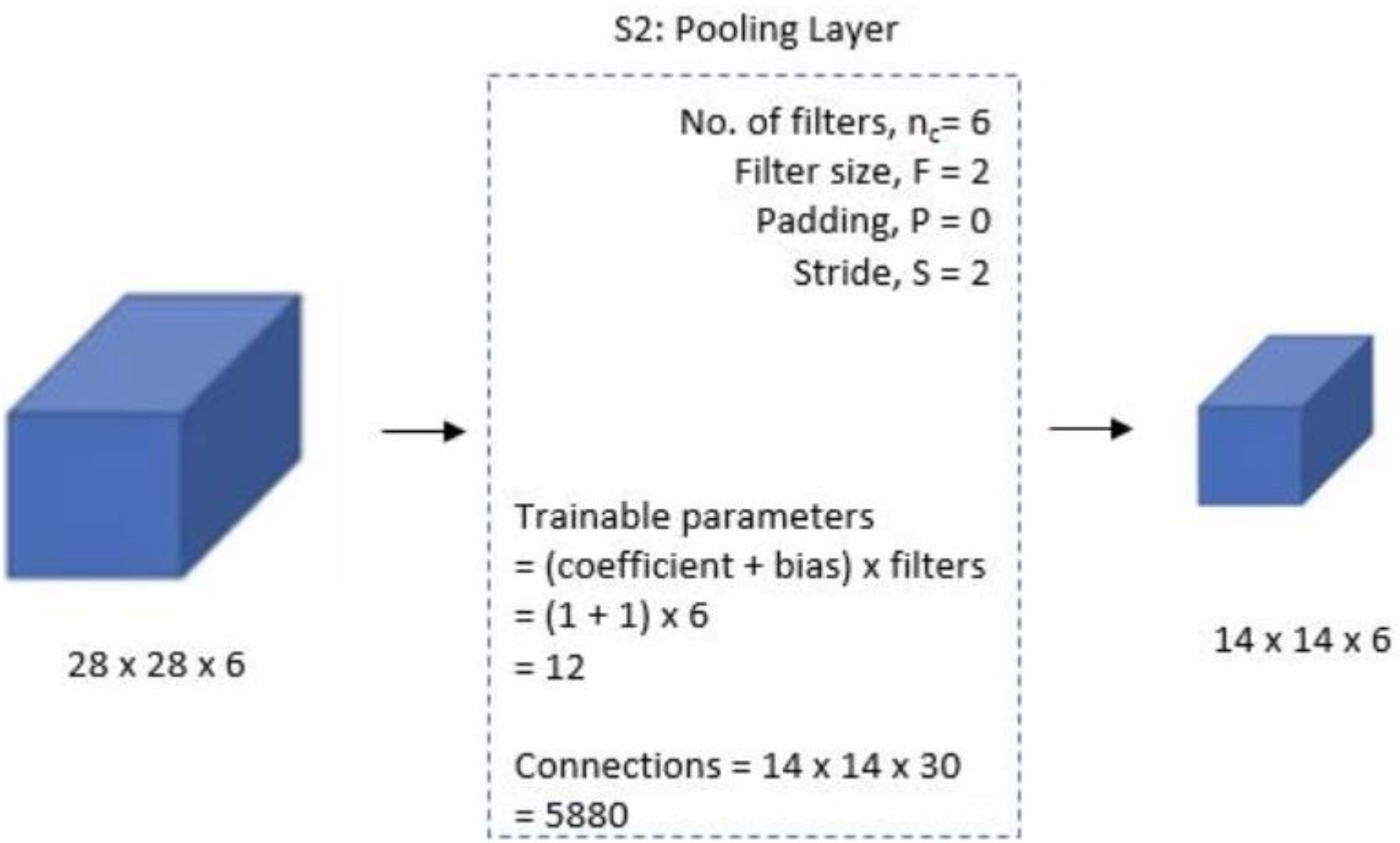
- Yann LeCun et al., proposed a CNN for handwritten and machine-printed character recognition in 1990's which they called LeNet-5.



C1

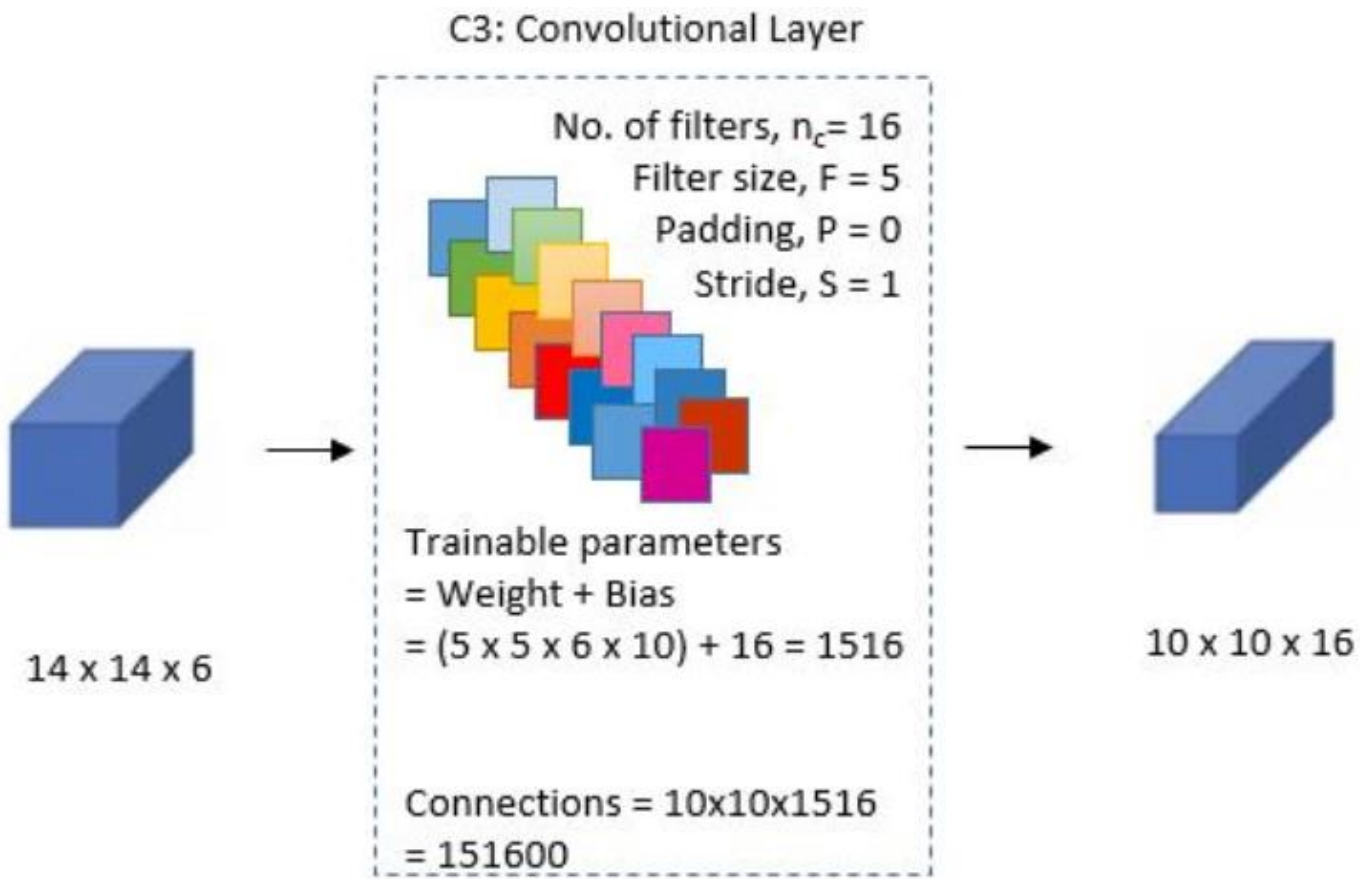

C1: Convolutional Layer
No. of filters, n_c= 6
Filter size, F = 5
Padding, P = 0
Stride, S = 1
Input image
32 x 32 x 1
Trainable parameters
= Weight + Bias
= F x F x n_c^{l-1} x n_c^l + n_c^l
= 5 x 5 x 1 x 6 + 6 = 156
Connections = 28 x 28 x 156
= 122304
28 x 28 x 6

S2



S2: Average Pooling Layer

C3



C3: Convolutional Layer

C3

S2	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

این نوع اتصال برهم زننده تقارن موجود در معماری است و تعداد پارامترها را کاهش داده است.

S4

S4: Pooling Layer

10 x 10 x 16

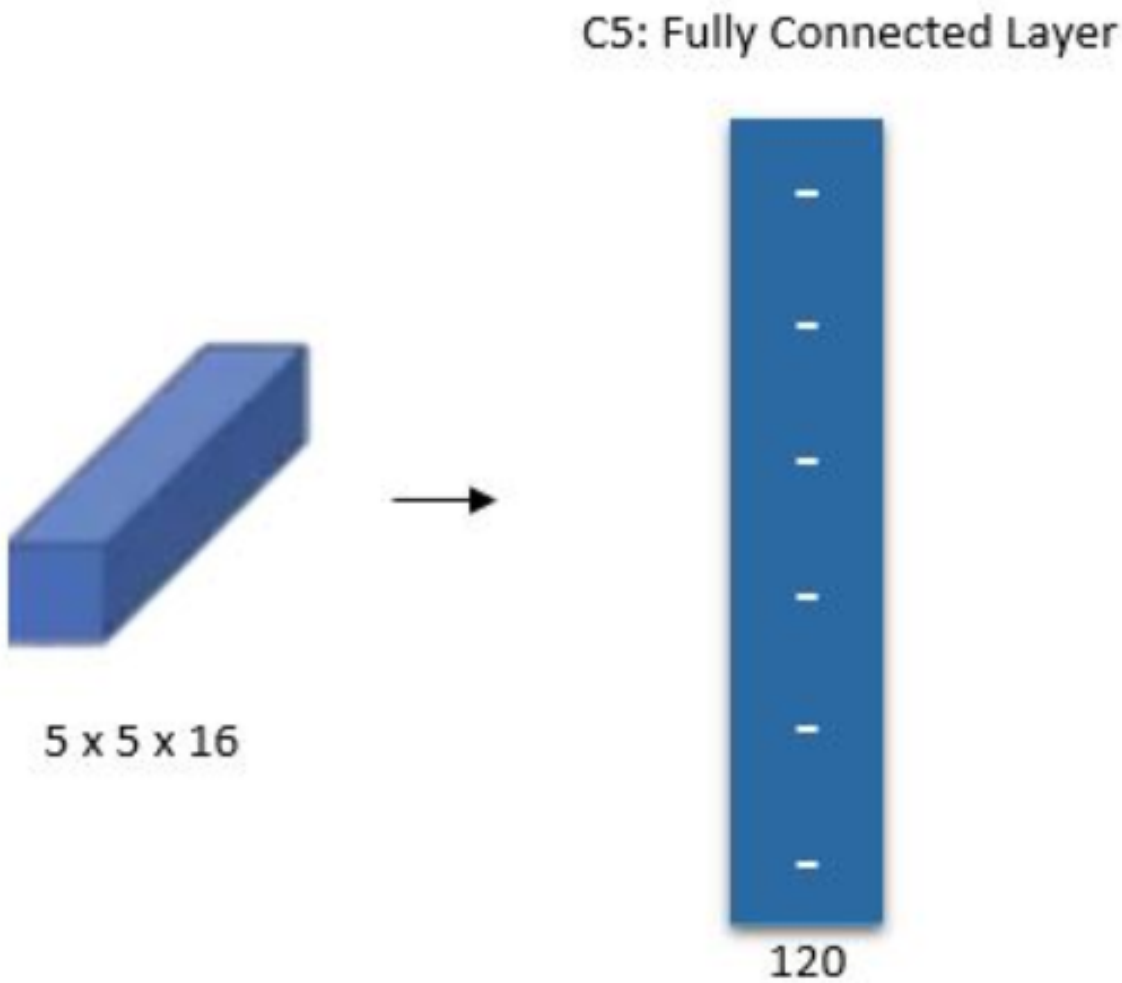
No. of filters, $n_c = 16$
Filter size, F = 2
Padding, P = 0
Stride, S = 2

Trainable parameters
= (coefficient + bias) x filters
= (1 + 1) x 16
= 32

Connections = 5 x 5 x 80
= 2000

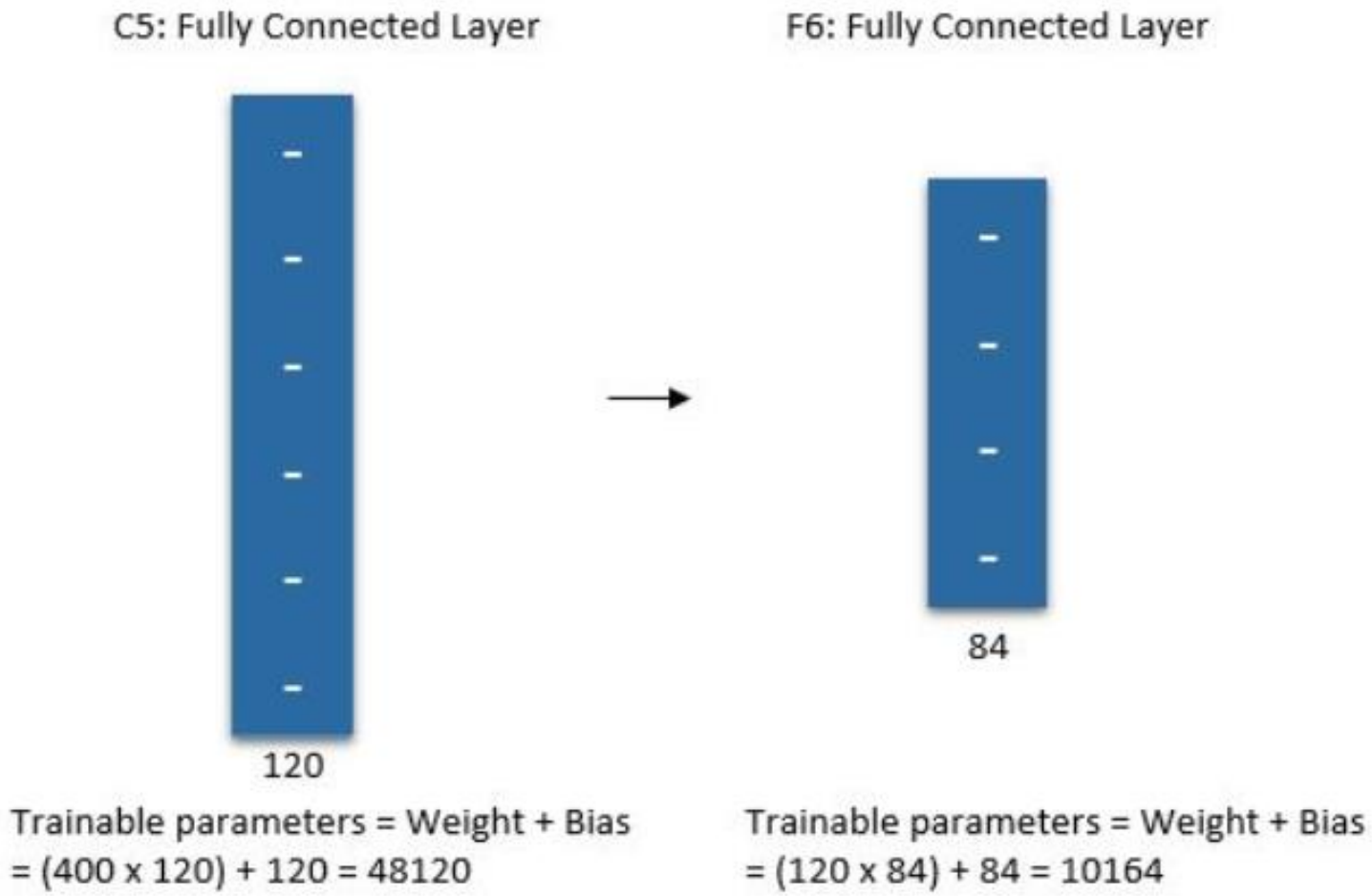
5 x 5 x 16

C5



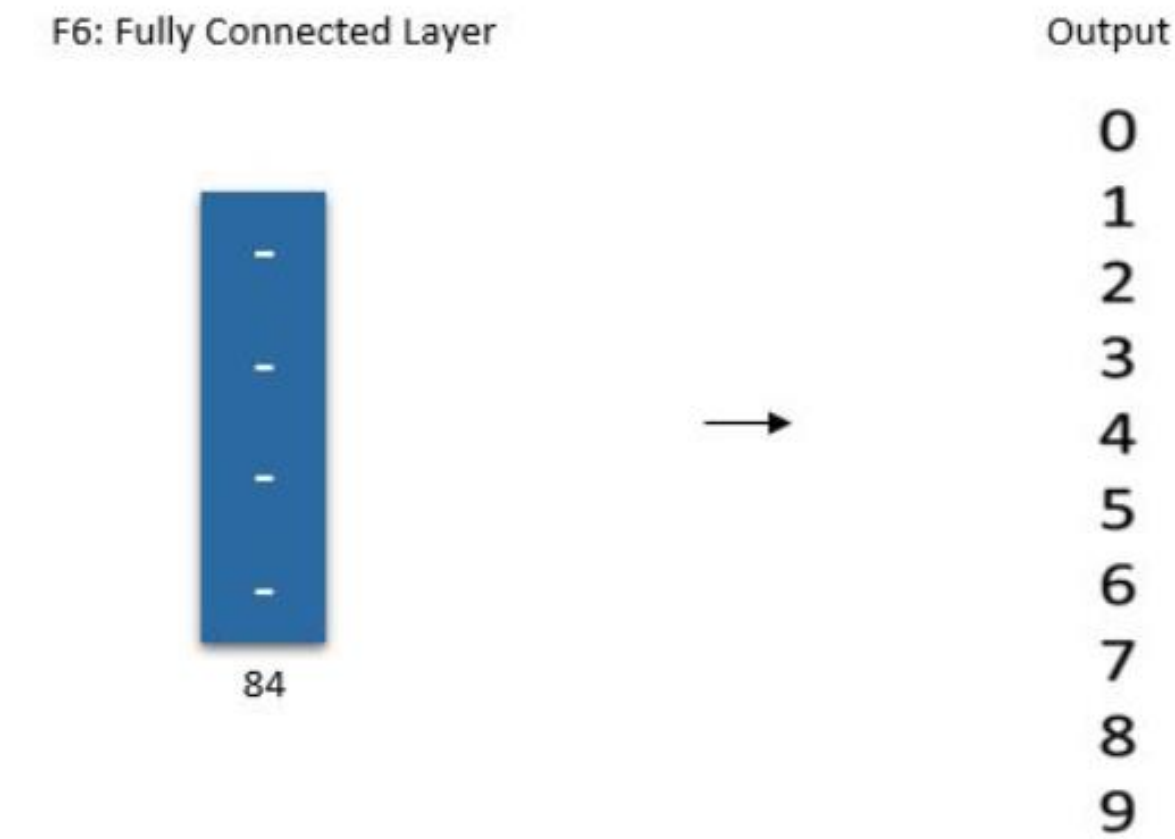
Trainable parameters = Weight + Bias
= (400 x 120) + 120 = 48120

F6



F6: Fully Connected Layer

Output Layer



Trainable parameters = Weight + Bias
= (120 x 84) + 84 = 10164

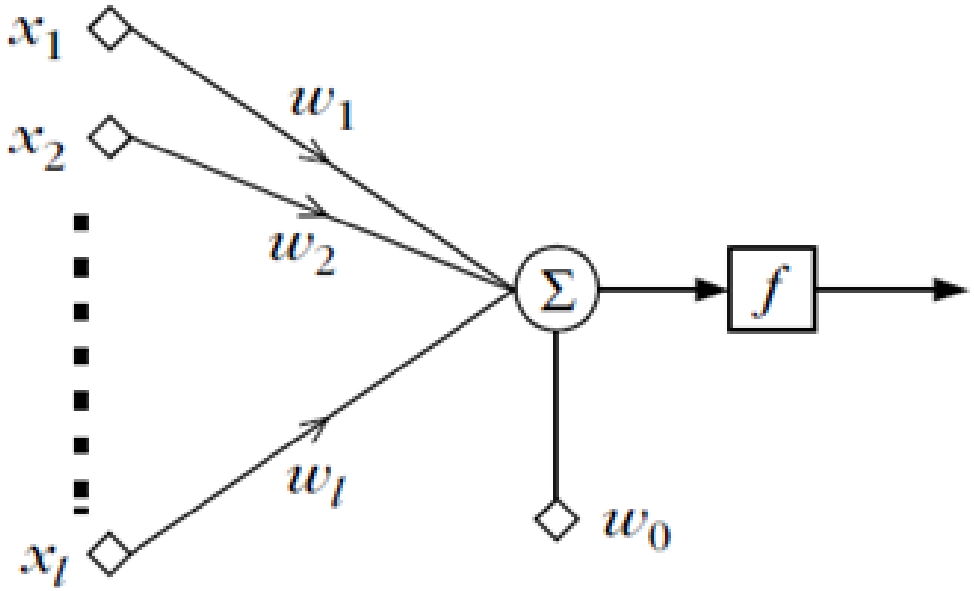
Fully Connected Output Layer

LeNet-5 Architecture Summarized Table

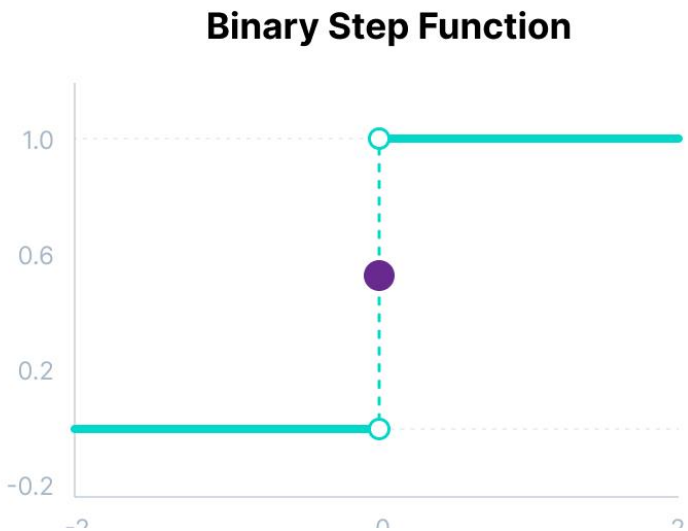
Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

تابع فعال‌سازی
ACTIVATION FUNCTION (f)

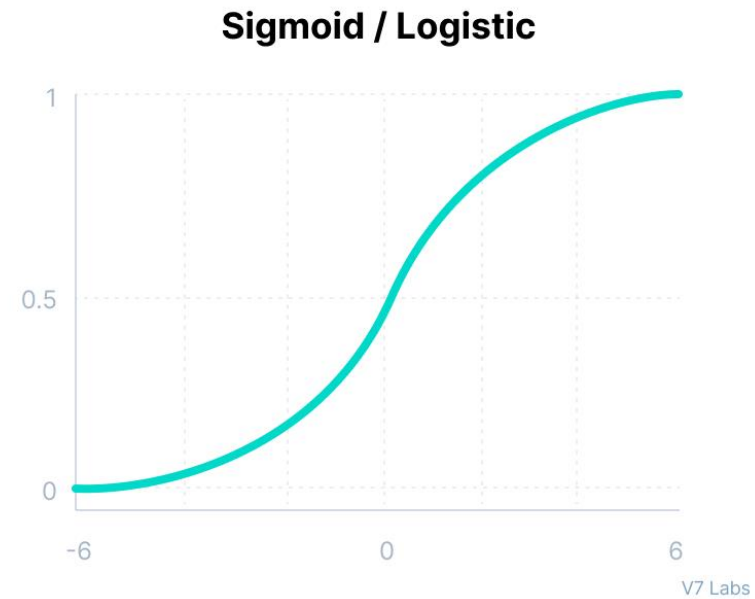
- تابع فعال‌سازی بخش مهمی از یک شبکه عصبی مصنوعی است.
- آن تصمیم می‌گیرد که آیا یک نورون باید فعال شود یا نه. بنابراین مقدار ورودی خالص را محدود می کند.
- تابع فعال سازی یک تبدیل غیر خطی است (تنها عنصر غیرخطی در شبکه کانولوشن).
- بعد از لایه‌های یادگیرنده مانند تماما متصل و کانولوشنی قرار دارد.



انواع تابع فعال‌سازی (۱)

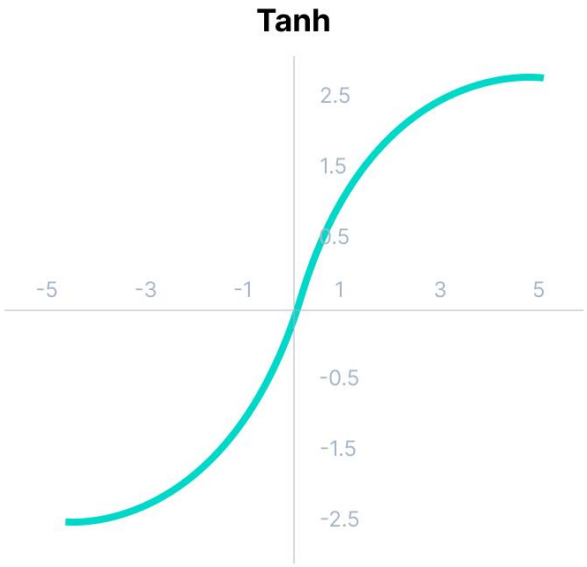


$$f(x) = 1, if\ x \geq 0$$
$$f(x) = 0\ if\ x < 0$$

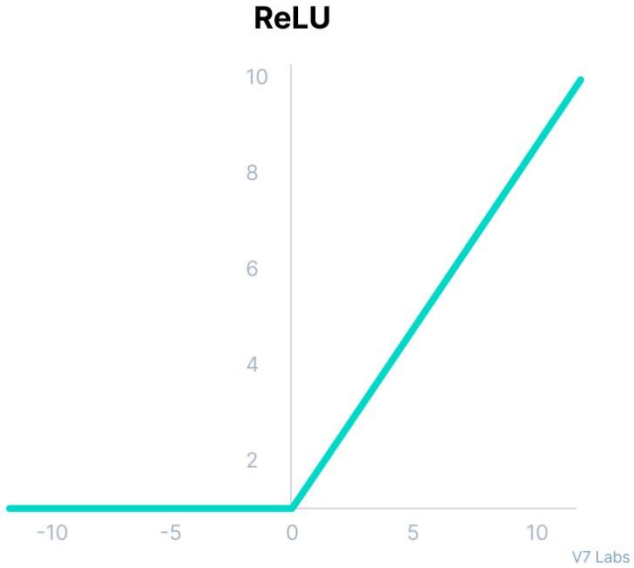


$$f(x) = \frac{1}{1 + e^{-x}}$$

انواع تابع فعال‌سازی (۲)

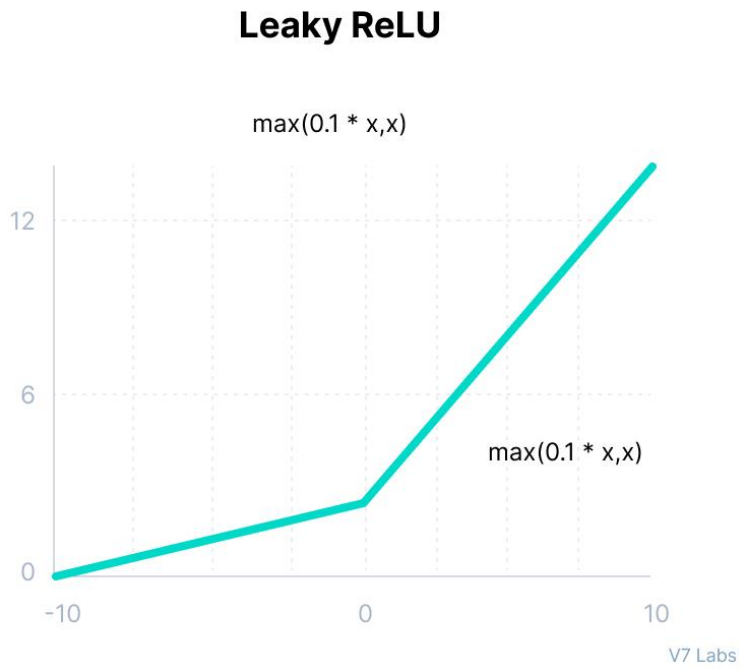


$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



$$f(x) = \max(0, x)$$

انواع تابع فعال‌سازی (۳)



$$f(x) = x, if\ x \geq 0$$
$$f(x) = ax, if\ x < 0$$

$$for\ example\ a\ =\ 0.1$$

سافتمکس
SOFTMAX

- یک تابع فعال‌سازی حیاتی در لایه نهایی شبکه های عصبی کانولوشن است.
- آن امتیازات خام تولید شده توسط آخرین لایه یک شبکه عصبی را به یک توزیع احتمال تبدیل می‌کند.
- این تابع هر عدد را به توان e می‌رساند، سپس نرمال‌سازی انجام می‌دهد؛ اطمینان ایجاد می‌کند که مقادیر خروجی بین ۰ و ۱ قرار می‌گیرند و جمع آنها ۱ است. این باعث می شود خروجی به عنوان احتمال تعلق به کلاس تفسیر شود.

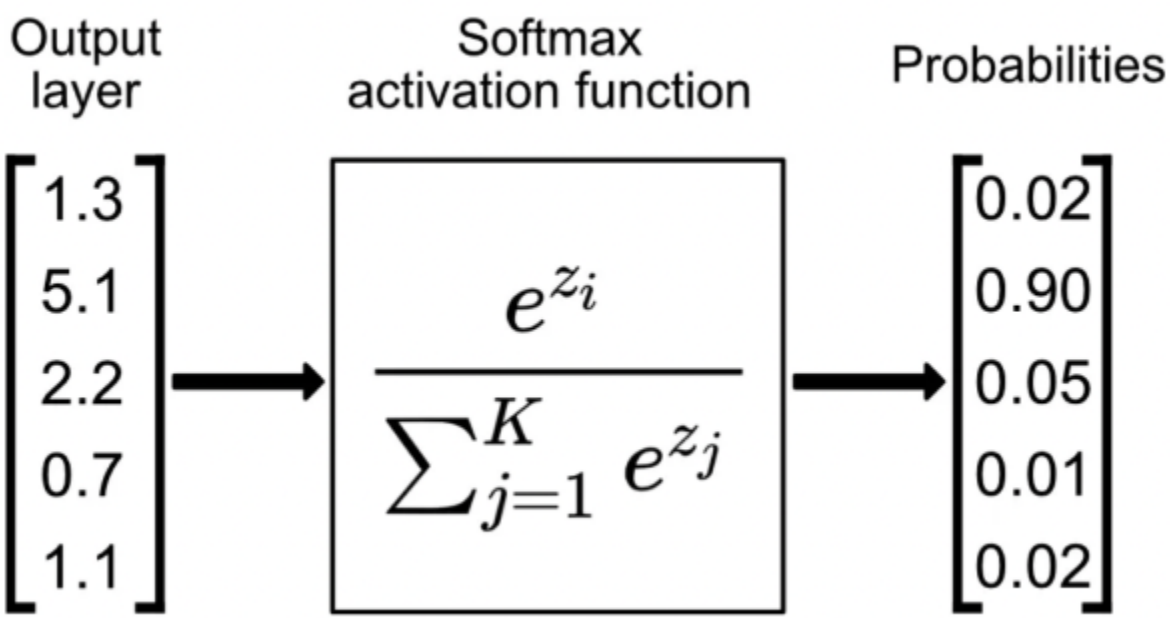
$$softmax(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

$$k = number\ of\ classes$$

- مثال

$$softmax(1, 2, 8) = (0.001, 0.002, 0.997)$$

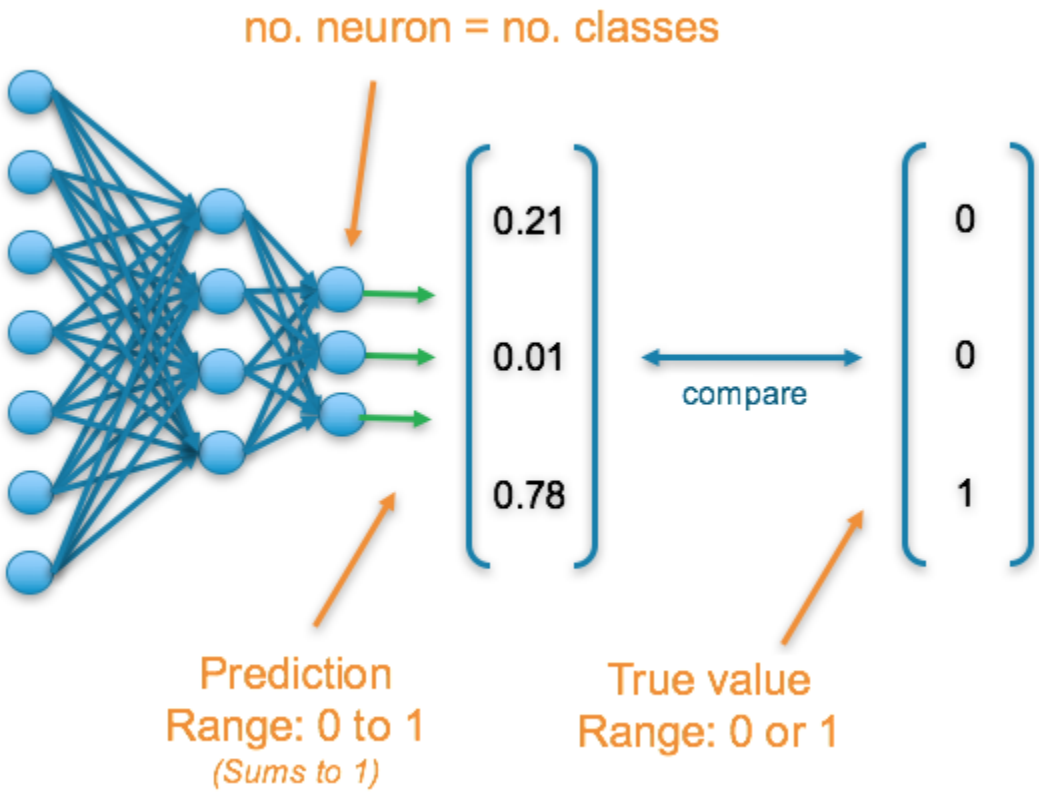
SOFTMAX


Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

Loss Function

- The final classification is achieved from the output layer.
- Some loss functions are utilized in the output layer to calculate the predicted error created across the training samples in the CNN model.
- This error reveals the difference between the actual output and the predicted one.
- Next, it will be optimized through the CNN learning process.
- Several types of loss function are employed in various problem types.

Loss Function



- A loss function is used to measure model performance by calculating difference between the actual output (ground truth) and the predicted one.
- Next, it will be optimized through the CNN learning process.
- Optimizing adjusts model parameters to minimize the loss function
- Several types of loss function are employed in various problem types.

Loss Function: Cross-Entropy

- Cross-Entropy or Softmax Loss Function or Log Loss Function
- Its output is the probability $p \in \{0, 1\}$.
- It is usually employed as a substitution of the square error loss function in multi-class classification problems.
- In the output layer, it employs the softmax activations to generate the output within a probability distribution.
- $p_i = \frac{e_i^a}{\sum_{k=1}^{N} e_k^a}$
- $H(p, y) = -\sum_i y_i \log(p_i)\,, i \in [1, N]$
  - e_i^a: the non-normalized output from the preceding layer
  - N: the number of neurons in the output layer
- For binary classification (where labels are either 0 or 1), the loss function simplifies to:

$$H(p, y) = -(\mathrm{y}\log(p) + (1 - y)\log(1 - p))$$

Loss Function: Euclidean Loss Function

- $H(p, y) = \frac{1}{2N}\sum_{i=1}^{N}(p_i - y_i)^2$
  - This function is widely used in regression problems.
  - It is also the so-called mean square error.

مراجع

1. https://www.historyofdatascience.com/yann-lecun/
2. K. O'Shea, R. Nash, An introduction to convolutional neural networks, "CoRR", vol. abs/1511.08458, 2015. doi: arXiv:1511.08458.
3. L Deng, D Yu, Deep learning: methods and applications, "Foundations and Trends in Signal Processing" ,7, (3–4), 197-387, 2014.
4. AlexNet: Alex Krizhevsky, I. Sutskever, G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Neural Information Processing Systems*, 2012.
5. VGG16: K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *ICLR 2015*.
6. GoogLeNet: C Szegedy et al. "Going Deeper with Convolutions", *CVPR2015*.
7. https://d2l.ai/chapter_convolutional-modern/googlenet.html
8. RezNet: K. He; X. Zhang; S. Ren; J. Sun "Deep Residual Learning for Image Recognition", CVPR 2016.
9. S. Ioffe, C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift", *arXiv preprint arXiv:1502.03167*.
10. DenseNet: G Huang et. al, "Densely Connected Convolutional Networks", CVPR 2017.