

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



یادگیری عمیق

جلسه ۱۳

شبکه‌های عصبی کانوولوشنال

Convolutional Neural Networks (CNN)

کاظم فولادی قلعه
دانشکده مهندسی، پردیس فارابی
دانشگاه تهران

<http://courses.fouladi.ir/deep>

شبکه‌های عصبی کانولوشنال

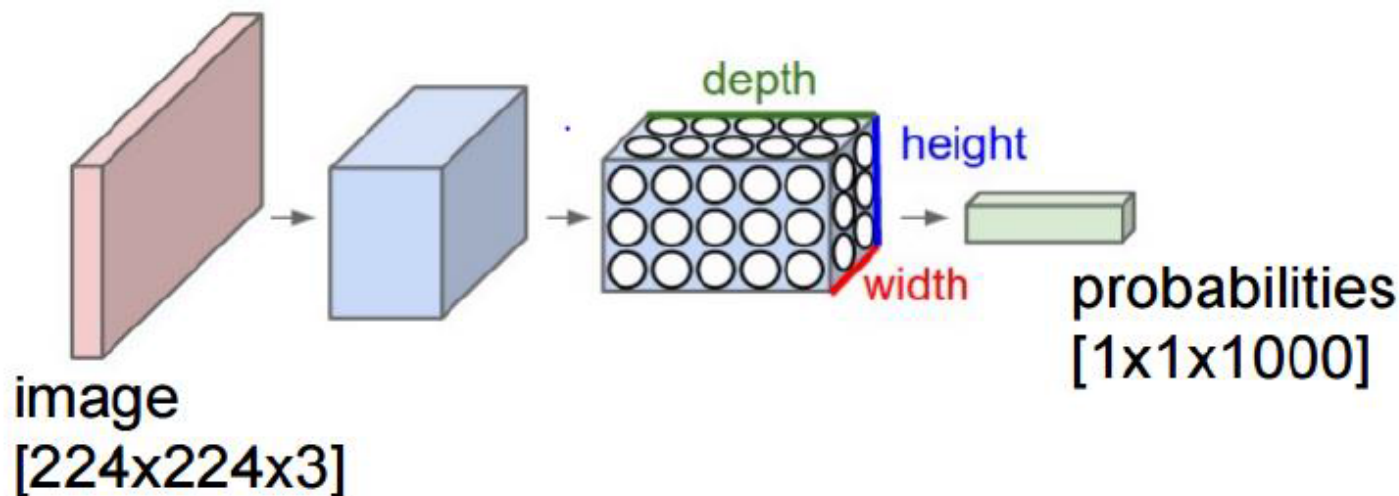


مقدمات

شبکه‌های عصبی کانولوشنال

CONVOLUTIONAL NEURAL NETWORKS (CNN)

شبکه‌های عصبی کانولوشنال، یک بازنمایی پیچیده از داده‌های دیداری را با استفاده از مقدار حجیمی از داده‌ها یاد می‌گیرند. این شبکه‌ها از سیستم بینایی انسان الهام گرفته شده‌اند و چندین لایه از تبدیلات را یاد می‌گیرند، که هر یک بر روی دیگری اعمال می‌شود و به صورت پیش‌رونده، بازنمایی پیچیده‌تری از ورودی را استخراج می‌کنند.



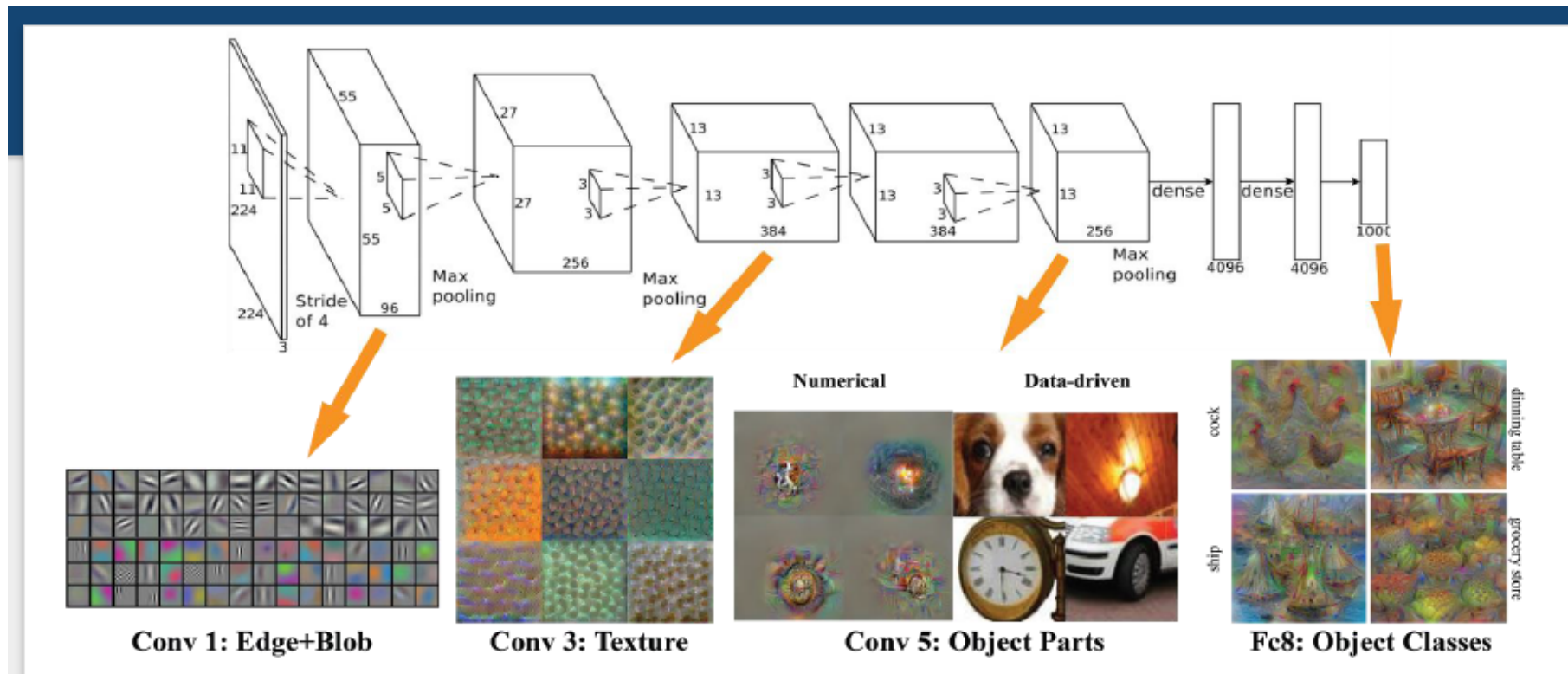
مثال از کاربرد: دسته‌بندی تصاویر (Image Categorization)

هر لایه از یک CNN یک حجم سه-بعدی از اعداد را دریافت می‌کند و یک حجم سه-بعدی از اعداد را خروجی می‌دهد. برای مثال: تصویر یک مکعب $224 \times 224 \times 3$ (RGB) است و به یک بردار 1000 تایی از احتمالات تبدیل می‌شود.

شبکه‌های عصبی کانولوشنال

لایه‌های اصلی

CONVOLUTIONAL NEURAL NETWORKS (CNN)



لایه‌ی کانولوشن، یک آشکارساز ویژگی است که به صورت خودکار و جادویی یاد می‌گیرد که با استفاده از کرنل کانولوشن اطلاعات غیر لازم از یک ورودی را فیلتر کند.

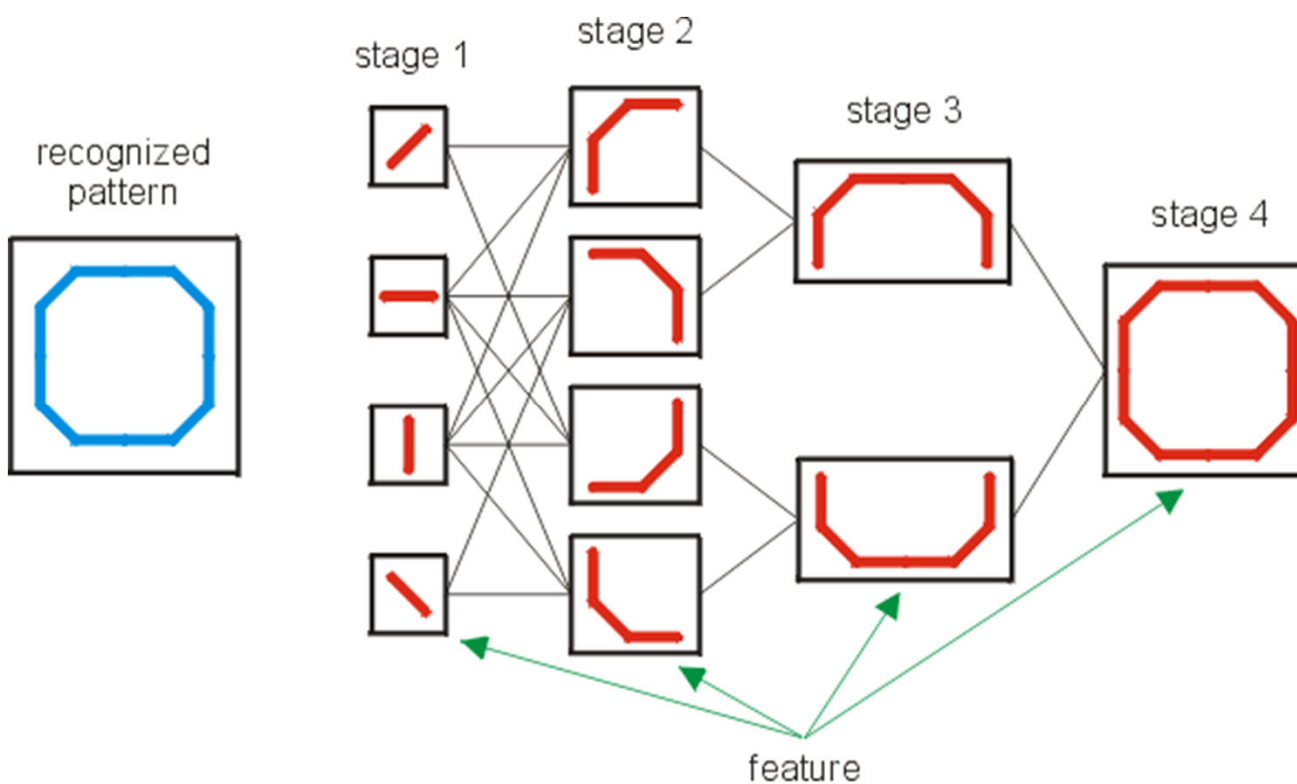
لایه‌ی تلفیق، ماکزیمم یا مقدار متوسط ویژگی‌های خاص بر روی یک ناحیه از داده‌های ورودی را محاسبه می‌کند (↔ کاهش اندازه‌ی تصاویر ورودی). همچنین به آشکارسازی اشیا در برخی مکان‌های نامعمول کمک می‌کند و اندازه‌ی حافظه را کاهش می‌دهد.

لایه‌ی کانولوشن
Convolution Layer

لایه‌ی تلفیق
Pooling Layer

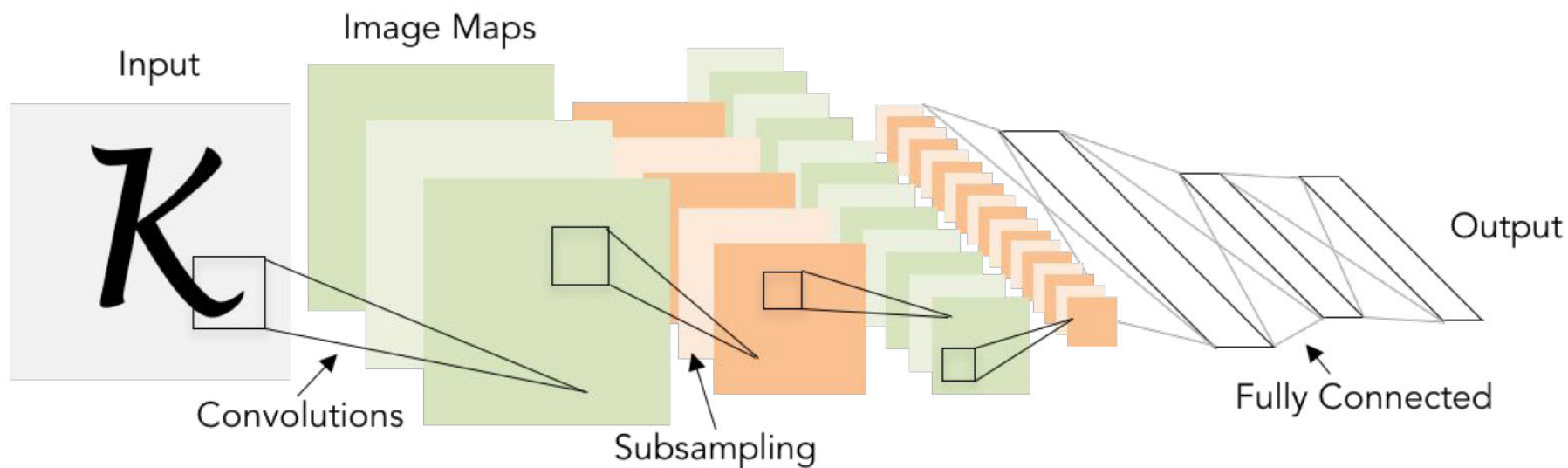
شبکه‌های عصبی کانولوشنال

کارکرد نوعی لایه‌ها



شبکه‌های عصبی کانولوشنال

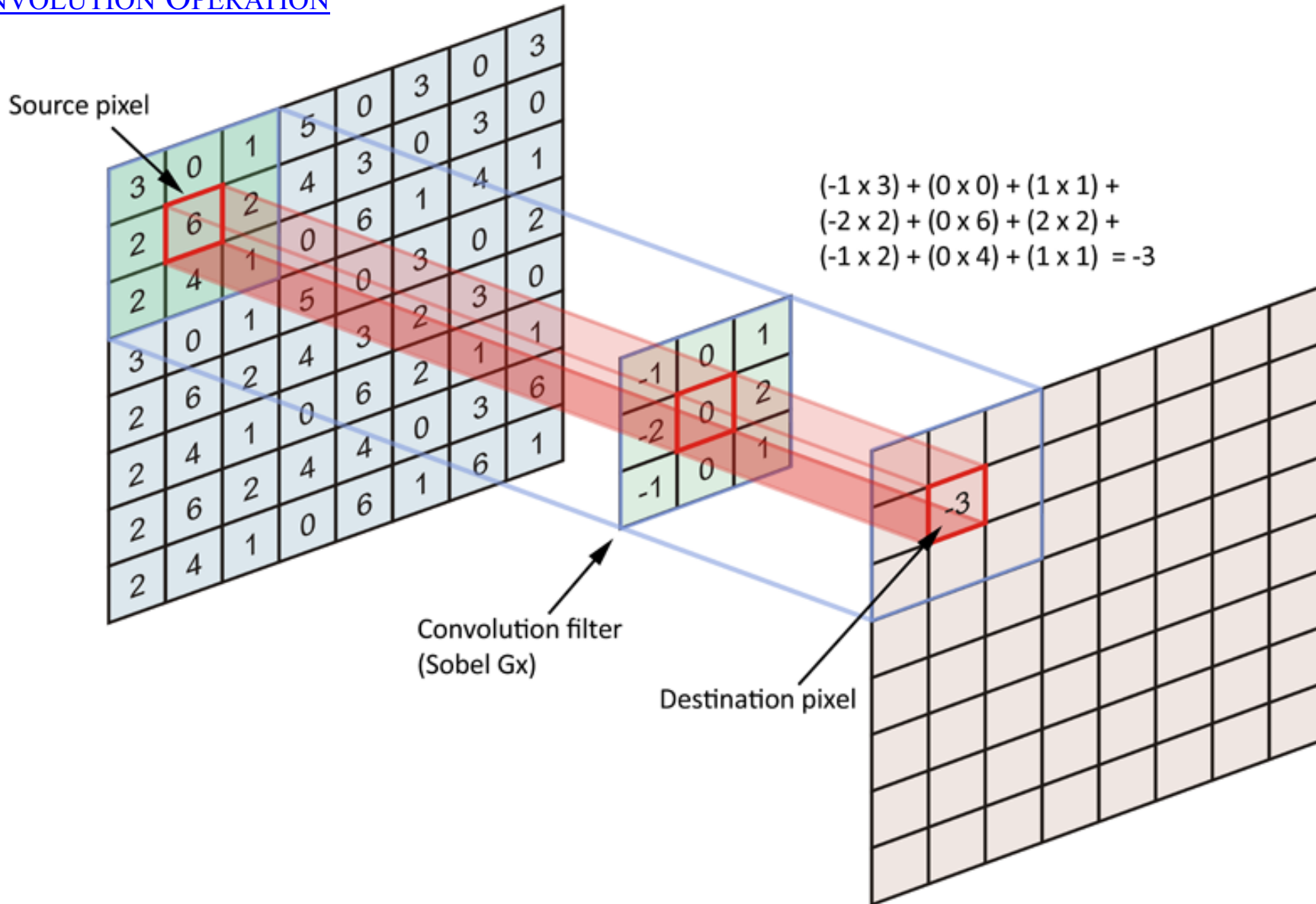
معماری

CONVOLUTIONAL NEURAL NETWORKS

شبکه‌های عصبی کانولوشنال

عملیات کانولوشن

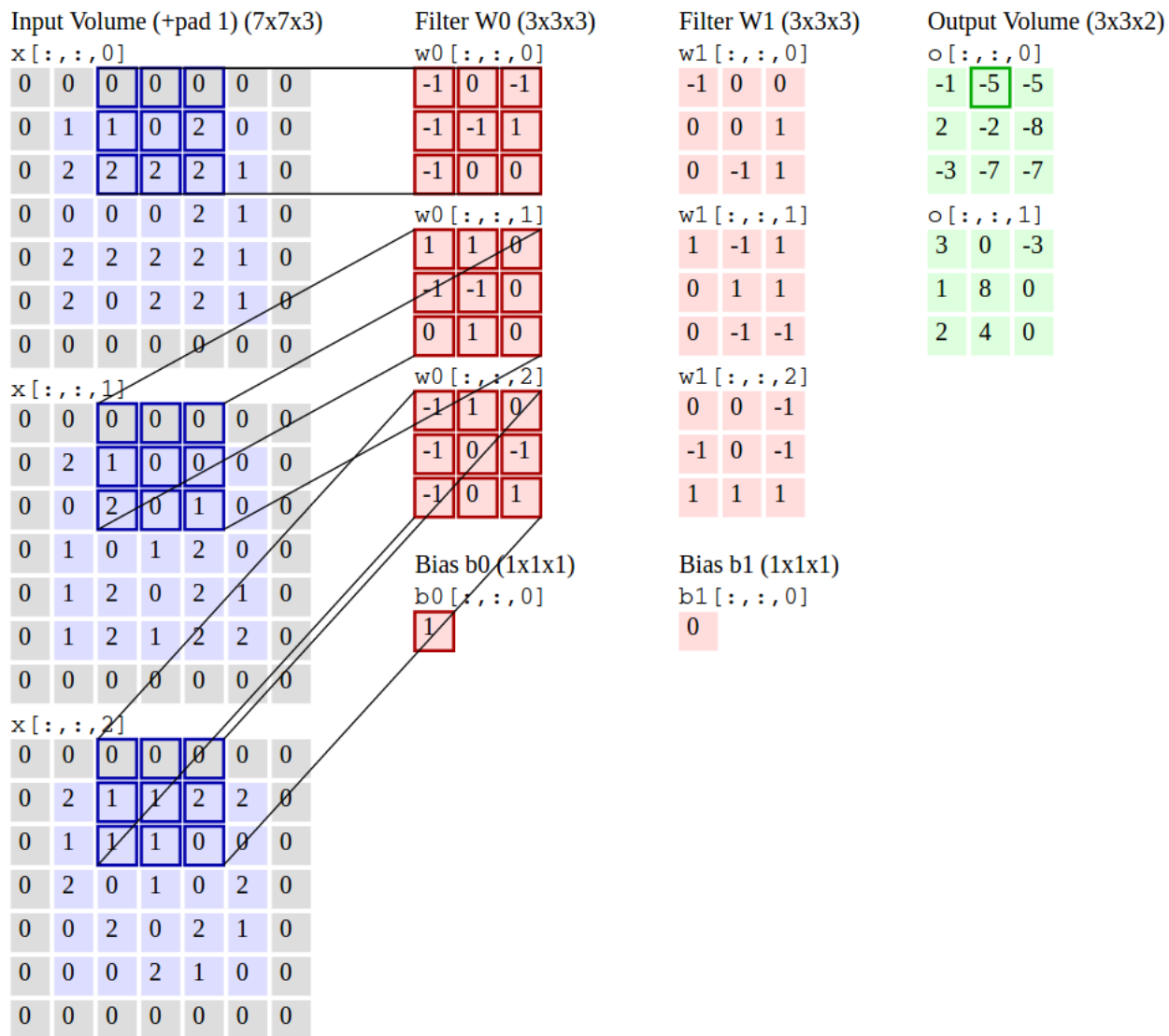
CONVOLUTION OPERATION



شبکه‌های عصبی کانولوشنال

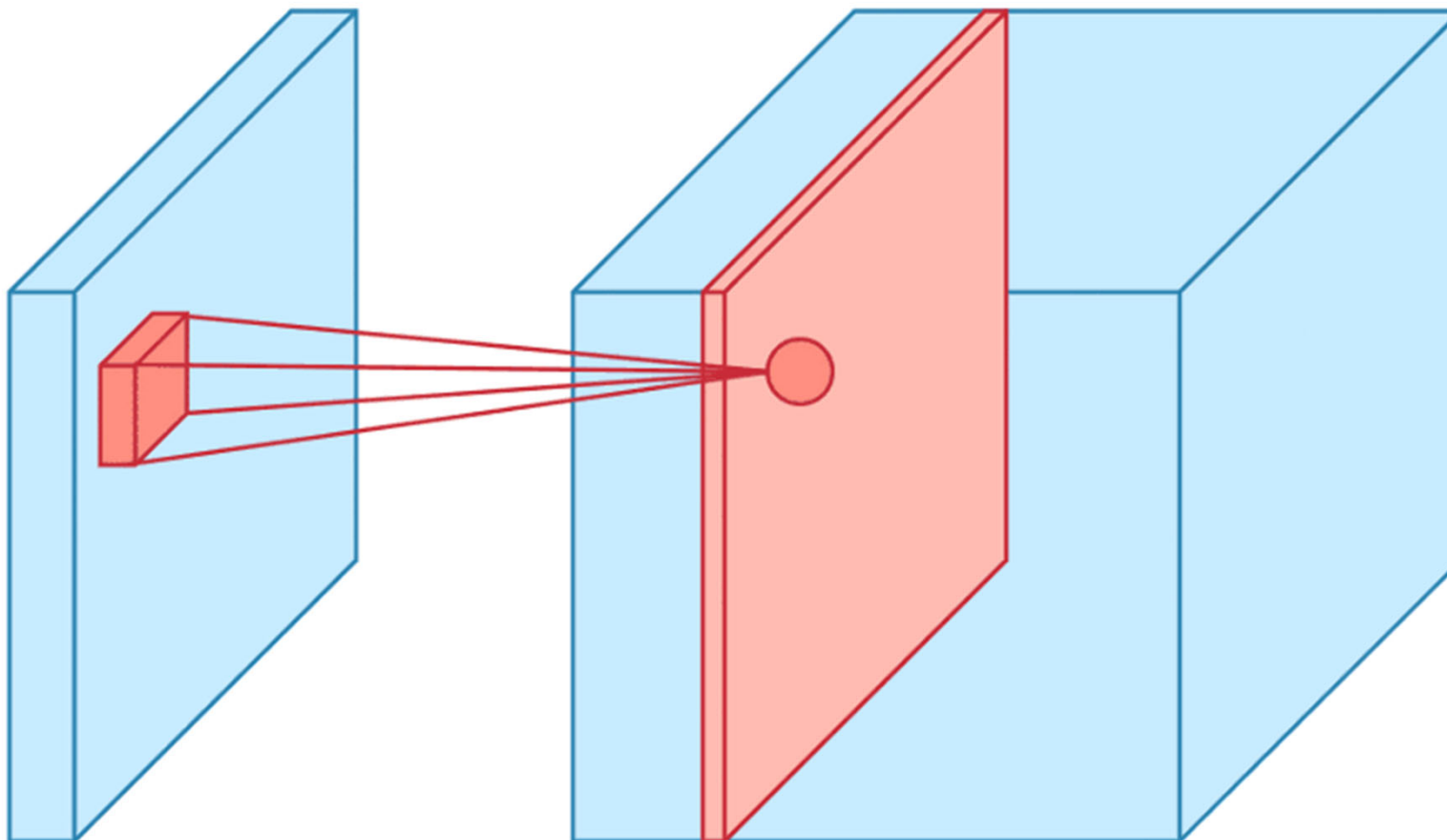
عملیات کانولوشن

CONVOLUTION OPERATION



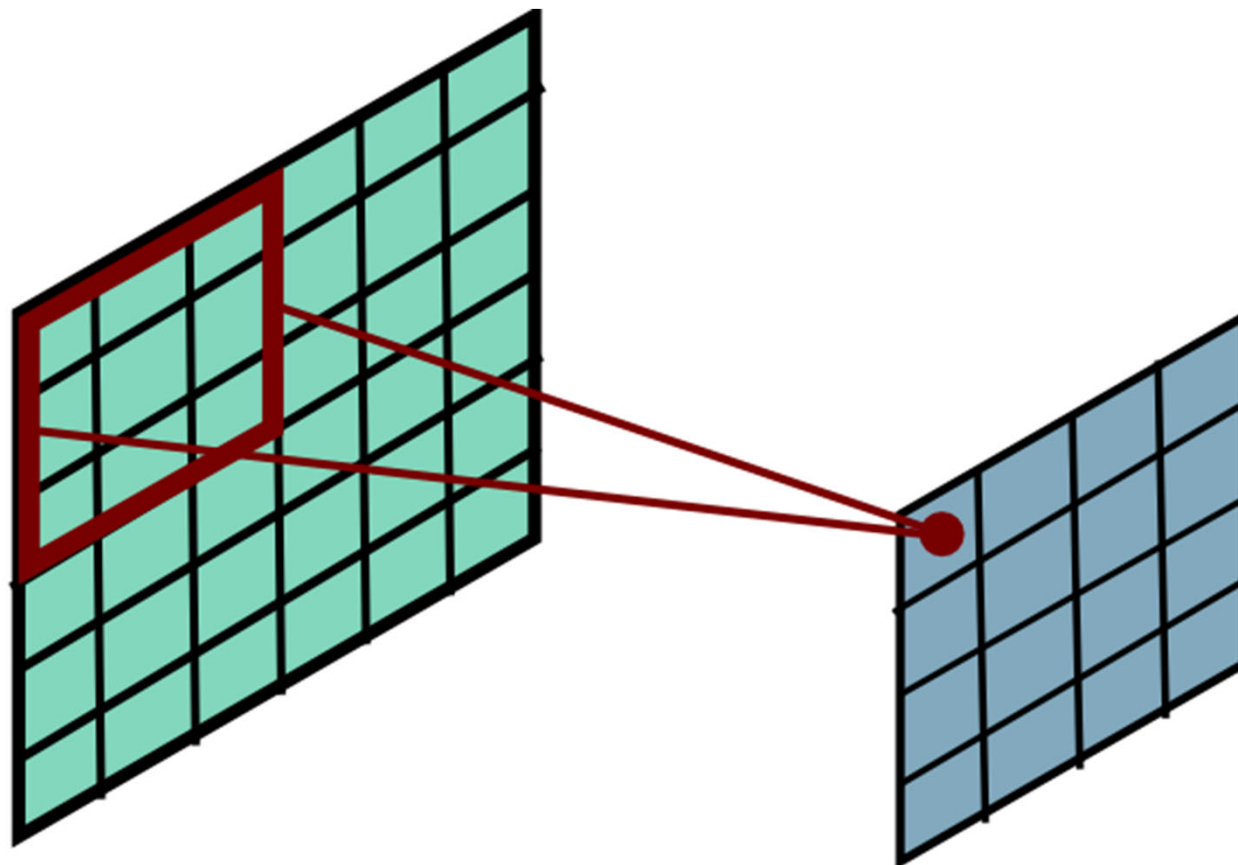
شبکه‌های عصبی کانولوشنال

عملیات کانولوشن

CONVOLUTION OPERATION

شبکه‌های عصبی کانولوشنال

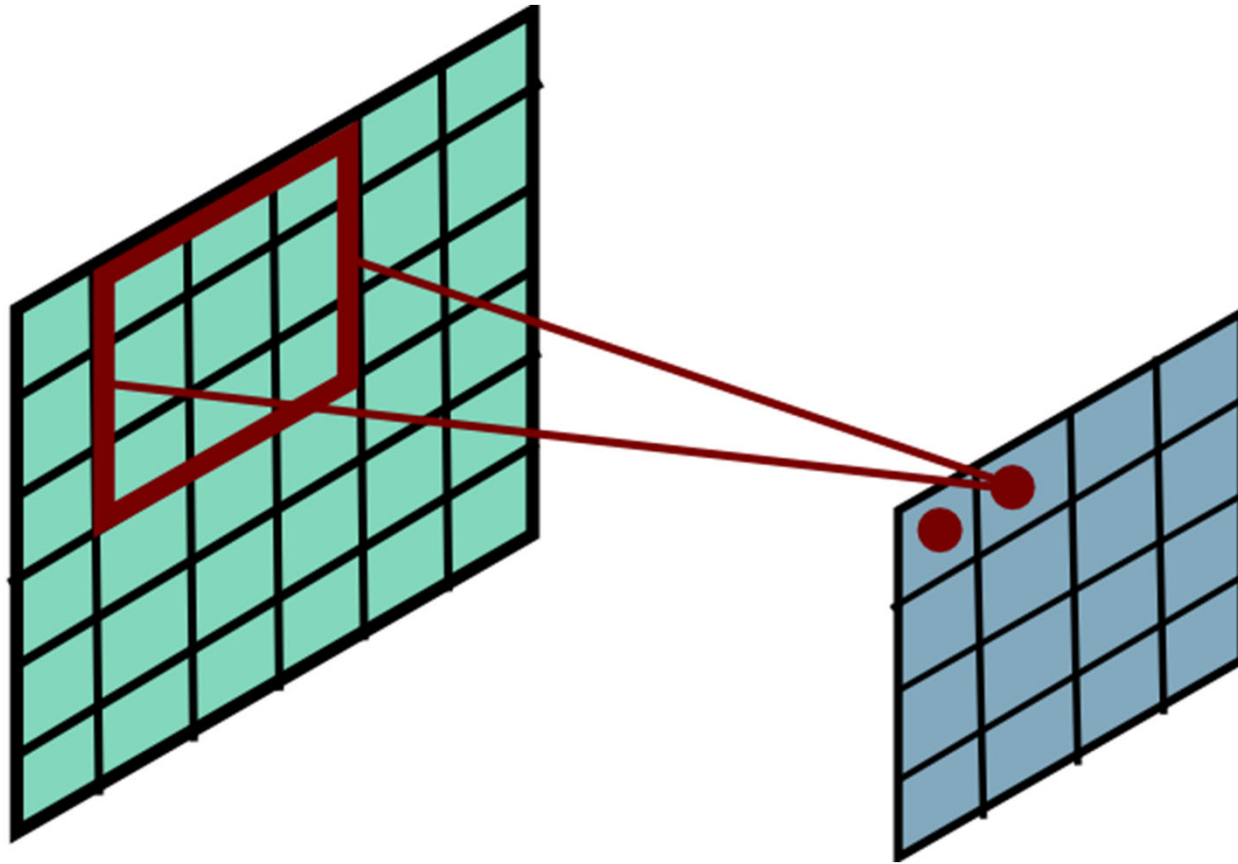
عملیات کانولوشن (۱ از ۱۶)

CONVOLUTION OPERATION

شبکه‌های عصبی کانولوشنال

عملیات کانولوشن (۲ از ۱۶)

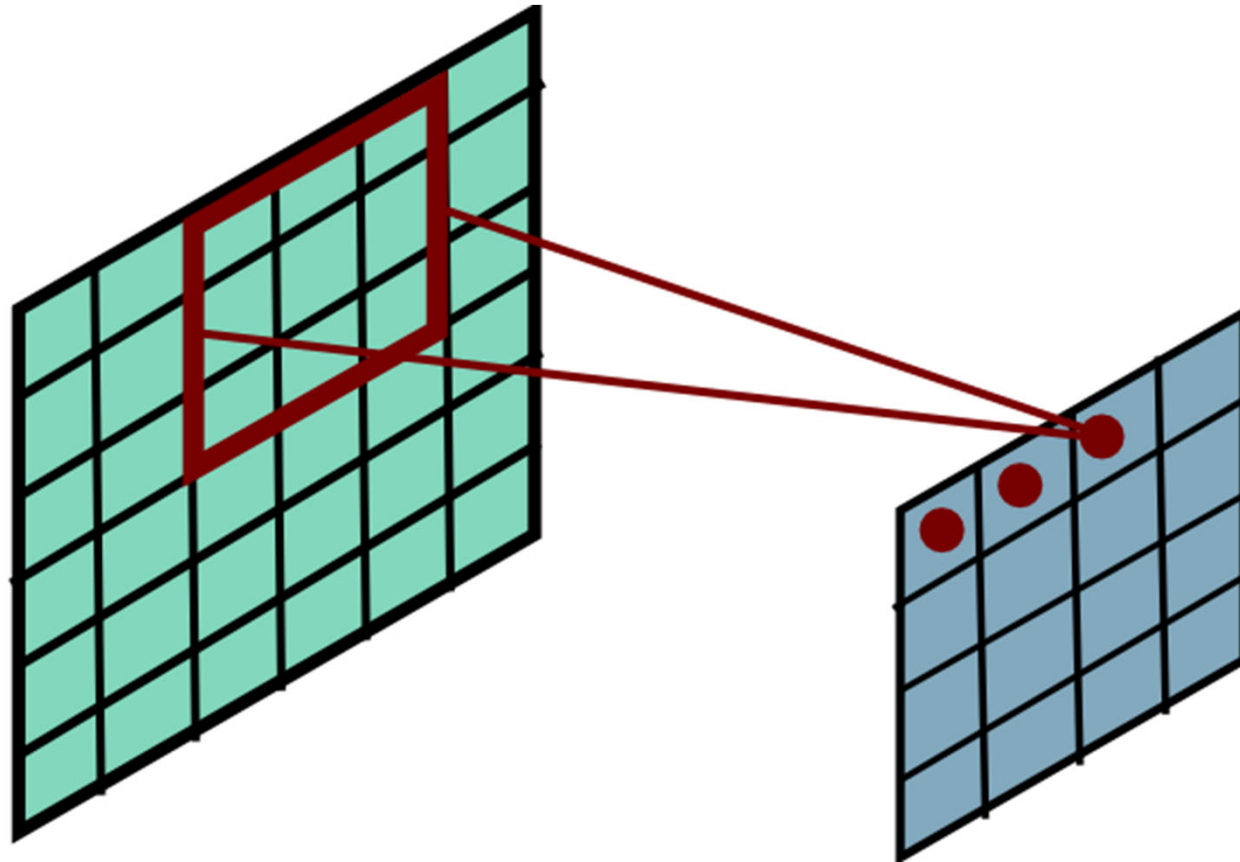
CONVOLUTION OPERATION



شبکه‌های عصبی کانولوشنال

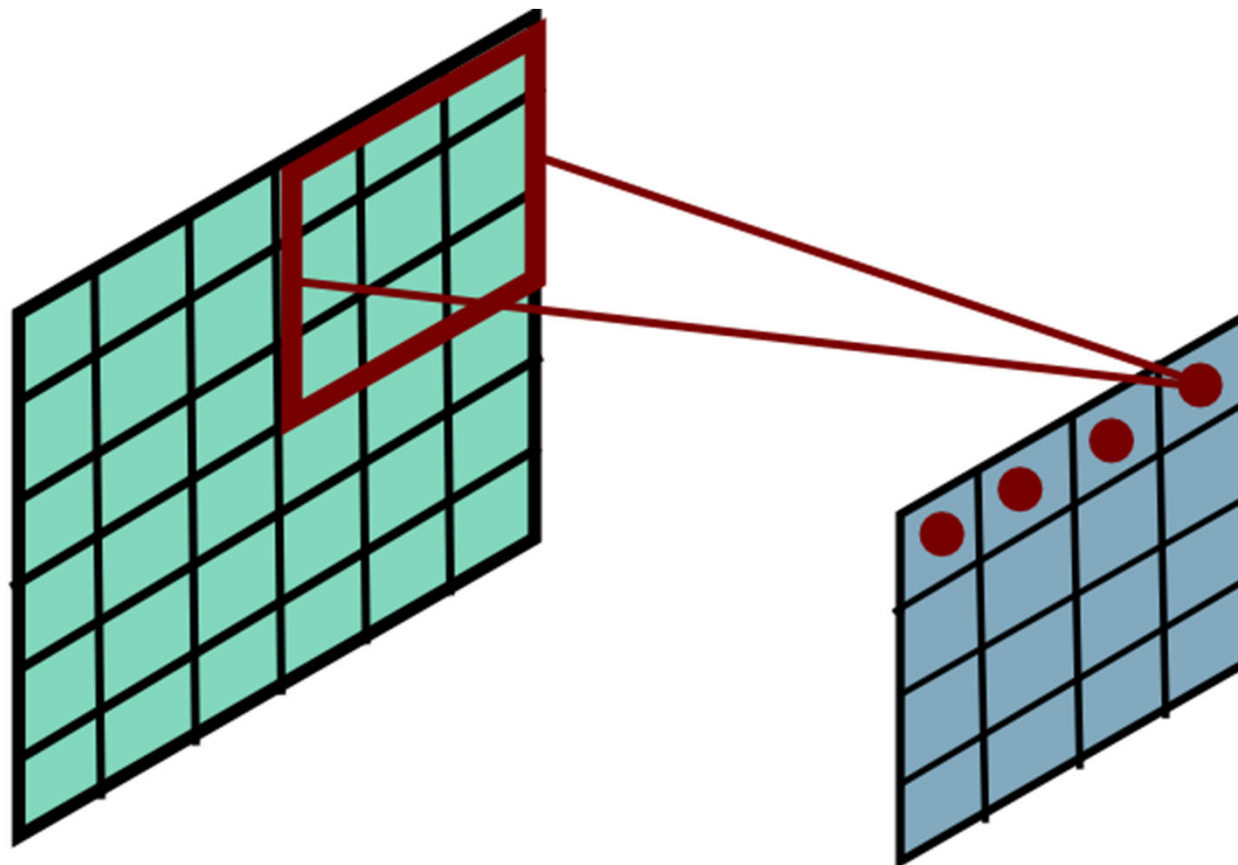
عملیات کانولوشن (۳ از ۱۶)

CONVOLUTION OPERATION



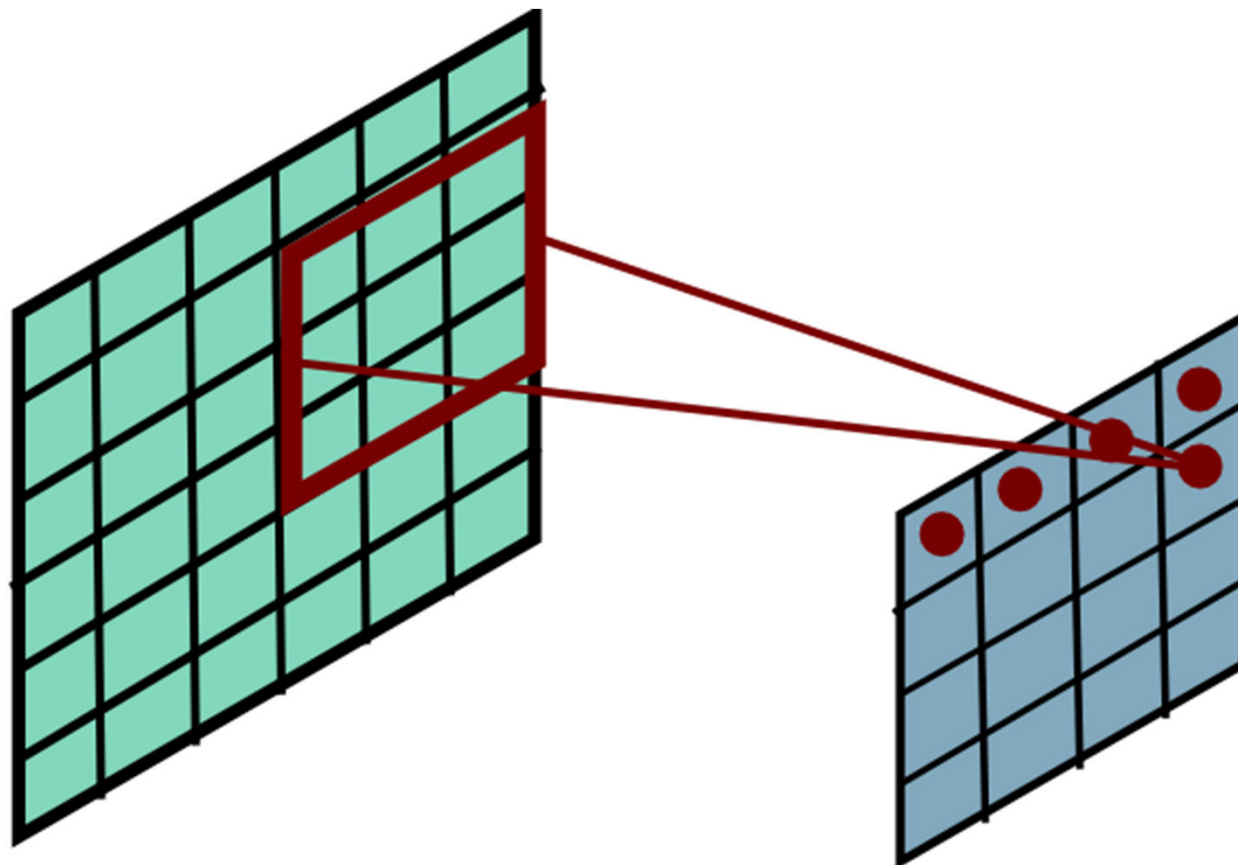
شبکه‌های عصبی کانولوشنال

عملیات کانولوشن (۴ از ۱۶)

CONVOLUTION OPERATION

شبکه‌های عصبی کانولوشنال

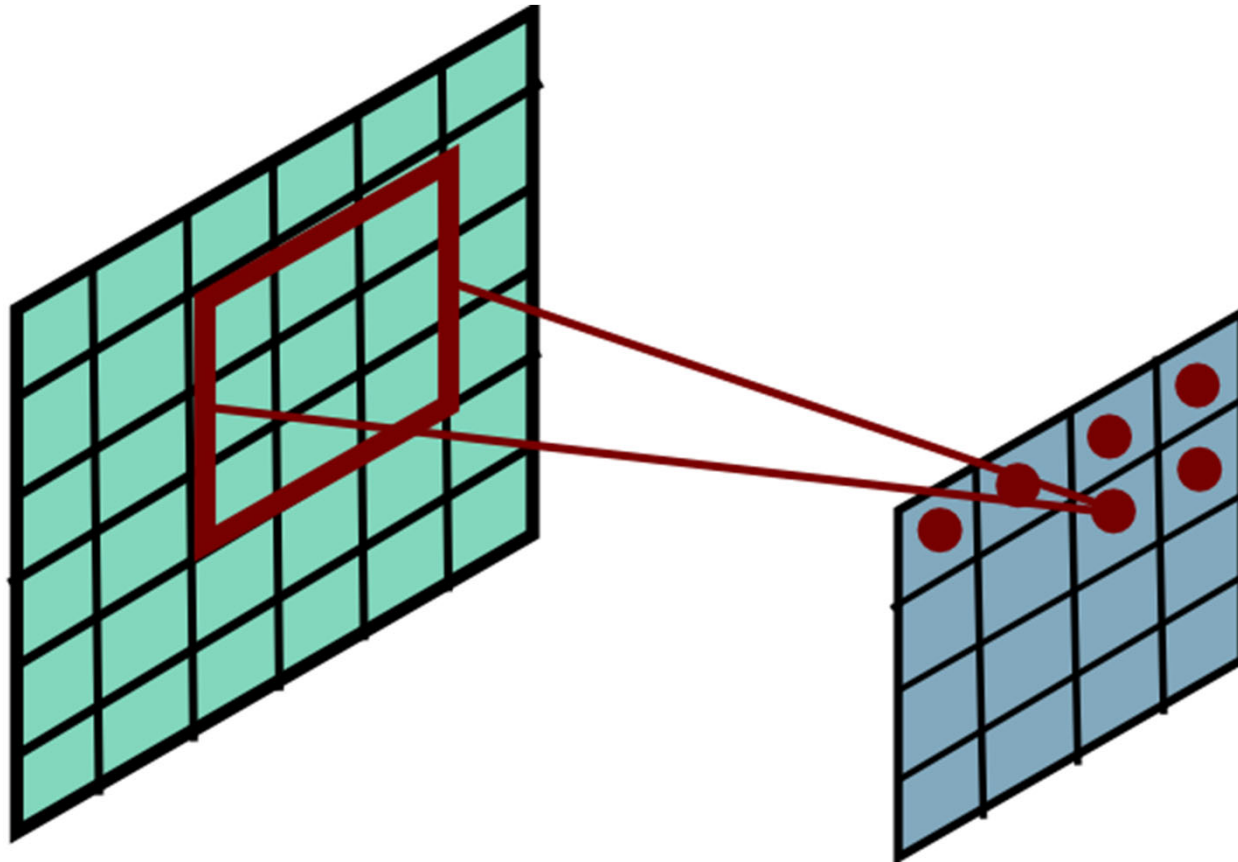
عملیات کانولوشن (۵ از ۱۶)

CONVOLUTION OPERATION

شبکه‌های عصبی کانولوشنال

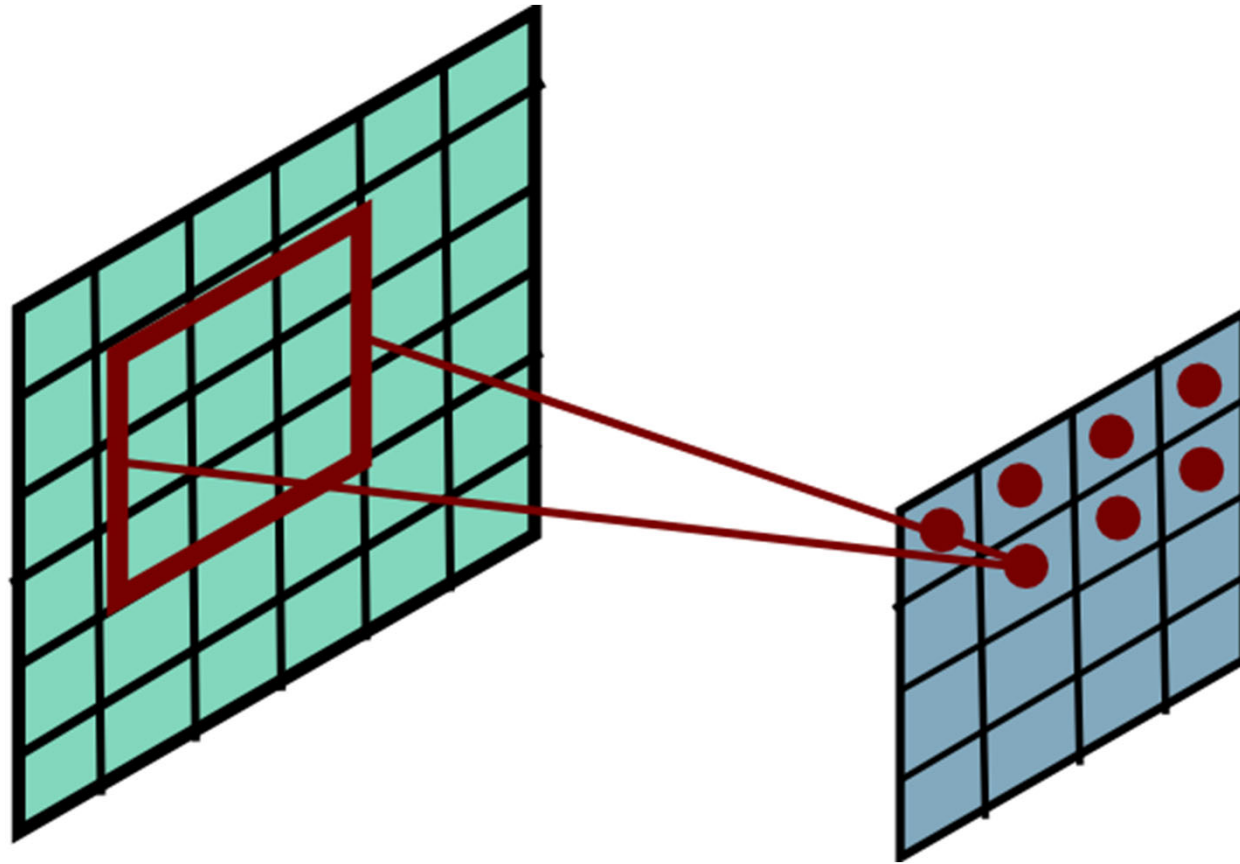
عملیات کانولوشن (۶ از ۱۶)

CONVOLUTION OPERATION



شبکه‌های عصبی کانولوشنال

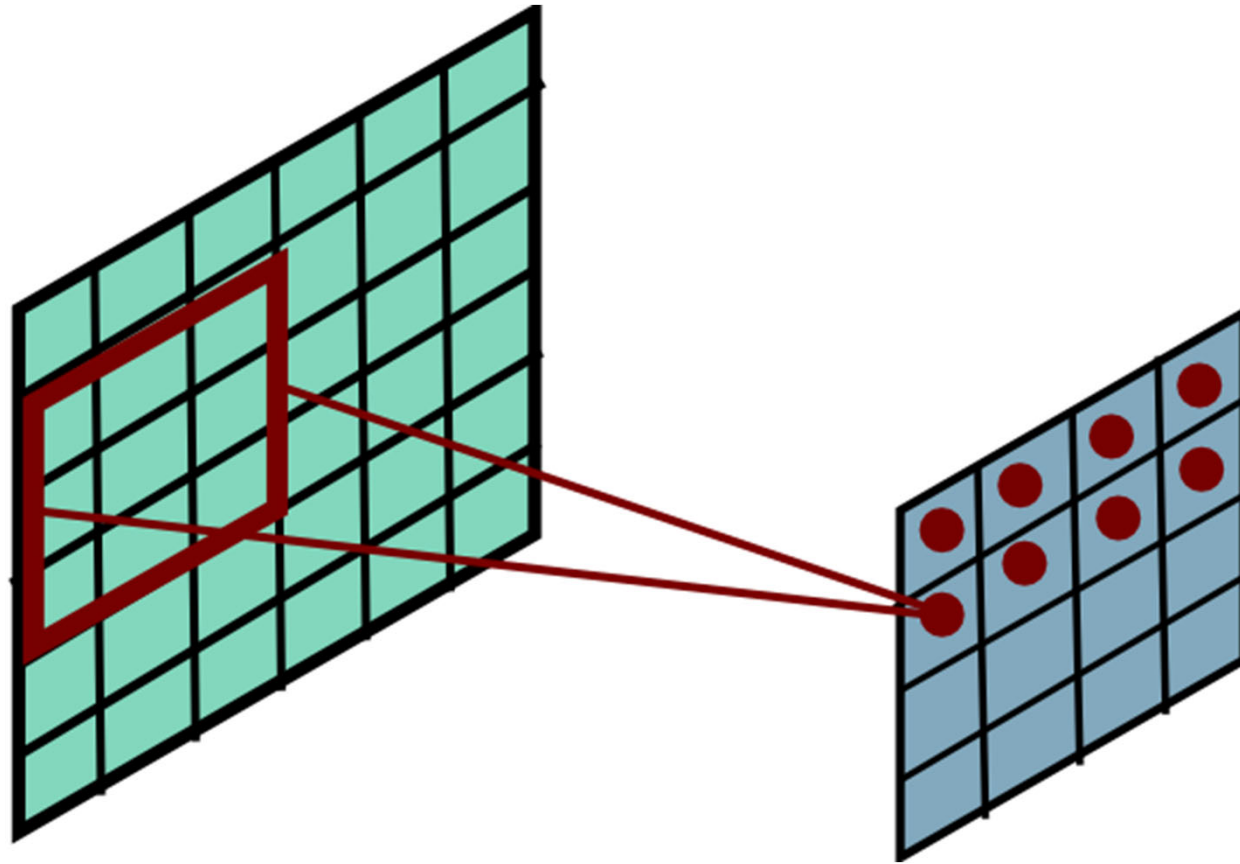
عملیات کانولوشن (۷ از ۱۶)

CONVOLUTION OPERATION

شبکه‌های عصبی کانولوشنال

عملیات کانولوشن (۸ از ۱۶)

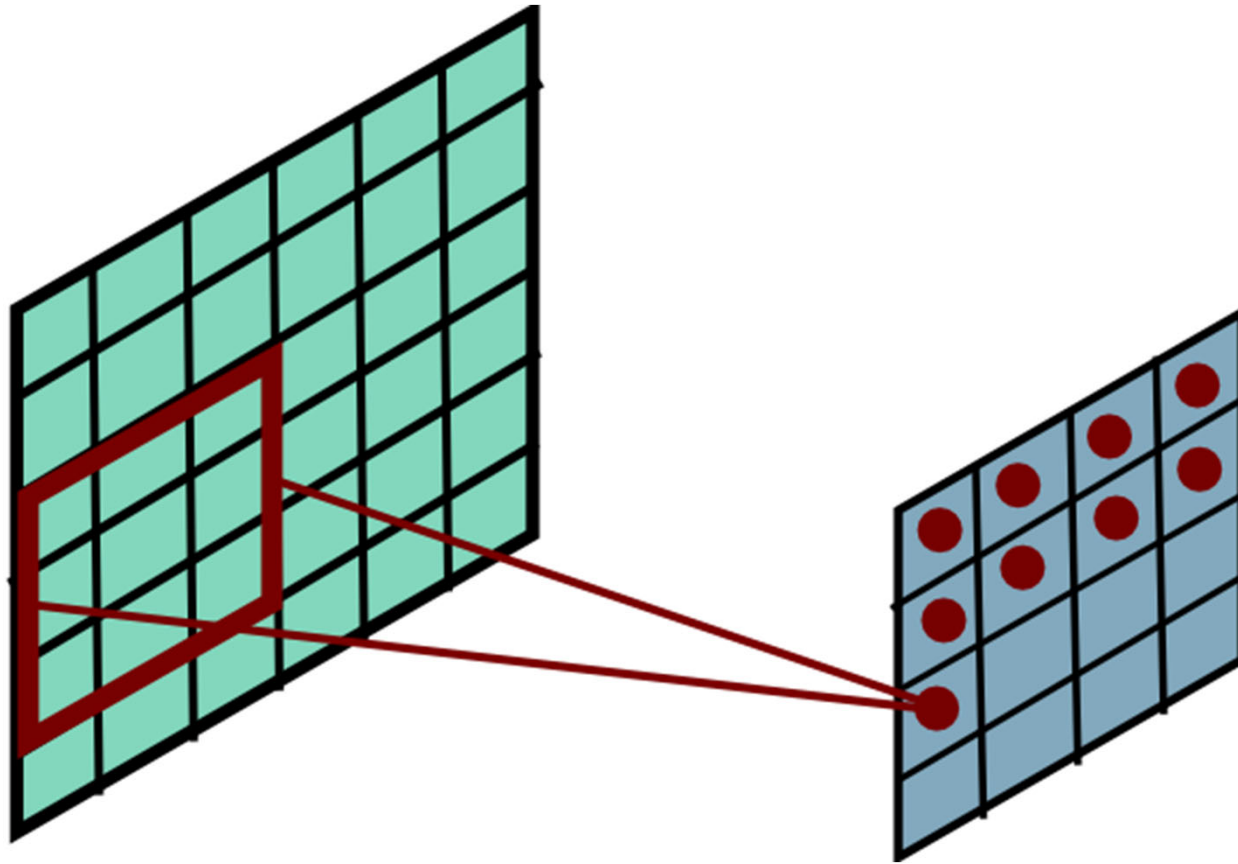
CONVOLUTION OPERATION



شبکه‌های عصبی کانولوشنال

عملیات کانولوشن (۹ از ۱۶)

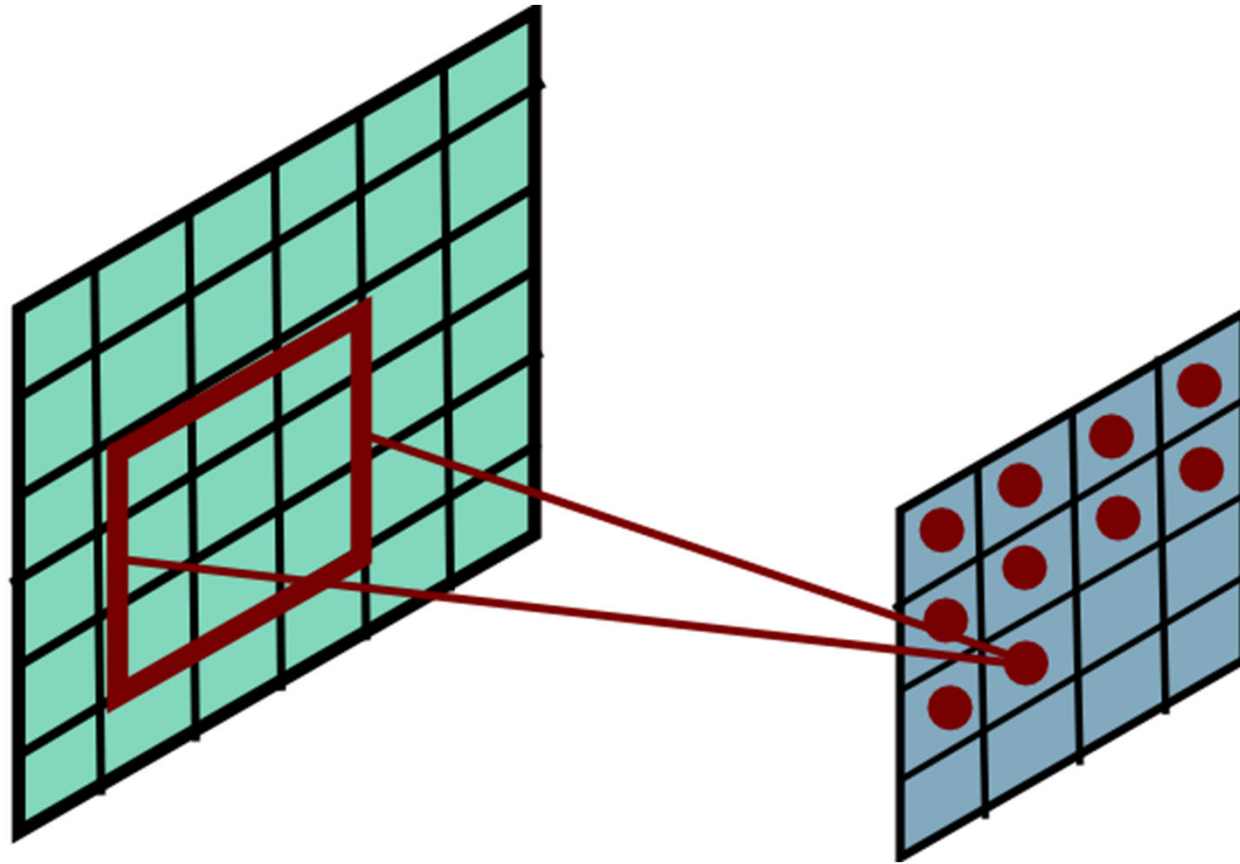
CONVOLUTION OPERATION



شبکه‌های عصبی کانولوشنال

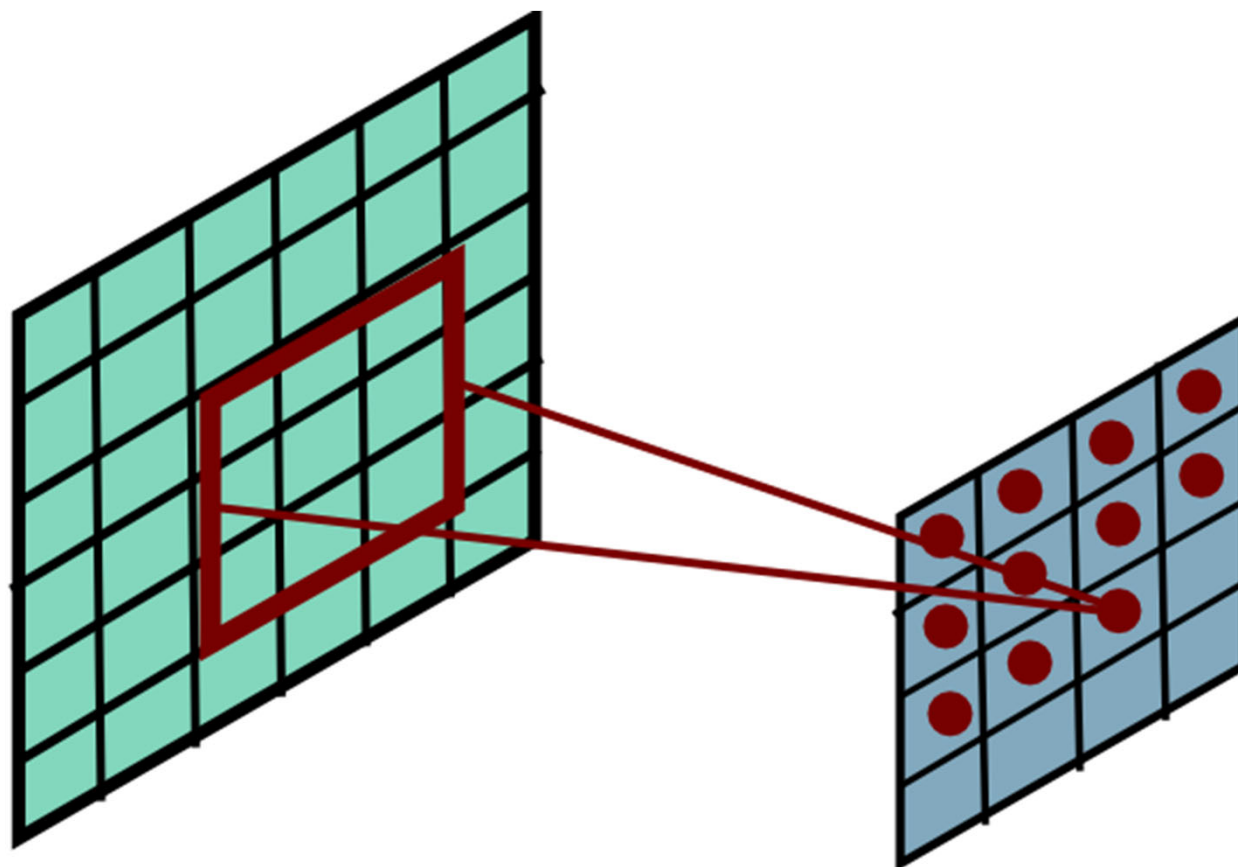
عملیات کانولوشن (۱۰ از ۱۶)

CONVOLUTION OPERATION



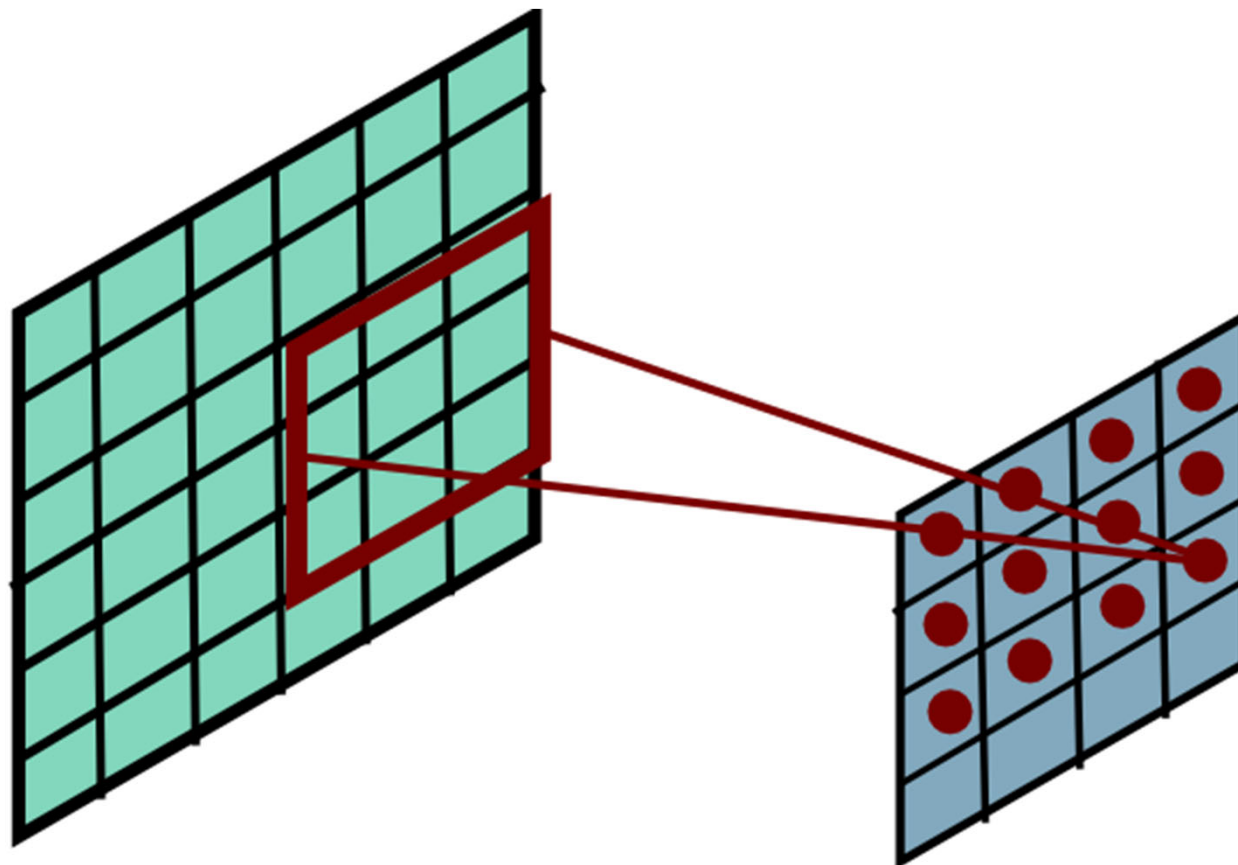
شبکه‌های عصبی کانولوشنال

عملیات کانولوشن (۱۱ از ۱۶)

CONVOLUTION OPERATION

شبکه‌های عصبی کانولوشنال

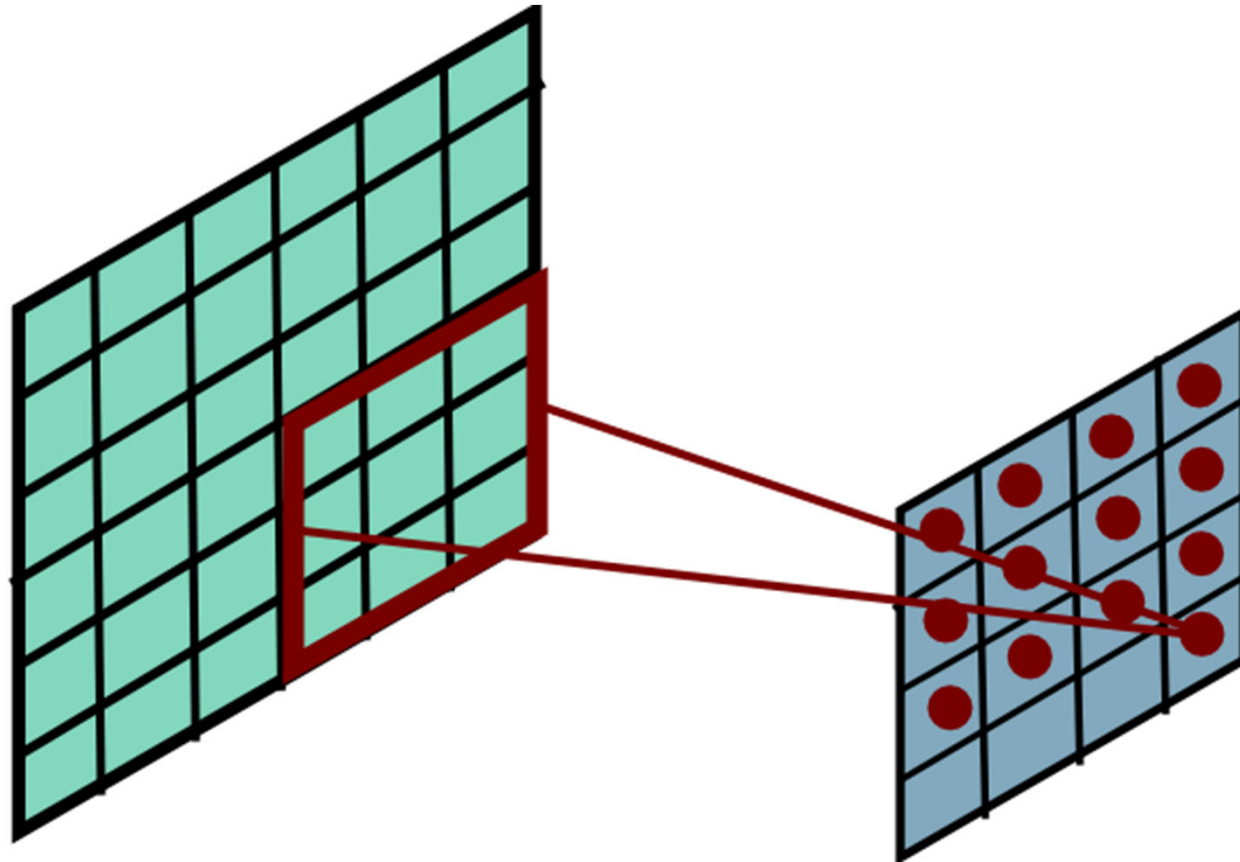
عملیات کانولوشن (۱۲ از ۱۶)

CONVOLUTION OPERATION

شبکه‌های عصبی کانولوشنال

عملیات کانولوشن (۱۳ از ۱۶)

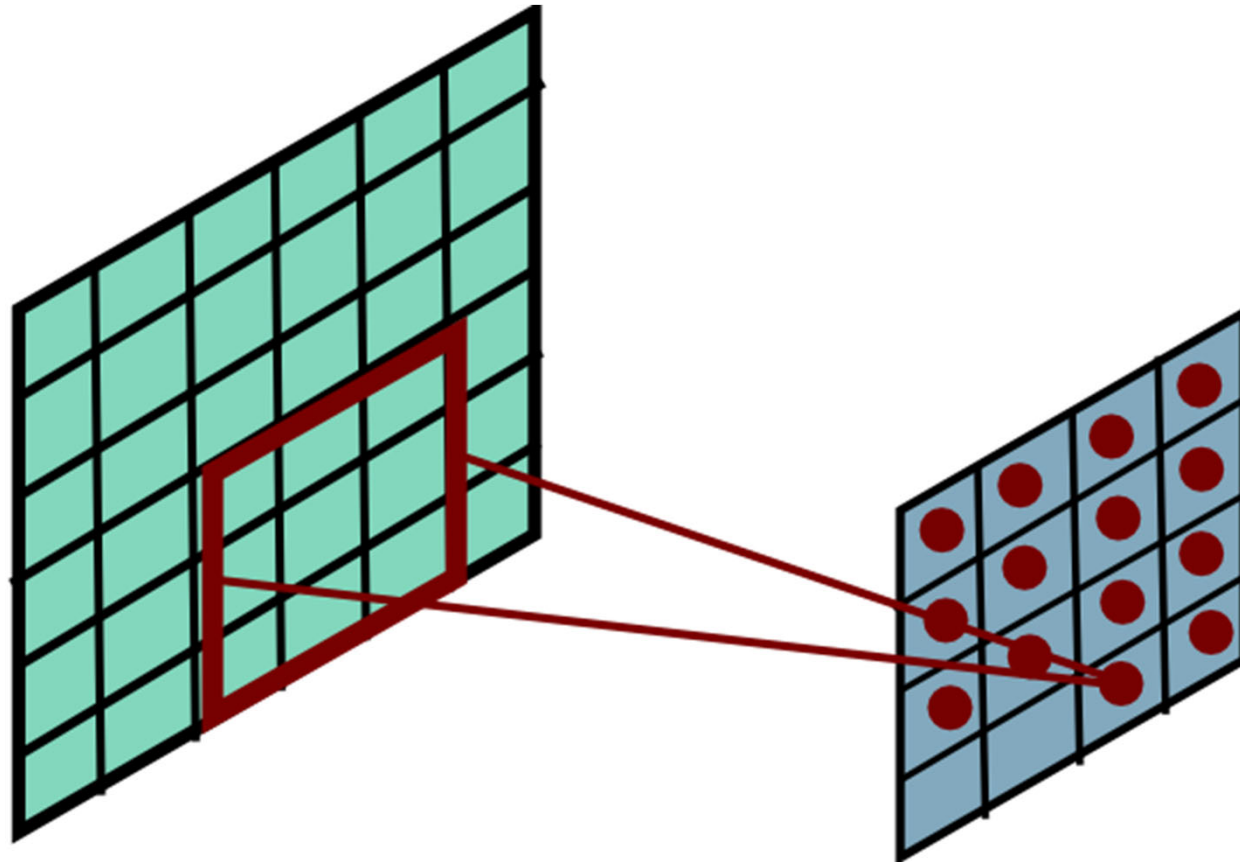
CONVOLUTION OPERATION



شبکه‌های عصبی کانولوشنال

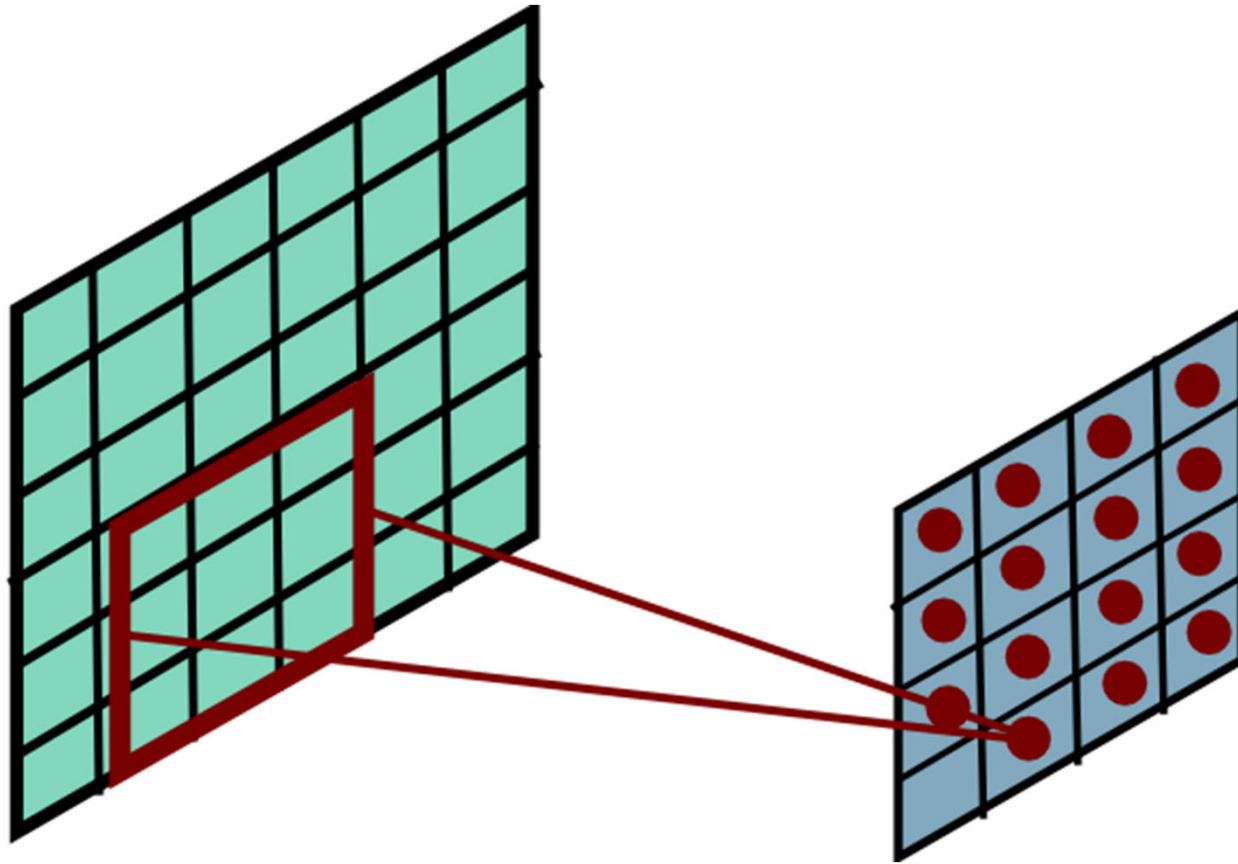
عملیات کانولوشن (۱۴ از ۱۶)

CONVOLUTION OPERATION



شبکه‌های عصبی کانولوشنال

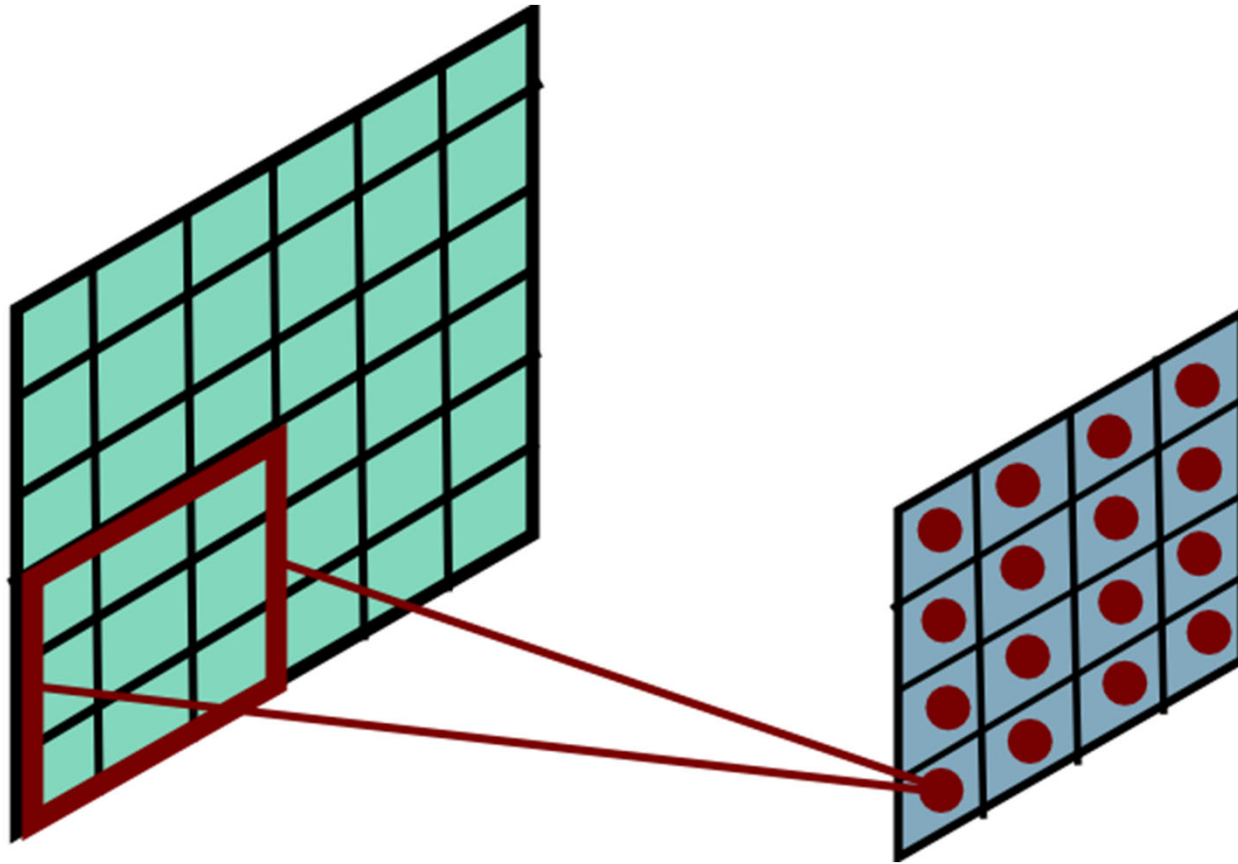
عملیات کانولوشن (۱۵ از ۱۶)

CONVOLUTION OPERATION

شبکه‌های عصبی کانولوشنال

عملیات کانولوشن (۱۶ از ۱۶)

CONVOLUTION OPERATION

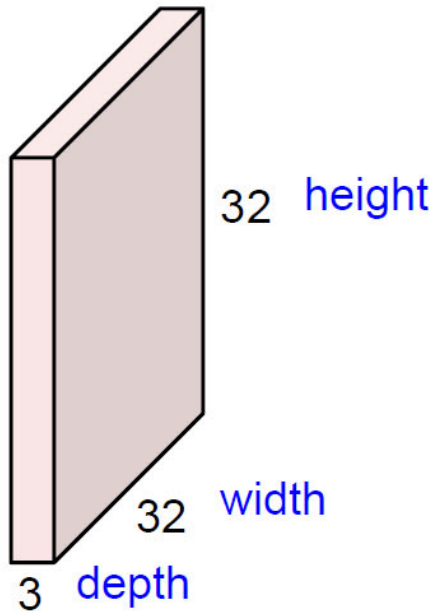


شبکه‌های عصبی کانولوشنال

لایه‌ی کانولوشن

CONVOLUTION LAYER

32x32x3 image -> preserve spatial structure

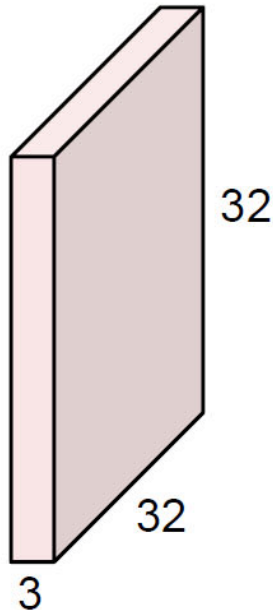


شبکه‌های عصبی کانولوشنال

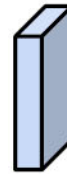
لایه‌ی کانولوشن

CONVOLUTION LAYER

32x32x3 image



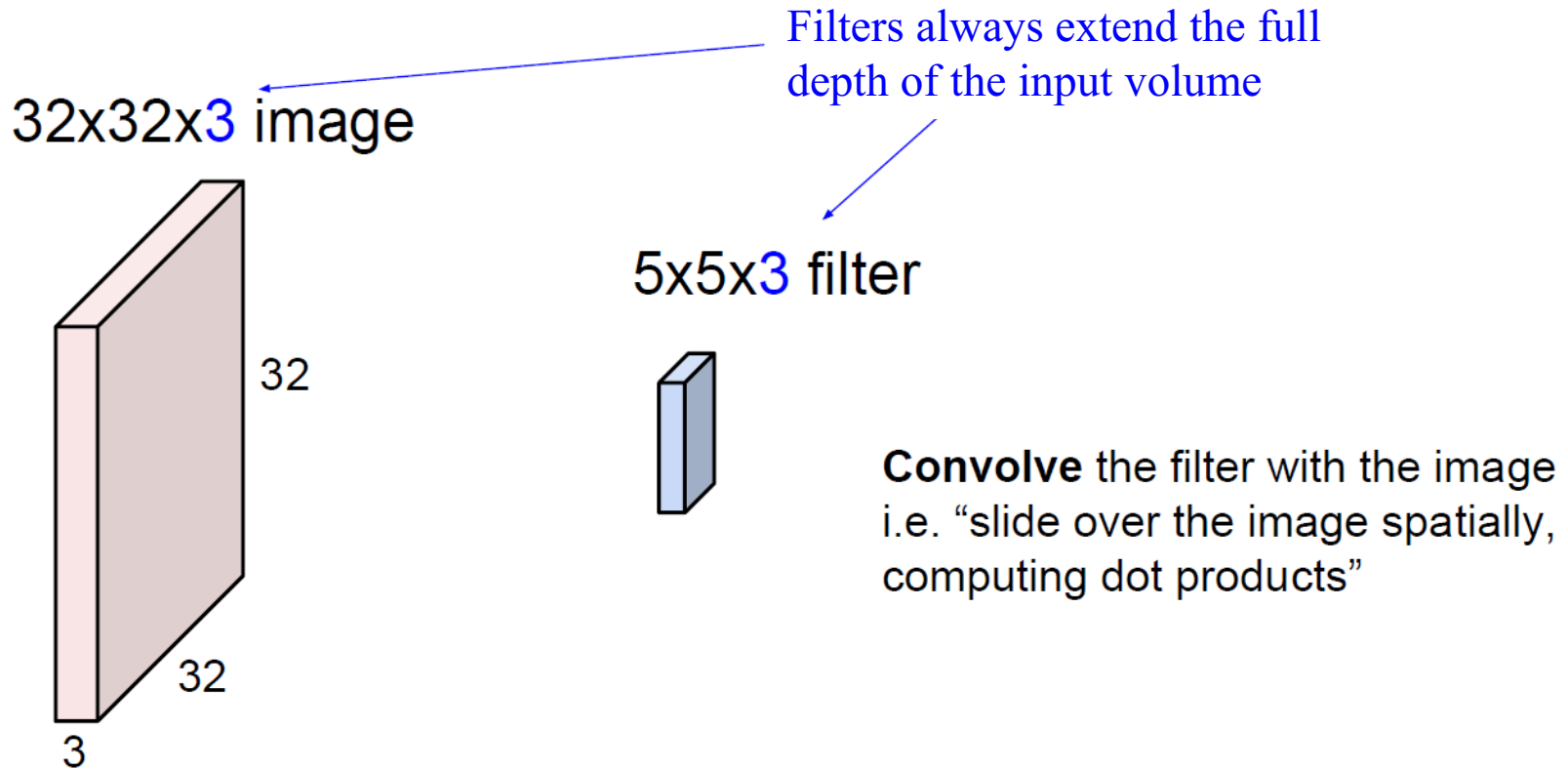
5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

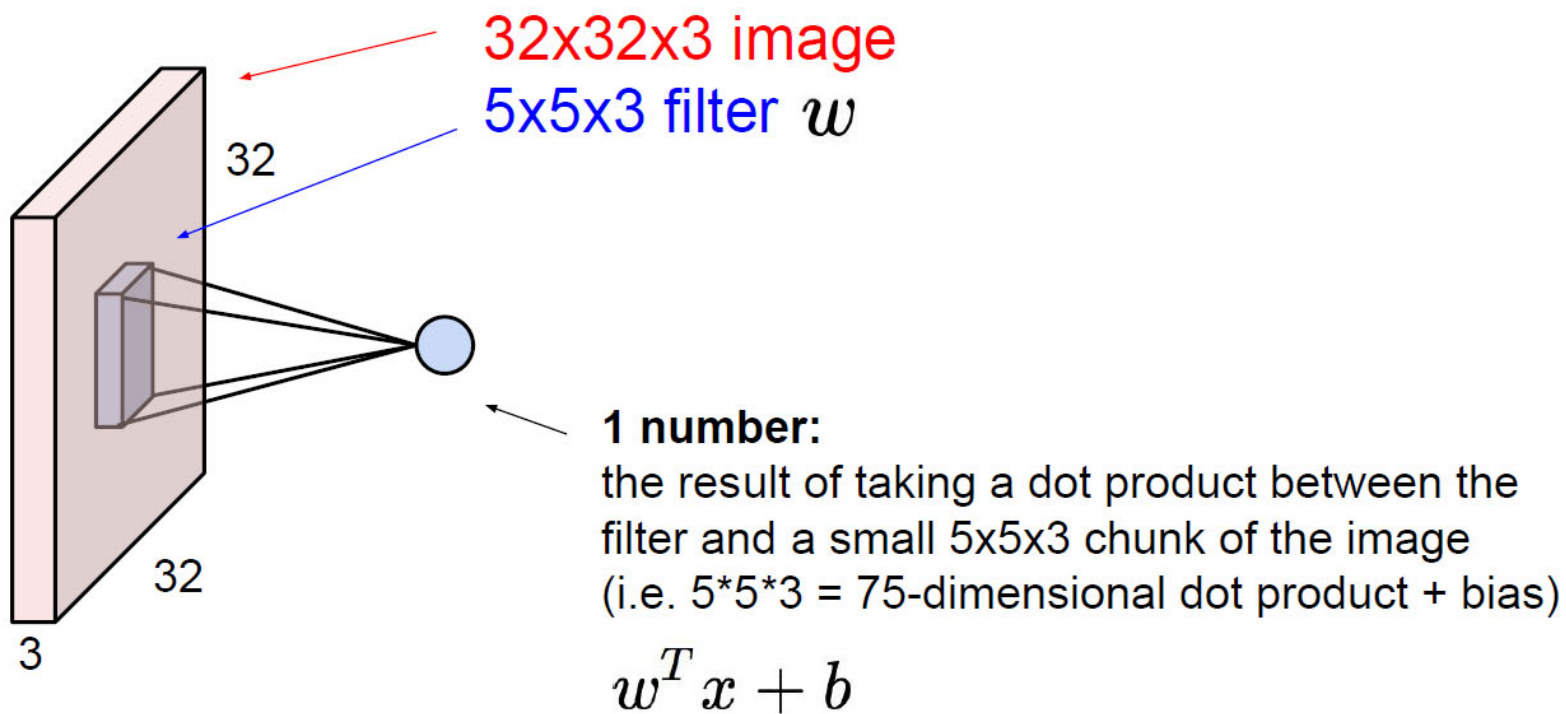
شبکه‌های عصبی کانولوشنال

لایه‌ی کانولوشن

CONVOLUTION LAYER

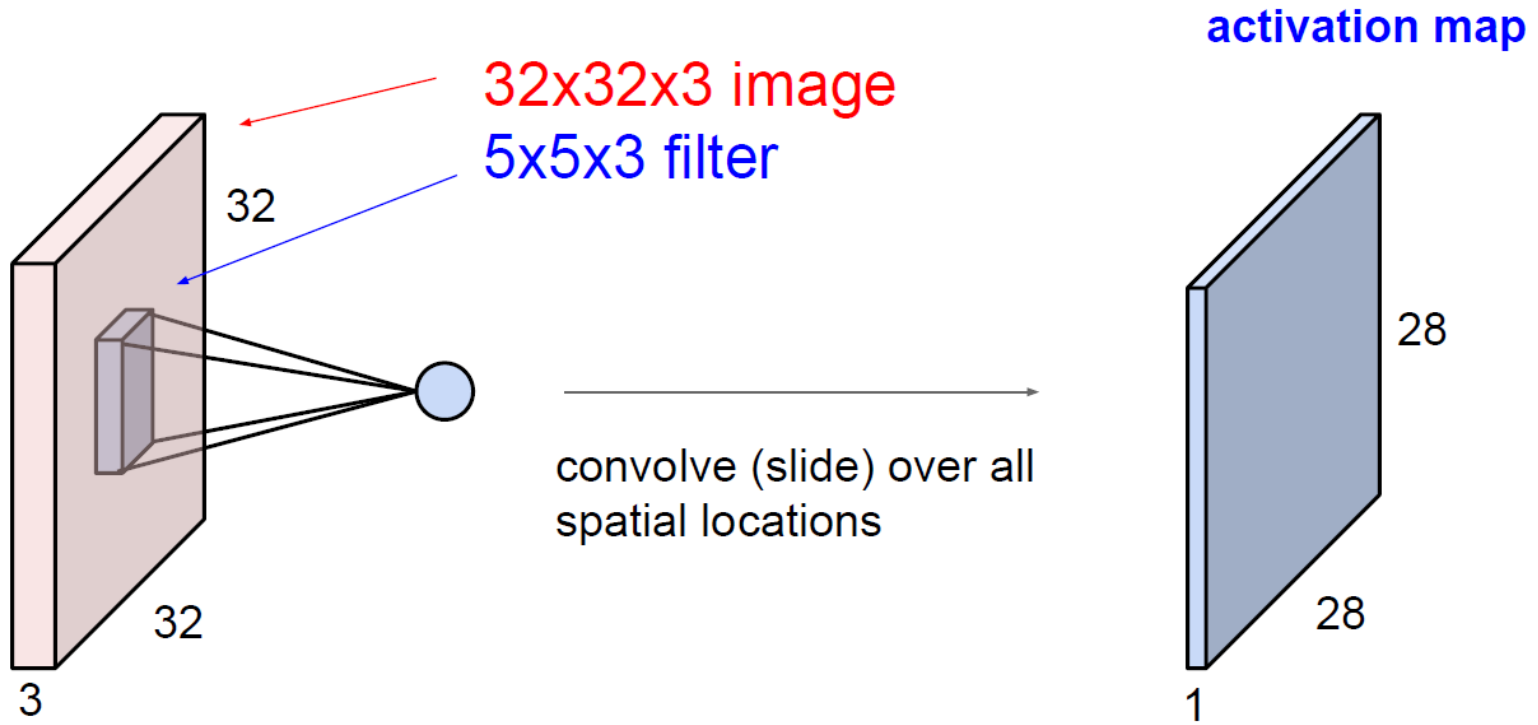
شبکه‌های عصبی کانولوشنال

لایه‌ی کانولوشن

CONVOLUTION LAYER

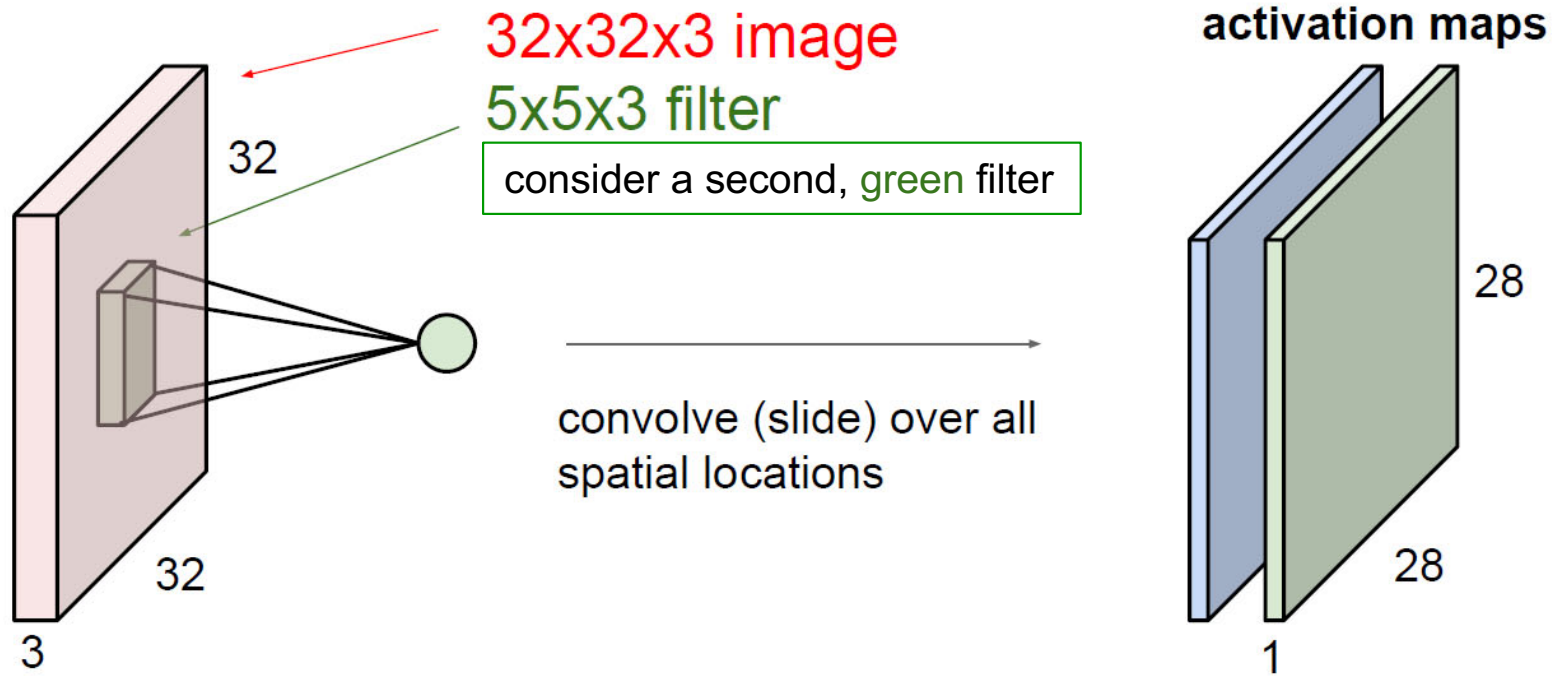
شبکه‌های عصبی کانولوشنال

لایه‌ی کانولوشن

CONVOLUTION LAYER

شبکه‌های عصبی کانولوشنال

لایه‌ی کانولوشن

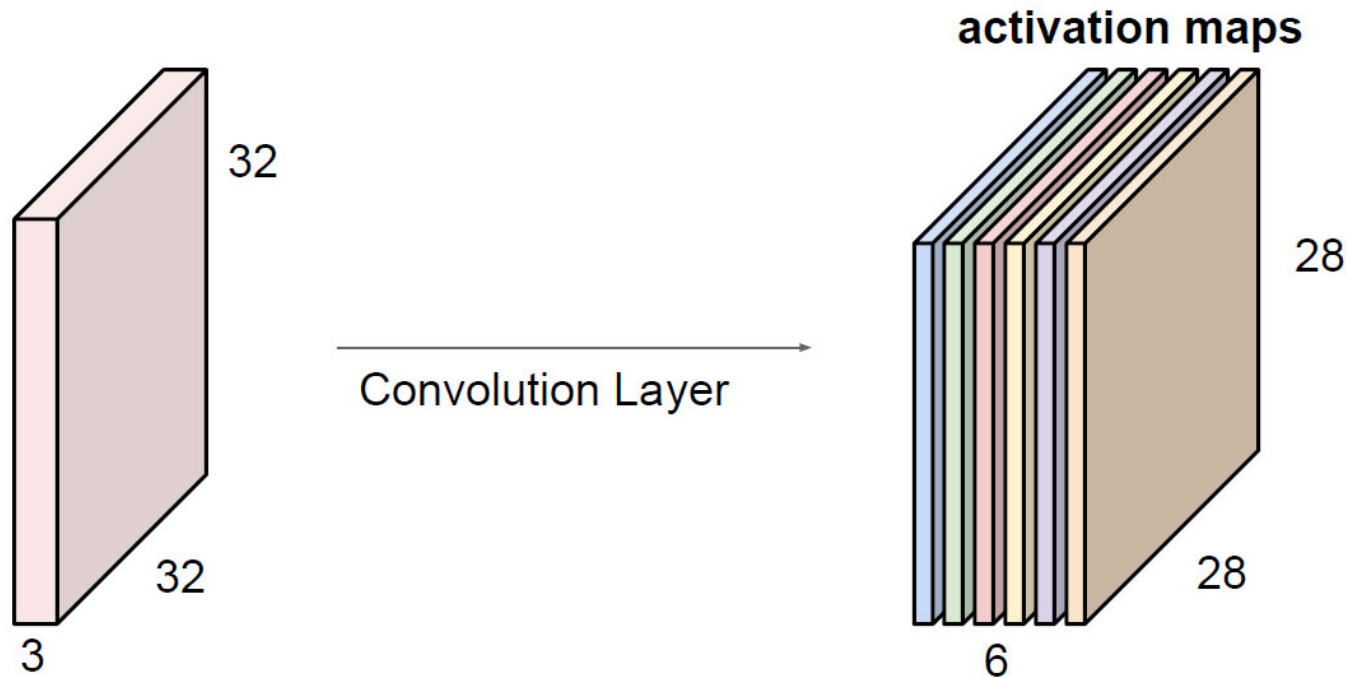
CONVOLUTION LAYER

شبکه‌های عصبی کانولوشنال

لایه‌ی کانولوشن

CONVOLUTION LAYER

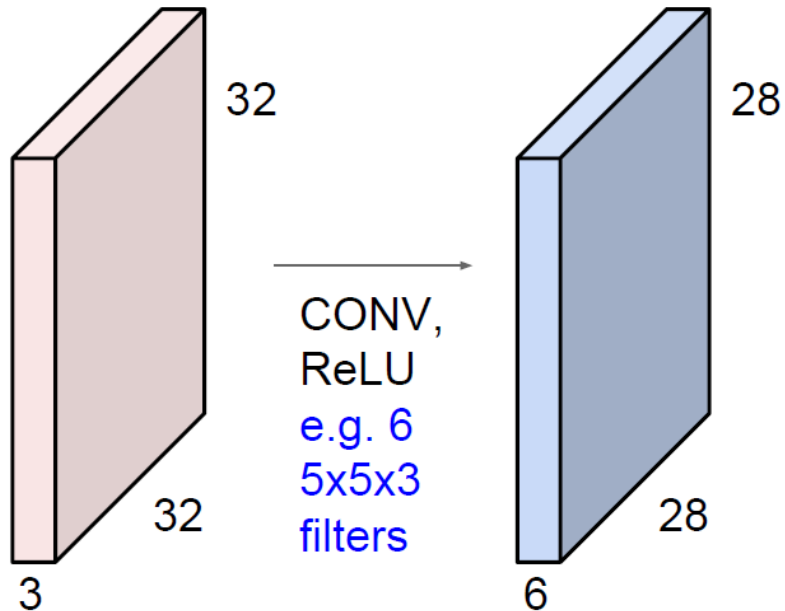
For example, if we had 6 (5x5 filters), we'll get 6 separate activation maps:



We stack these up to get a “new image” of size 28x28x6!

شبکه‌های عصبی کانولوشنال

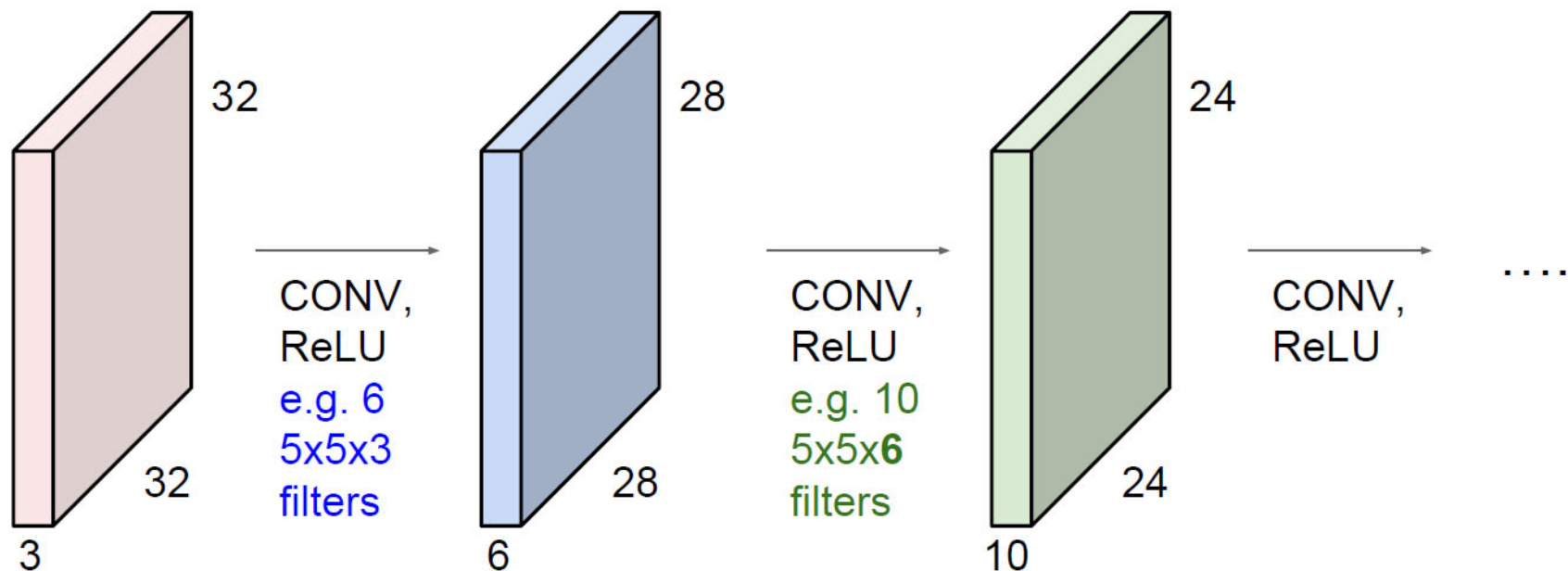
لایه‌ی کانولوشن

CONVOLUTION LAYER

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

شبکه‌های عصبی کانولوشنال

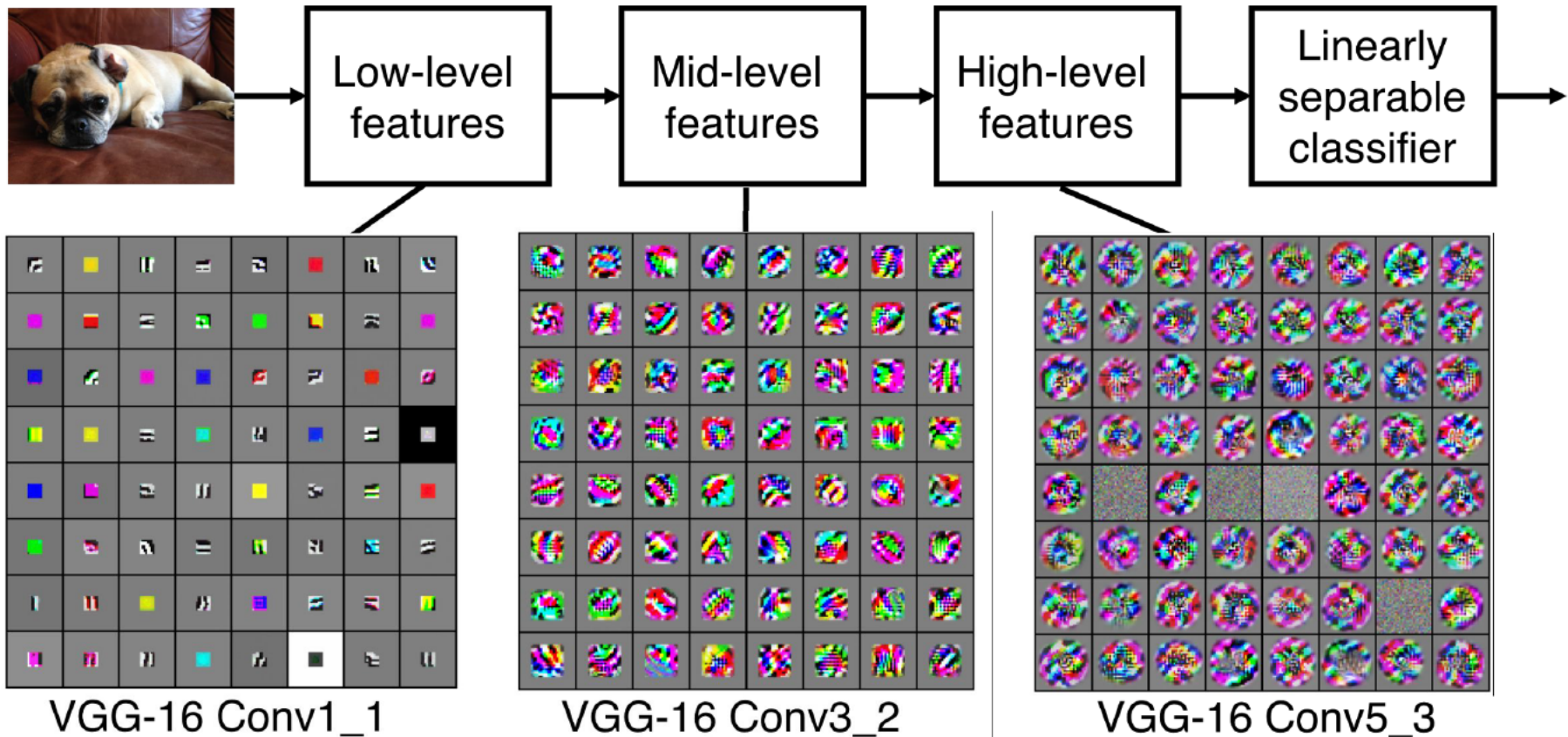
لایه‌ی کانولوشن

CONVOLUTION LAYER

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

شبکه‌های عصبی کانولوشنال

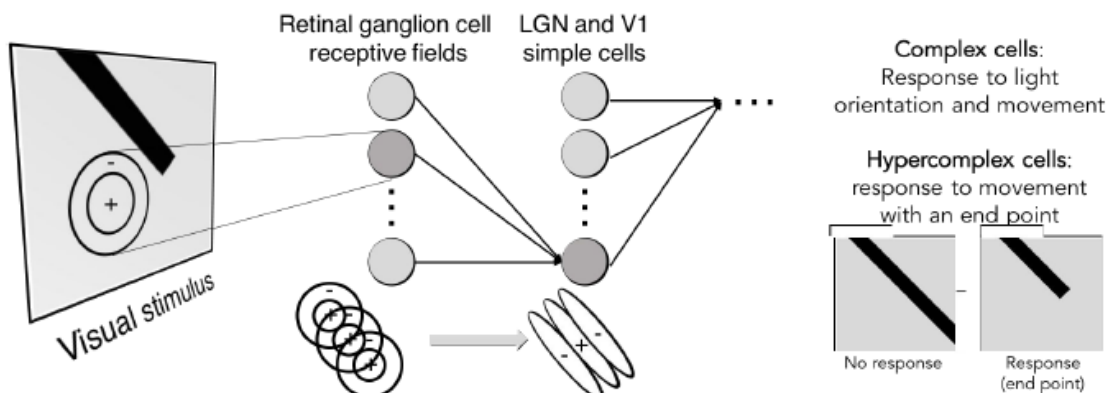
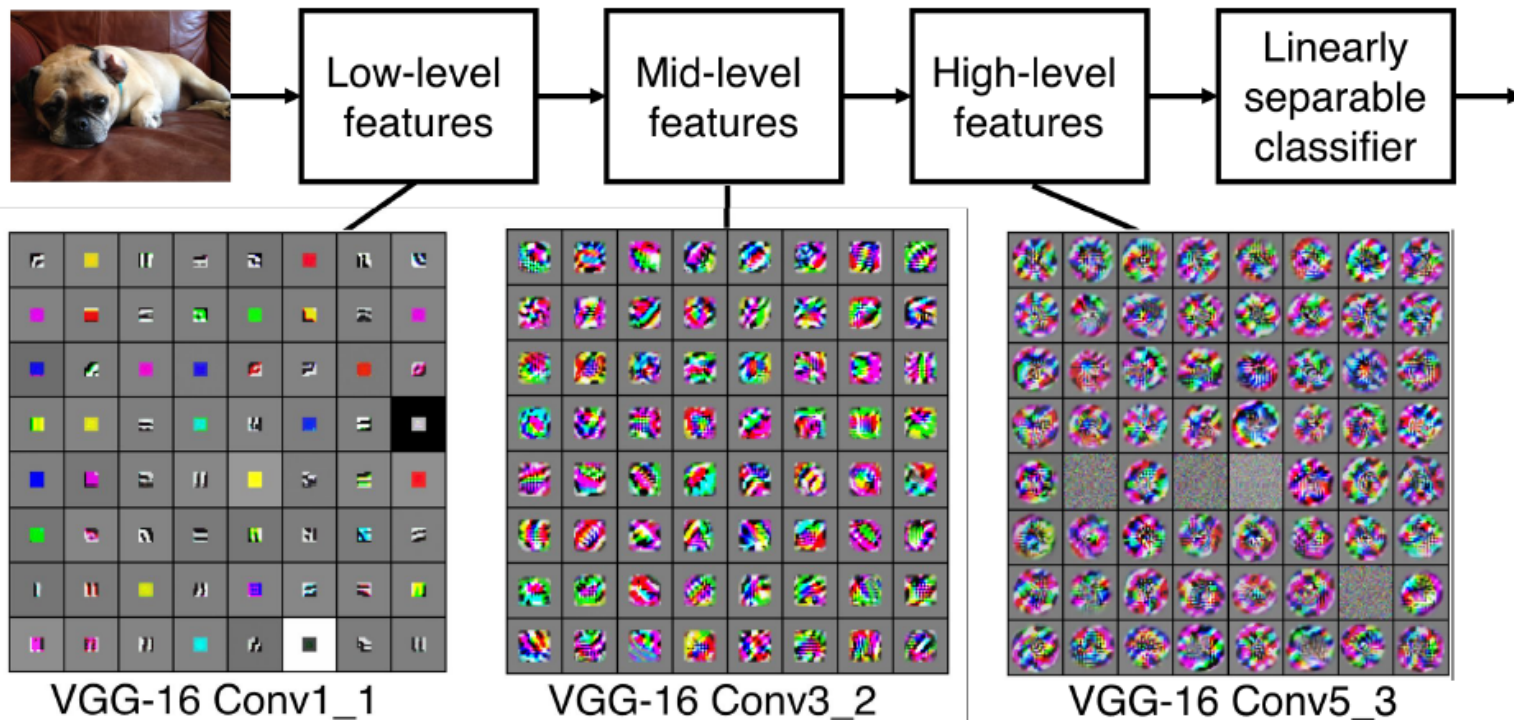
تعبیر لایه‌ها



[Simonyan and Zisserman 2014]

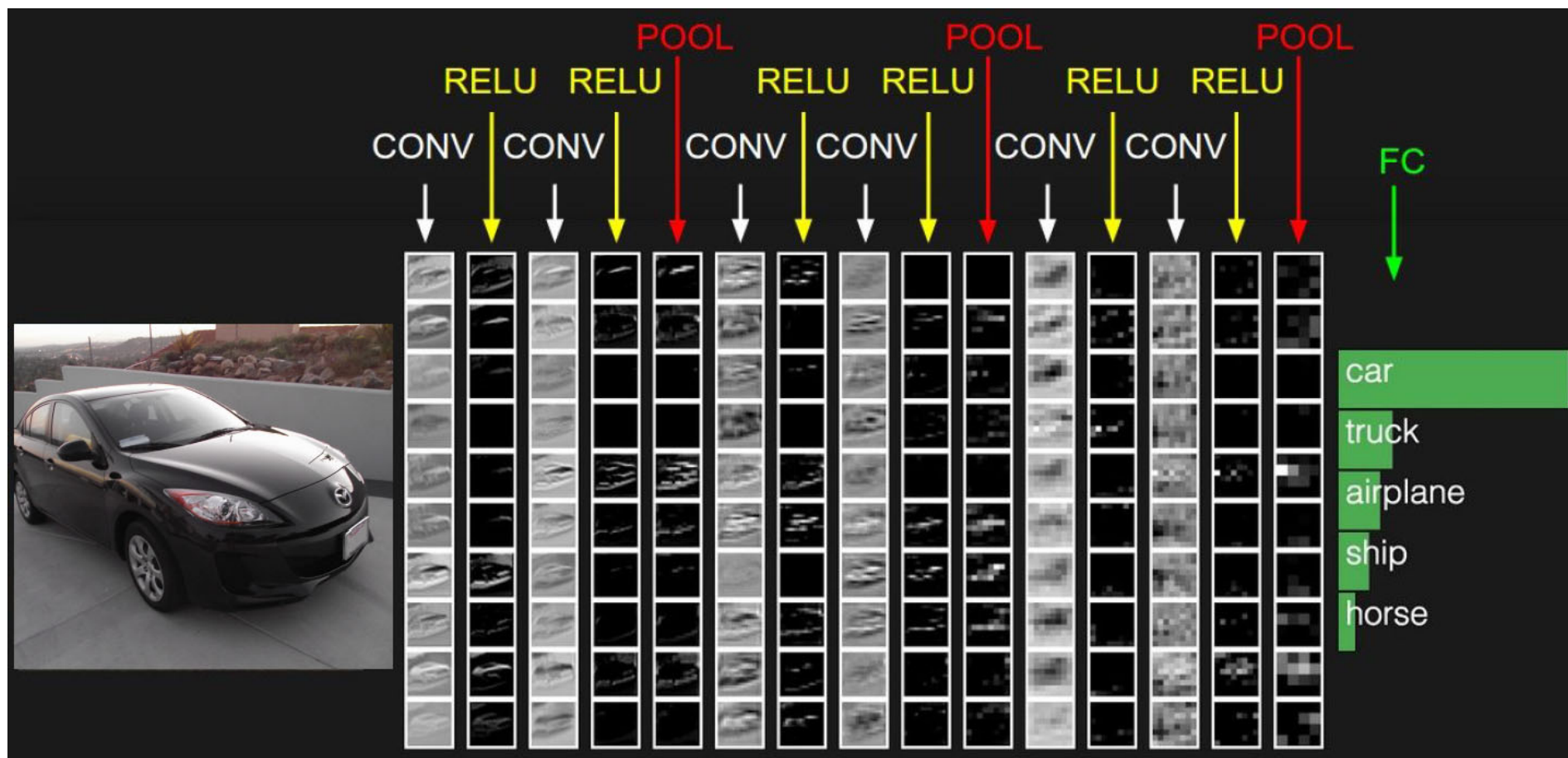
شبکه‌های عصبی کانولوشنال

تعبیر لایه‌ها



شبکه‌های عصبی کانولوشنال

تعبیر لایه‌ها

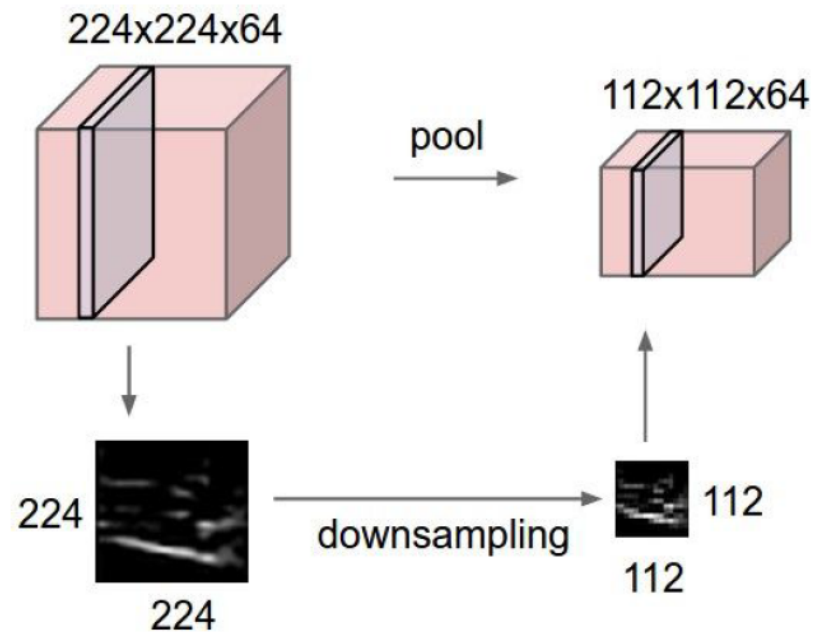


شبکه‌های عصبی کانولوشنال

لایه‌ی تلفیق

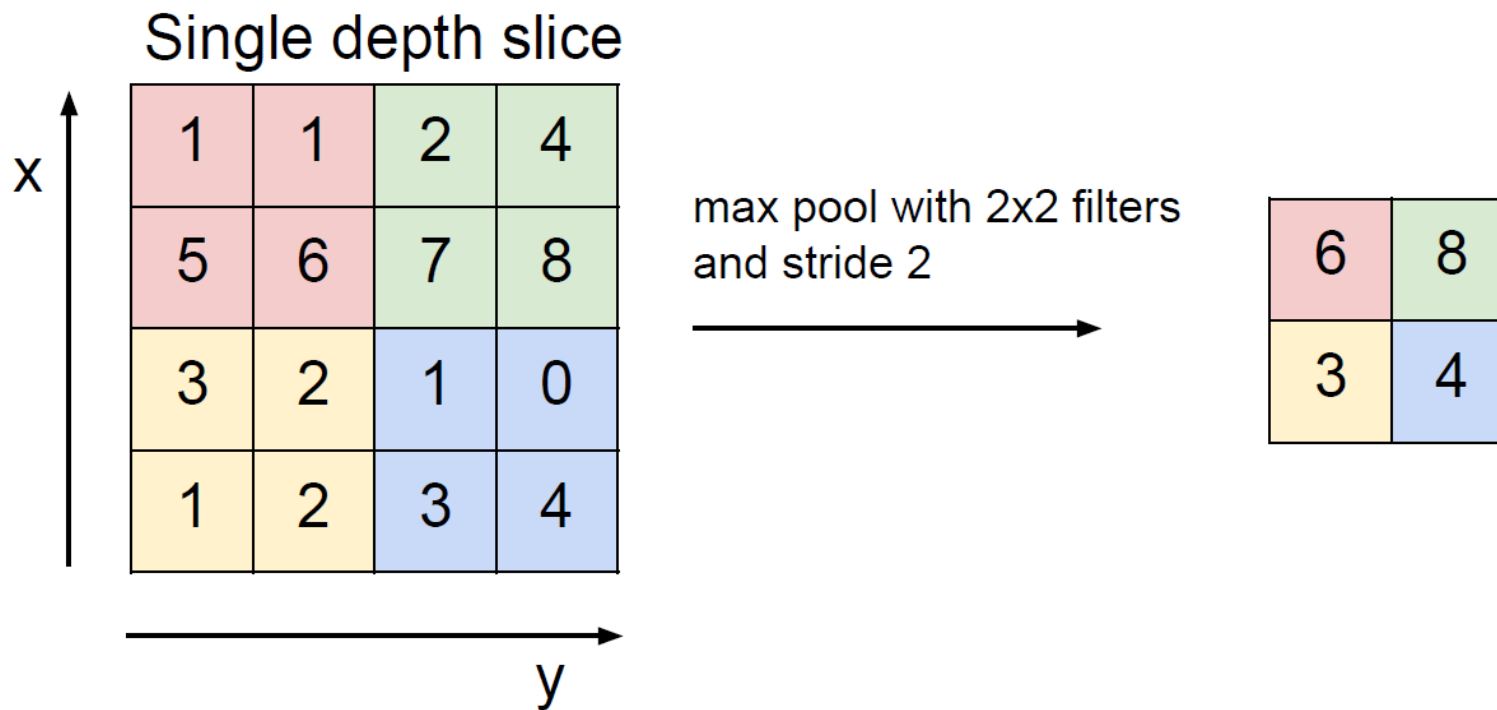
POOLING LAYER

- makes the representations smaller and more manageable
- operates over each activation map independently:



شبکه‌های عصبی کانولوشنال

لایه‌ی تلفیق: مثال

POOLING LAYER

شبکه‌های عصبی کانولوشنال

لایه‌ی تلفیق: روش‌ها

POOLING LAYER

Max-pooling:

$$h_i^n(r, c) = \max_{\bar{r} \in N(r), \bar{c} \in N(c)} h_i^{n-1}(\bar{r}, \bar{c})$$

Average-pooling:

$$h_i^n(r, c) = \text{mean}_{\bar{r} \in N(r), \bar{c} \in N(c)} h_i^{n-1}(\bar{r}, \bar{c})$$

L2-pooling:

$$h_i^n(r, c) = \sqrt{\sum_{\bar{r} \in N(r), \bar{c} \in N(c)} h_i^{n-1}(\bar{r}, \bar{c})^2}$$

L2-pooling over features:

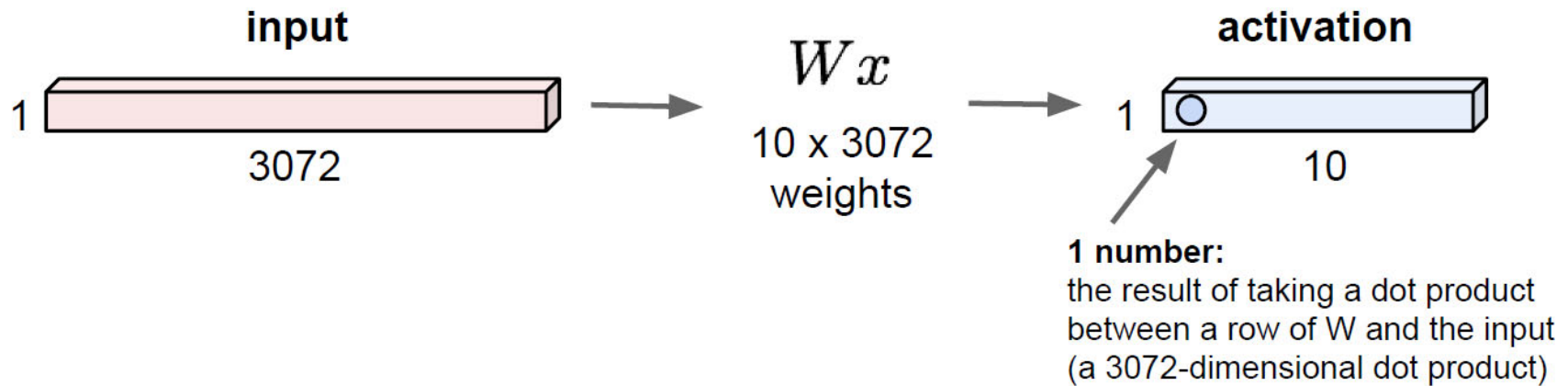
$$h_i^n(r, c) = \sqrt{\sum_{j \in N(i)} h_i^{n-1}(r, c)^2}$$

شبکه‌های عصبی کانولوشنال

لایه‌ی تماماً متصل

FULLY CONNECTED LAYER

32x32x3 image -> stretch to 3072 x 1



Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

[ConvNetJS demo: training on CIFAR-10]

ConvNetJS CIFAR-10 demo

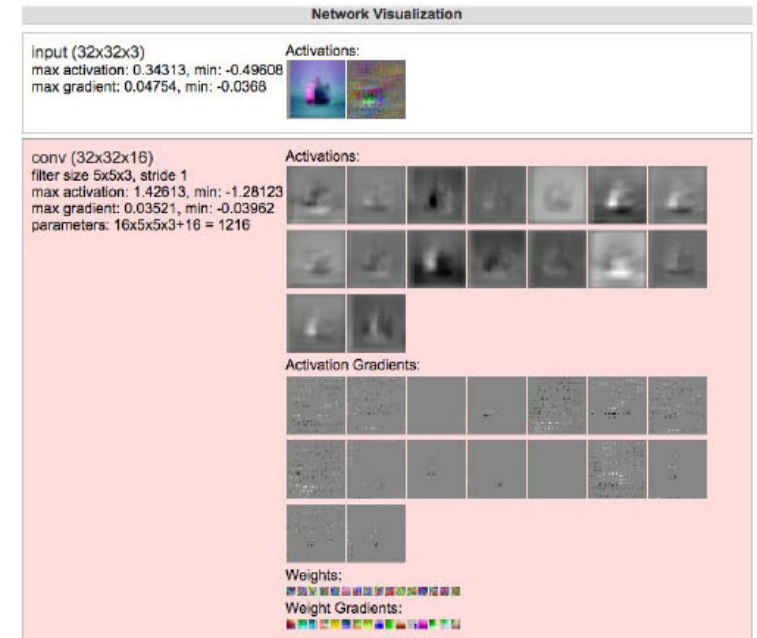
Description

This demo trains a Convolutional Neural Network on the [CIFAR-10 dataset](#) in your browser, with nothing but Javascript. The state of the art on this dataset is about 90% accuracy and human performance is at about 94% (not perfect as the dataset can be a bit ambiguous). I used [this python script](#) to parse the [original files](#) (python version) into batches of images that can be easily loaded into page DOM with img tags.

This dataset is more difficult and it takes longer to train a network. Data augmentation includes random flipping and random image shifts by up to 2px horizontally and vertically.

By default, in this demo we're using Adadelata which is one of per-parameter adaptive step size methods, so we don't have to worry about changing learning rates or momentum over time. However, I still included the text fields for changing these if you'd like to play around with SGD+Momentum trainer.

Report questions/bugs/suggestions to [@karpathy](#).



<http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html>

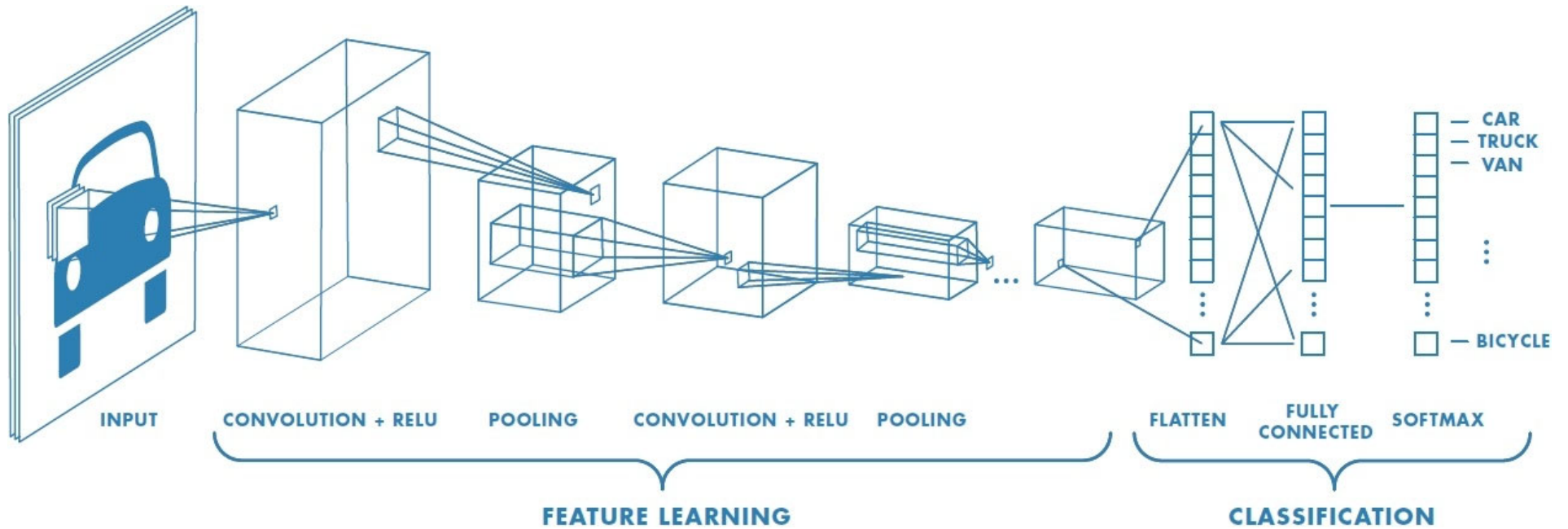
شبکه‌های عصبی کانولوشنال

۲

نمونه‌ها

شبکه‌های عصبی کانولوشنال

معماری

ARCHITECTURE

شبکه‌های عصبی کانولوشنال برای بازشناسی ارقام دست‌نویس

CONVOLUTIONAL NETWORKS FOR OCR

- **Task:** Correctly recognize the image of **hand-written digits**.
- **Dataset:** MNIST (28*28 image), **60000** Training samples, **10000** Test.



LeNet-5

CONVOLUTIONAL NETWORKS FOR OCR

- Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner, 1998]
- Three key ideas:
 - Local Receptive Fields (**C**onvolution, Filters),
 - Shared Weights,
 - Sub-sampling (**P**ooling).

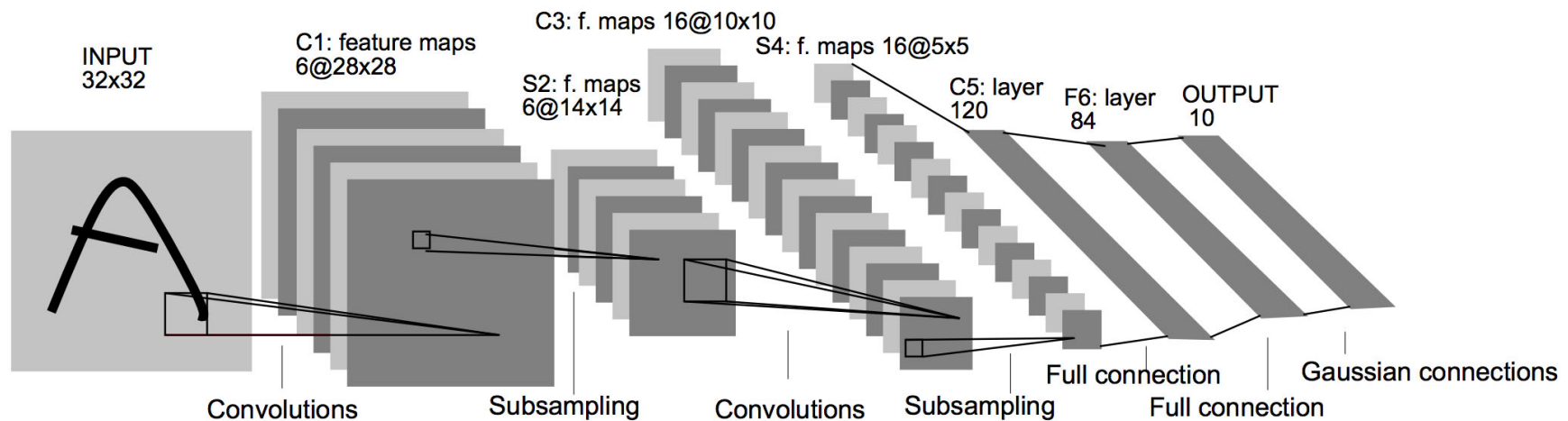


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

شبکه‌های عصبی کانولوشنال برای بازشناسی ارقام دست‌نویس

LeNet-5

INPUT

- **Input:** 32×32 pixel image.
- Largest character is 20×20 (All important info should be in the center of the receptive field of the highest level feature detectors)
- Black and White pixel values are normalized:
E.g. White = -0.1 , Black = 1.175 (Mean of pixels = 0 , Std of pixels = 1)

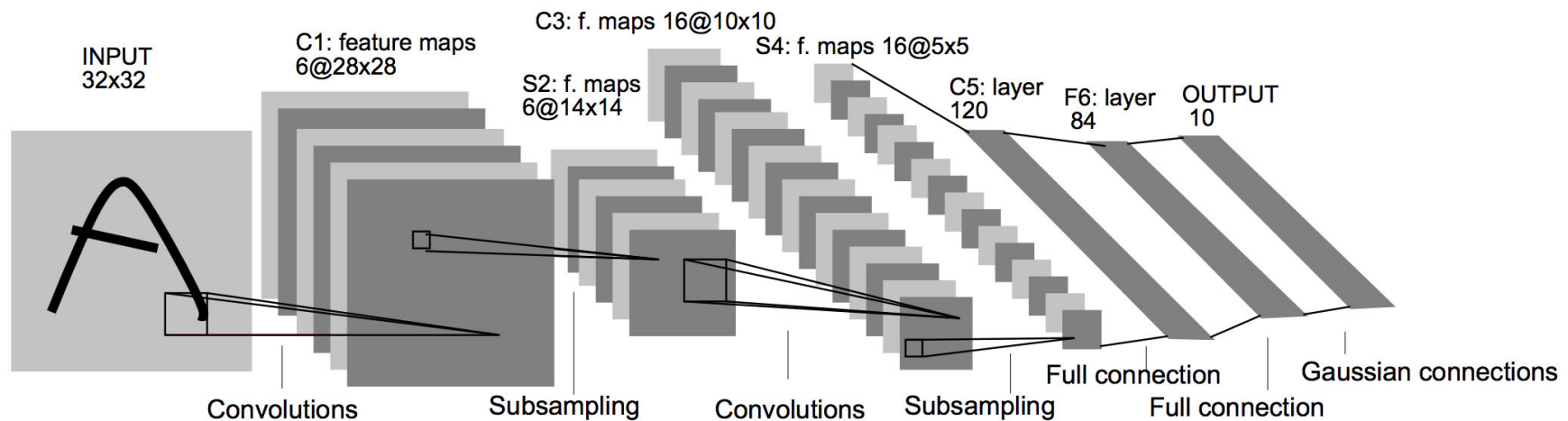


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

شبکه‌های عصبی کانولوشنال برای بازشناسی ارقام دست‌نویس

LeNet-5

LAYER C1

- **C1**: Convolutional layer with 6 feature maps of size 28×28 . $C1_k$ ($k = 1..6$)
 - Formula for output size = $(N - F + 2P) / S + 1$
 - N : input size, F : size of the filter, P : Padding, S : Stride
- Each unit of C1 has a 5×5 receptive field in the input layer.
- $(5 \times 5 + 1) \times 6 = 156$ parameters to learn
- Connections: $28 \times 28 \times (5 \times 5 + 1) \times 6 = 122304$
- If it was fully connected we had $(32 \times 32 + 1) \times (28 \times 28) \times 6 = 4,821,600$ parameters

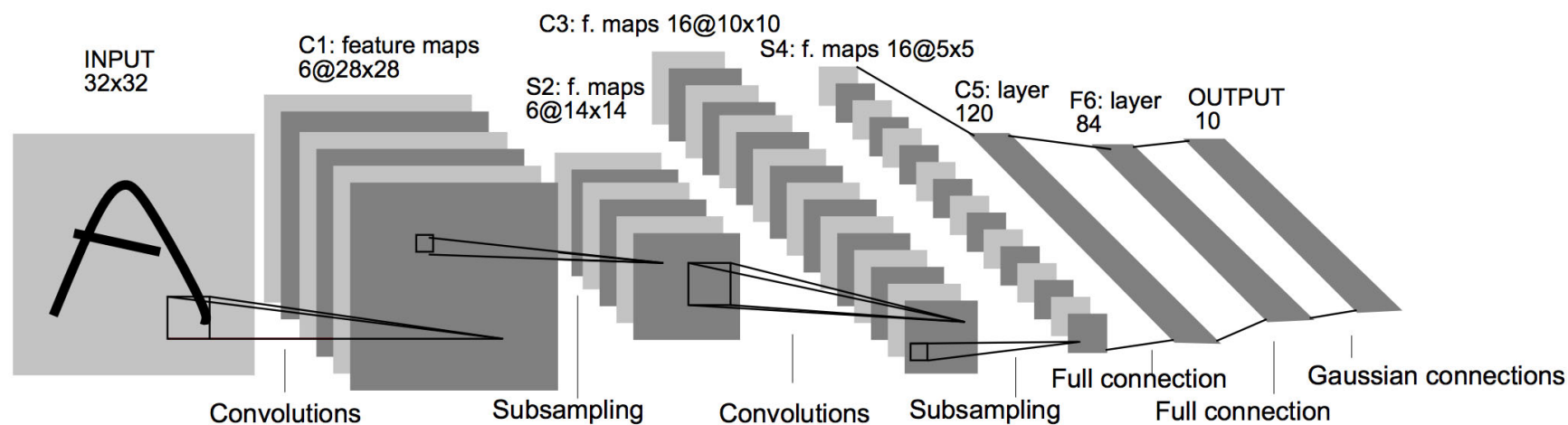


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

شبکه‌های عصبی کانولوشنال برای بازشناسی ارقام دست‌نویس

LeNet-5

LAYER S2

- **S2**: Subsampling layer with 6 feature maps of size 14×14 , 2×2 non overlapping receptive fields in C1 Layer
- Averages the input, multiplies by a coefficient and adds a bias.
 - Both these coefficient and bias are learnt.
- **S2**: $6 \times 2 = 12$ trainable parameters.
- Connections: $14 \times 14 \times (2 \times 2 + 1) \times 6 = 5880$

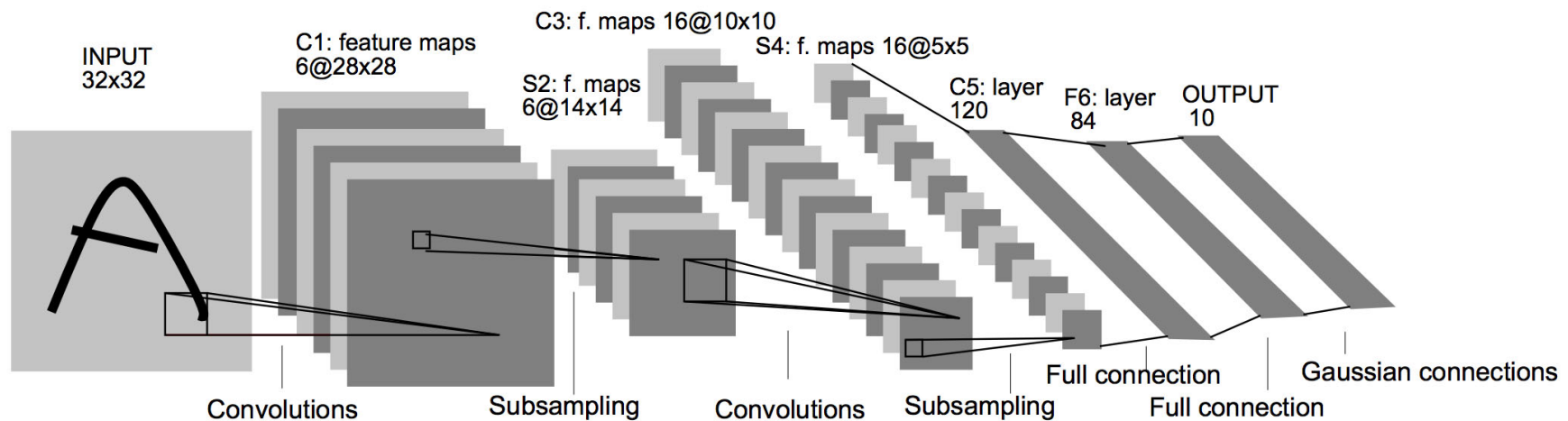


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

شبکه‌های عصبی کانولوشنال برای بازشناسی ارقام دست‌نویس

LeNet-5

LAYER C3

- **C3**: Convolutional layer with 16 feature maps of size 10×10
- Each unit in C3 is connected to several! 5×5 receptive fields at identical locations in S2
- Layer C3: 1516 trainable parameters.
- Connections: 151600

	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		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.

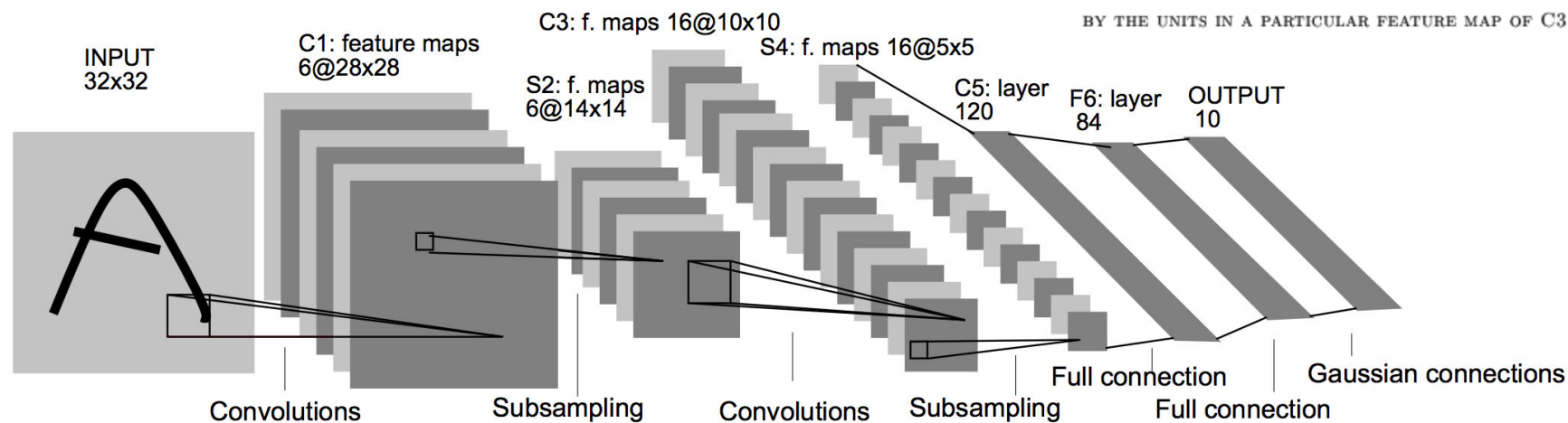


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

شبکه‌های عصبی کانولوشنال برای بازشناسی ارقام دست‌نویس

LeNet-5

LAYER S4

- **S4**: Subsampling layer with 16 feature maps of size 5×5
- Each unit in S4 is connected to the corresponding 2×2 receptive field at C3
- Layer S4: $16 \times 2 = 32$ trainable parameters.
- Connections: $5 \times 5 \times (2 \times 2 + 1) \times 16 = 2000$

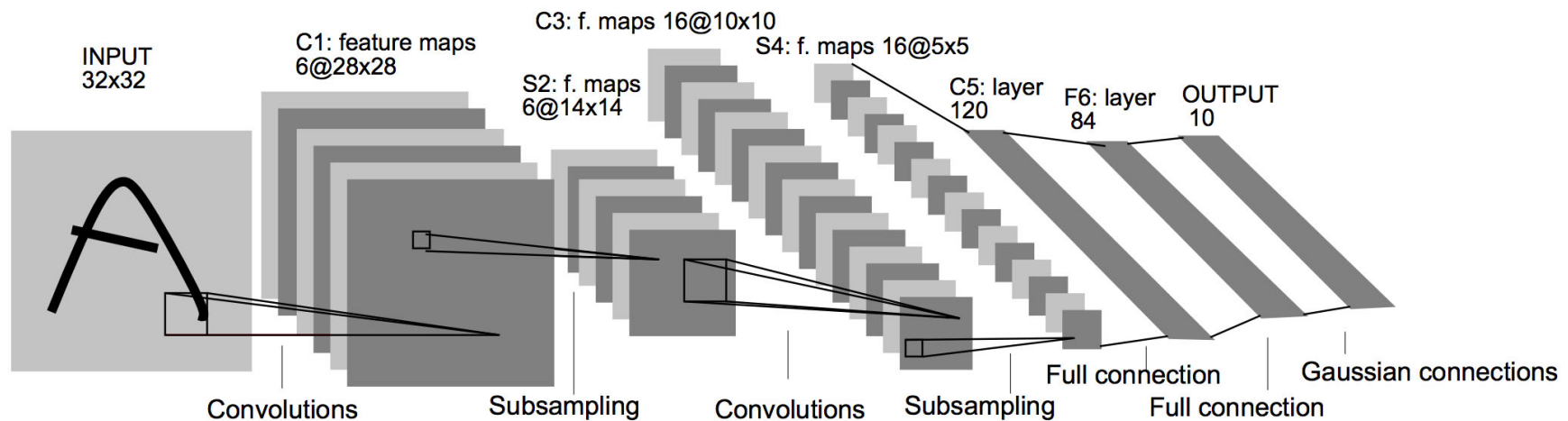


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

شبکه‌های عصبی کانولوشنال برای بازشناسی ارقام دست‌نویس

LeNet-5

LAYER C5

- **C5**: Fully connected layer with 120 feature maps of size 1×1
- Each unit in C5 is connected to all 16, 5×5 receptive fields in S4
- Layer C5: $120 \times (16 \times 25 + 1) = 48120$ trainable parameters and connections (Fully connected)

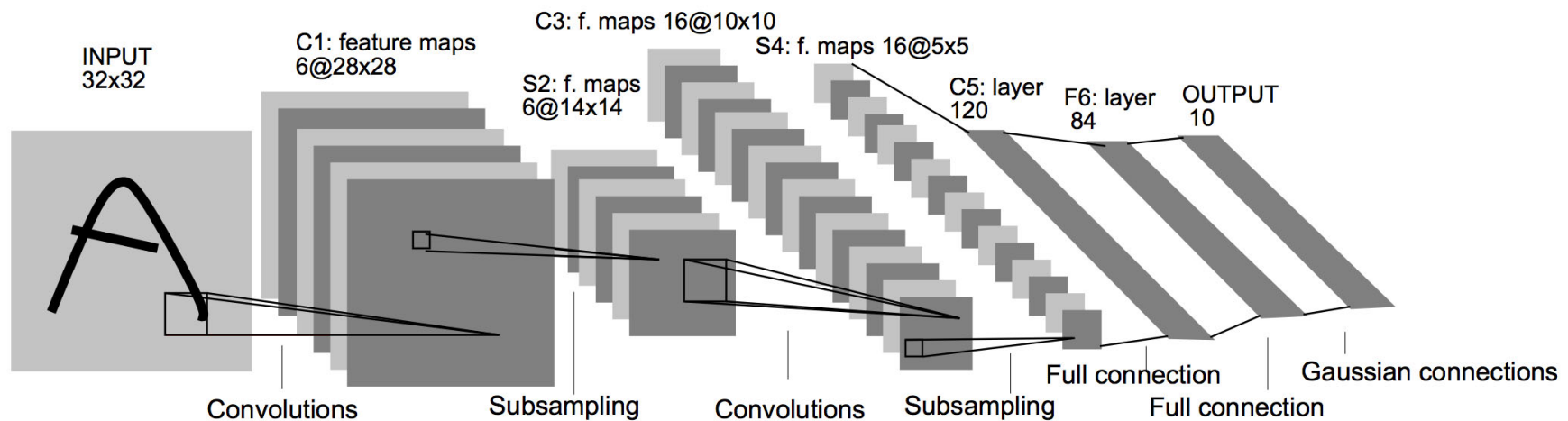


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

شبکه‌های عصبی کانولوشنال برای بازشناسی ارقام دست‌نویس

LeNet-5

LAYER F6

- **F6:** 84 fully connected units.
 $84 \times (120 + 1) = 10164$ trainable parameters and connections.
- Output layer: 10 RBF (One for each digit) $84 = 7 \times 12$, stylized image
- Weight update: **Backpropagation**

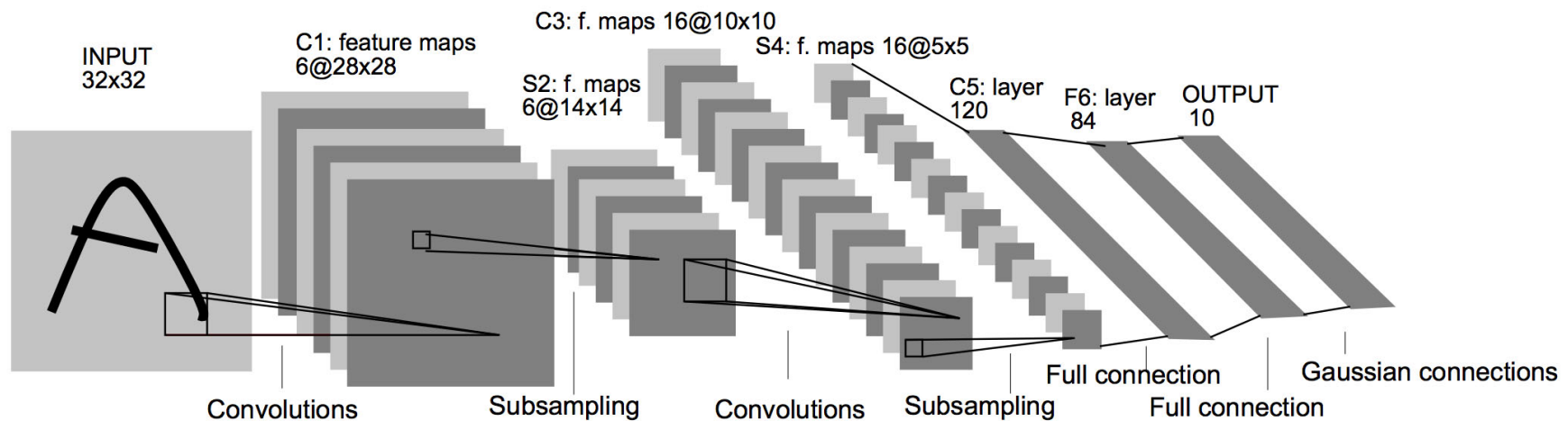


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

شبکه‌های عصبی کانولوشنال برای بازشناسی ارقام دست‌نویس

LeNet-5

RESULTS

- **LeNet** achieves **0.8%** error rate on test set.
- SVM got 1.1%, a slightly modified version known as v-SVM got 0.8%. but the computation cost is very high. It is almost twice as expensive as LeNet.
- LeNet was trained for 2-3 days on CPU (200 MHz processor).
- Currently it can be trained in less than 30 minutes.

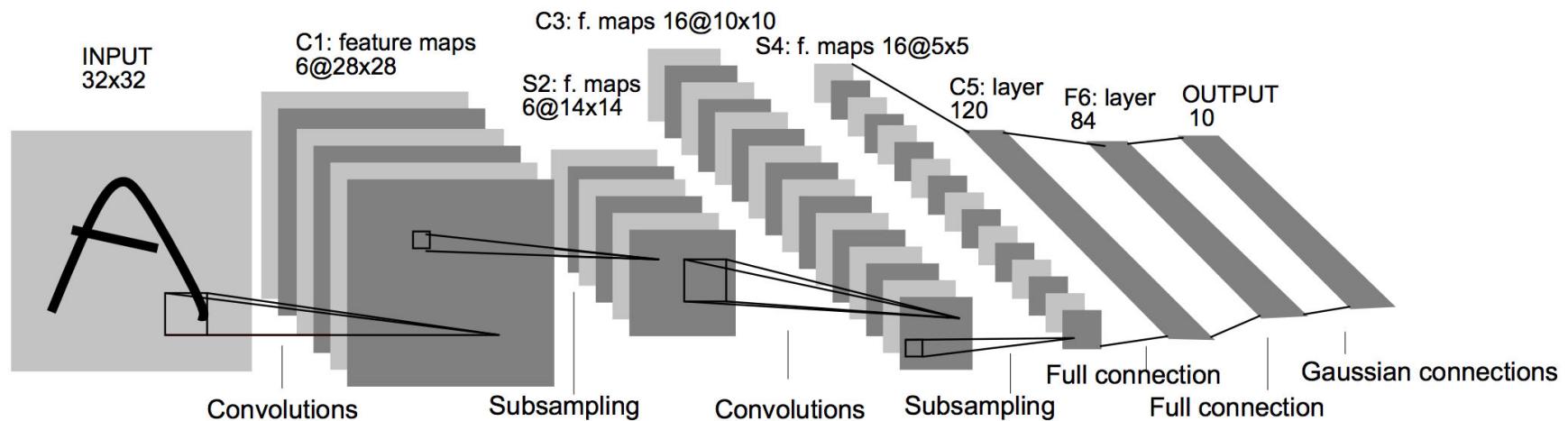


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

ImageNet

CONVOLUTIONAL NETWORKS FOR IMAGE CLASSIFICATION

هدف: بازشناسی اشیای موجود در یک تصویر



Tags	Confidence
racer, race car, racing car	0.7962
sports car, sport car	0.1341
cab, hack, taxi, taxicab	0.0278
convertible	0.0165
car wheel	0.0080

شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

IMAGENET

- Over 15M labeled high resolution images.
- Roughly 22K categories
- Collected from web and labeled by Amazon Mechanical Turk.



شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

ImageNet

IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC)

- Annual competition of image classification at large scale.
- 1.2M training images in 1K categories.
- 50K validation images, 150K testing images.
- Classification: make 1 (Top-1 error) /5 (Top-5 error) guesses about the image label.



شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

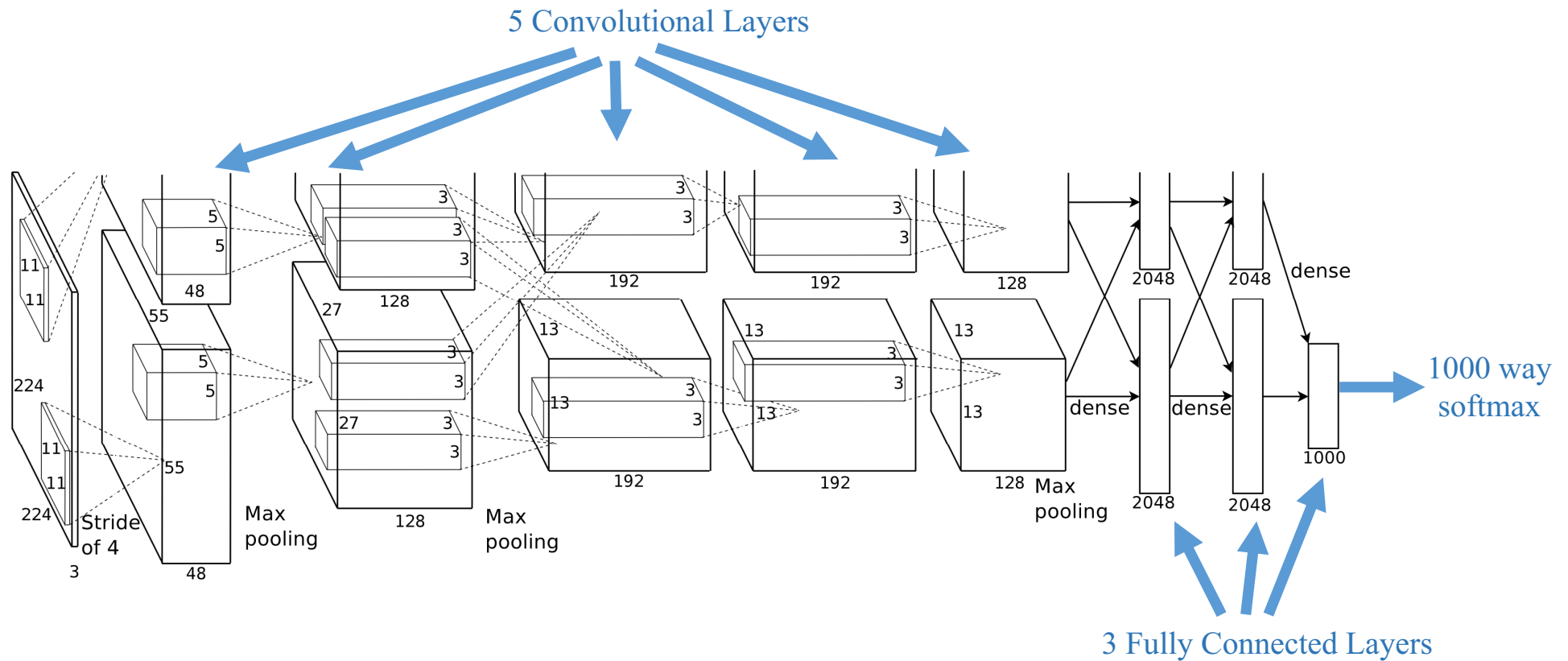
ImageNet

ALEXNET

- Similar framework to LeCun'98 but,
- Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
- More data (10^6 vs. 10^3 images)
- GPU implementation (50x speedup over CPU)
- Trained on two GPUs for a week
- Better regularization for training (DropOut)

شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

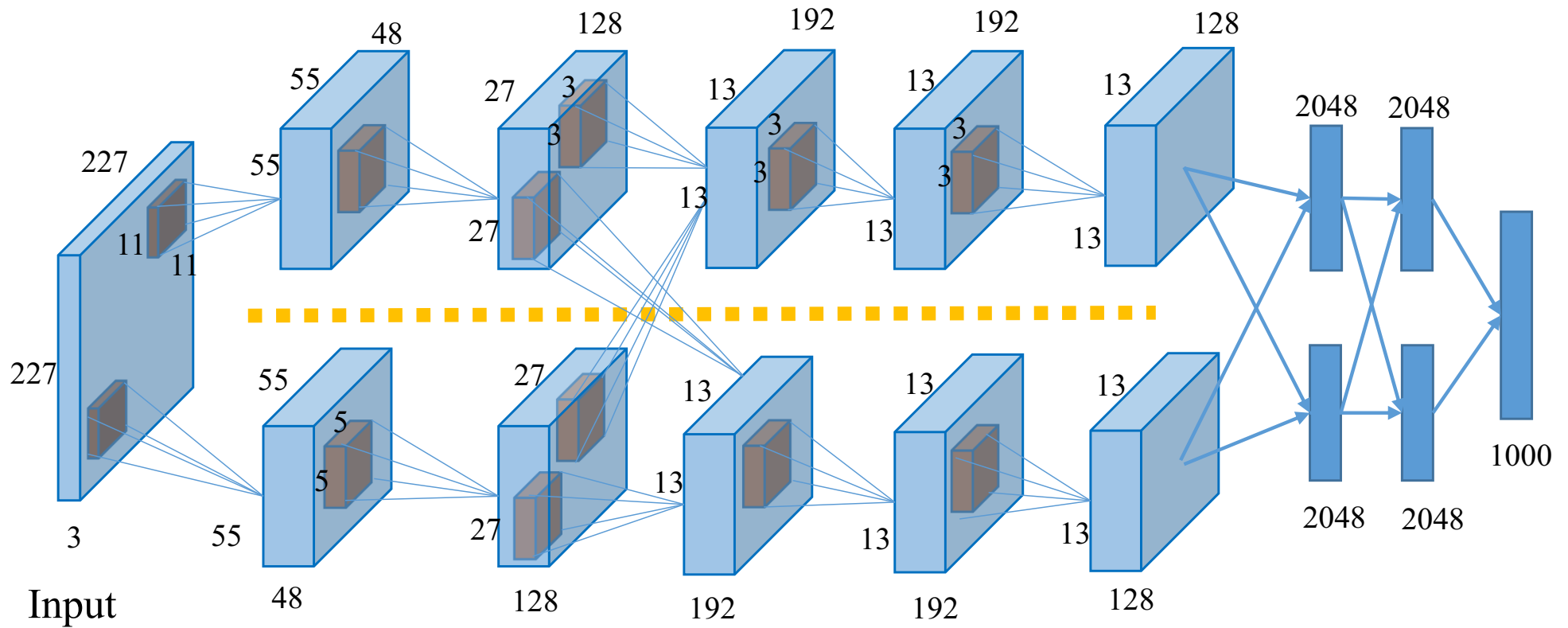
ImageNet

ALEXNET: ARCHITECTURE

شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

ImageNet

ALEXNET: ARCHITECTURE

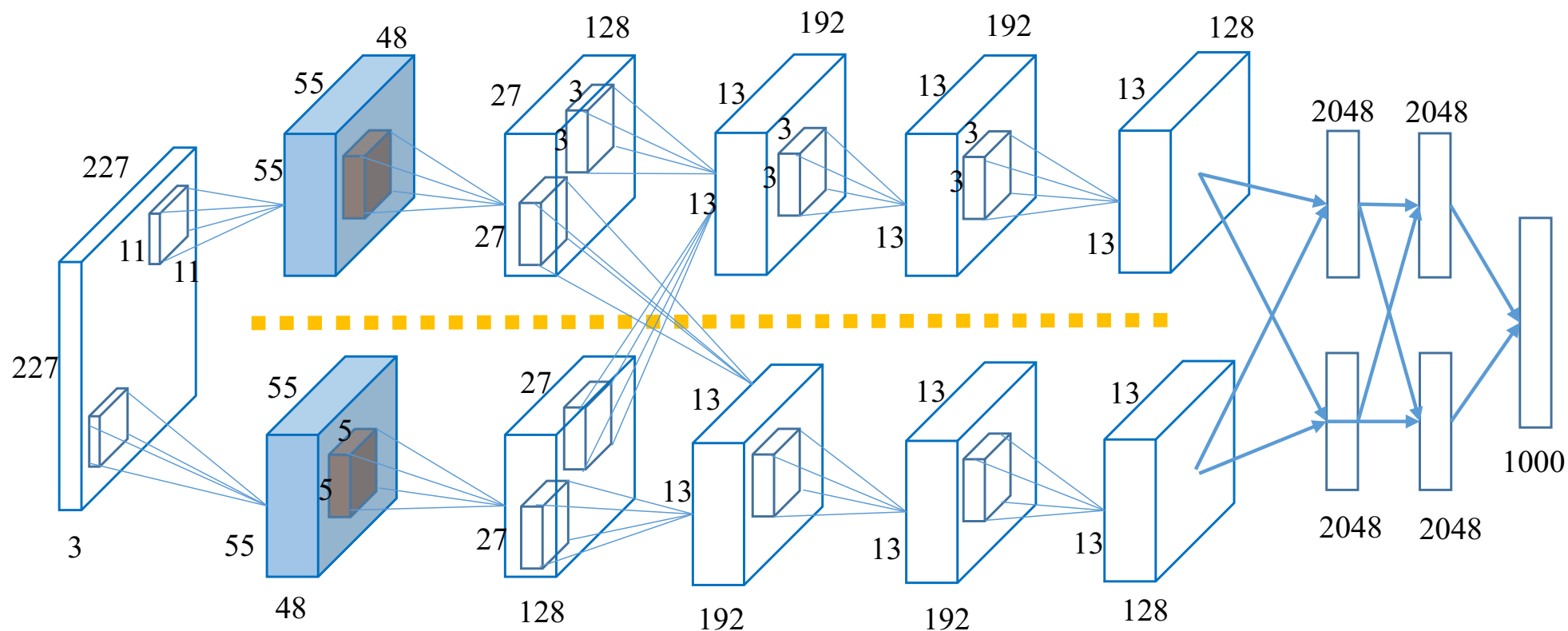


شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

ImageNet

ALEXNET: ARCHITECTURE

- $55 \times 55 \times 96 = 290,400$ neurons, each having $11 \times 11 \times 3 = 363$ weights + 1 bias
- $290400 \times 364 = 105,705,600$ parameters in first layer alone if fully connected.

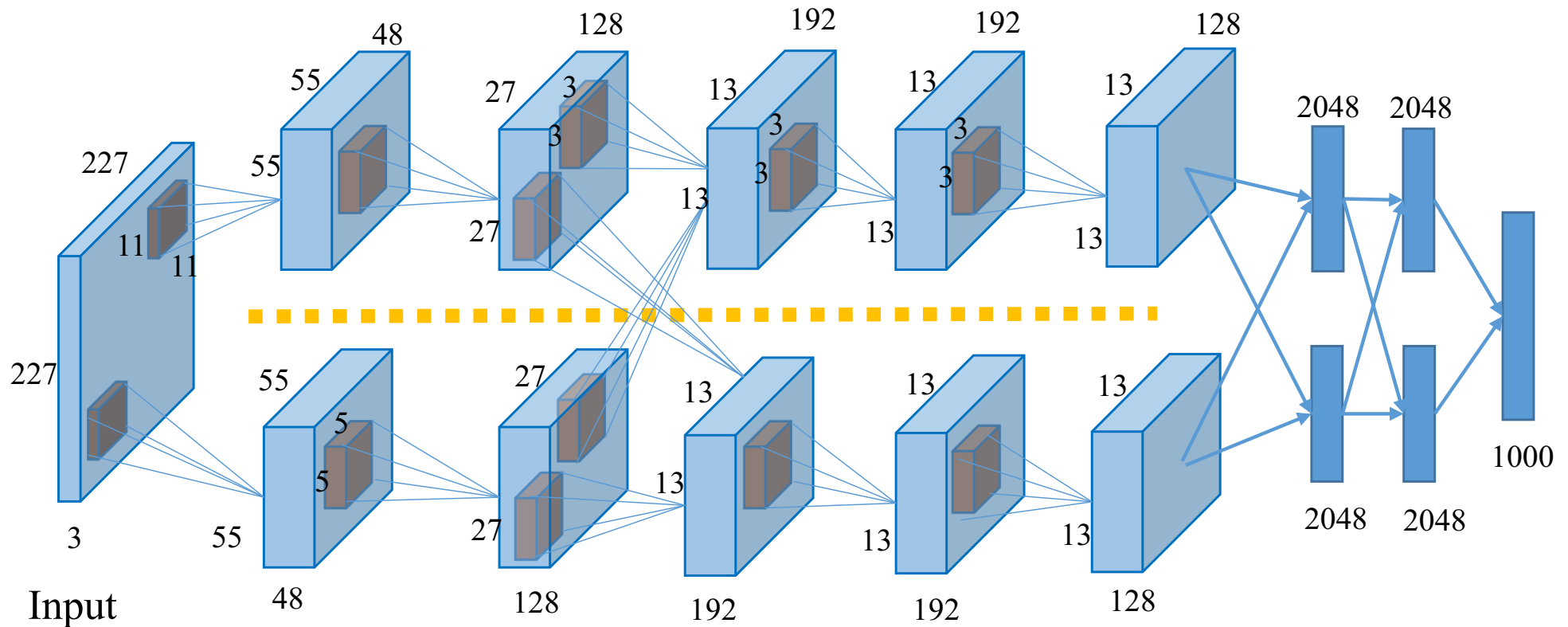


شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

ImageNet

ALEXNET: ARCHITECTURE

for layer details refer to:

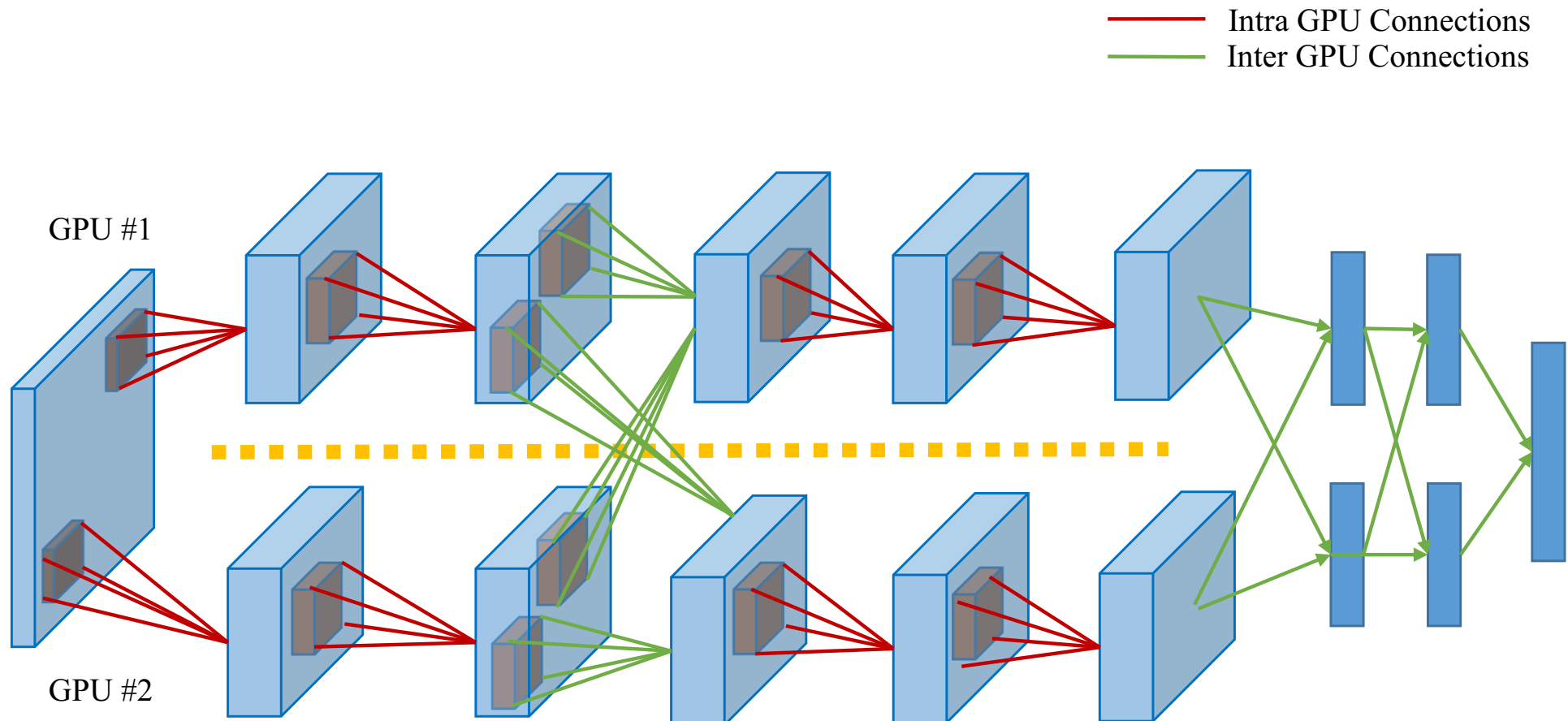
https://github.com/BVLC/caffe/blob/master/models/bvlc_alexnet/deploy.prototxt

شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

ImageNet

ALEXNET: ARCHITECTURE

Top-1 and Top-5 error rates decreases by 1.7% and 1.2% respectively, comparing to the net trained with one GPU and half neurons

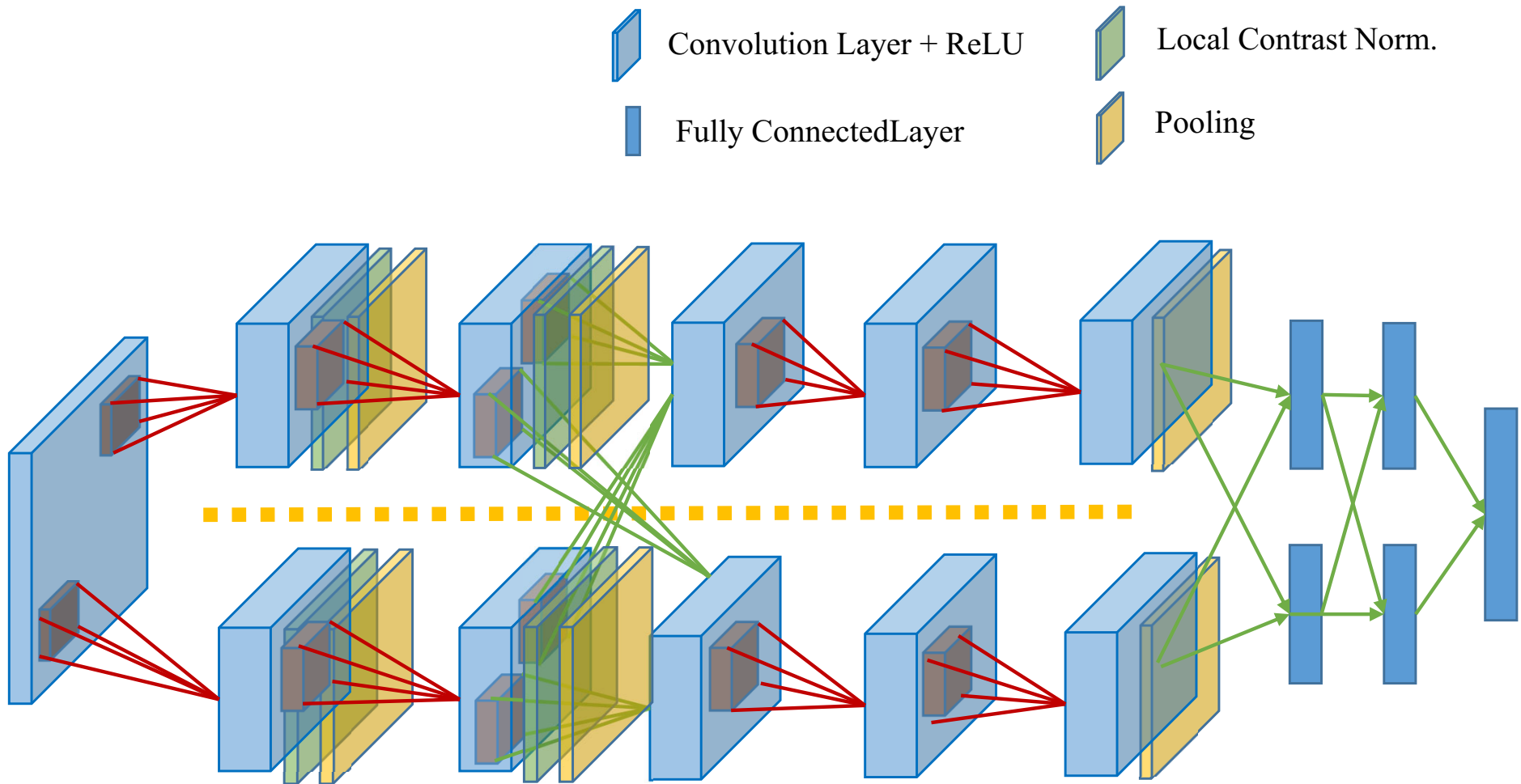


GPU #2

GPU #1

شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

ImageNet

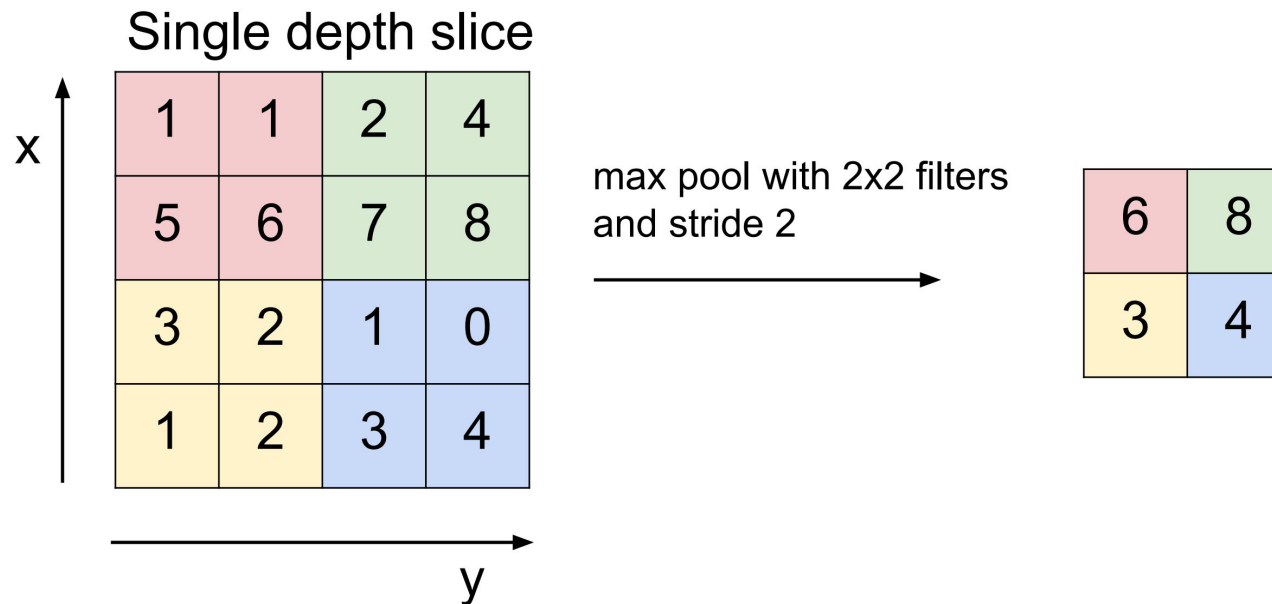
ALEXNET: ARCHITECTURE

شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

ImageNet

ALEXNET: MAX POOLING

Makes the representation smaller and more manageable without losing too much information.



شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

ImageNet

ALEXNET: DATA AUGMENTATION

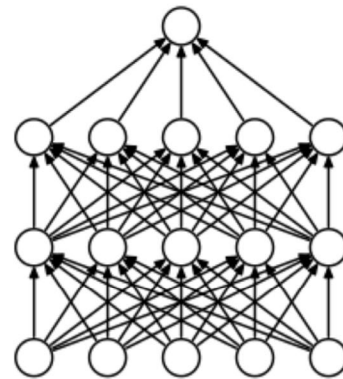
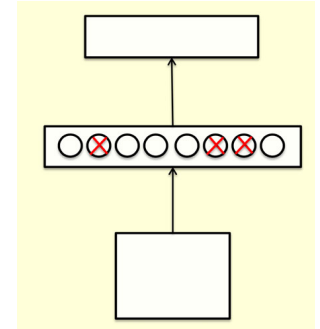
- Easiest way to reduce overfitting on image data is to **artificially enlarge the dataset** using **label-preserving transformations**.
- Two forms of data augmentation:
 - Image Translation and Horizontal Reflection.
 - Changing RGB intensities.

شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

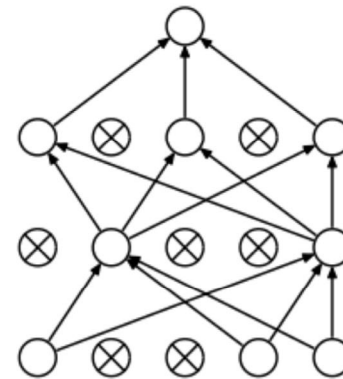
ImageNet

ALEXNET: DROPOUTS : AN EFFICIENT WAY TO AVERAGE MANY LARGE NEURAL NETS

- Consider a neural net with one hidden layer.
- Each time we present a training example, we randomly omit each hidden unit with probability 0.5.
- Equivalent to randomly sampling from 2^h different units.
- All architectures share weights.
- We sample from 2^h models i.e. only a few of the models ever get trained. They only get one training example
- Due to sharing of weights, the model is strongly regularized.
 - Pulls the weights towards what other models want.
 - Better than L2 and L1 that pull weights towards zero.



Standard Neural Net



After applying dropout.

شبکه‌های عصبی کانولوشنال برای طبقه‌بندی تصاویر

ImageNet

ALEXNET: RESULTS

six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.



شبکه‌های عصبی کانولوشنال

۳

یادگیری
انتقالی
با
CNN

یادگیری انتقالی با CNN

TRANSFER LEARNING WITH CNN

انتقال مدل یادگیری شده برای یک وظیفه برای انجام وظیفه‌ای دیگر

یادگیری انتقالی
Transfer Learning

می‌توان یک شبکه‌ی عصبی کانوولوشنال که برای یک وظیفه‌ی خاص آموزش دیده است را با مقدار کمی تغییر برای یک وظیفه‌ی دیگر استفاده کرد.

Transfer Learning

“You need a lot of a data if you want to train/use CNNs”

Transfer Learning with CNNs

1. Train on Imagenet



Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

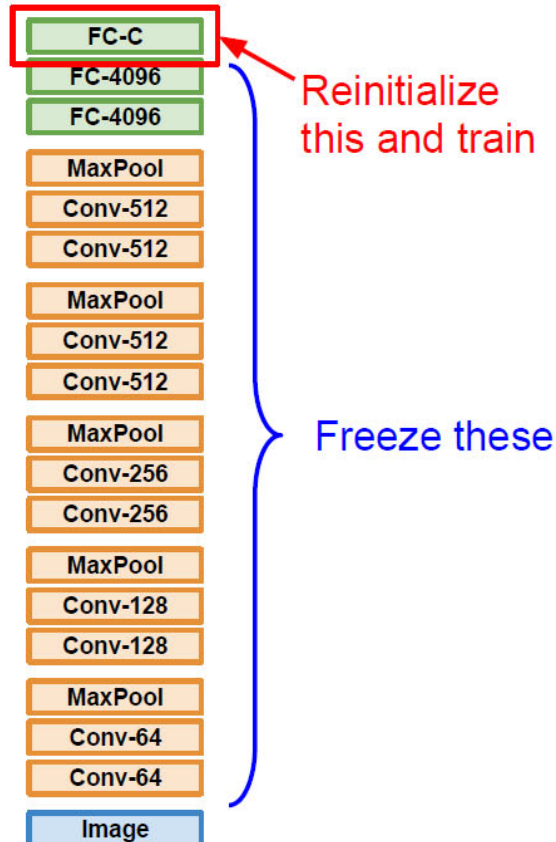
Transfer Learning with CNNs

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

1. Train on Imagenet



2. Small Dataset (C classes)



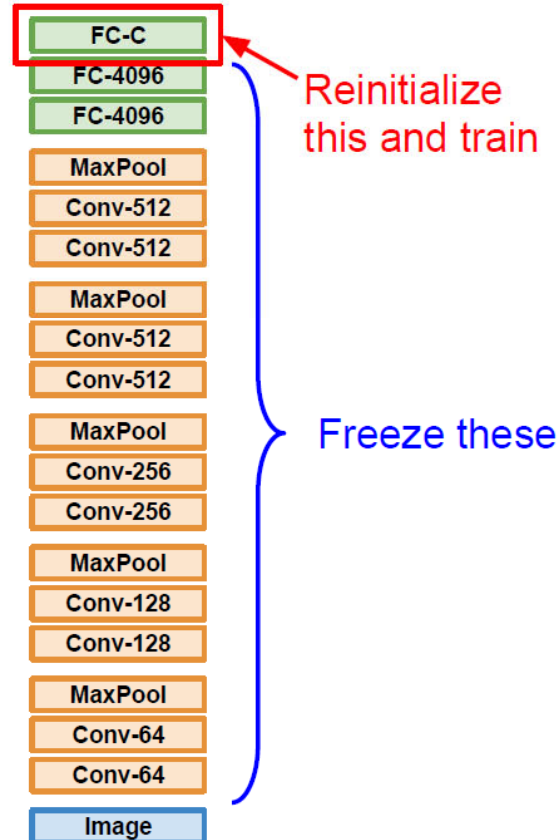
Transfer Learning with CNNs

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

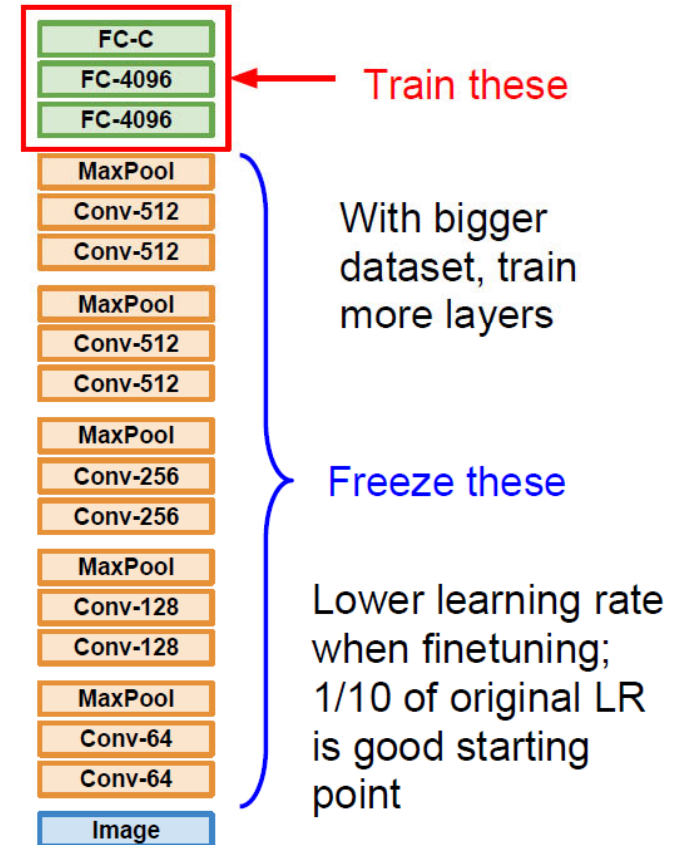
1. Train on Imagenet

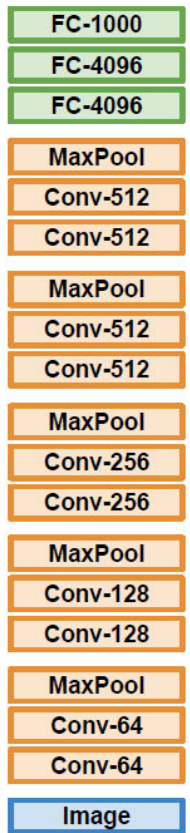


2. Small Dataset (C classes)



3. Bigger dataset

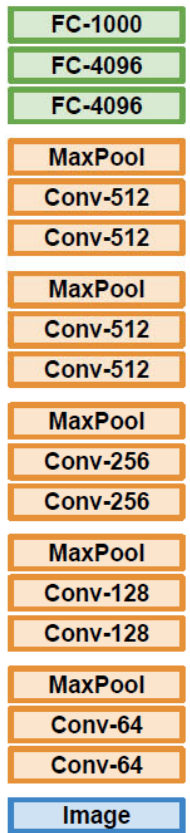




More specific

More generic

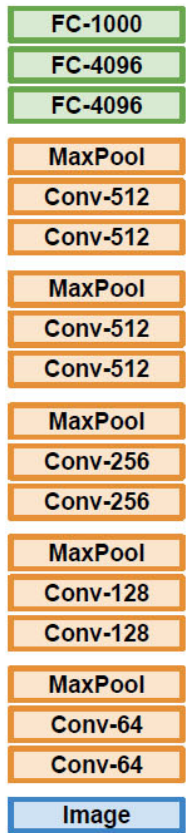
	very similar dataset	very different dataset
very little data	?	?
quite a lot of data	?	?



More specific

More generic

	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	?
quite a lot of data	Finetune a few layers	?



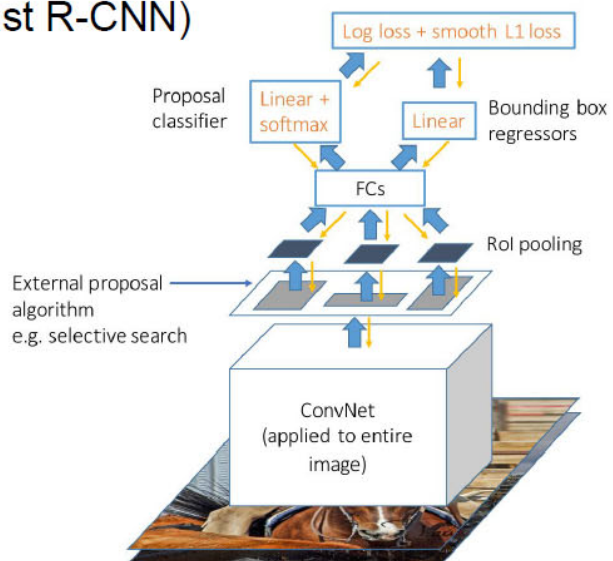
More specific

More generic

	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble... Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

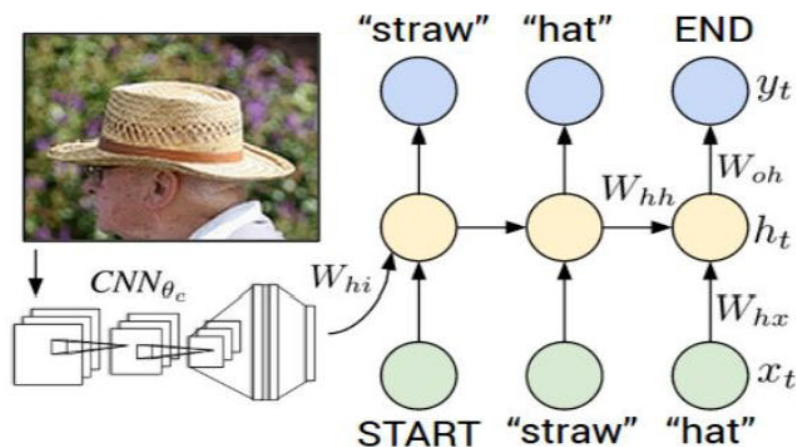
Transfer learning with CNNs is pervasive... (it's the norm, not an exception)

Object Detection (Fast R-CNN)



Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

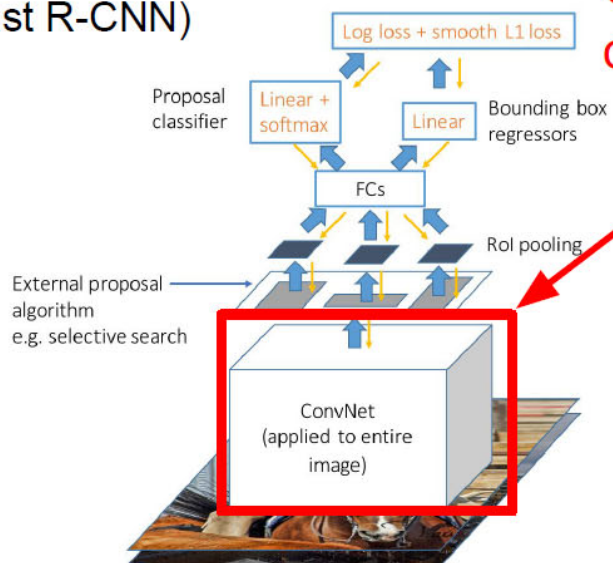
Image Captioning: CNN + RNN



Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for
Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

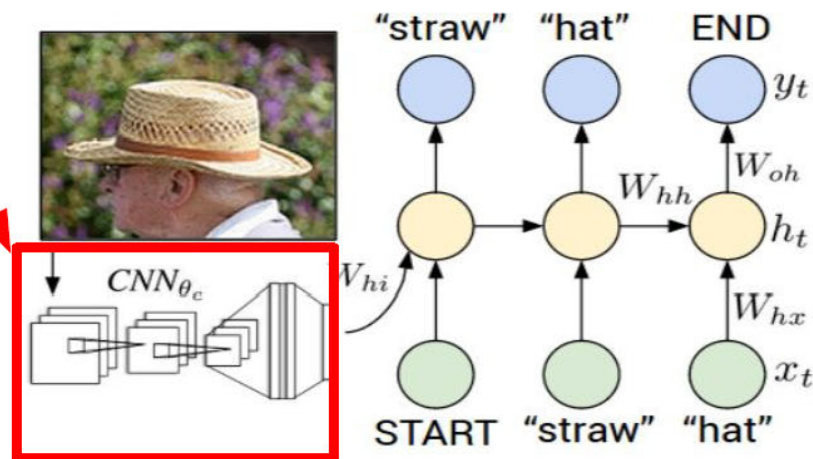
Transfer learning with CNNs is pervasive... (it's the norm, not an exception)

Object Detection
(Fast R-CNN)



CNN pretrained
on ImageNet

Image Captioning: CNN + RNN

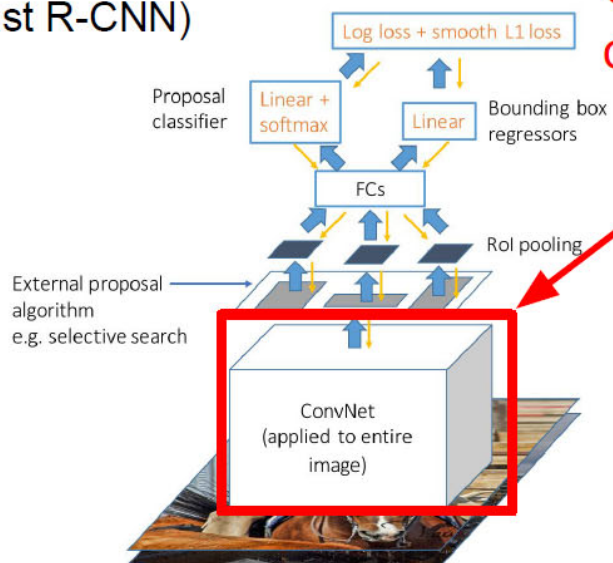


Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for
Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

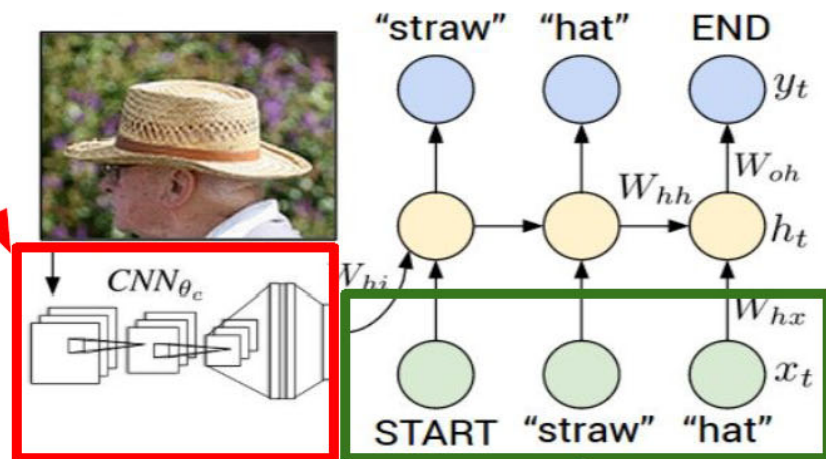
Transfer learning with CNNs is pervasive... (it's the norm, not an exception)

Object Detection
(Fast R-CNN)



CNN pretrained
on ImageNet

Image Captioning: CNN + RNN



Word vectors pretrained
with word2vec

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for
Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

Takeaway for your projects and beyond:

Have some dataset of interest but it has $< \sim 1\text{M}$ images?

1. Find a very large dataset that has similar data, train a big ConvNet there
2. Transfer learn to your dataset

Deep learning frameworks provide a “Model Zoo” of pretrained models so you don’t need to train your own

Caffe: <https://github.com/BVLC/caffe/wiki/Model-Zoo>

TensorFlow: <https://github.com/tensorflow/models>


PyTorch: <https://github.com/pytorch/vision>

شبکه‌های عصبی کانولوشنال

۴

منابع


منبع اصلی

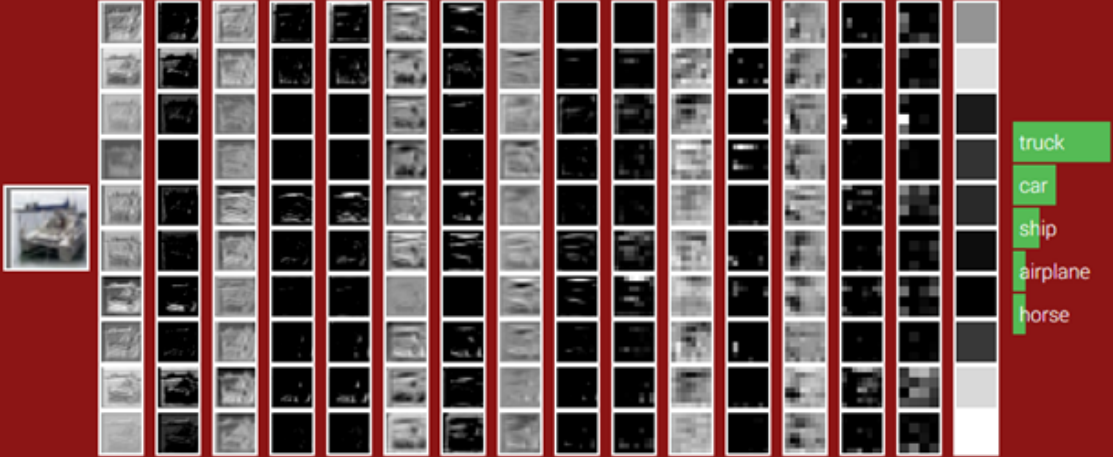


CS231n: Convolutional Neural Networks for Visual Recognition

Spring 2019

Previous Years: [\[Winter 2015\]](#) [\[Winter 2016\]](#) [\[Spring 2017\]](#) [\[Spring 2018\]](#)





*This network is running live in your browser

Course Description

Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification, localization and detection. Recent developments in neural network (aka "deep learning") approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. This course is a deep dive into details of the deep learning architectures with a focus on learning end-to-end models for these tasks, particularly image classification. During the 10-week course, students will learn to implement, train and debug their own neural networks and gain a detailed understanding of cutting-edge research in computer vision. The final assignment will involve training a multi-million parameter convolutional neural network and applying it on the largest image classification dataset (ImageNet). We will focus on teaching how to set up the problem of image recognition, the learning algorithms (e.g. backpropagation), practical engineering tricks for training and fine-tuning the networks and guide the students through hands-on assignments and a final course project. Much of the background and materials of this course will be drawn from the [ImageNet Challenge](#).

<http://cs231n.stanford.edu>

<http://cs231n.github.io/convolutional-networks/>

منبع کمکی

ECE 6504 Deep Learning for Perception

Deep Learning for Perception

Virginia Tech, Electrical and Computer Engineering

Fall 2015: ECE 6504

[Course Information](#) [Schedule](#) [Grading](#) [Late Policy](#) [Prerequisites](#) [FAQs](#) [Reviews](#) [Presentations](#) [Projects](#) [Related Classes](#)

Course Information

This is an exciting time to be studying (Deep) Machine Learning, or Representation Learning, or for lack of a better term, simply Deep Learning!

Deep Learning is rapidly emerging as one of the most successful and widely applicable set of techniques across a range of applications (vision, language, speech, computational biology, robotics, etc), leading to some [pretty significant commercial success](#).

This course will expose students to cutting-edge research — starting from a refresher in basics of neural networks, to recent developments. The emphasis will be on student-led paper presentations and discussions. Each “module” will begin with instructor lectures to present context and background material.

NOTE: Only the lectures delivered by the course instructor are made public. We regret that student presentations cannot be made public due to privacy reasons.

Instructor	Dhruv Batra
Office Hour	Fri 3-4pm, Whittemore 468
Teaching Assistants	Abhishek Das Ashwin Kalyan
Office Hours	Ashwin Kalyan: Mon 3-4pm, Whittemore 415 Abhishek Das: Fri 3-4pm, Whittemore 415
Class meets	Tue, Thu 5:00 - 6:15, SURGE 107
Scholar Site	https://scholar.vt.edu/portal/site/f15ece6504
Q&A and Discussions	Scholar Discussion Forum
Staff Mailing List	staff-f15ece6504-g@vt.edu

<https://computing.ece.vt.edu/~f15ece6504/>