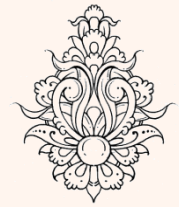
یادگیری عمیق
بخش چهارم
شبکه‌های عصبی کانولوشنی

دانشگاه شهید بهشتی
بهار ۱۴۰۱
احمد محمودی ازناوه

http://faculties.sbu.ac.ir/~a_mahmoudi/

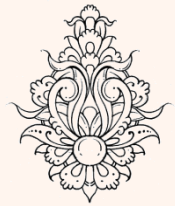
فهرست مطالب

- مقدمه‌ای بر یادگیری عمیق
- آشنایی با شبکه‌های عصبی کانولوشنی دوبعدی
- مروری بر مفهوم کانولوشن
  - لایه کانولوشن
  - لایه ادغام (تجمیع)
- بررسی ساختارهای مختلف
- روش‌های غلبه بر بیش‌برازش
- افزایش سرعت بهینه‌سازی



معرفی

- «یادگیری عمیق» باعث ایجاد پیشرفت‌های چشمگیری در زمینه‌های گوناگونی مانند بینایی ماشین، پردازش صوت و پردازش زبان طبیعی شده است.
  - در مواردی کارایی مدل‌های به دست آمده فراتر از کاربر انسانی بوده است.
- این پیشرفت‌های خیره‌کننده تا حد زیادی مدیون افزایش قدرت سیستم‌های پردازشگر می‌باشد.
  - استفاده از پردازشگر گرافیکی برای محاسبات

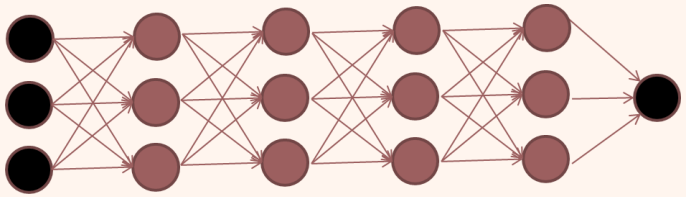


یادگیری عمیق چیست؟

- این شبکه به نوعی از نحوه‌ی پردازش اطلاعات در مغز انسان تقلید می‌کند، چندین لایه که وظیفه‌ی استخراج ویژگی و شناسایی را به صورت همزمان انجام می‌دهند.

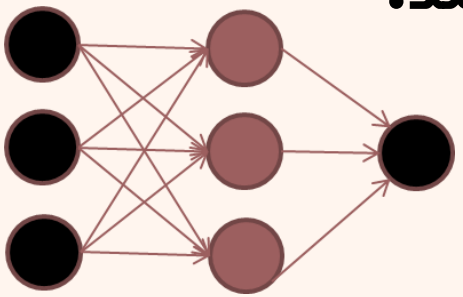
End 2 end learning

- یک شبکه‌ی عصبی با چندین لایه بین ورودی و خروجی
- شبکه‌های چندلایه سال‌هاست که شناخته شده‌اند، اما تنها ویژگی شبکه‌های عمیق داشتن چندین لایه‌ی مخفی نیست!

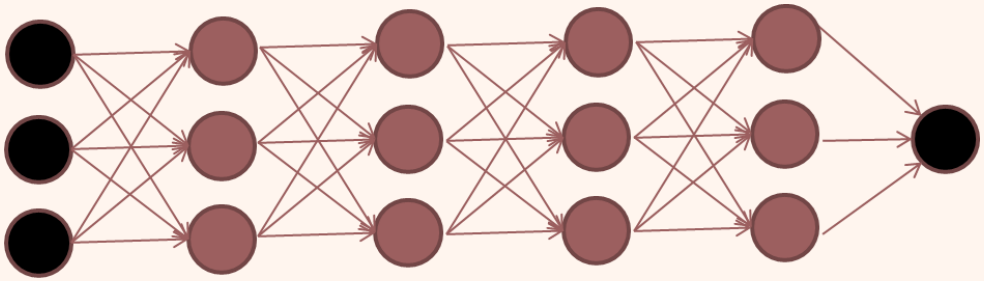


یادگیری عمیق

- شبکه‌ی عصبی با تعداد لایه‌های مخفی محدود به راحتی آموزش می‌بینند:



- اما در صورت افزایش تعداد لایه‌ها، الگوریتم‌های یادگیری کارایی لازم را نخواهند داشت.



یادگیری عمیق(ادامه...)

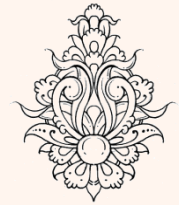
- در واقع، مهمترین تفاوت فوت و فن‌هایی است که برای آموزش این شبکه‌ها به کار گرفته شده است:
- کنترل پارامترهای آزاد
  - استفاده از پارامترهای مشترک
  - اتصالات تنک
- یادگیری بی‌نظارت خصیصه‌ها
- منظم‌سازی

Weight sharing

Sparse connectivity

unsupervised feature learning

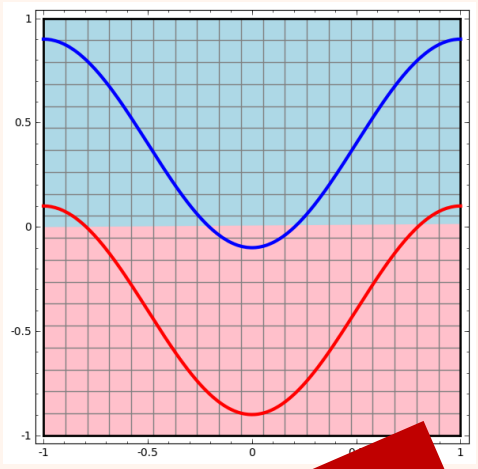
Regularization



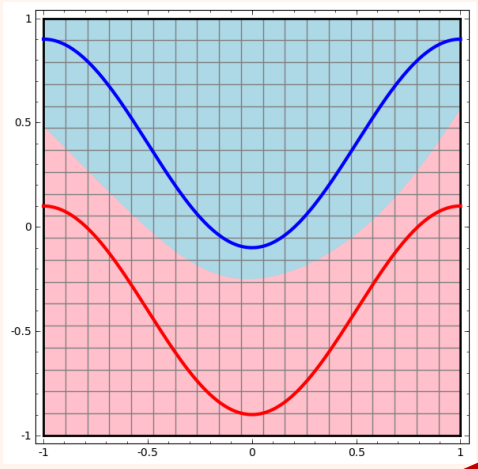
یادآوری(MLP)

- یک شبکه‌ی تک‌لایه توانایی دسته‌بندی داده‌های جدایی‌پذیر خطی را دارد.
- به ازای هر نورون بردار وزن‌ها در بردار ورودی ضرب نقطه‌ای می‌شود.
- می‌دانیم که یک شبکه‌ی عصبی با تنها دو لایه‌ی مخفی توانایی جداسازی پیچیده‌ترین اشکال را دارد.
  - لایه‌ی آخر بازهم نقش یک جداساز خطی را دارد.
  - می‌توان گفت لایه‌های مخفی وظیفه‌ی استخراج ویژگی را بر عهده دارند.
- وزن‌های آموزش دیده، تفسیرپذیری خوبی ندارند.

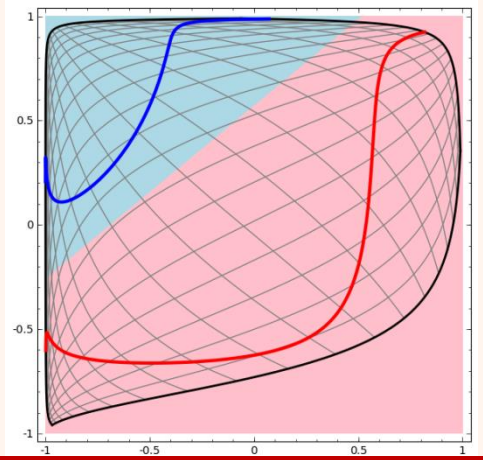
نقش لایه‌های مخفی



پرسپترون



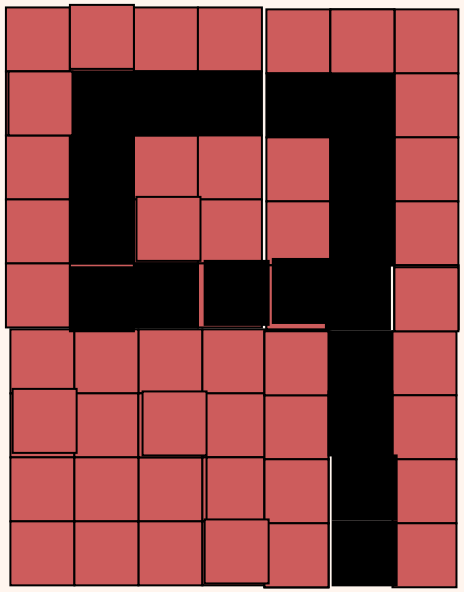
شبکه‌ی چندلایه

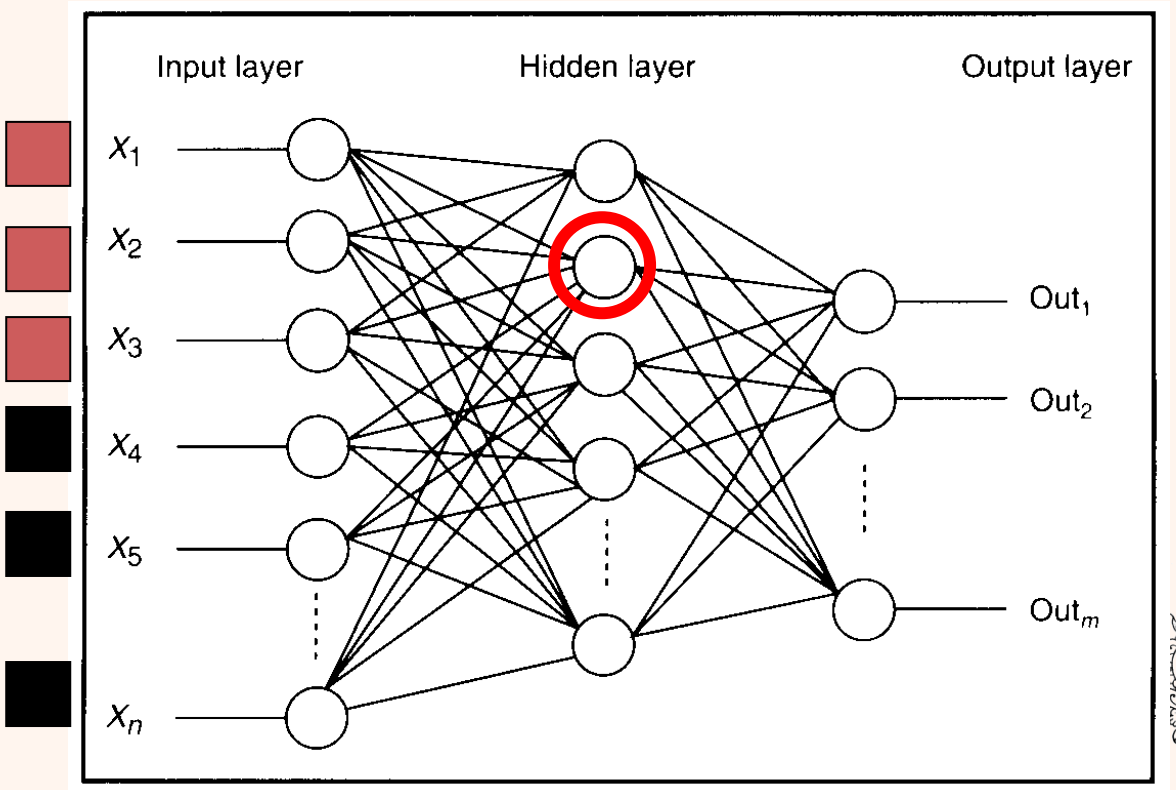


داده‌ی ورودی به لایه‌ی آخر

http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

استخراج ویژگی

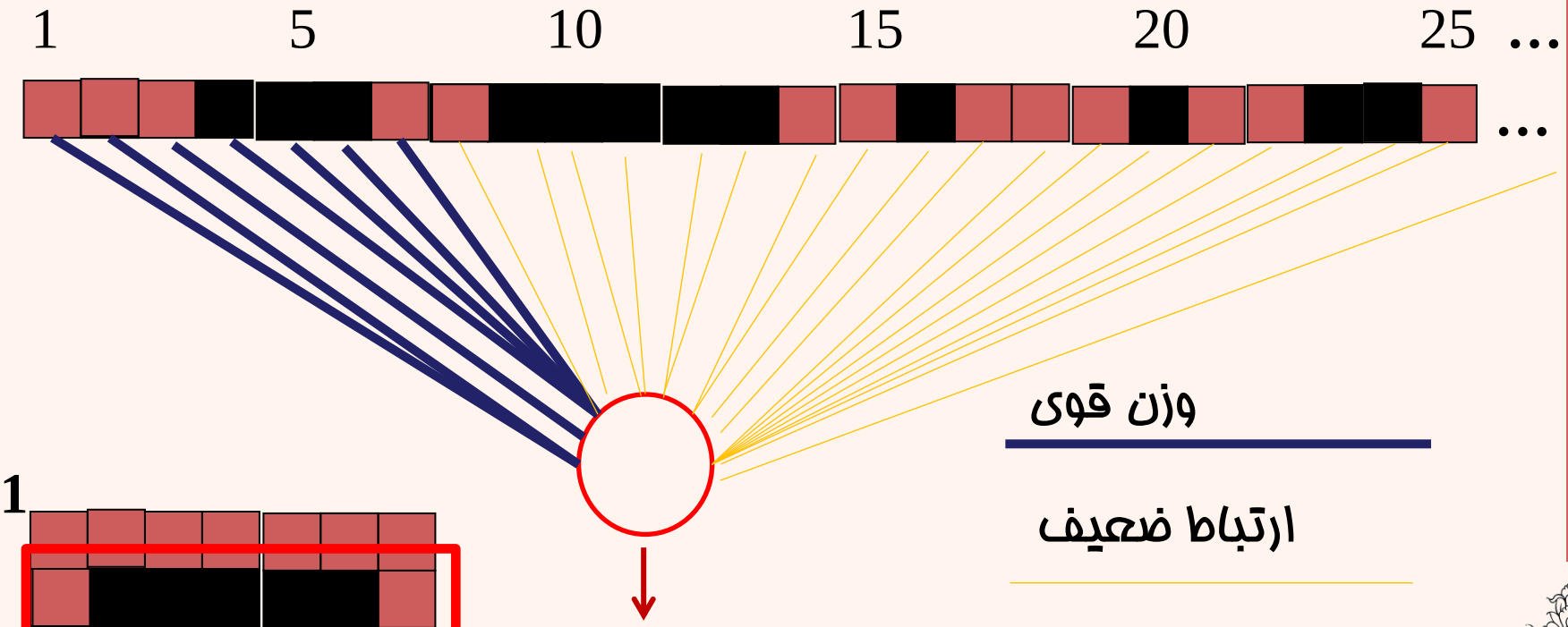


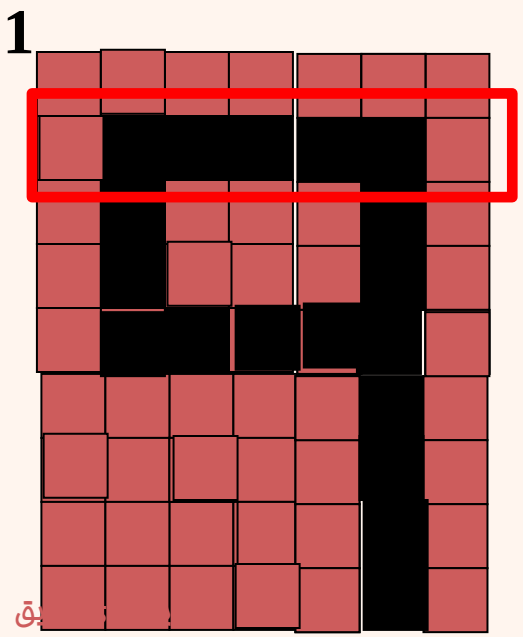


نقش نورون‌های لایه‌ی مخفی چیست؟

Adopted from Professor Corne slides

استخراج ویژگی(ادامه...)

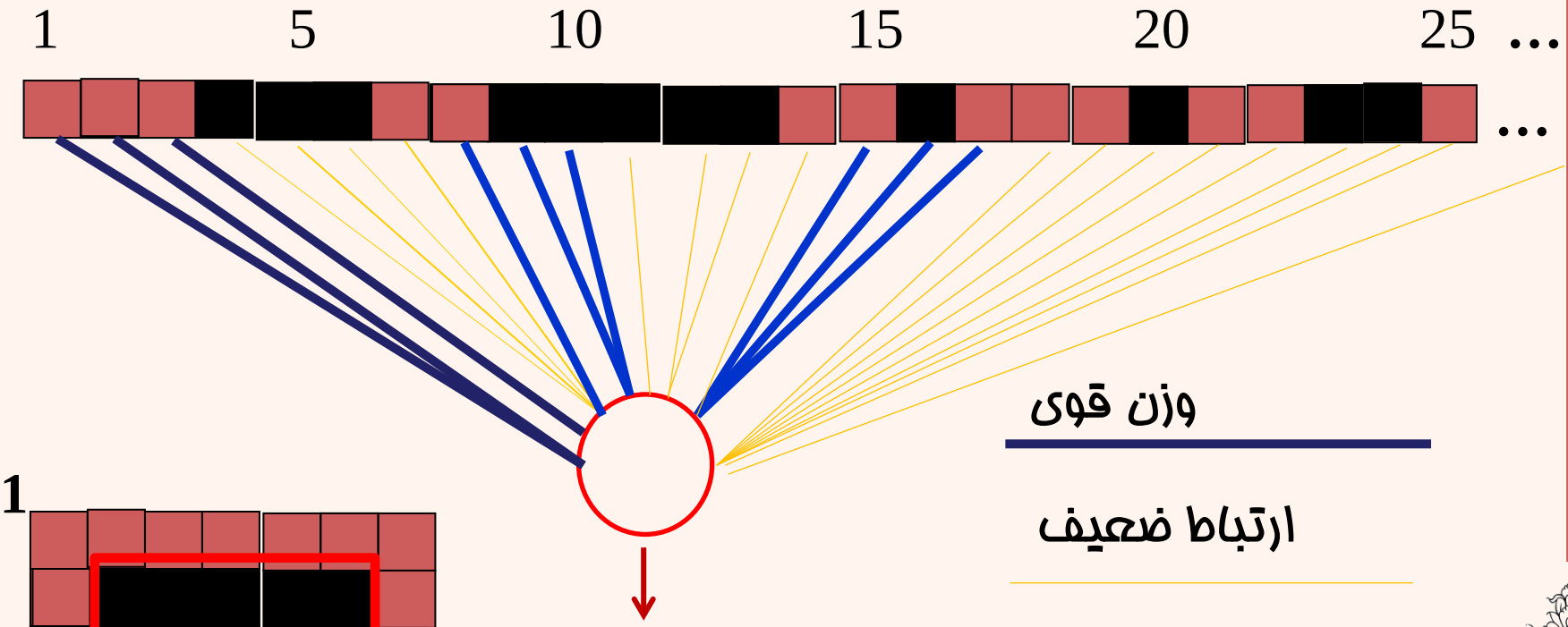


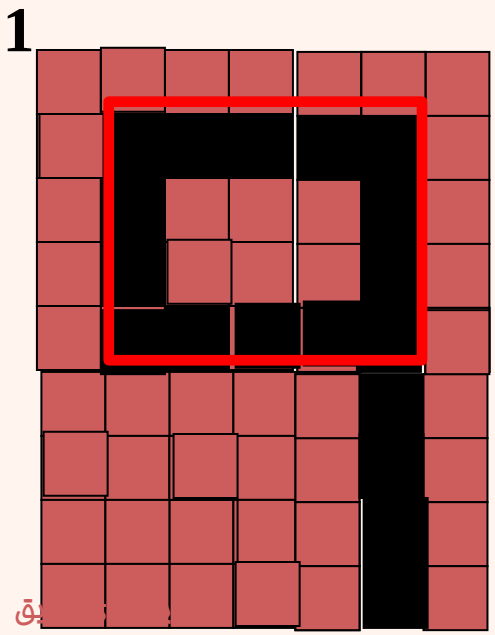


بیانگر یک لبه‌ی قوی افقی

Adopted from Professor Corne slides

استخراج ویژگی(ادامه...)





یک ناحیه مدور در بالای صفحه

استخراج ویژگی(ادامه...)

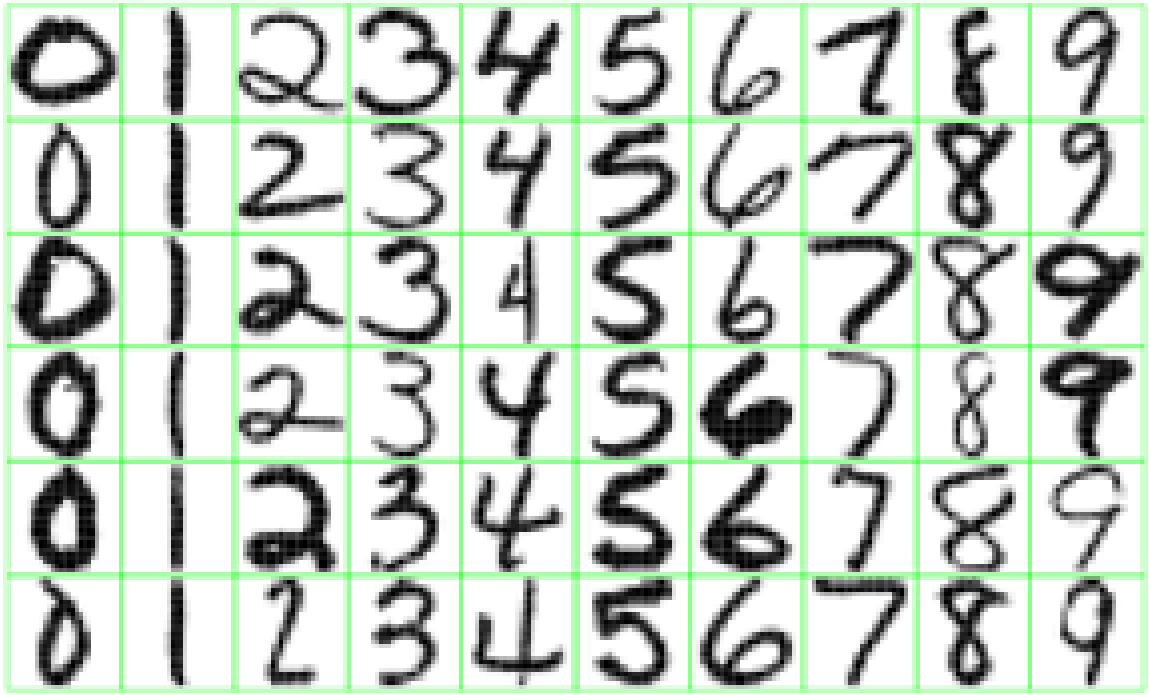


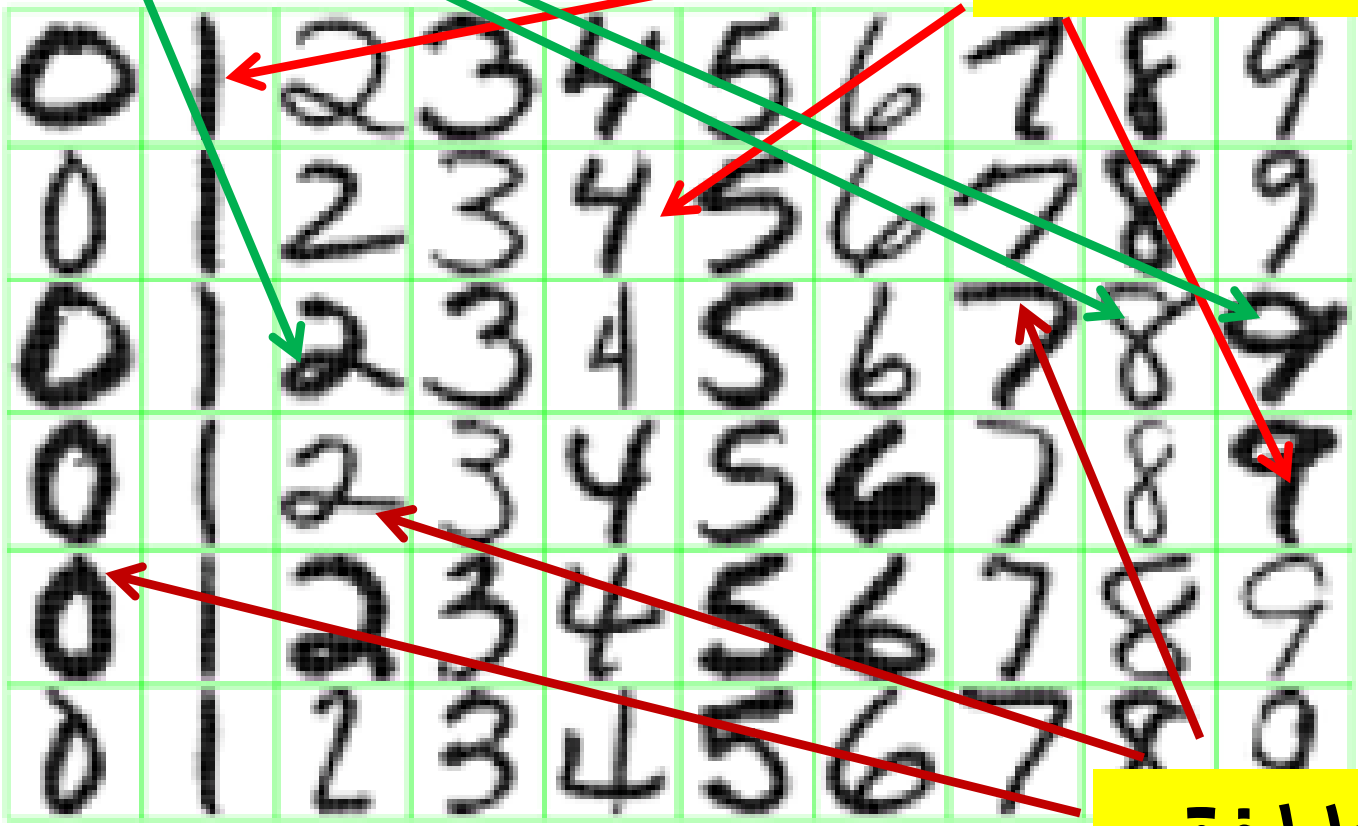
Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

شبکه‌ی عصبی چه نوع خصیصه‌ای برای تشخیص کاراکتر یاد می‌گیرد؟

استخراج ویژگی(ادامه...)

دایره‌های کوچک

خطوط عمودی



خطوط افقی

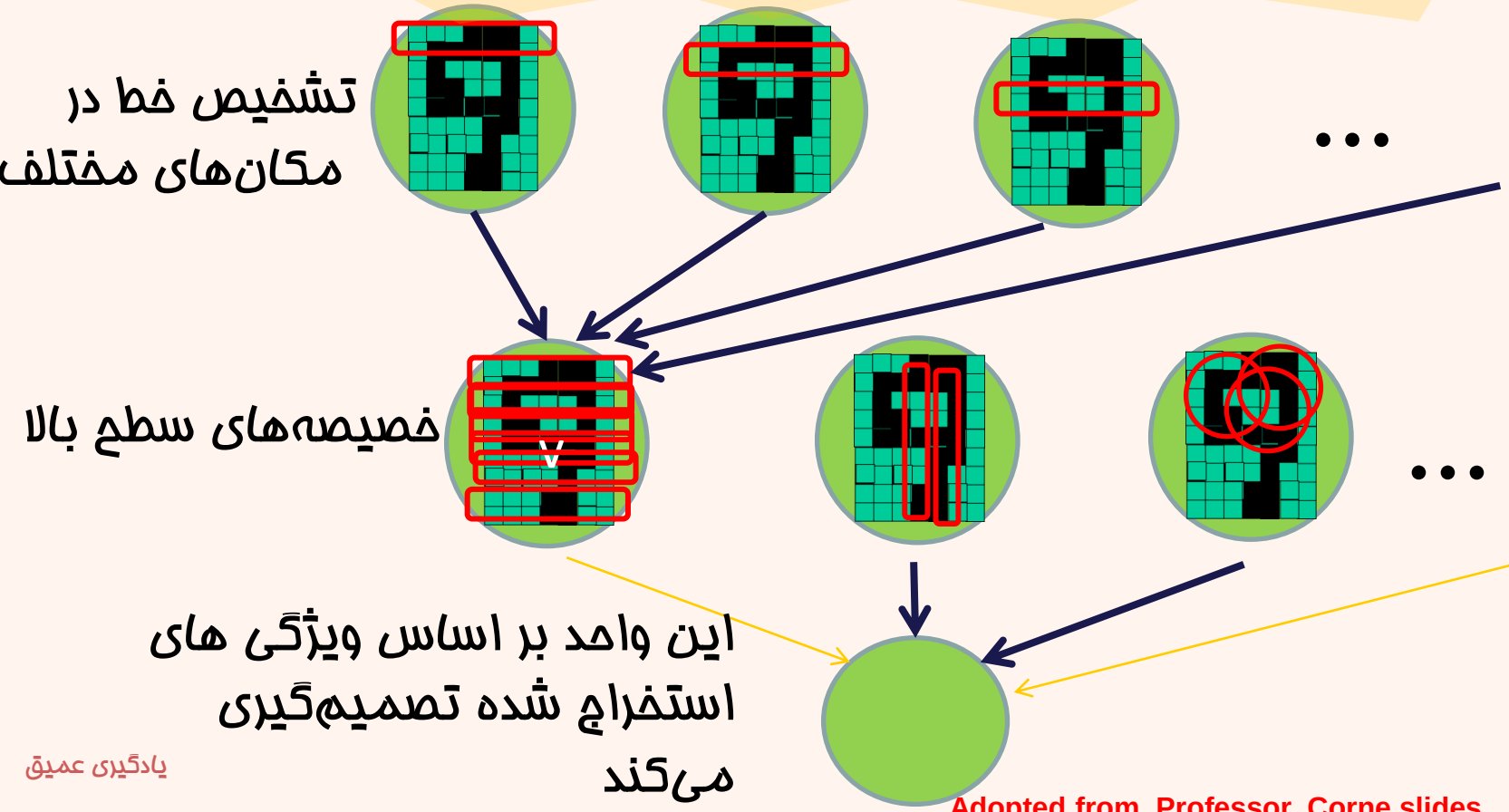
Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

خصیصه‌های استخراج شده نسبت به مکان حساس هستند!

Adopted from Professor Corne slides

استخراج ویژگی(ادامه...)

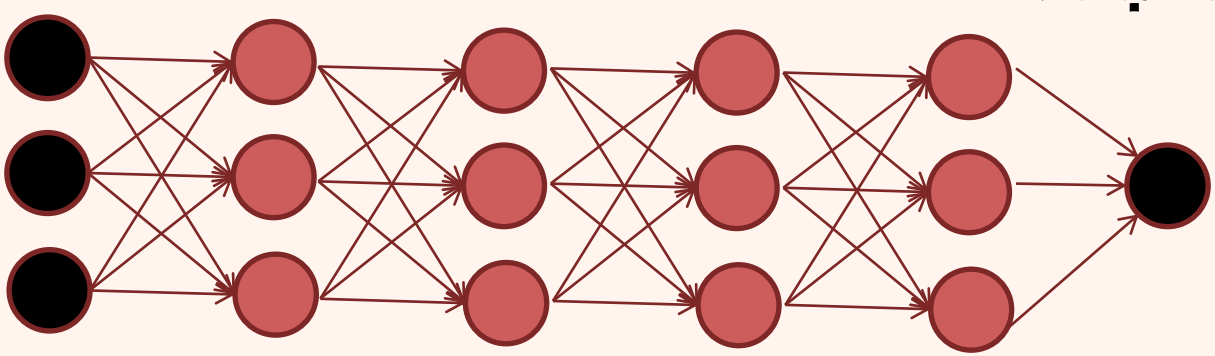
- لایه‌های مخفی می‌توانند خصیصه‌های سطح بالا استخراج کنند.



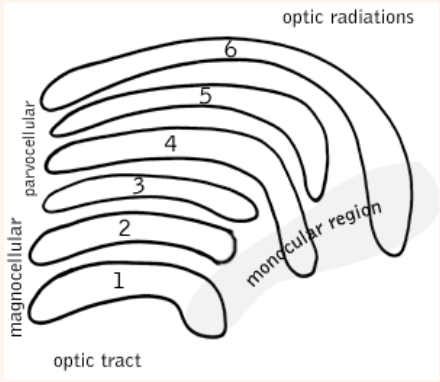
Adopted from Professor Corne slides

استخراج ویژگی(ادامه...)

- در این حالت استفاده از چندین لایه‌ی مختلف توجیه‌پذیر است.

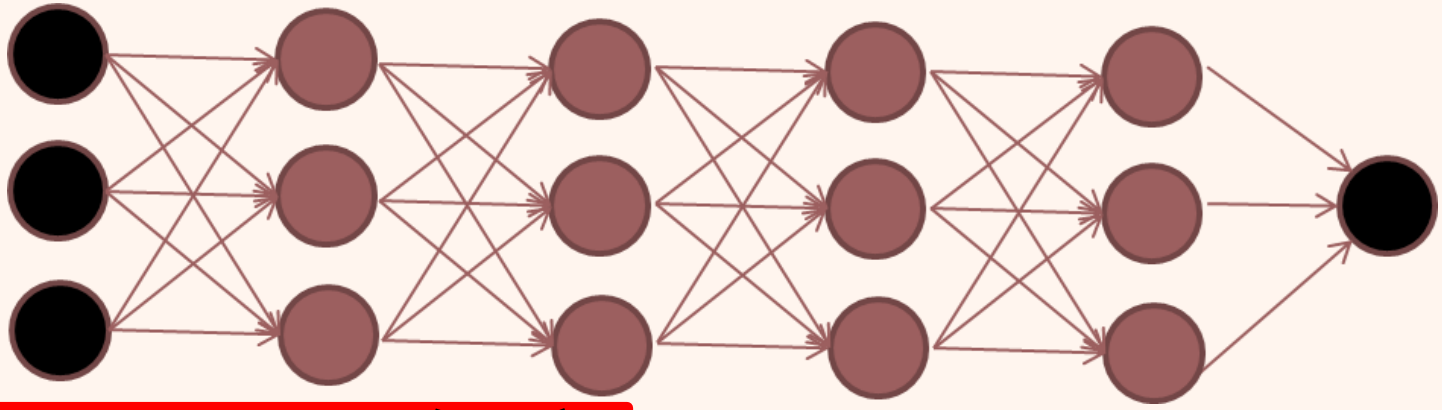


- درست مانند مغز

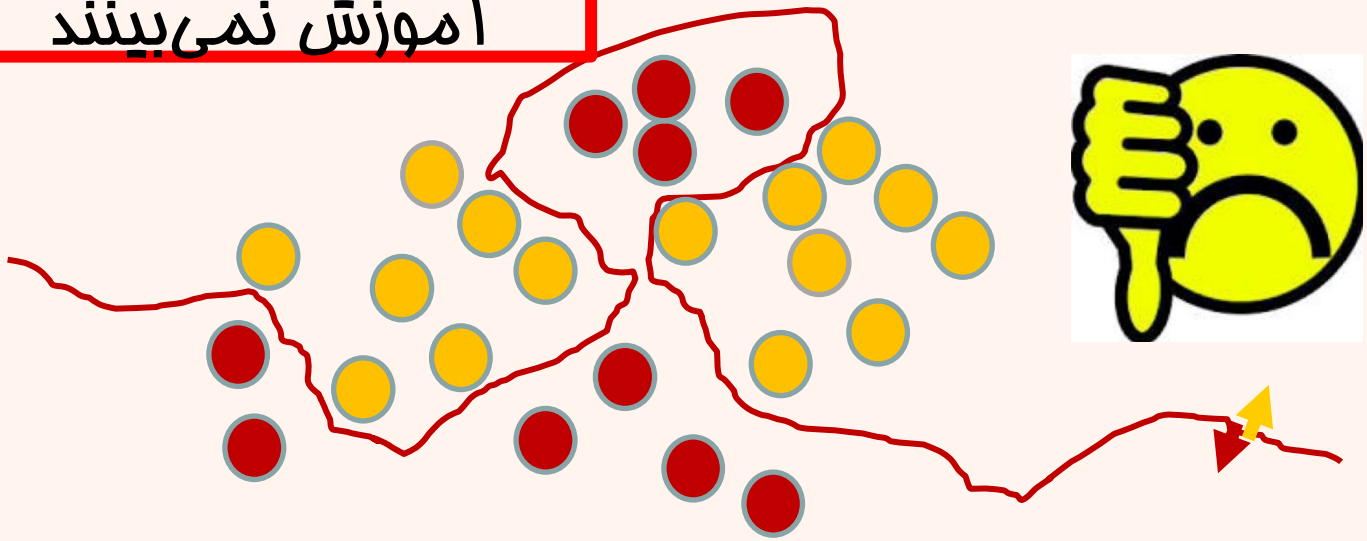


- لایه‌های مخفی باید بتوانند خصیصه‌های مناسب را استخراج کنند.

استخراج ویژگی(ادامه...)

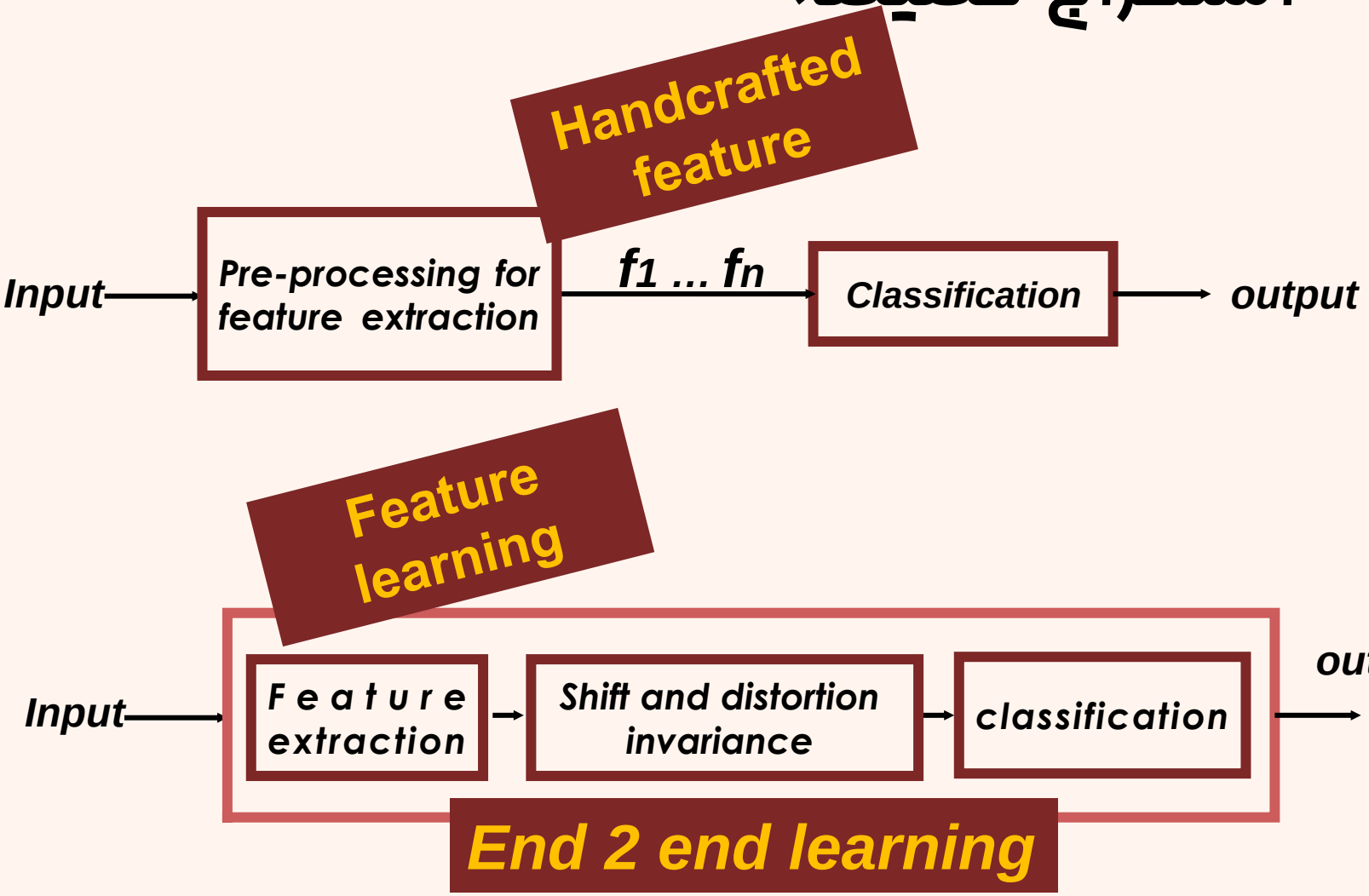


لایه‌های نخست خوب آموزش نمی‌بینند



مشکل بیش‌برازش

استخراج خصیصه



شبکه‌های عصبی کانولوشنی

- در سال ۱۹۹۵ توسط LeCun و Bengio مطرح شد.
- در لایه‌های کانولوشنی به جای ضرب داخلی بردار ورودی در بردار وزن‌ها عملگر کانولوشن انجام می‌شود.
- به شبکه‌ای که حداقل یک «لایه کانولوشنی» داشته باشد، «شبکه کانولوشنی» گفته می‌شود.
- برای آموزش این شبکه‌ها نیز از الگوریتم پس‌انتشار خطا استفاده می‌شود.

اعمال فیلتر دوبعدی

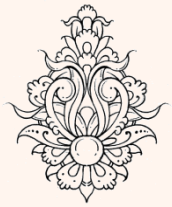
- فیلتر دوبعدی
- Kernel (mask), kernel coefficients
- Size: 3×3, 5×5, 7×7, ….

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

Input

1	2	0	1	3	
2	1	4	2	2	
1	0	1	0	1	
1	2	1	0	2	
2	5	3	1	2	

Output



اعمال فیلتر دوبعدی

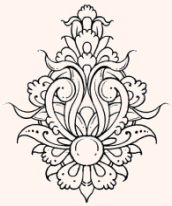
1	0	-1
2	0	-2
1	0	-1

Input

1	2	0	1	3	
2	1	4	2	2	
1	0	1	0	1	
1	2	1	0	2	
2	5	3	1	2	

Output

	-4				



عملگر همبستگی و کانولوشن

- گفته می‌شود بین فیلتر و تصویر ورودی، «عملگر همبستگی» اعمال شده است.

Correlation

- عملگر همبستگی به صورت زیر تعریف می‌شود:

$$O(x,y) = I(x,y) \circ h(x,y) = \sum_{j=-n}^{n} \sum_{i=-m}^{m} I(i,j)h(x+i,y+j)$$

- عملگر کانولوشن به همین ترتیب عمل می‌کند، به جز این ابتدا فیلتر قرینه می‌شود:

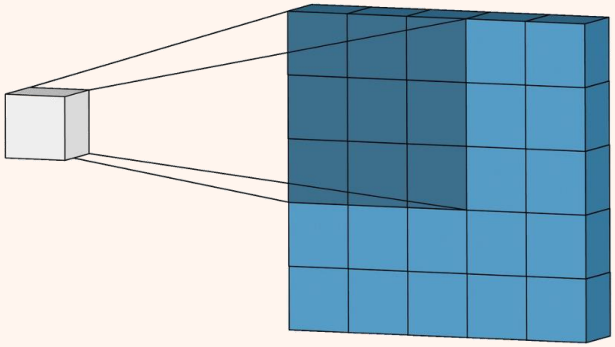
convolution

$$O(x,y) = I(x,y) * h(x,y) = \sum_{j=-n}^{n} \sum_{i=-m}^{m} I(i,j)h(x-i,y-j)$$

هنگامی‌که فیلتر متقارن باشد، پاسخ کورولیشن همانند کانوولوشن است

عملگر همبستگی و کانولوشن

- عملگر کانولوشن یک عملگر خطی است که برای یافتن پاسخ یک سیستم LTI که پاسخ ضربه آن مشخص است استفاده می‌شود.
- همبستگی میزان شباهت دو سیگنال متفاوت را نشان می‌دهد.
- هر چند که این دو مفهوم شباهت زیادی با هم دارند آن‌چه در شبکه‌های کانولوشنی لایه‌ی کانولوشن نامیده می‌شود، در واقع میزان شباهت یک ناحیه محلی به فیلتر را نشان می‌دهد.



بر اساس نوع فیلتر ویژگی‌های خاصی از نواحی محلی تصویر استخراج می‌شود.

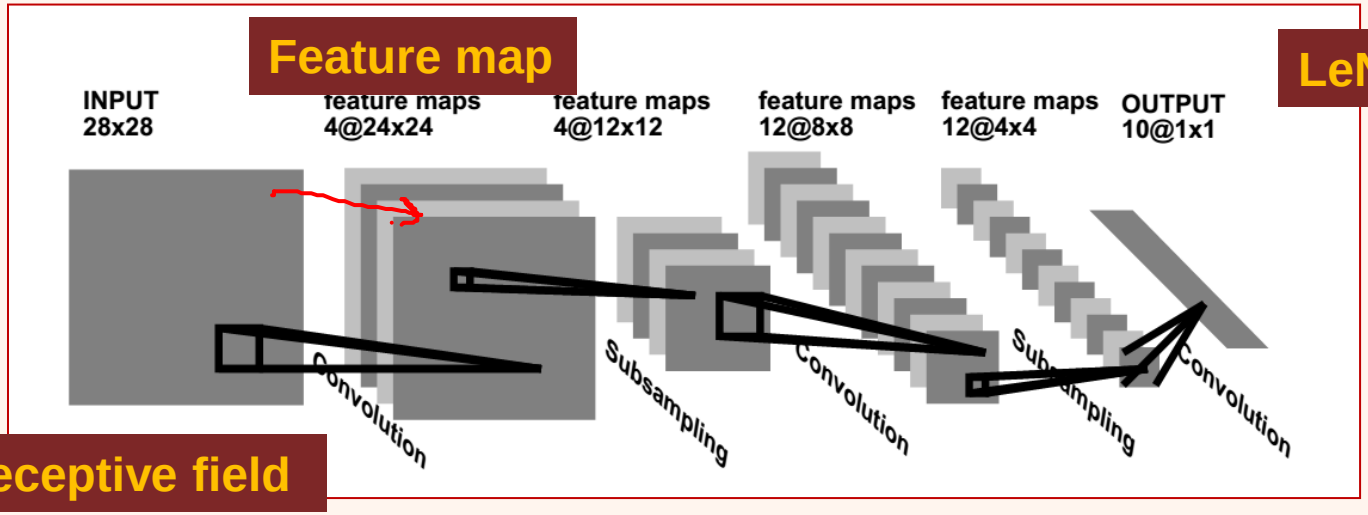
*

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

Convolutional Neural Networks

Feature map

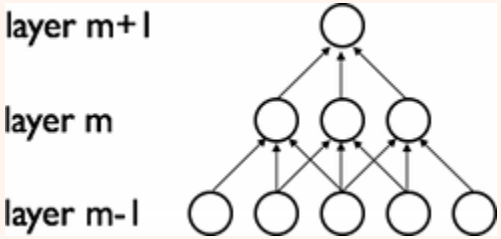
LeNet1(3000/100000)



receptive field

- این شبکه در ابتدا برای داده‌های دوبعدی طراحی شد.
- برای داده‌هایی مناسب است که به صورت محلی ساختار منظمی دارند.
  - اتصالات بین نورون‌ها به صورت محلی است.

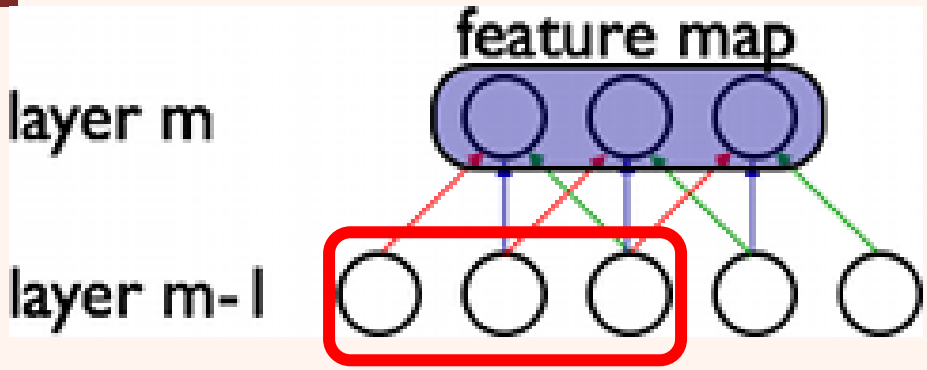
Sparse Connectivity



لایه‌ی کانولوشن

- وزن‌های متصل به لایه‌ی feature-map یکسان است.

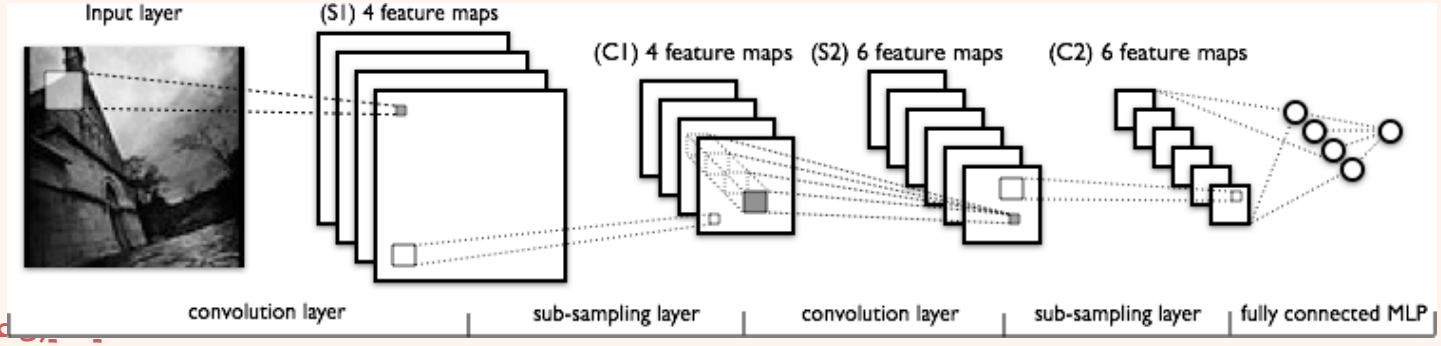
Shared Weights



- بدین‌ترتیب تعداد پارامترهای آزاد شبکه به شدت کاهش یابد.
- ترکیب خطی محدوده‌ی محلی از ورودی در لایه‌ی بعدی محاسبه می‌شود(عملگر کانولوشن).
- حاصل با یک مقدار بایاس جمع شده و از یک تابع فعال‌سازی غیرخطی عبور می‌کند.

لایه‌ی کانولوشن

- در این حالت خروجی این لایه در مواردی که شبیه کرنل باشد، مقدار بالایی خواهد بود.
- در لایه‌های بعدی، براساس این خصیصه‌ها، خصیصه‌های جدید استخراج می‌شود.
- با توجه به مشابهت محاسبه‌ی هر گره در این لایه با کانولوشن معمولی، به چنین لایه‌ای «لایه‌ی کانولوشنی» می‌گویند.



لایه‌ی کانولوشن

- این لایه الگوها را به صورت سلسله مراتبی استخراج می‌کند.
- در واقع فیلترهایی که برای استخراج ویژگی استفاده می‌شوند، از طریق یادگیری به دست می‌آیند.
  - در نخستین لایه الگوهای ساده مانند لبه‌ها تشخیص داده می‌شود و در لایه‌های بعدی الگوهای پیچده‌تری که از ترکیب الگوهای ساده به دست می‌آید.

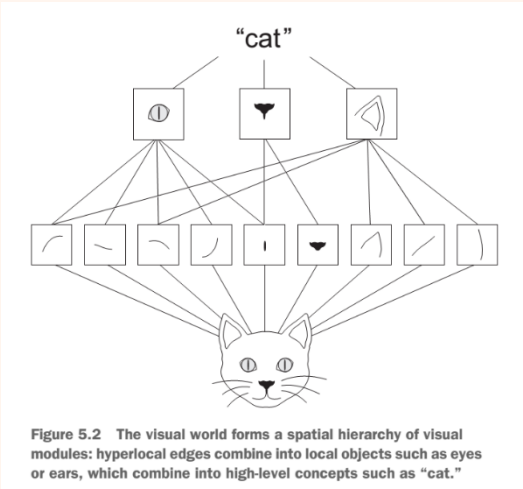
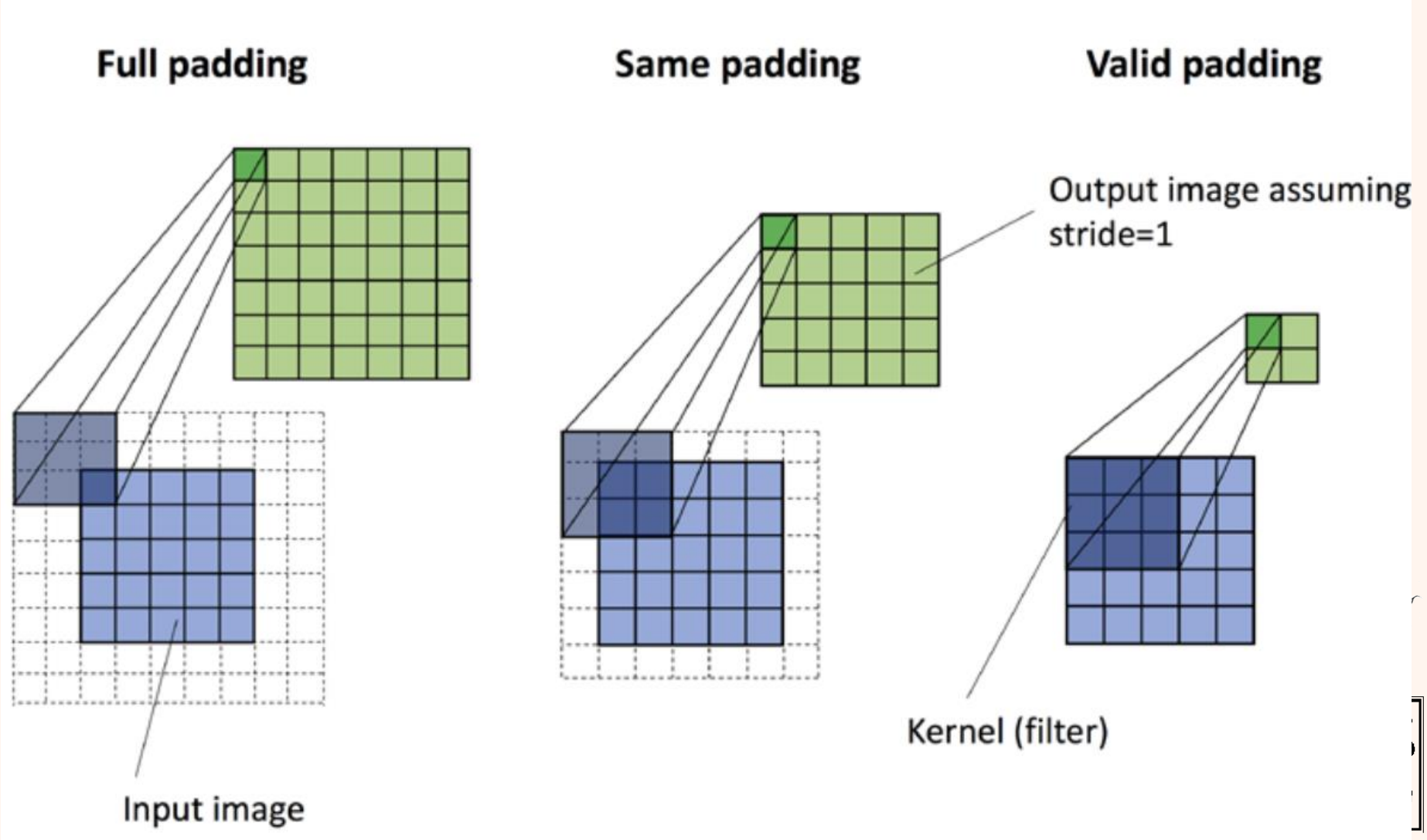


Figure 5.2 The visual world forms a spatial hierarchy of visual modules: hyperlocal edges combine into local objects such as eyes or ears, which combine into high-level concepts such as "cat."

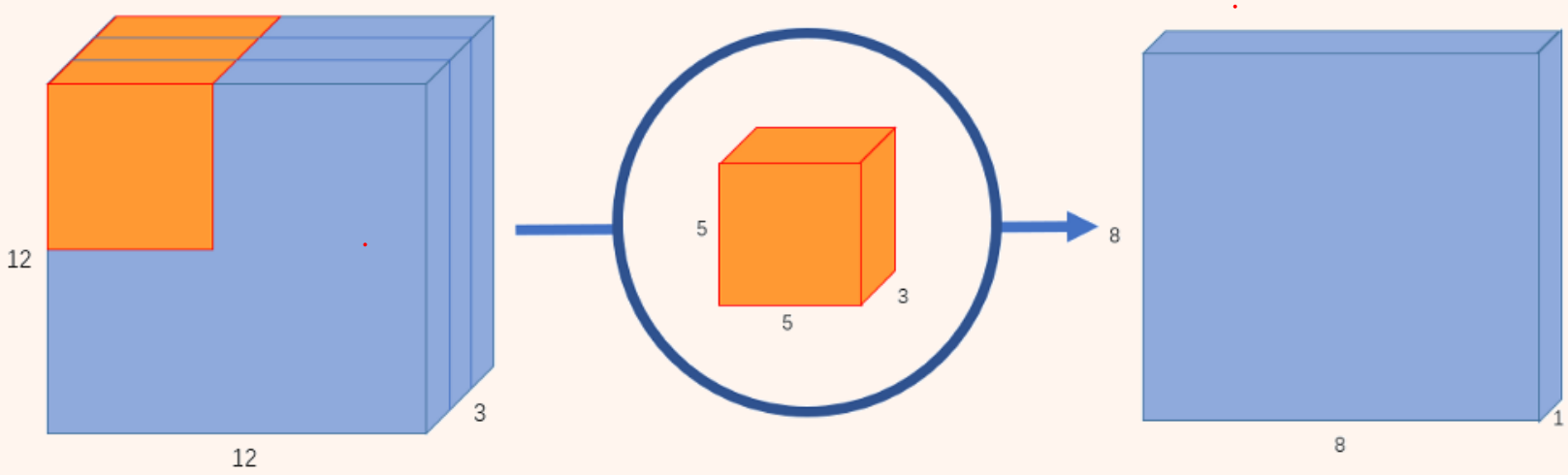
Chollet, F. *Deep Learning with Python*

padding

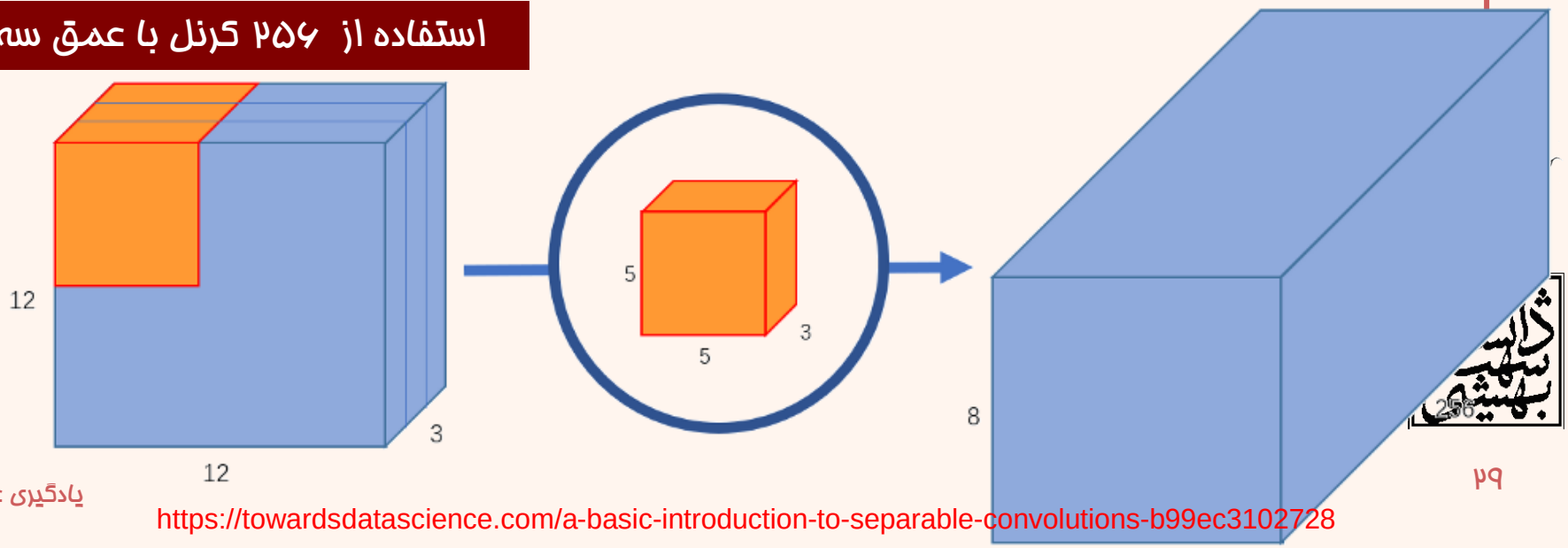

Full padding
Same padding
Valid padding
Output image assuming stride=1
Kernel (filter)
Input image

تصاویر سه کانله

استفاده از کرنل با عمق سه



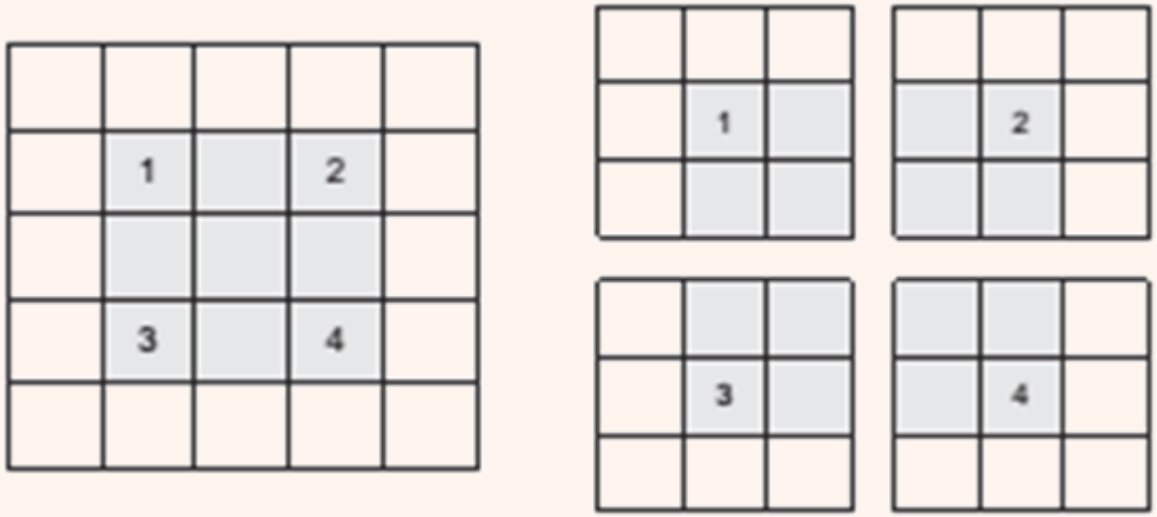
استفاده از ۲۵۶ کرنل با عمق سه

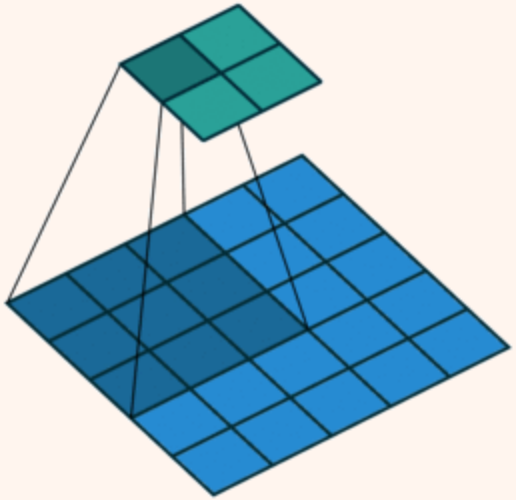


https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

استفاده از گام بیش از یک پیکسل

stride





Stride 2

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Sub-Sampling (Pooling)

- عملا لایه‌ی کانولوشن باعث می‌شود تعداد خصیصه‌ها افزایش یابد.

Average Pooling

- در این لایه ابعاد با روش‌هایی چون میانگین‌گیری یا جایگزین کردن مقدار بیشینه کاهش می‌یابد.

Max Pooling

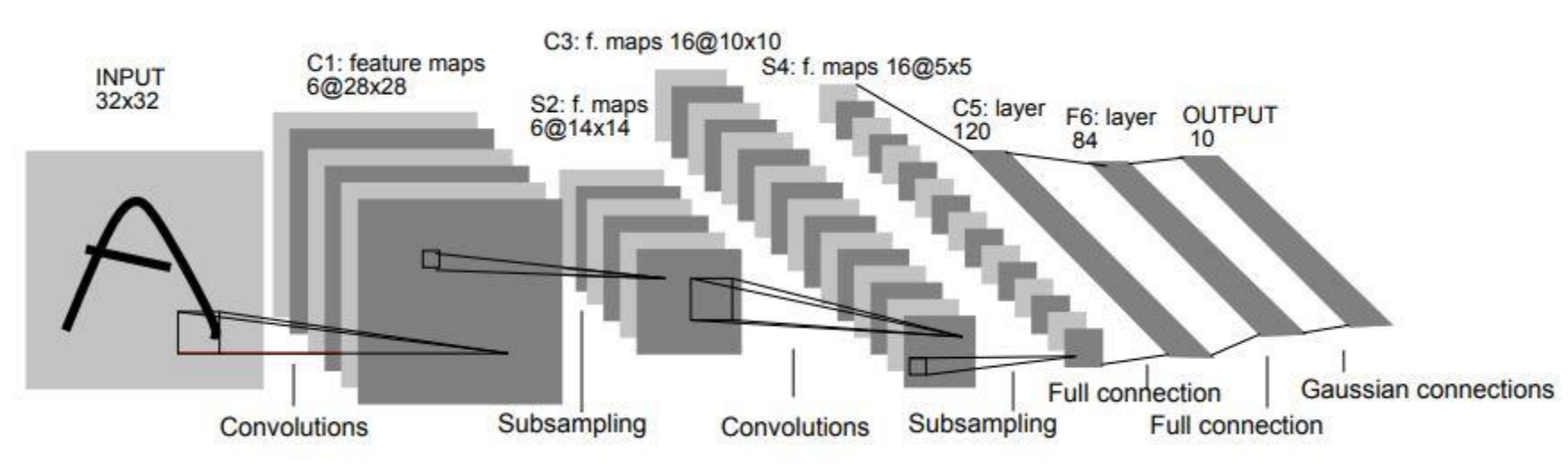
  - بدین ترتیب وابستگی خصیصه‌ی استخراج شده به مکان کاهش می‌یابد.
  - با توجه به این که خصیصه‌های استخراج شده بیان‌گر وجود الگوهای خاص در یک ناحیه هستند، استفاده از مقدار بیشینه معقول‌تر است.

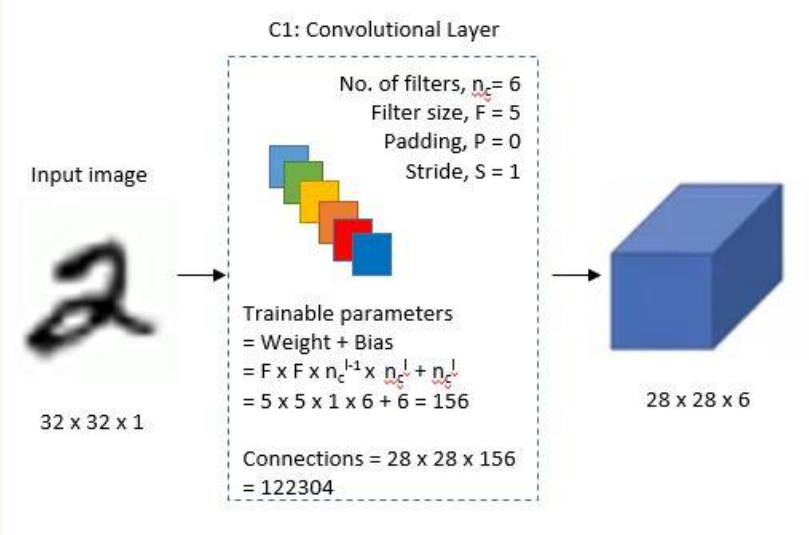
- حذف این لایه به ساختار سلسله‌مراتبی لطمه می‌زند.

LeNet-5 architecture

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

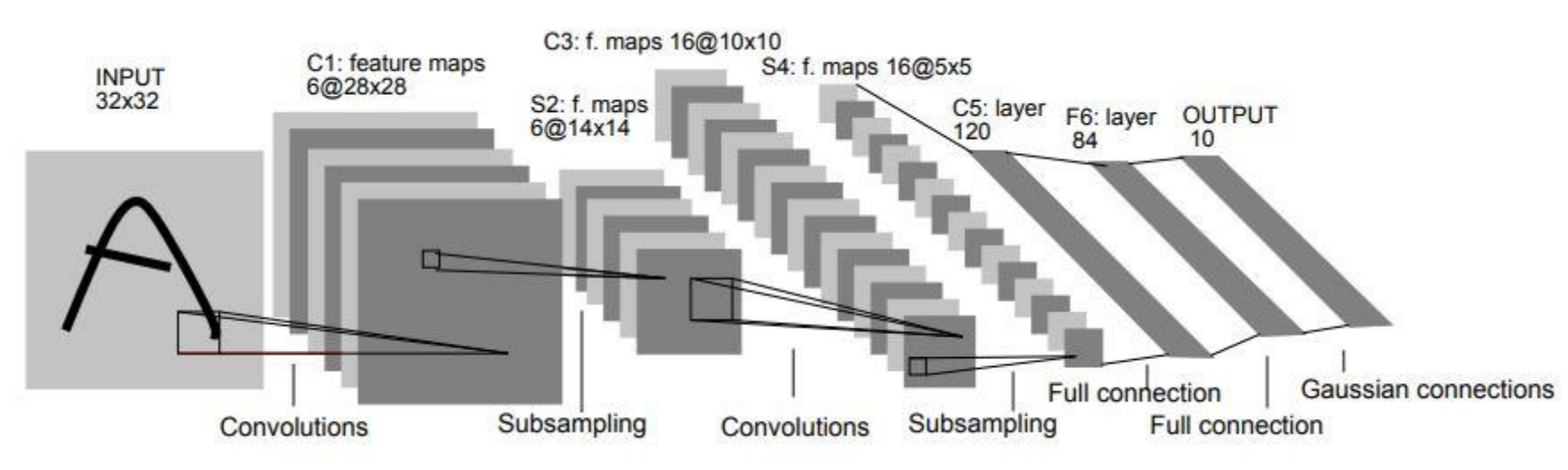
LeNet5

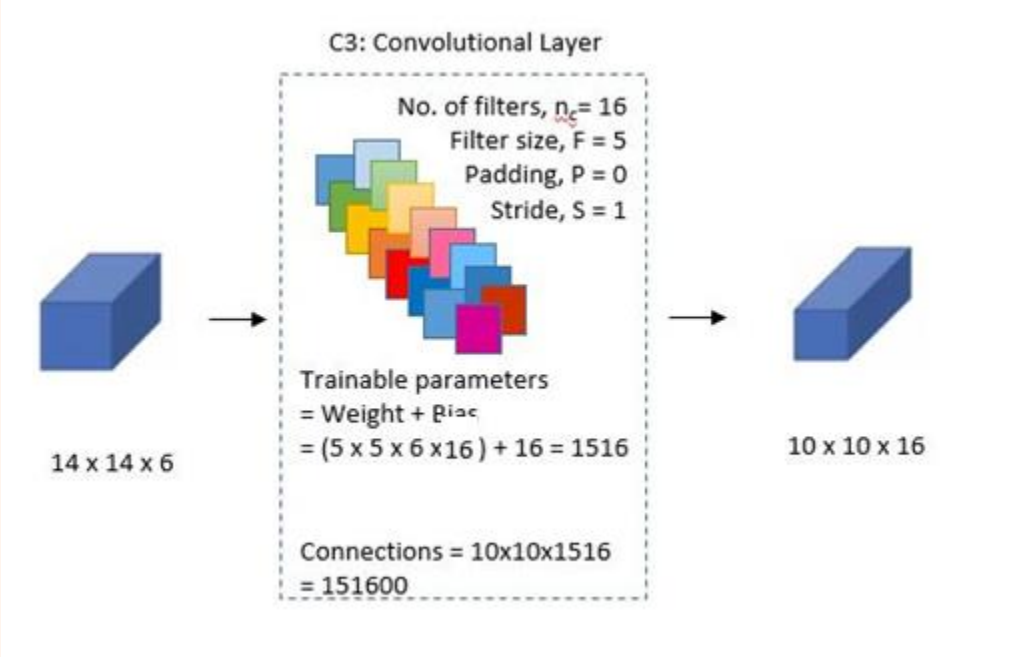




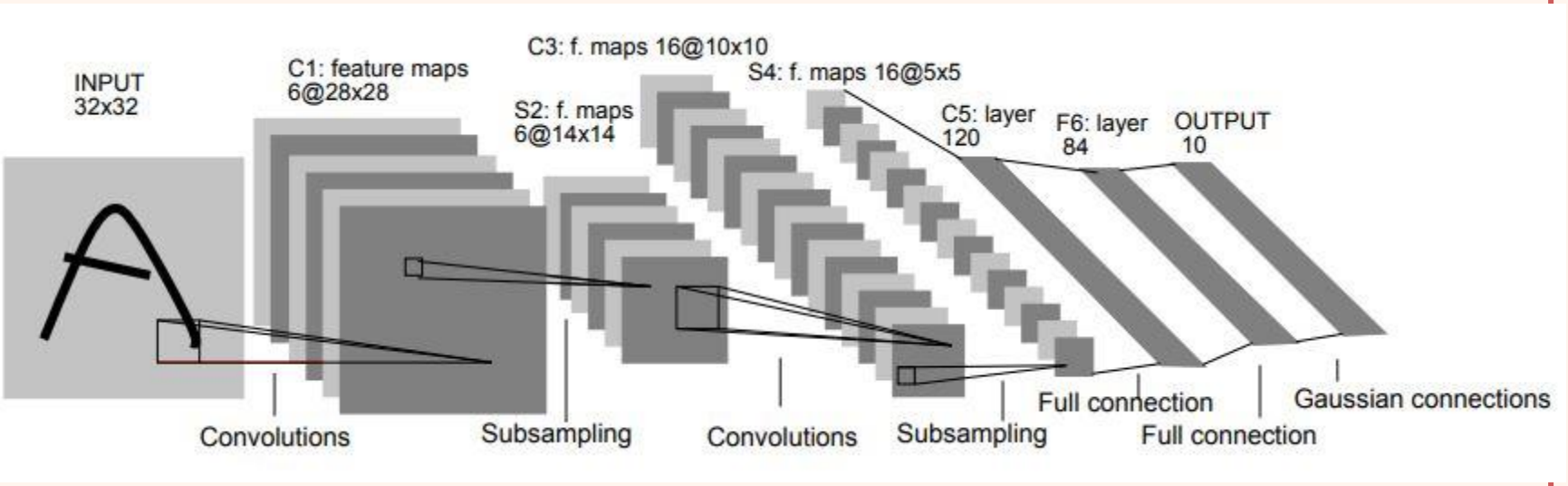
https://engmrk.com/lenet-5-a-classic-cnn-architecture/

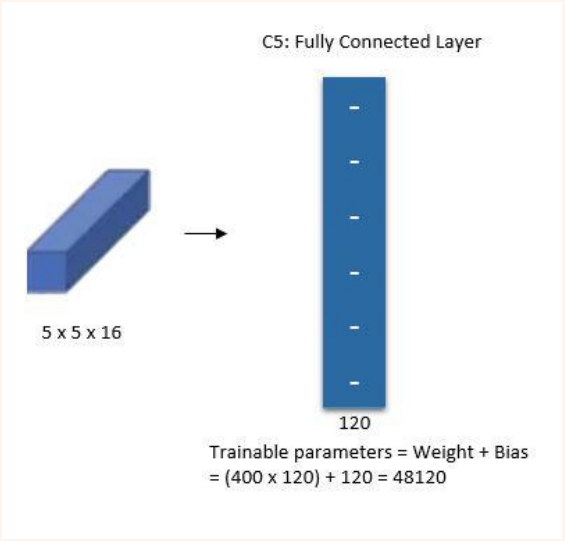
LeNet5

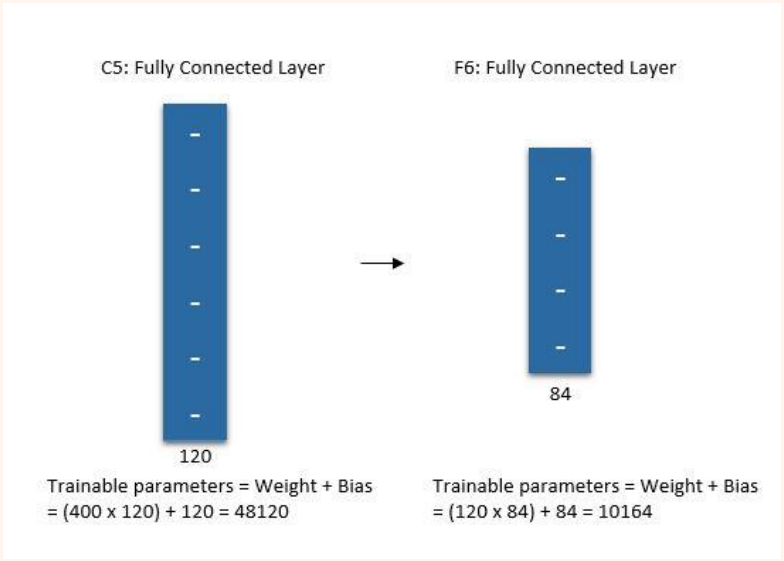

INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections


C3: Convolutional Layer
No. of filters, n_c = 16
Filter size, F = 5
Padding, P = 0
Stride, S = 1
Trainable parameters
= Weight + Bias
= (5 x 5 x 6 x16) + 16 = 1516
Connections = 10x10x1516
= 151600
14 x 14 x 6
10 x 10 x 16

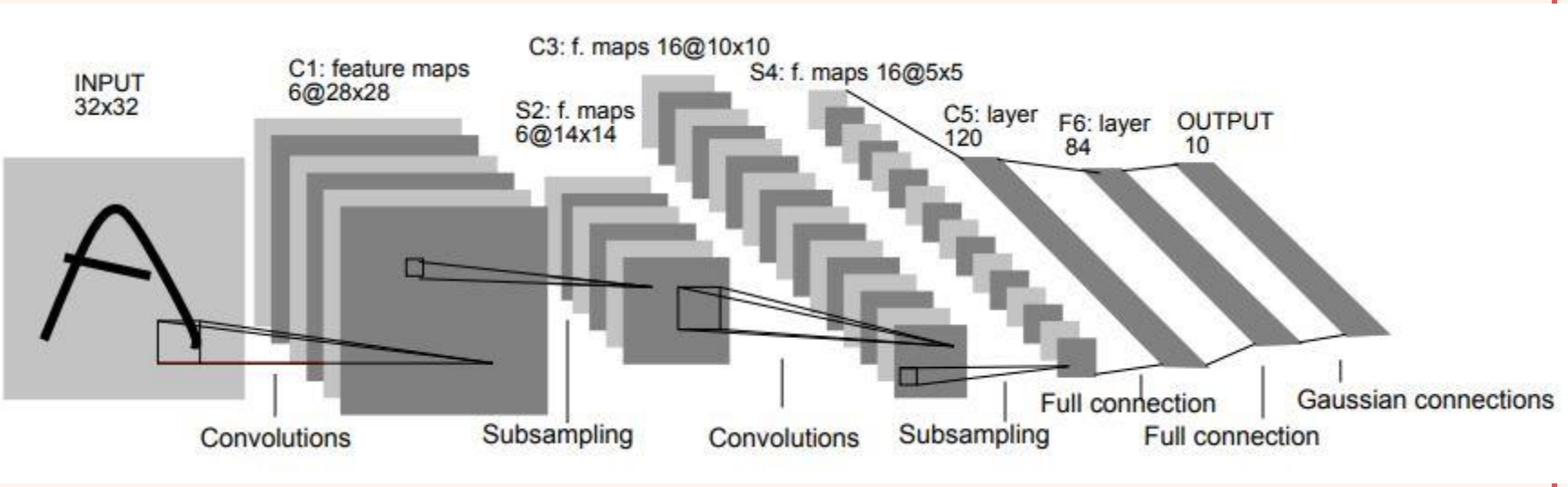
LeNet5

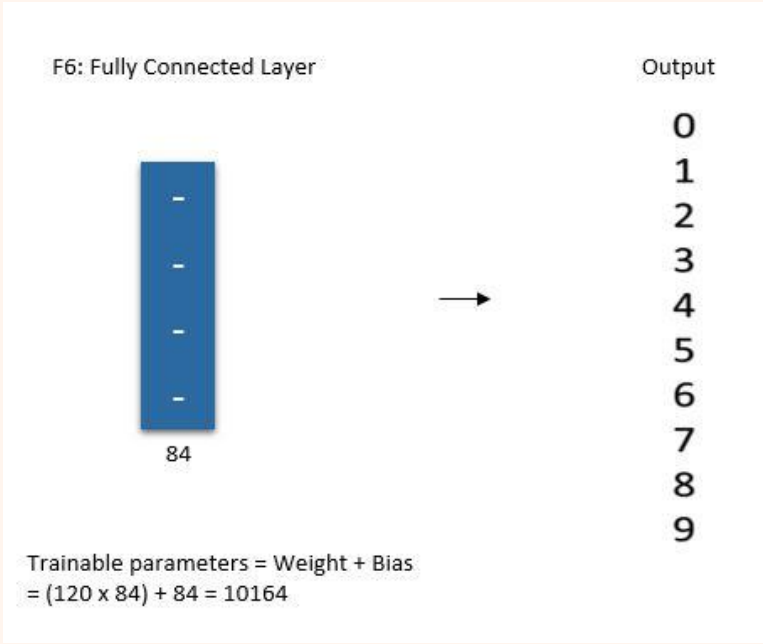

INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

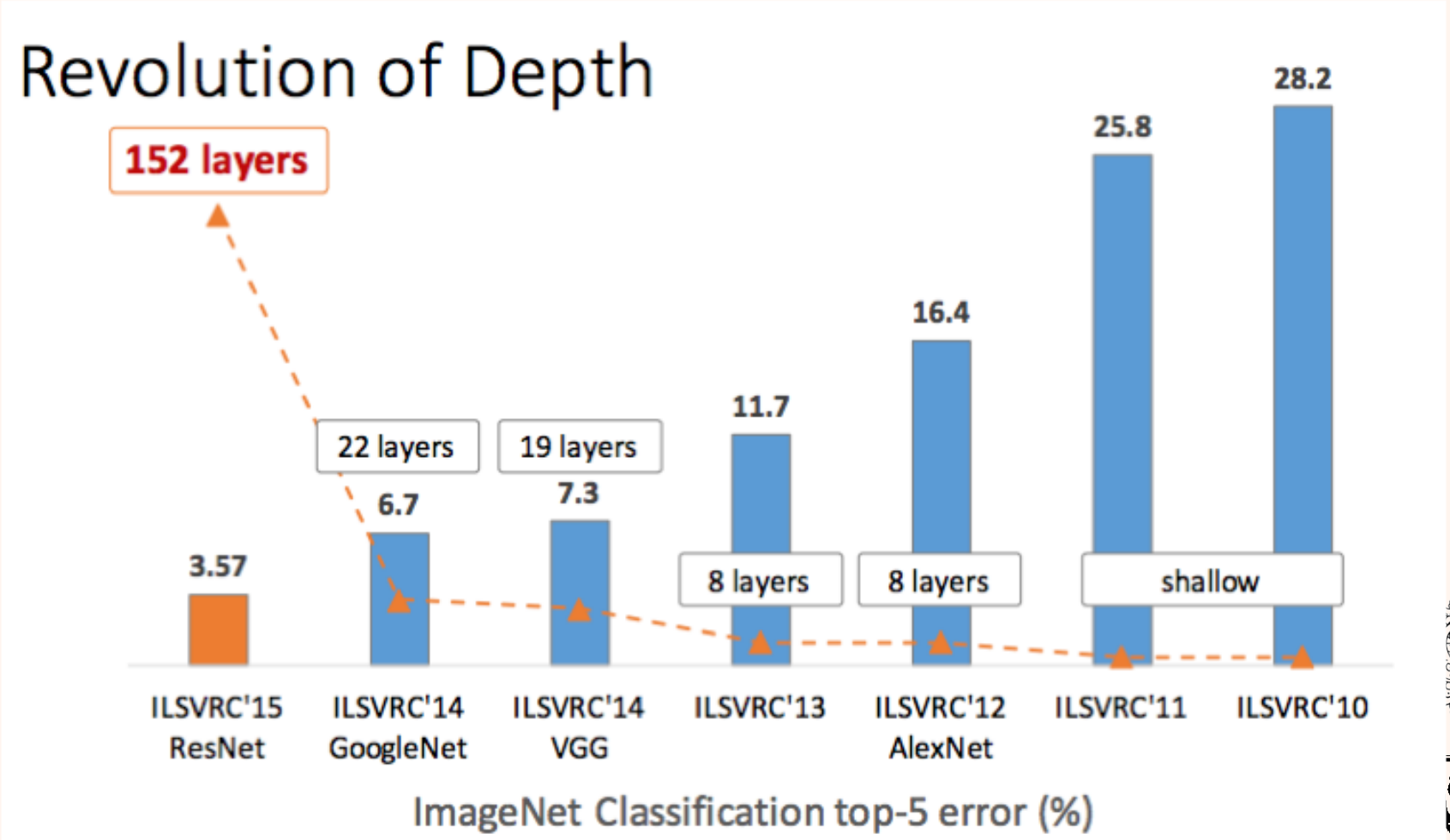

C5: Fully Connected Layer
5 x 5 x 16
120
Trainable parameters = Weight + Bias
= (400 x 120) + 120 = 48120


C5: Fully Connected Layer
F6: Fully Connected Layer
120
84
Trainable parameters = Weight + Bias
= (400 x 120) + 120 = 48120
Trainable parameters = Weight + Bias
= (120 x 84) + 84 = 10164

LeNet5

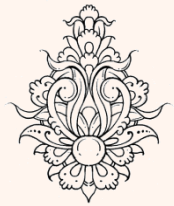

INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections


F6: Fully Connected Layer
Output
84
0
1
2
3
4
5
6
7
8
9
Trainable parameters = Weight + Bias
= (120 x 84) + 84 = 10164


Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

ImageNet Large Scale Visual Recognition Challenge, or ILSVRC

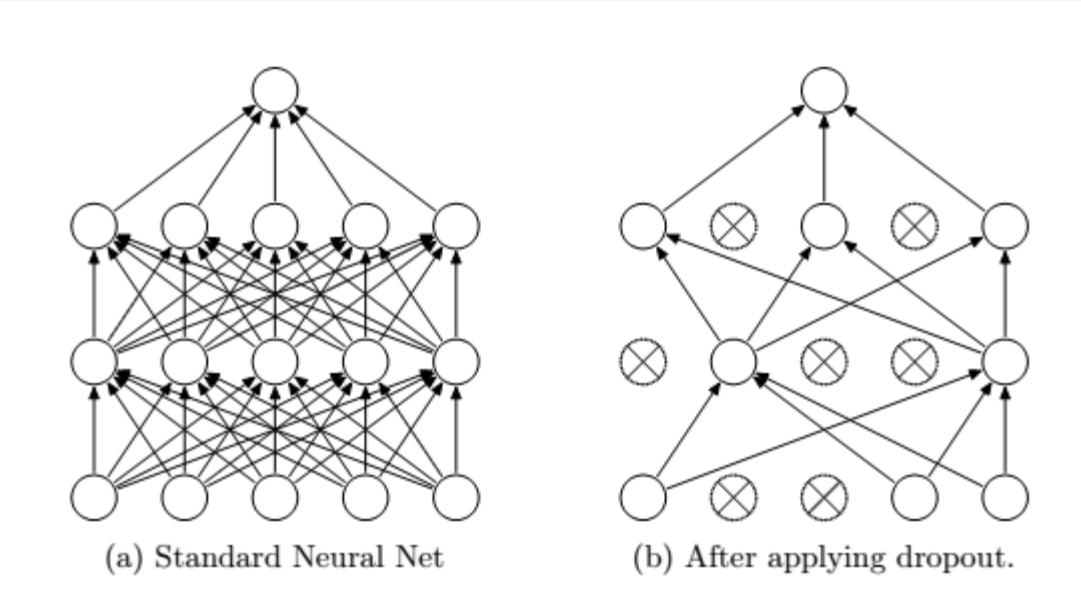
- یک مجموعه‌داده برچسب‌خورده برای بینایی ماشین است.
- ۱۴ میلیون تصویر در این مجموعه داده وجود دارد.
- حدود ۲۱ هزار کلاس و گروه(synset)
- در حدود یک میلیون محدوده اشیا با یک مستطیل مشخص شده‌اند.
- ILSVRC بین سال‌های ۲۰۱۰ و ۲۰۱۷ برگزار شد.



AlexNet architecture

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13 × 13	3 × 3	1	SAME	ReLU
C6	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
C5	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
S4	Max Pooling	256	13 × 13	3 × 3	2	VALID	–
C3	Convolution	256	27 × 27	5 × 5	1	SAME	ReLU
S2	Max Pooling	96	27 × 27	3 × 3	2	VALID	–
C1	Convolution	96	55 × 55	11 × 11	4	VALID	ReLU
In	Input	3 (RGB)	227 × 227	–	–	–	–

dropout



شیوه‌ای کارا برای جلوگیری از بیش‌برازش است

به صورت تصادفی خروجی برخی نرون‌ها را صفر در نظر می‌گیرد

مانع از این می‌شود که نورون با الگوی خاصی تطبیق پیدا کنند.

Journal of Machine Learning Research 15 (2014) 1929-1958 Submitted 11/13; Published 6/14

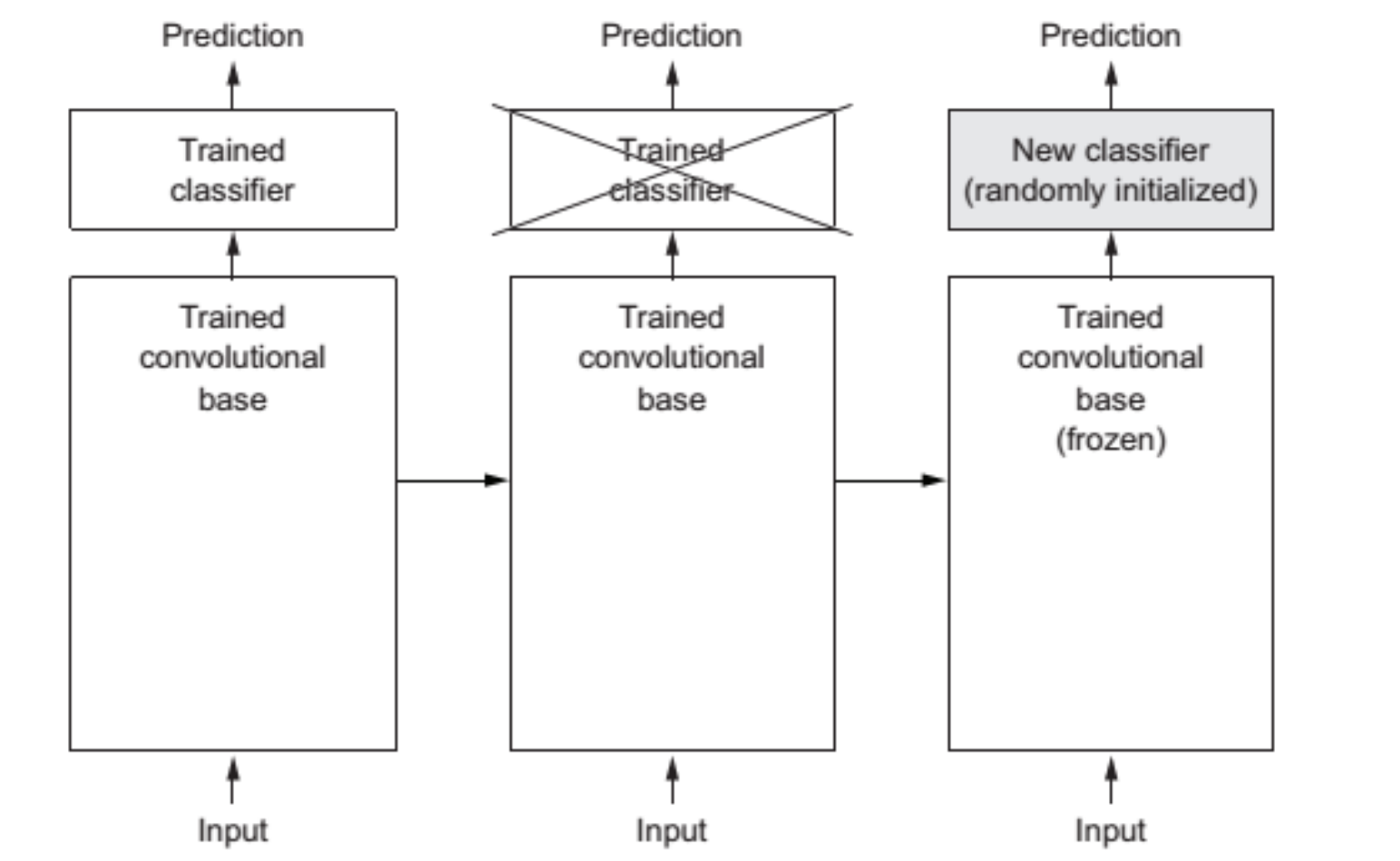
Dropout: A Simple Way to Prevent Neural Networks from Overfitting

data augmentation

شیوه‌ی دیگری برای جلوگیری از بیش‌برازش است

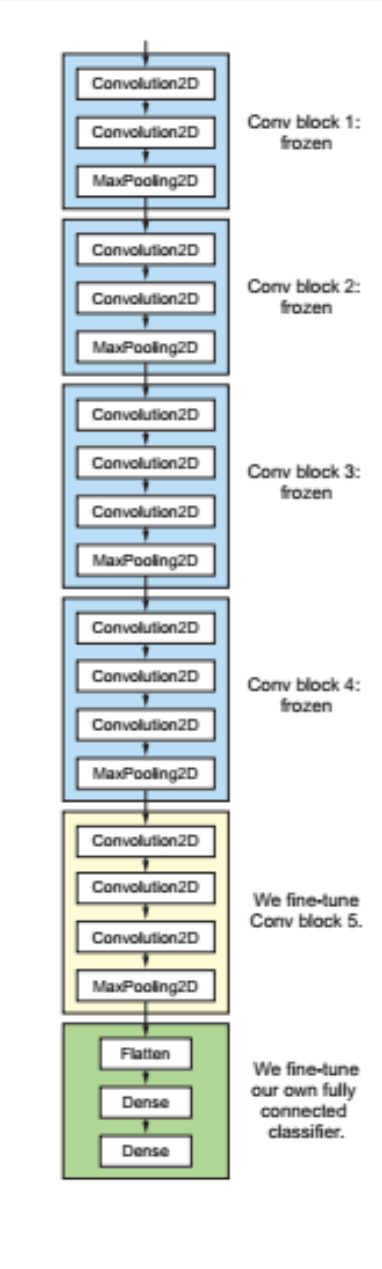
این شیوه با استفاده از تغییرات بر روی داده‌های آموزشی، به نوعی باعث تنوع داده‌های مورد استفاده در آموزش می‌شود.

استفاده از مدل‌های آماده



Transfer Learning

تنظیم دقیق



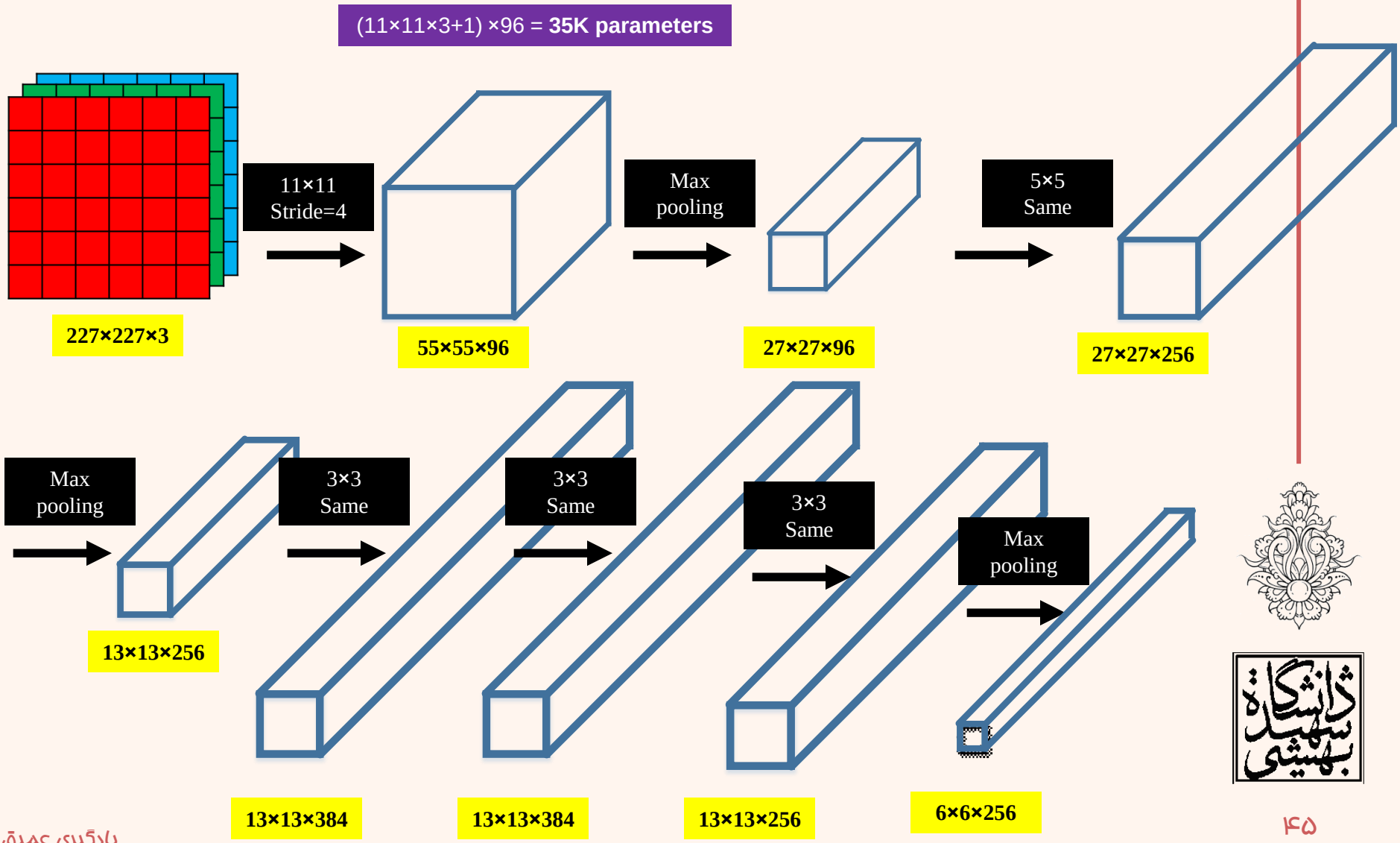
- هنگامی که از fine tune استفاده می‌کنیم لایه‌های نخست که خصوصیات عمومی را استخراج می‌کنند دست نخورده باقی می‌مانند.

- اگر تمامی لایه‌ها بنا باشد آموزش ببینند، به دلیل این که تعداد پارامترهای آزاد بالا است، احتمال بیش‌برازش بالا می‌رود.

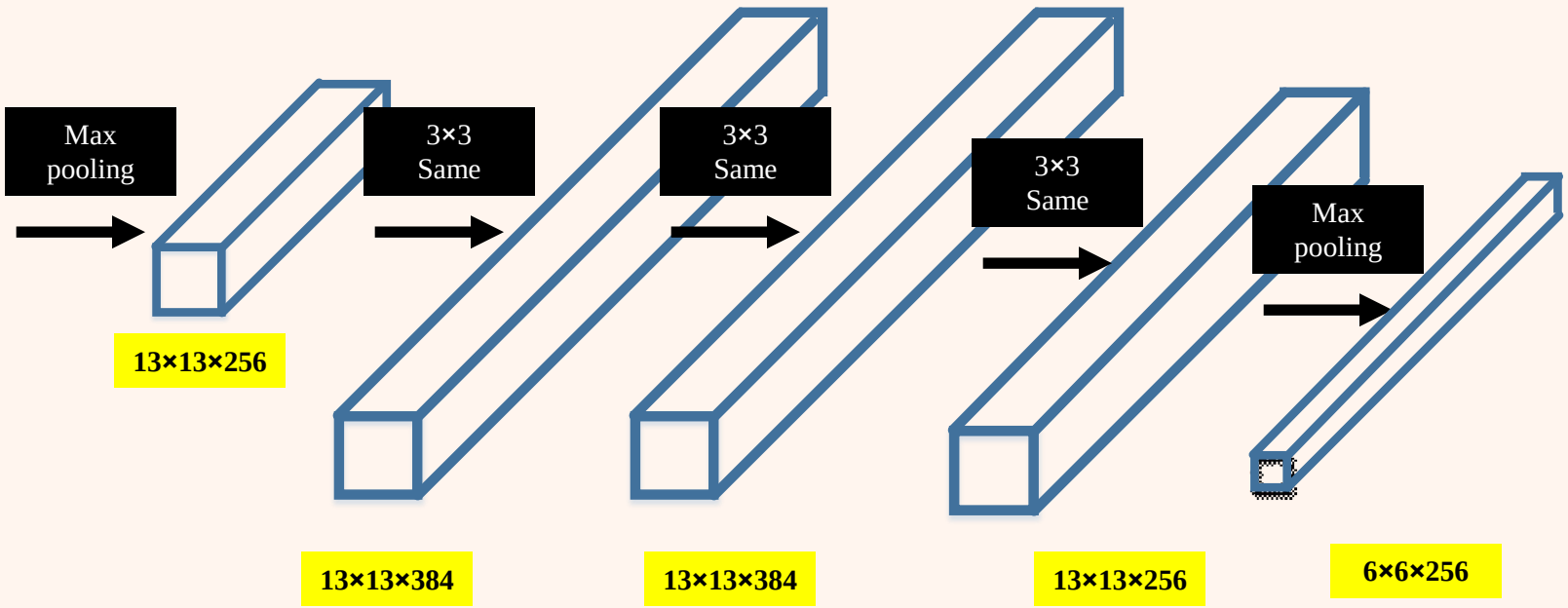
- شبکه‌ای که تعداد پارامترهای آزاد آن بالاست وقتی با دیتاست کوچک آموزش ببیند احتمال عملکرد قابل قبول آن پایین است.

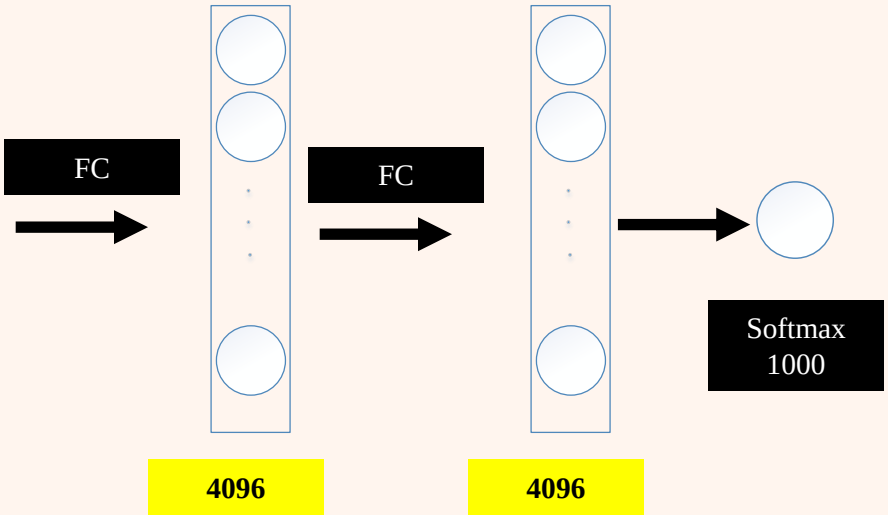
AlexNet

[Krizhevsky et al. 2012]



AlexNet


Max pooling
13×13×256
3×3 Same
13×13×384
3×3 Same
13×13×384
3×3 Same
13×13×256
Max pooling
6×6×256


FC
4096
FC
4096
Softmax 1000

AlexNet

50% dropout rate during training
to the outputs of layers F8 and F9.

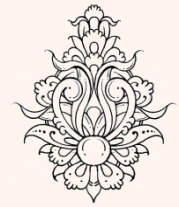
heavy data augmentation

batch size 128

SGD Momentum 0.9
Learning rate 1e-2, reduced by 10

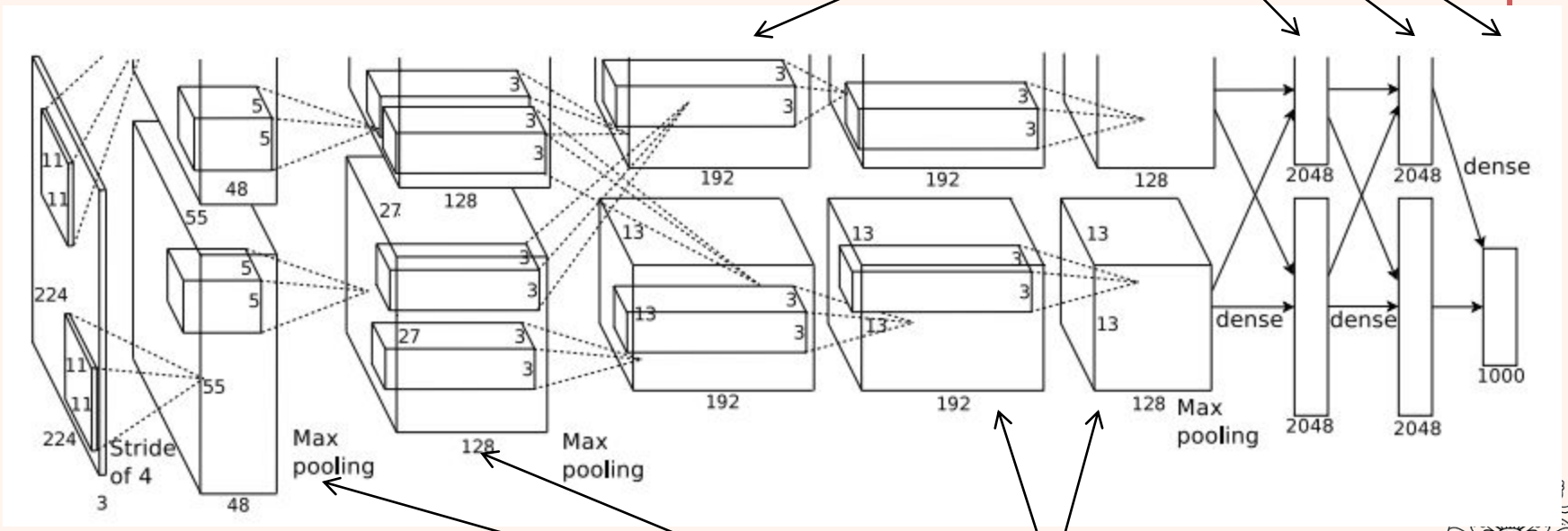
competitive normalization step immediately after the ReLU step
of layers C1 and C3, called *local response normalization*.

Trained on GTX 580 GPU with only 3 GB of memory.
Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.



AlexNet

لایه های کانولوشنی سه و لایه های FC به تمام داده های Feature map ها نیاز دارند بنابراین بین GPU ها ارتباط خواهند داشت.

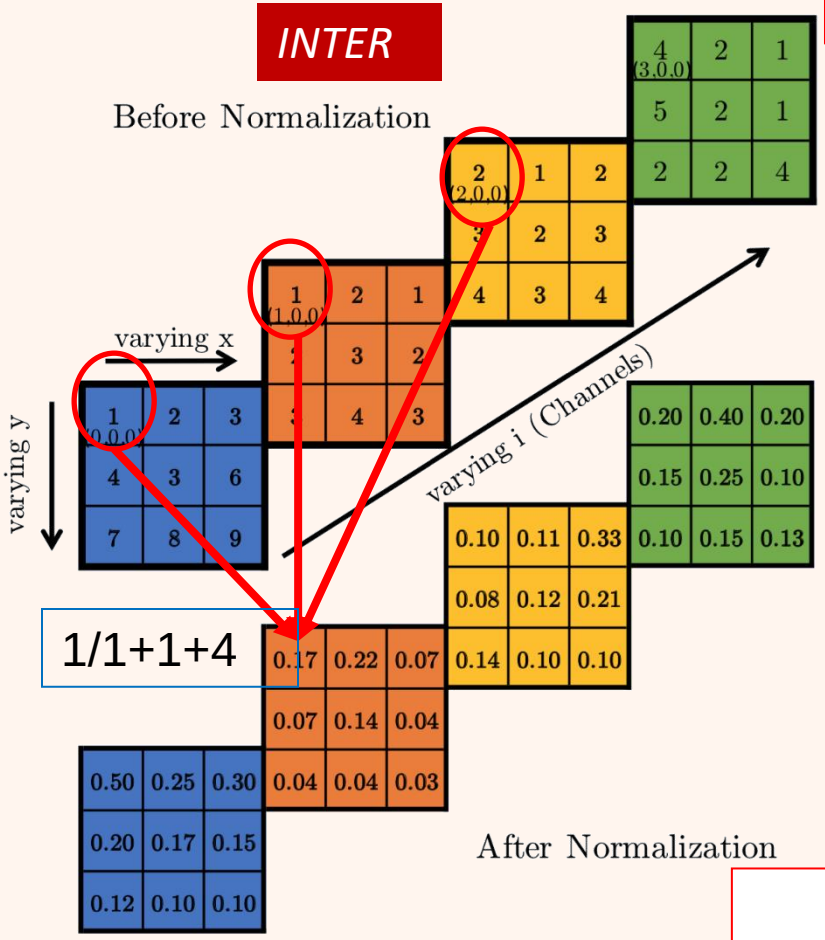


لایه های کانولوشنی یک دو چهار و پنج فقط با feature map ها یی در GPU یکسان در ارتباط هستند

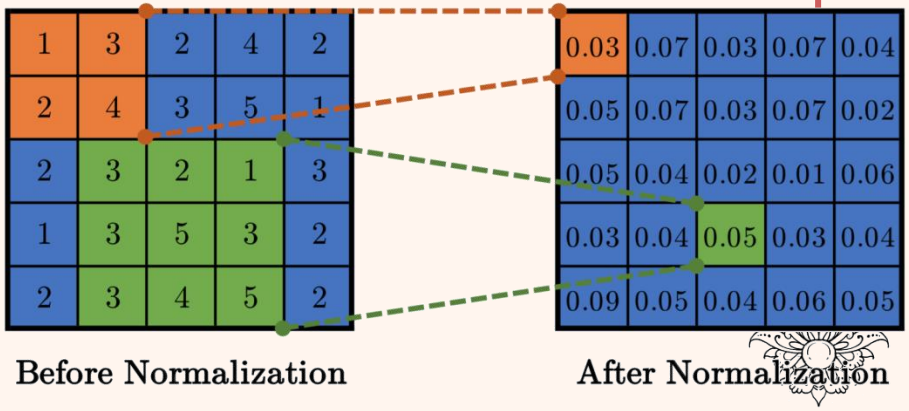
local response normalization

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

INTER



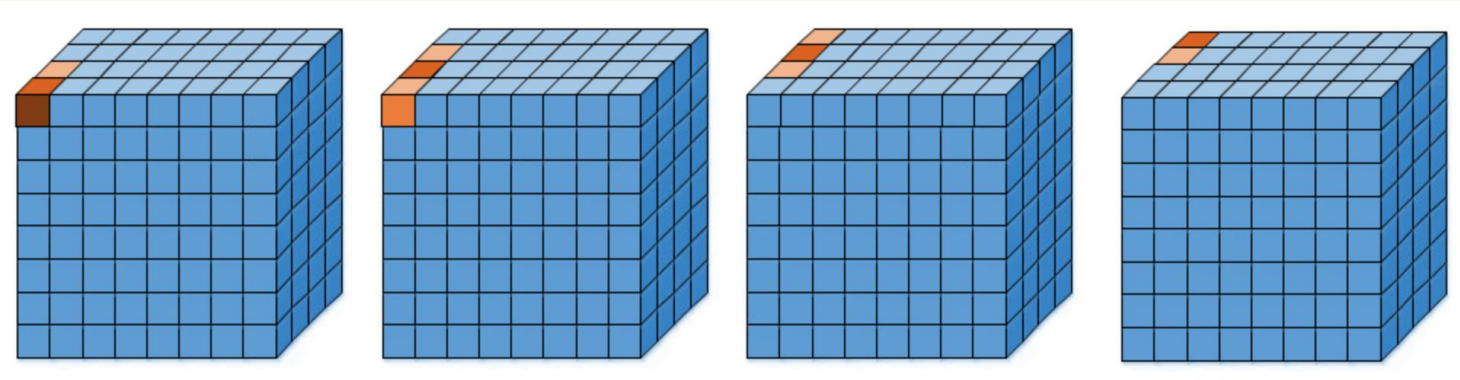
INTRA



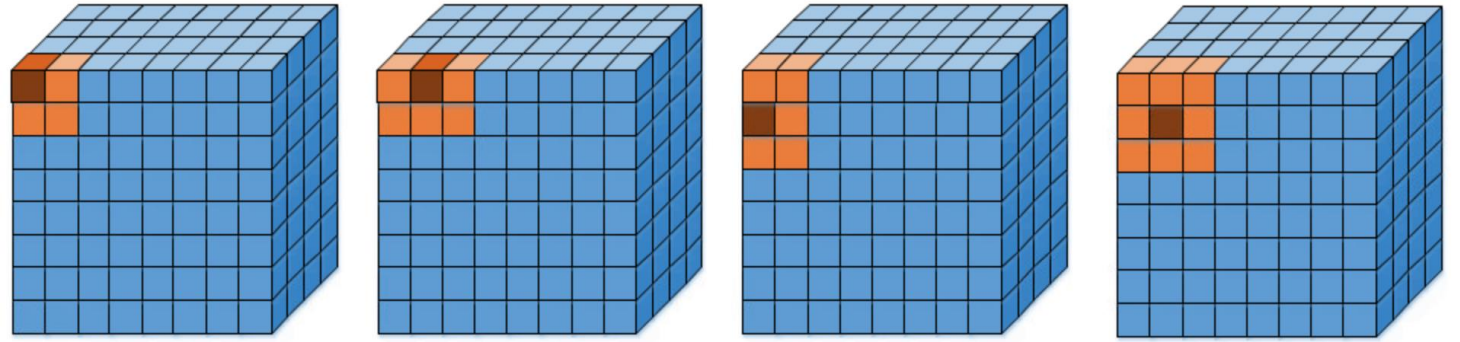
$$b^k_{x,y} = a^k_{x,y} / \left(k + \alpha \sum_{i=max(0,x-n/2)}^{min(W,x+n/2)} \sum_{j=max(0,y-n/2)}^{min(H,y+n/2)} (a^k_{i,j})^2 \right)^{\beta}$$

یادگیری عمیق

https://towardsdatascience.com/difference-between-local-response-normalization-and-batch-normalization-272308c034ac



a) Inter-Channel LRN (n=2)



b) Intra-Channel LRN (n=2)

Batch Normalization

The problem of input values changing

"internal covariate shift"

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

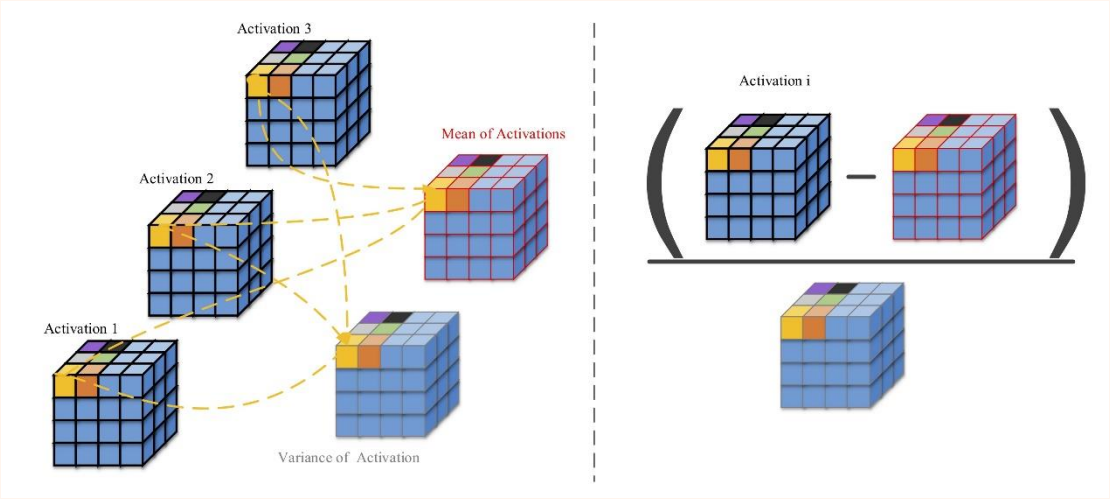
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., *sioffe@google.com*

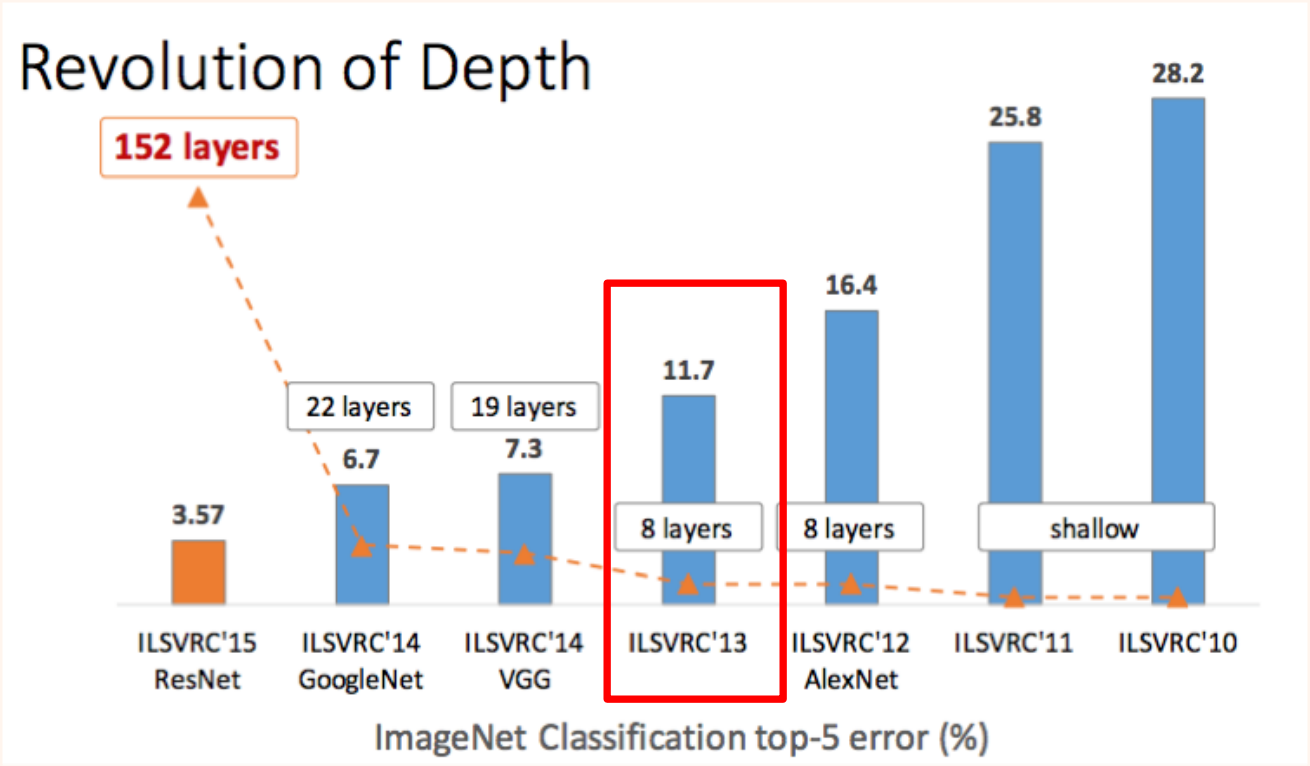
Christian Szegedy
Google Inc., *szegedy@google.com*

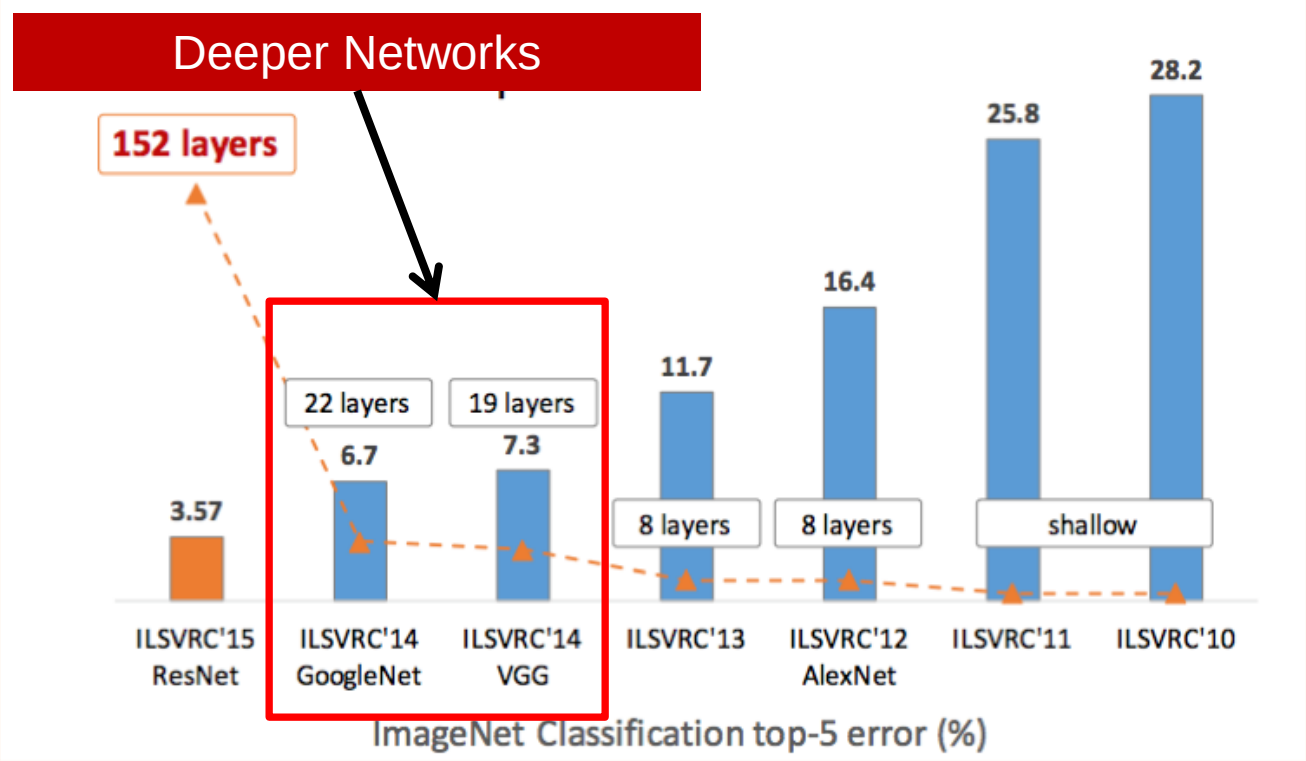


Comparison of the two normalization techniques in DNNs				
Norm Type	**Trainable**	**Training Parameters**	**Fouses on**	Regularization
LRN	No	0	Lateral Inhibition	No
BN	Yes	2 (Scale and Shift)	Internal Covariate Shift	Yes

ZF Net

AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

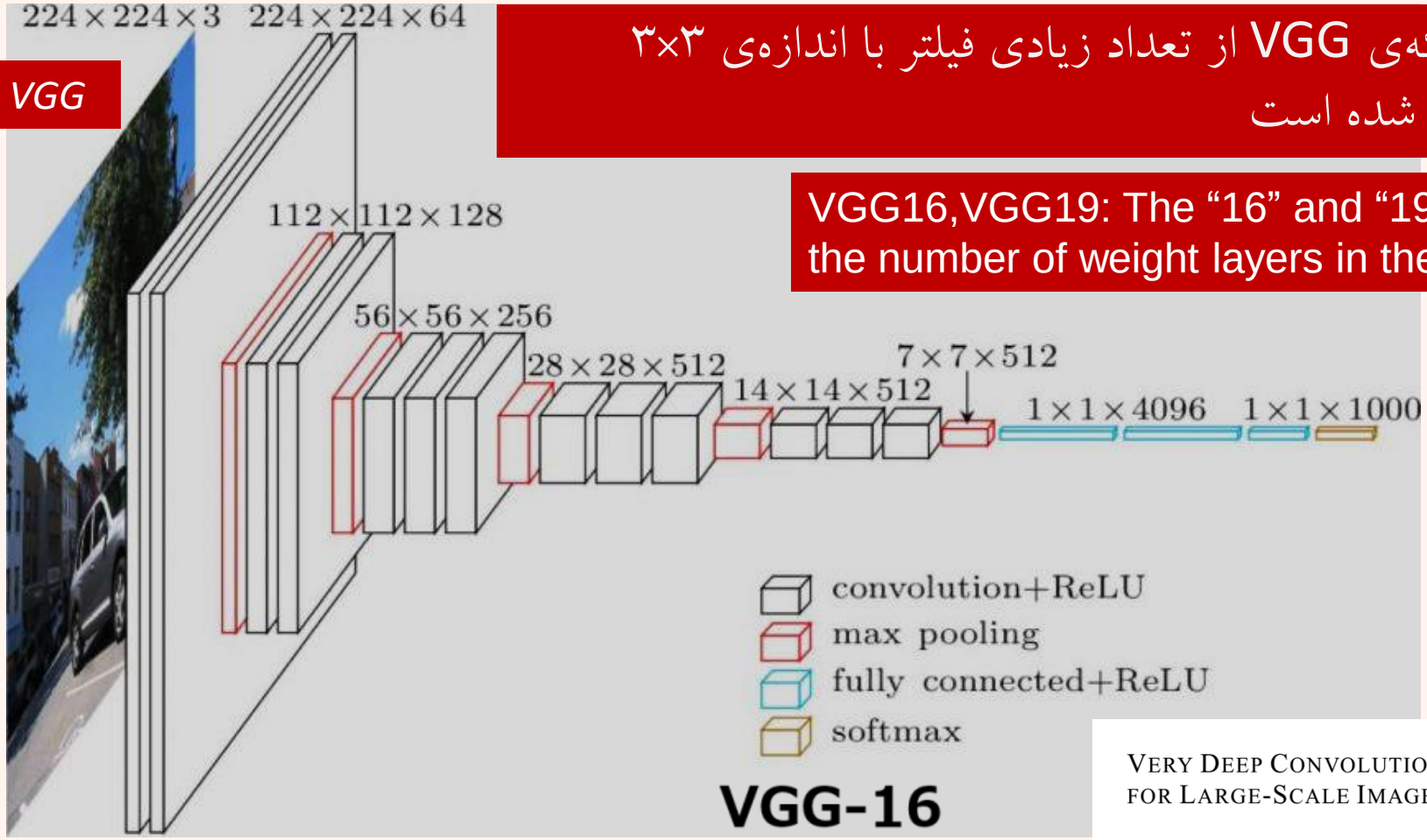



Deeper Networks
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

VGG

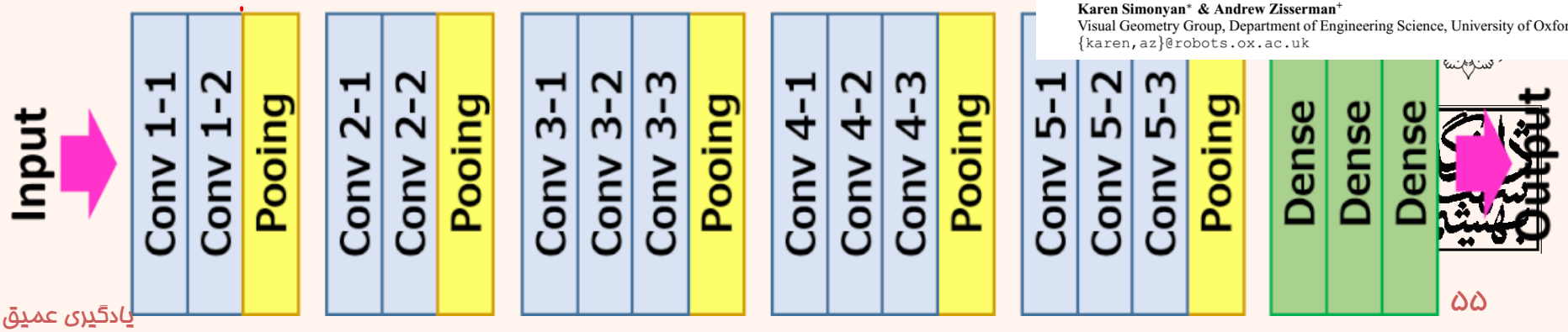
در شبکه‌ی VGG از تعداد زیادی فیلتر با اندازه‌ی ۳×۳ استفاده شده است

VGG16,VGG19: The “16” and “19” stand for the number of weight layers in the network



VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* & Andrew Zisserman+
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk



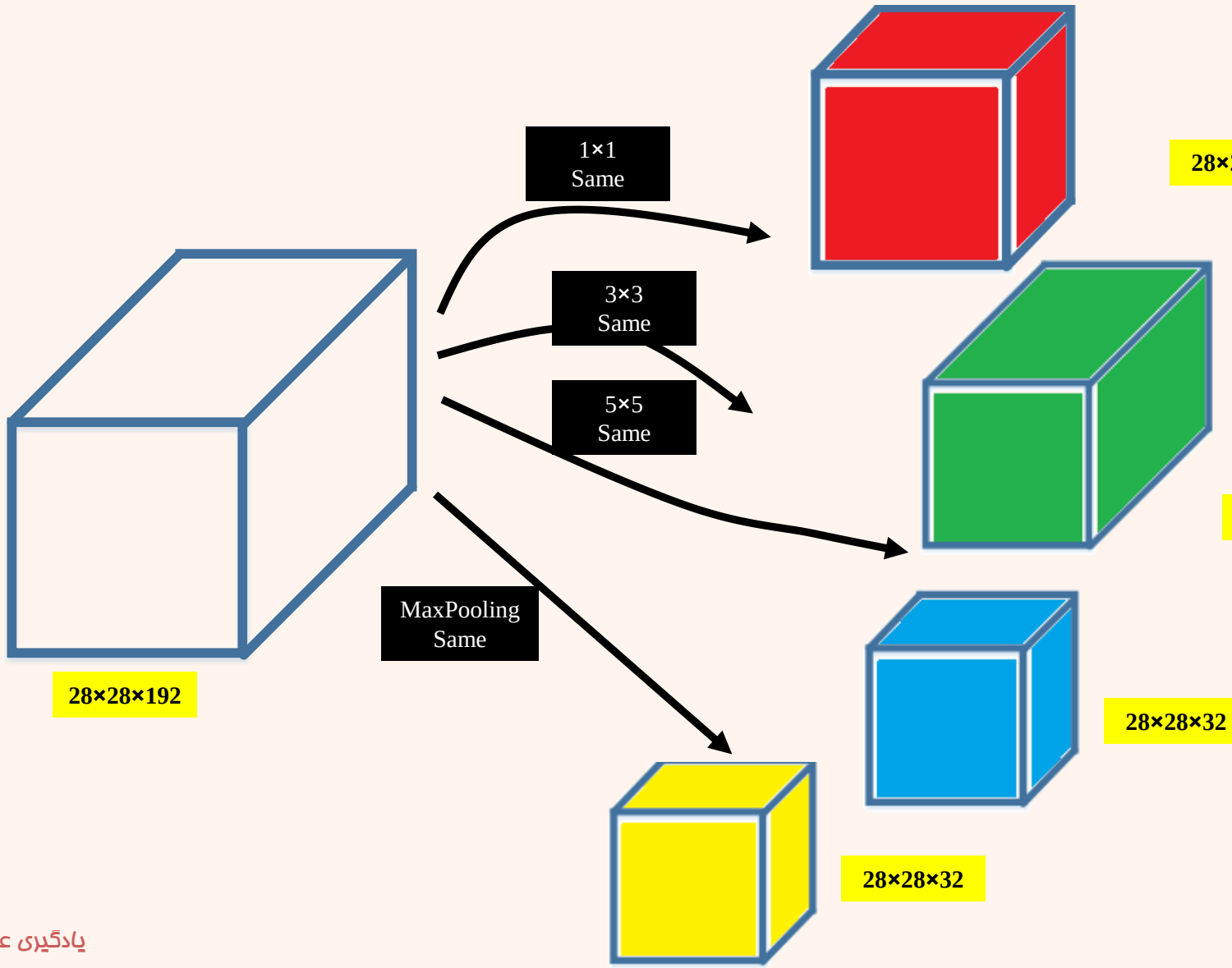
https://neurohive.io/en/popular-networks/vgg16/

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

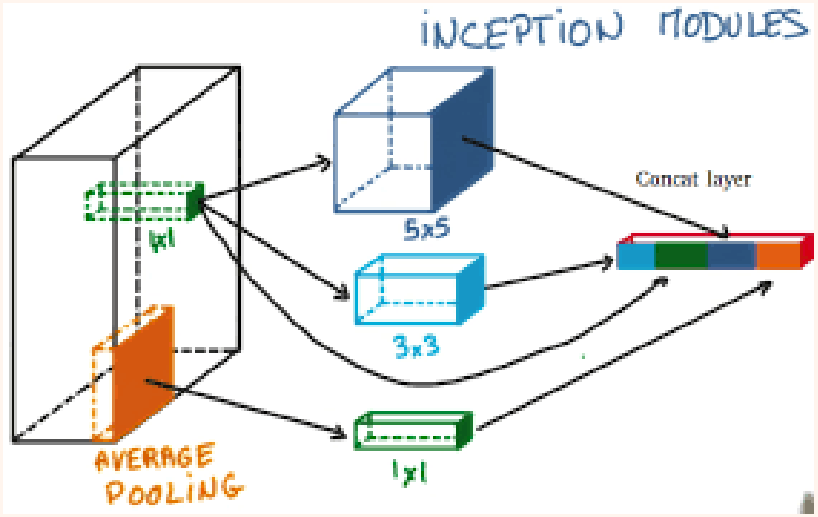
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

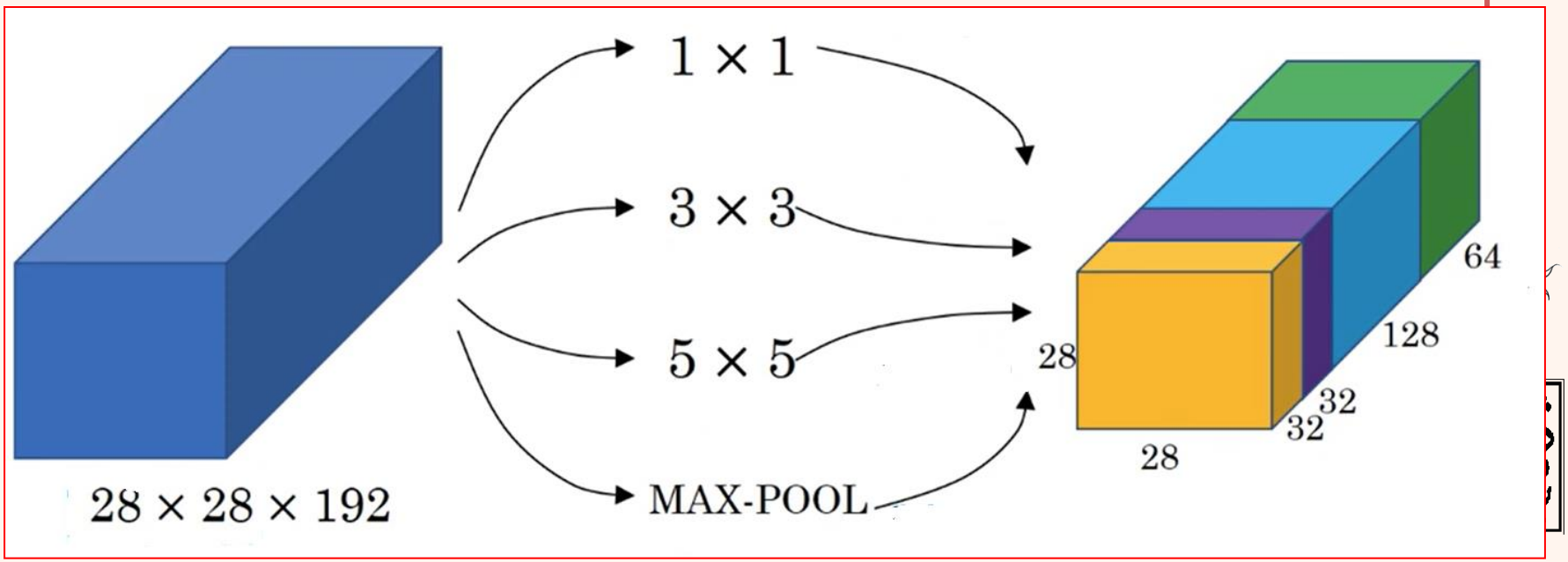
Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144


1×1
Same
3×3
Same
5×5
Same
MaxPooling
Same
28×28×192
28×28×64
28×28×128
28×28×32
28×28×32

Inception Module





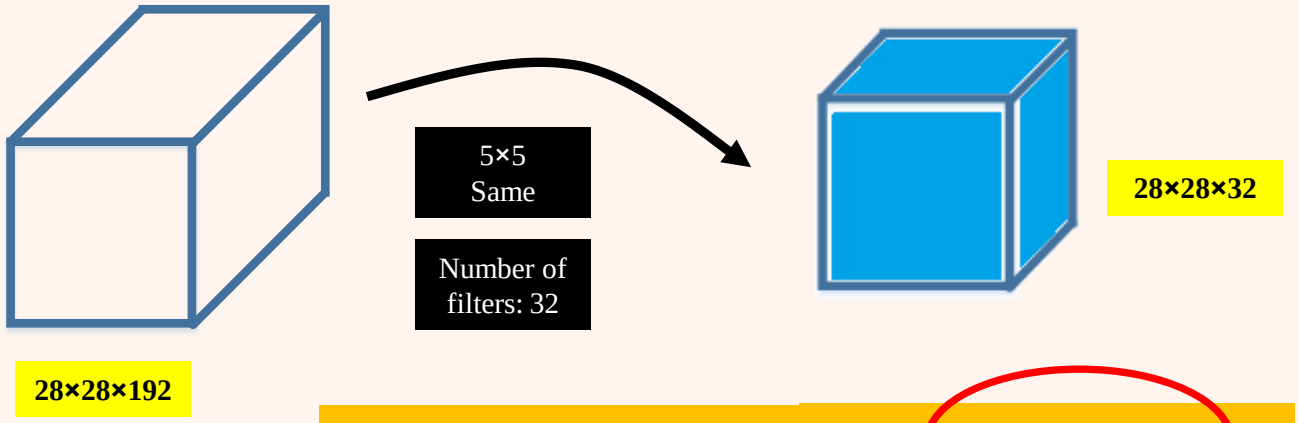
در این حالت اندازه‌ی مدل کاهش می‌یابد و این مساله به کاهش بیش‌برازش کمک می‌کند.

1.The 1×1 Convolution
2.Inception Module
3.Global Average Pooling
4.Overall Architecture
5.Auxiliary Classifiers for Training

In GoogLeNet, 1×1 convolution is used as a dimension reduction module to reduce the computation. By reducing the computation bottleneck, depth and width can be increased.

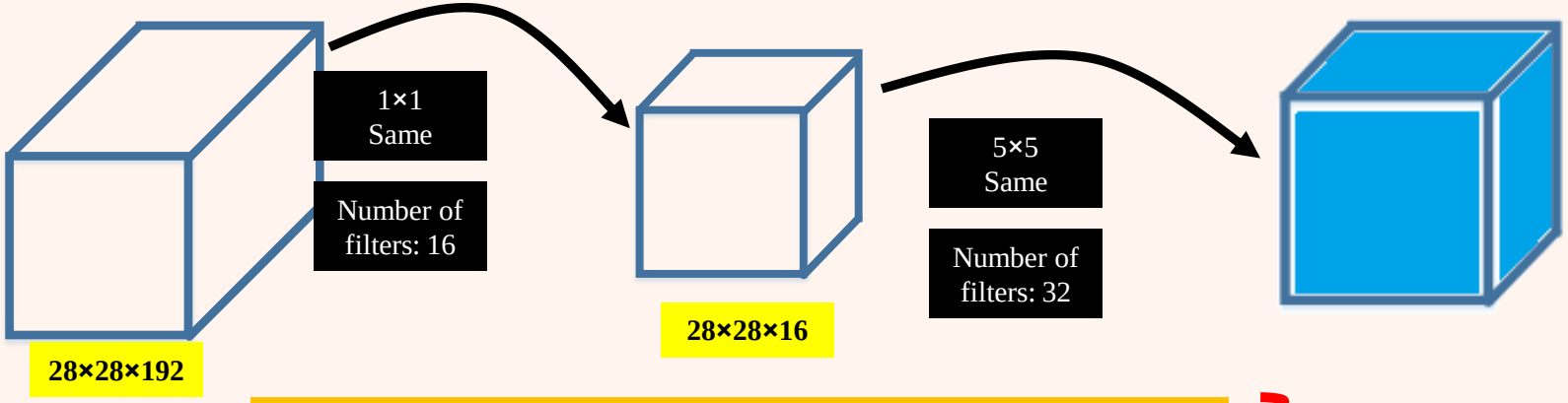
https://medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7

پیچیدگی محاسباتی



28×28×32×5×5×192 =120M

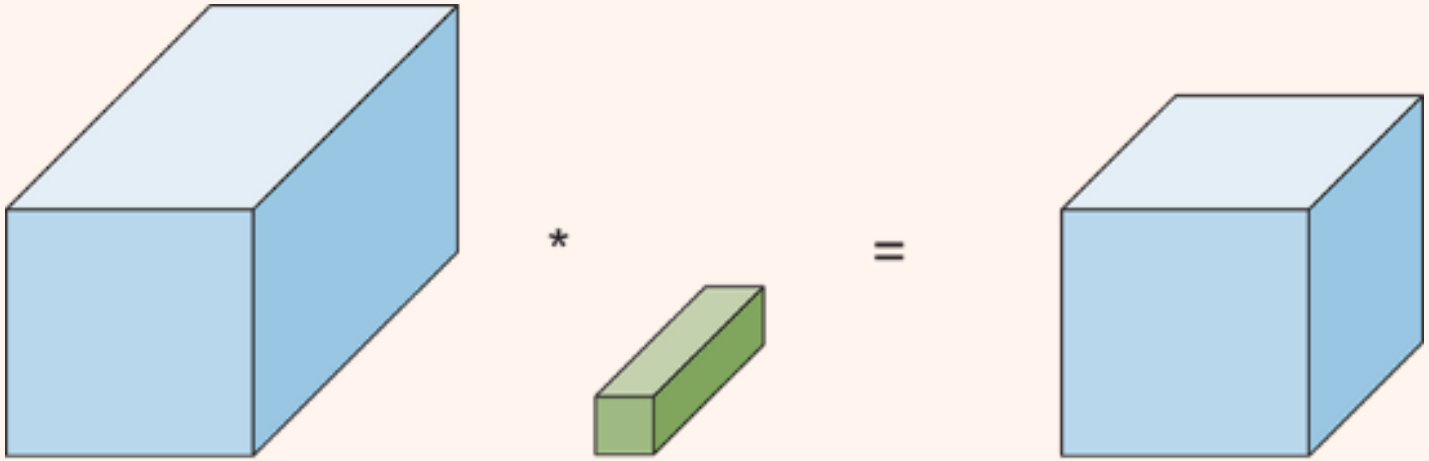
Using 1×1 Convolution

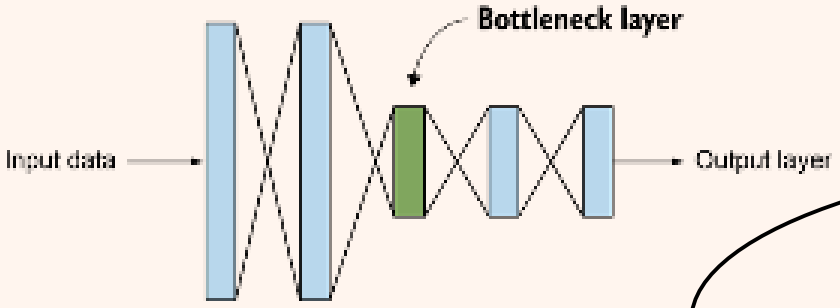


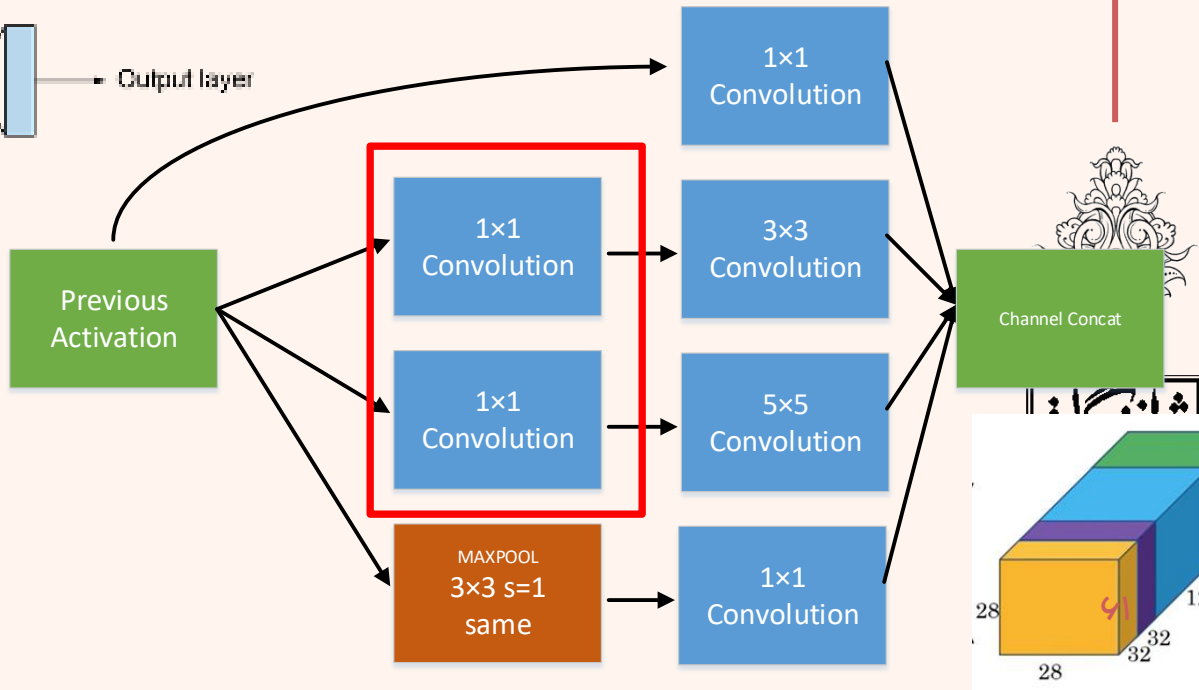
28×28×16×1×1×192 =2.4M

28×28×32×5×5×16 =10M

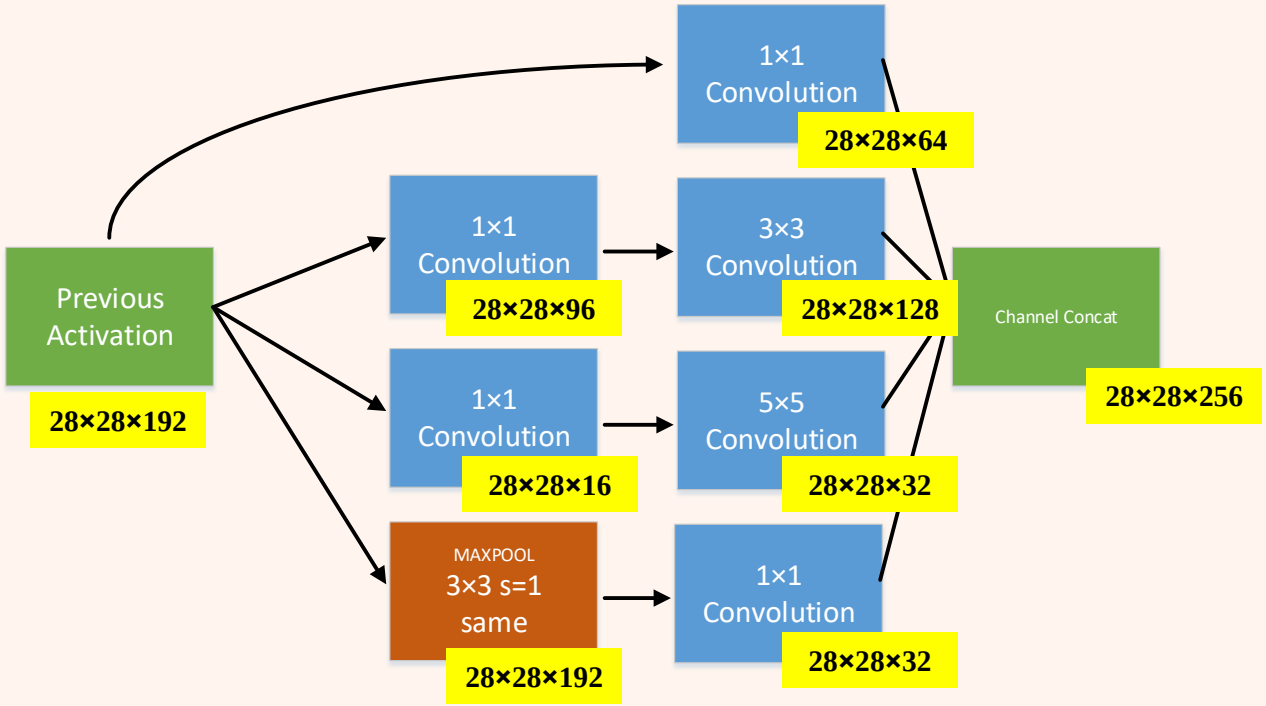
=12.4M



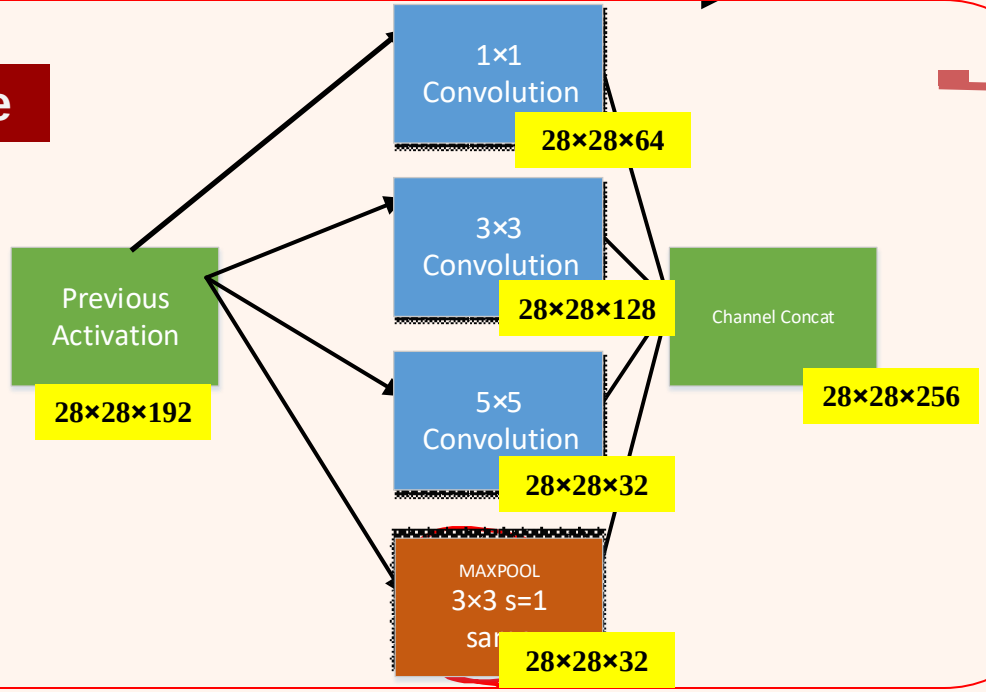




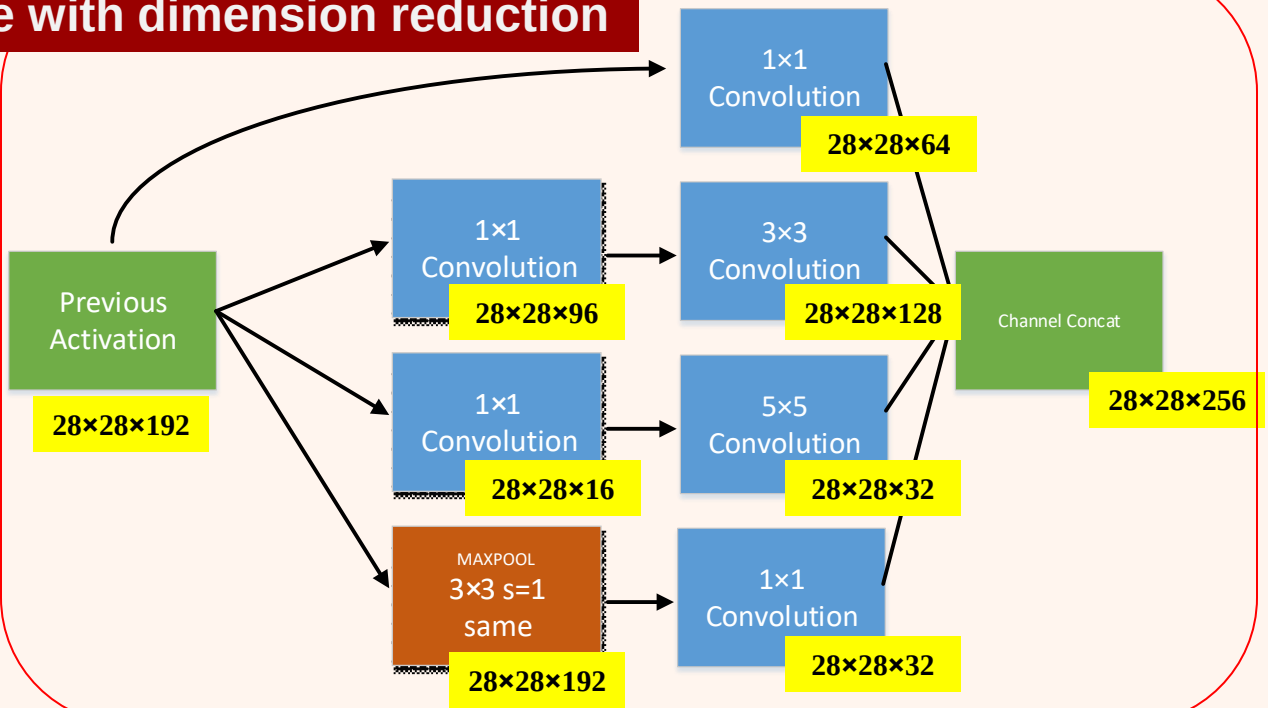
Inception Module



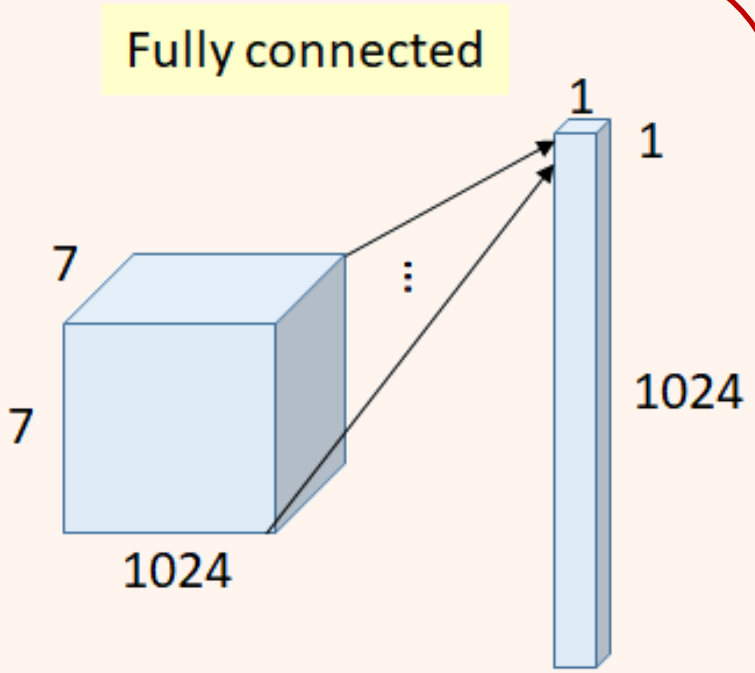
Naïve Inception Module



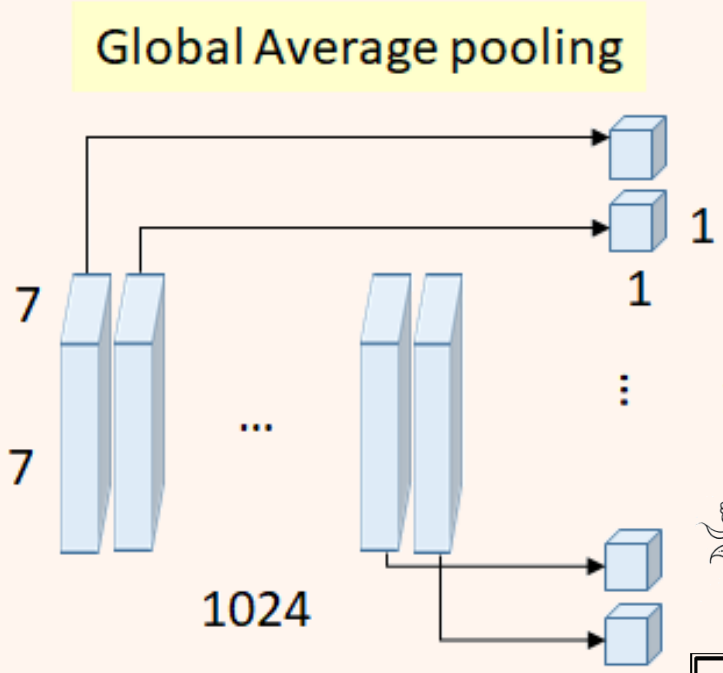
Inception Module with dimension reduction



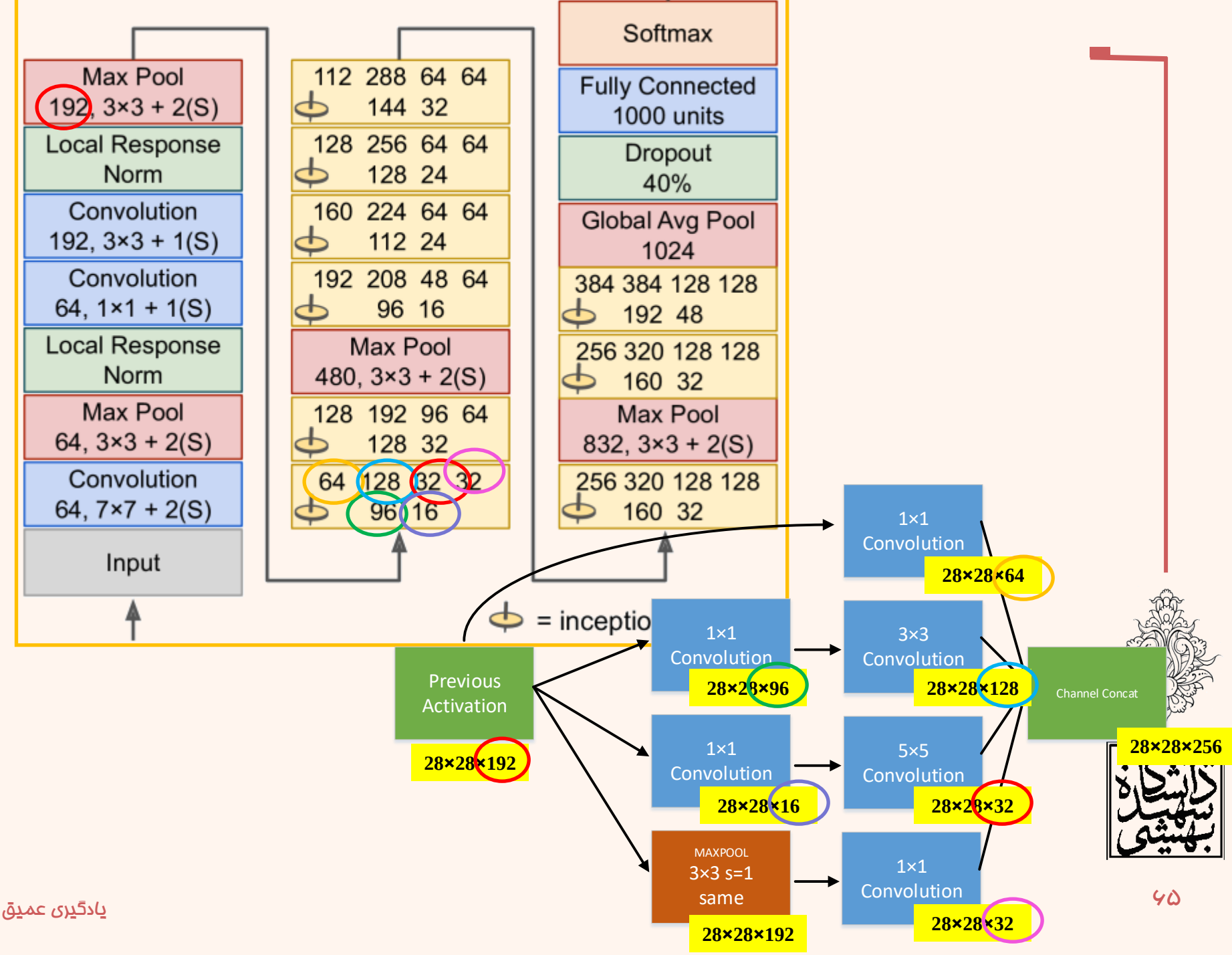
1.The 1×1 Convolution
2.Inception Module
3.Global Average Pooling
4.Overall Architecture
5.Auxiliary Classifiers for Training

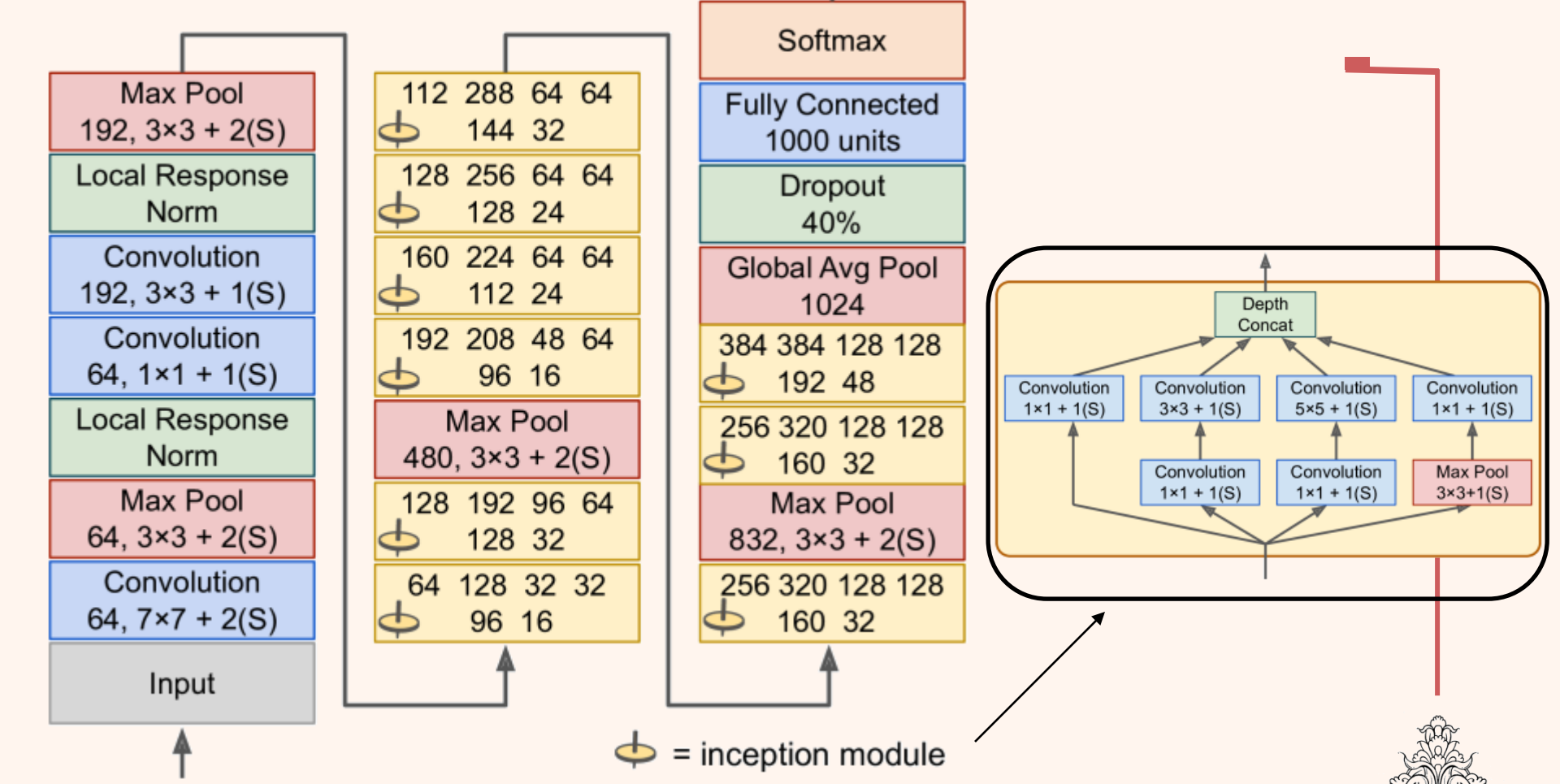


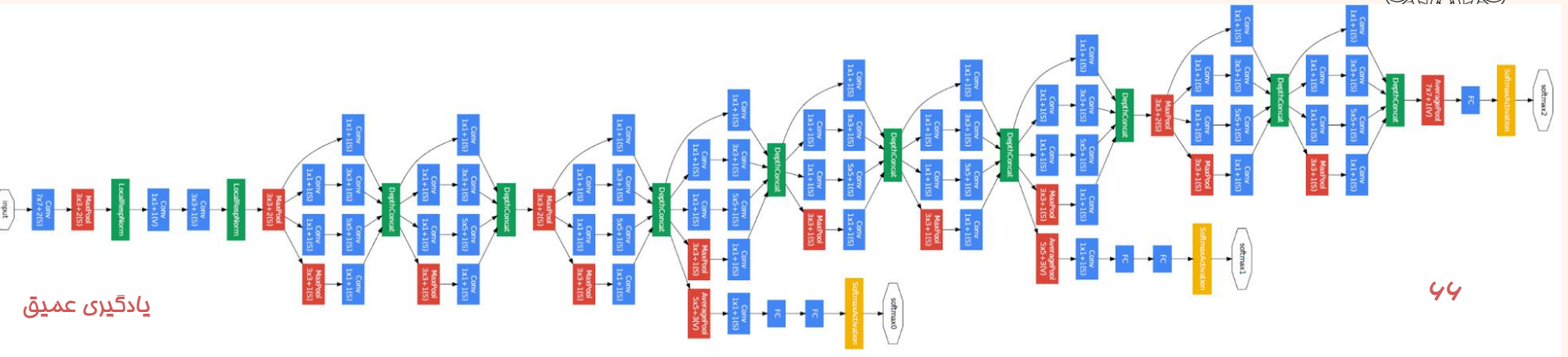
Number of weights (connections) = 7×7×1024×1024 = 51.3M



Number of weights = 0


Softmax
Fully Connected 1000 units
Dropout 40%
Global Avg Pool 1024
384 384 128 128
192 48
256 320 128 128
160 32
Max Pool 832, 3×3 + 2(S)
256 320 128 128
160 32
112 288 64 64
144 32
128 256 64 64
128 24
160 224 64 64
112 24
192 208 48 64
96 16
Max Pool 480, 3×3 + 2(S)
128 192 96 64
128 32
64 128 32 32
96 16
Max Pool 192, 3×3 + 2(S)
Local Response Norm
Convolution 192, 3×3 + 1(S)
Convolution 64, 1×1 + 1(S)
Local Response Norm
Max Pool 64, 3×3 + 2(S)
Convolution 64, 7×7 + 2(S)
Input
= inceptio
Previous Activation
28×28×192
1×1 Convolution
28×28×96
3×3 Convolution
28×28×128
1×1 Convolution
28×28×16
5×5 Convolution
28×28×32
MAXPOOL 3×3 s=1 same
28×28×192
1×1 Convolution
28×28×32
1×1 Convolution
28×28×64
Channel Concat
28×28×256


Max Pool
192, 3×3 + 2(S)
Local Response Norm
Convolution
192, 3×3 + 1(S)
Convolution
64, 1×1 + 1(S)
Local Response Norm
Max Pool
64, 3×3 + 2(S)
Convolution
64, 7×7 + 2(S)
Input
112 288 64 64
144 32
128 256 64 64
128 24
160 224 64 64
112 24
192 208 48 64
96 16
Max Pool
480, 3×3 + 2(S)
128 192 96 64
128 32
64 128 32 32
96 16
Softmax
Fully Connected
1000 units
Dropout
40%
Global Avg Pool
1024
384 384 128 128
192 48
256 320 128 128
160 32
Max Pool
832, 3×3 + 2(S)
256 320 128 128
160 32
= inception module
Depth Concat
Convolution 1×1 + 1(S)
Convolution 3×3 + 1(S)
Convolution 5×5 + 1(S)
Convolution 1×1 + 1(S)
Convolution 1×1 + 1(S)
Convolution 1×1 + 1(S)
Max Pool 3×3+1(S)



1.The 1×1 Convolution
2.Inception Module
3.Global Average Pooling
4.Overall Architecture
5.Auxiliary Classifiers for Training

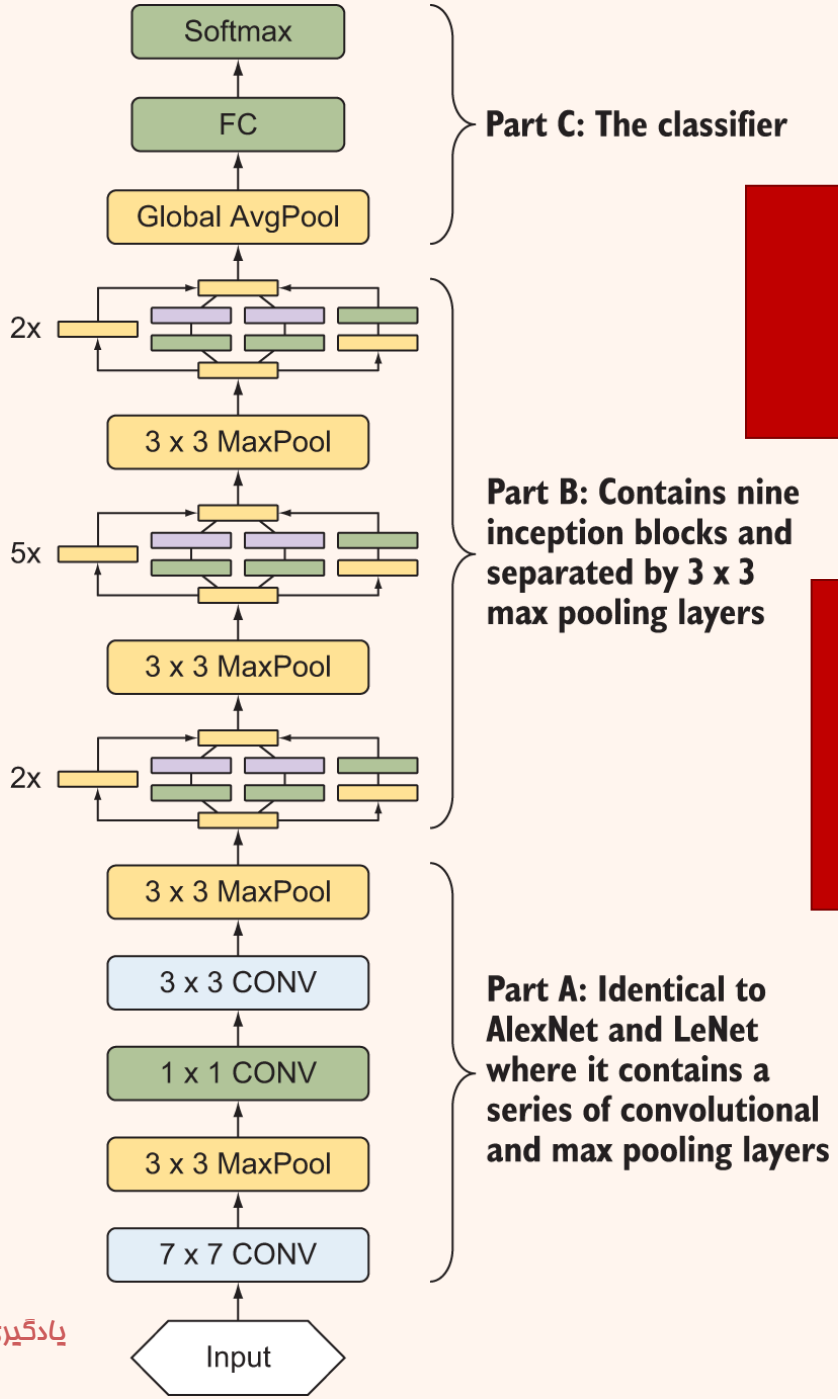
استفاده از دسته‌بندهای میانی در فاز آموزش می تواند در حل مساله‌ی gradient vanishing کمک کننده باشد.

هر کدام شامل موارد زیر است:

5×5 Average Pooling (Stride 3)
1×1 Conv (128 filters)
1024 FC
1000 FC
Softmax

The loss is added to the total loss, with weight 0.3.

هدف استفاده از این ماجول به منظور حل مشکل gradient vanishing و فراهم آوردن regularization بود. اما بعدها دیده شد که این اثر بسیار اندک است.

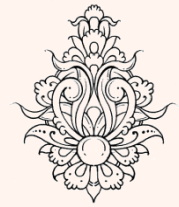


تعداد زیادی ماجول Inception برای عمیق تر شدن این شبکه به کار گرفته شده است.

GoogLeNet در مقایسه با VGG, AlexNet ساختار عمیق تری دارد. اما شبکه‌ای مانند ResNet عمیق تر از GoogLeNet است.

در مجموع GoogLeNet را می توان ۲۲ لایه‌ای در نظر گرفت.

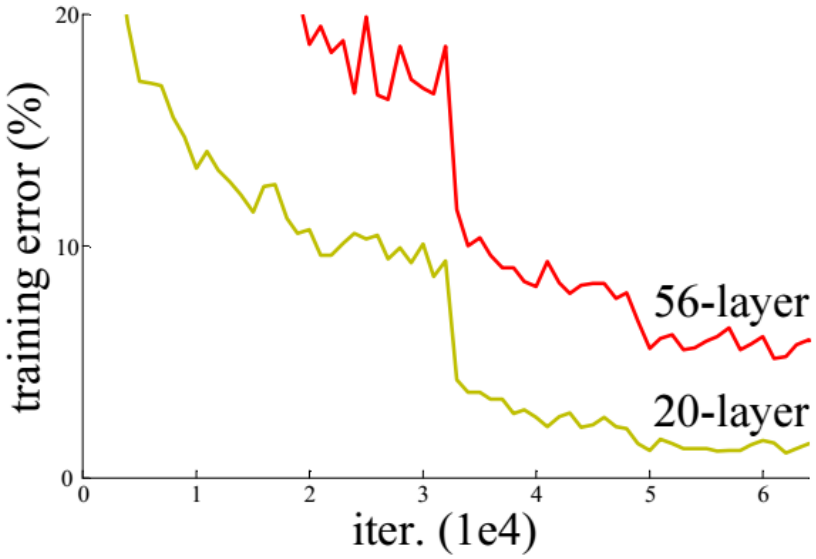
	type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
Part A	convolution	7×7/2	112×112×64	1							2.7K	34M
	max pool	3×3/2	56×56×64	0								
	convolution	3×3/1	56×56×192	2		64	192				112K	360M
	max pool	3×3/2	28×28×192	0								
Part B	inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
	inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
	max pool	3×3/2	14×14×480	0								
	inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
	inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
	inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
	inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
	inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
	max pool	3×3/2	7×7×832	0								
	inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
	inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
Part D	avg pool	7×7/1	1×1×1024	0								
	dropout (40%)		1×1×1024	0								
	linear		1×1×1000	1							1000K	1M
	softmax		1×1×1000	0								



Degradation Problem

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com



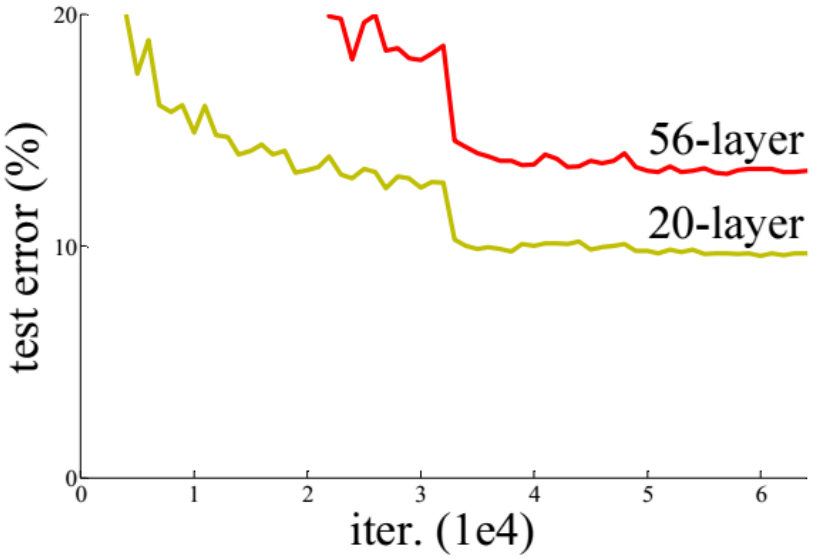


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.

آیا یک مدل کم عمق را به وسیله‌ی یک مدل عمیق می‌توان معادل‌سازی کرد؟

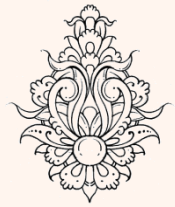
اگر پاسخ این سوال مثبت باشد پس باید بتوان دست کم به همان میزان دقت معماری کم عمق با استفاده از معماری عمیق دست یافت

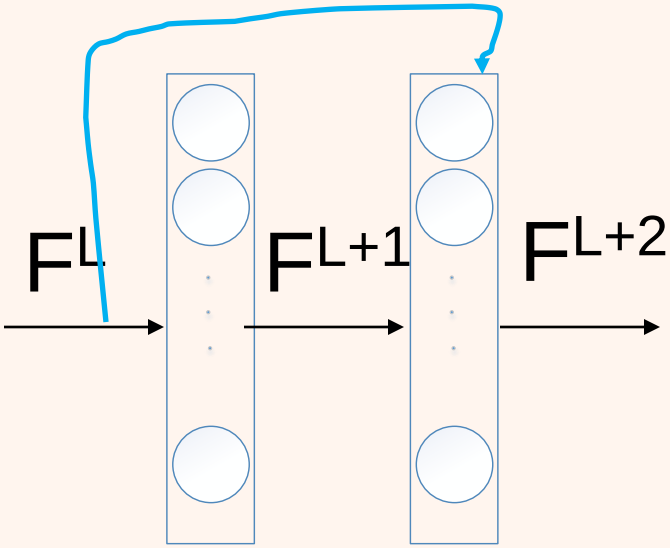
پس نباید با اضافه کردن لایه عملا میزان دقت کاهش یابد.

معماری عمیق = معماری کم عمق + لایه های همانی

این معماری با عمق ۱۵۲ لایه، در سال ۲۰۱۵ مطرح شده است

- با استفاده از Resnet می‌توان از عملکرد شبکه اطمینان داشت به گونه‌ای که عمیق تر شدن شبکه مشکلاتی مانند gradients vanishing یا gradients exploding را به دنبال نخواهد داشت.
- بلوک‌های Residual به شبکه توانایی نگهداری آن‌چیز که آموخته است را می‌دهد. این به صورتی است که اگر به یادگیری بیشتر نیازی نباشد با استفاده از identity mapping weight function خروجی همانند ورودی در نظر گرفته می‌شود، و در غیر این صورت اگر لایه چیزی را آموخته باشد، به آموخته هایش می افزاید.





RELU

$$Z^{L+1} = W^{L+1}F^{L} + b^{L+1}$$

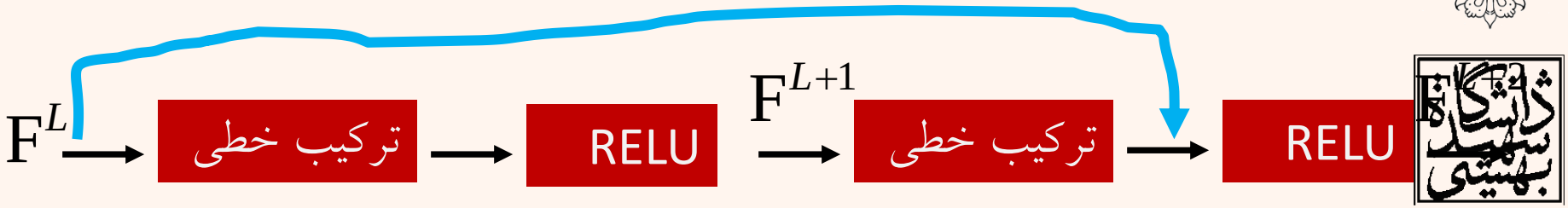
$$F^{L+1} = f(Z^{L+1})$$

$$Z^{L+2} = W^{L+2}F^{L+1} + b^{L+2}$$

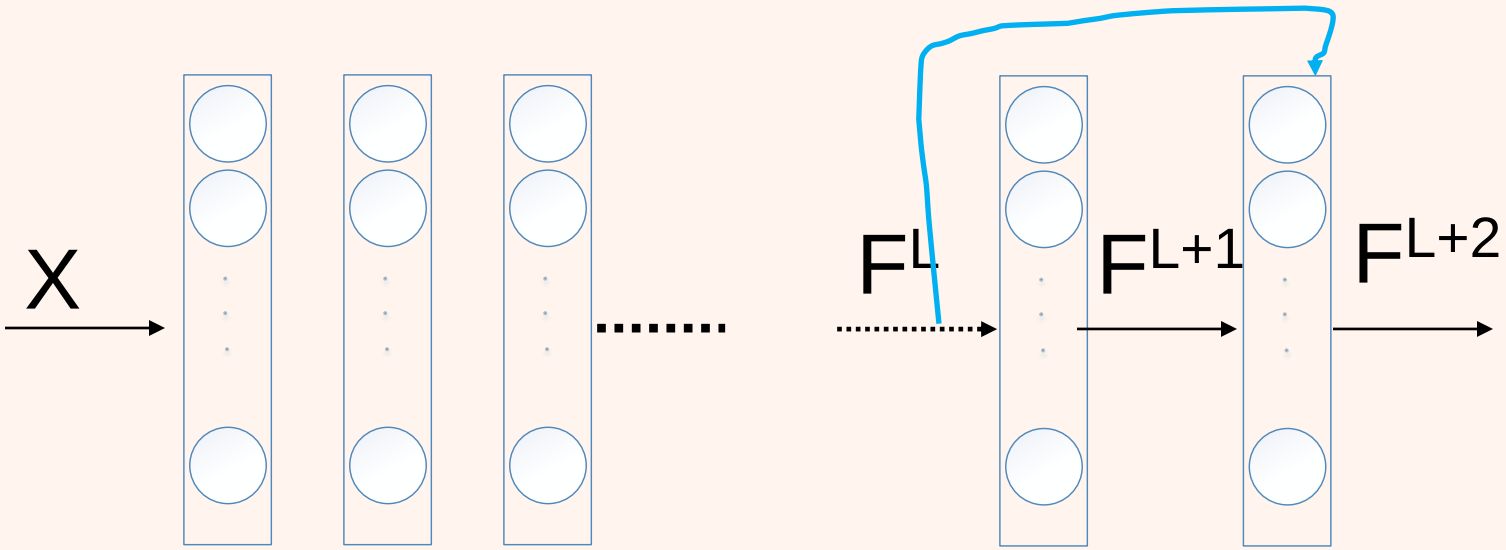
$$F^{L+2} = f(Z^{L+2})$$

Skip Connection

Shortcut



$$F^{L+2} = f(Z^{L+2} + F^{L})$$

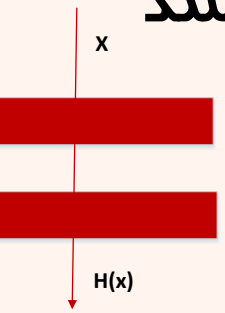


$$F^{L+2} = f(Z^{L+2} + F^{L})$$

$$F^{L+2} = f(W^{L+2}\mathrm{F}^{L+1} + \mathrm{b}^{L+2} + F^{L})$$

$$if \;\; W^{L+2} = 0 \text{ and } \mathrm{b}^{L+2} = 0$$

$$F^{L+2} = f(F^{L}) = F^{L}$$



- اگر بنا باشد لایه‌های مخفی اثر همانی داشته باشند خواهیم داشت:

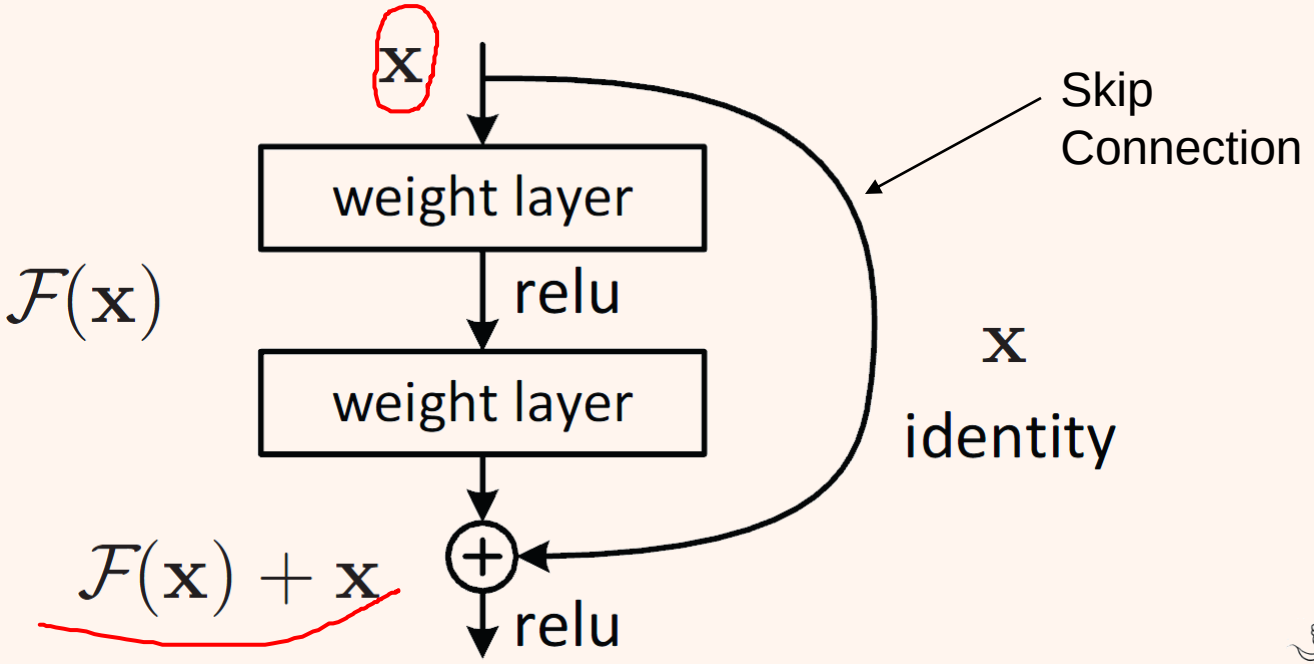
$$H(x) - x = 0$$

- در حقیقت می خواهیم معادله‌ی F(x) برای مواردی که نقش همانی دارد صفر شود یعنی معادله به صورت زیر است:

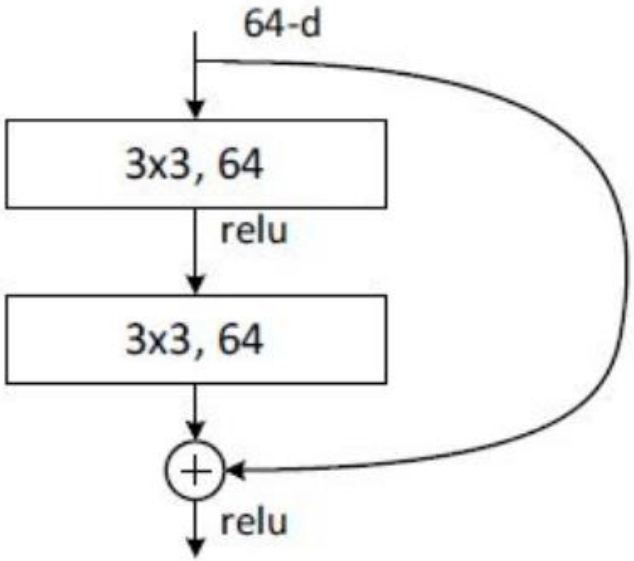
$$\mathrm{F}(x) = H(x) - x$$

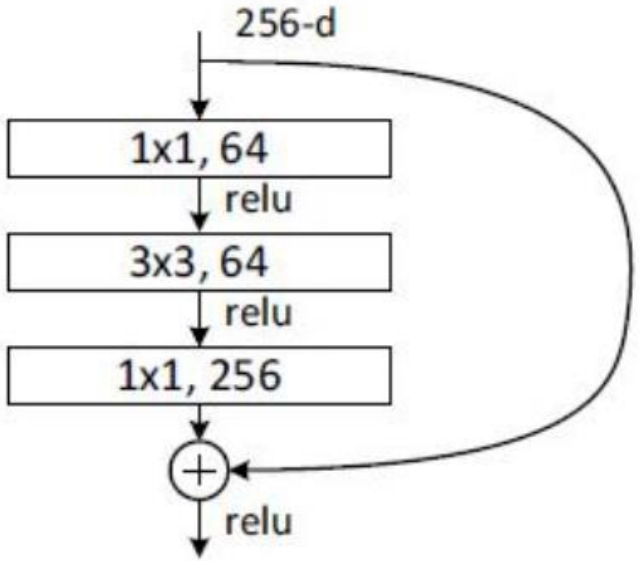
- حدس زدن H(x) دشوار است و دلیل آن عملکرد لایه‌های غیرخطی است.
- عملا به جای حدس زدن H(x)، رابطه‌ی زیر حدس زده می‌شود:

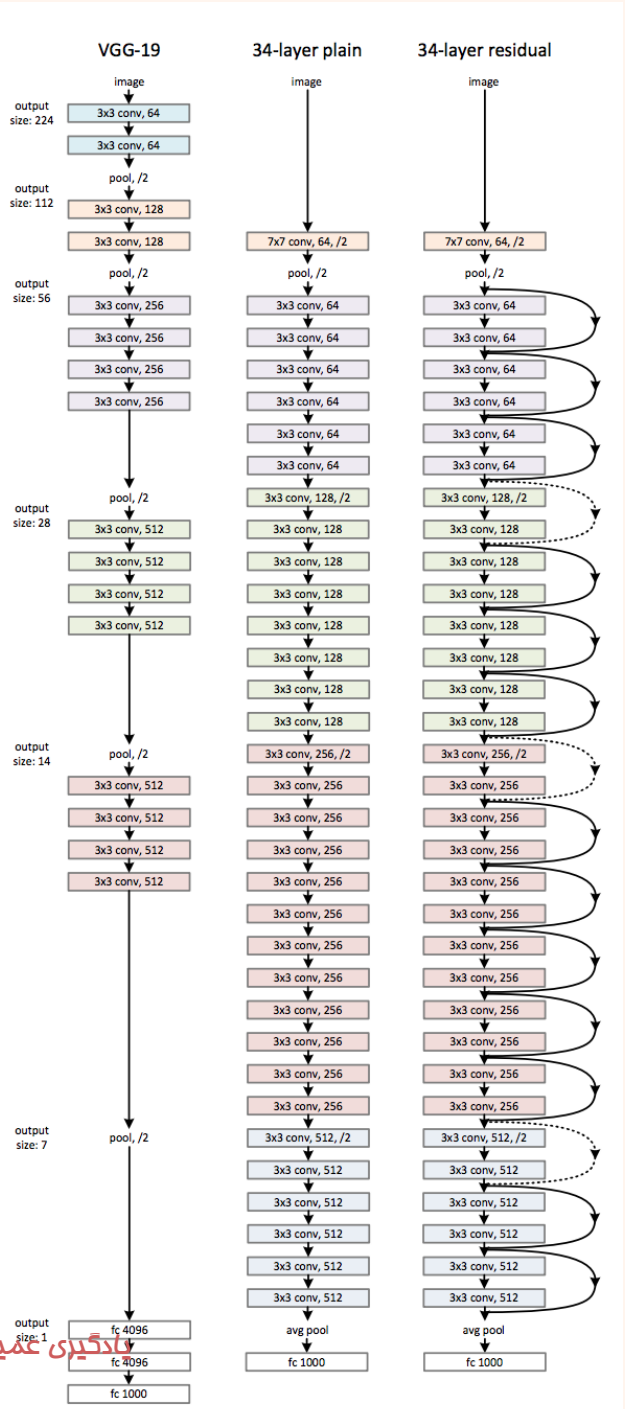
$$H(x) = F(x) + x$$


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
Skip
Connection
x
identity

$$H(x) = F(x) + x$$

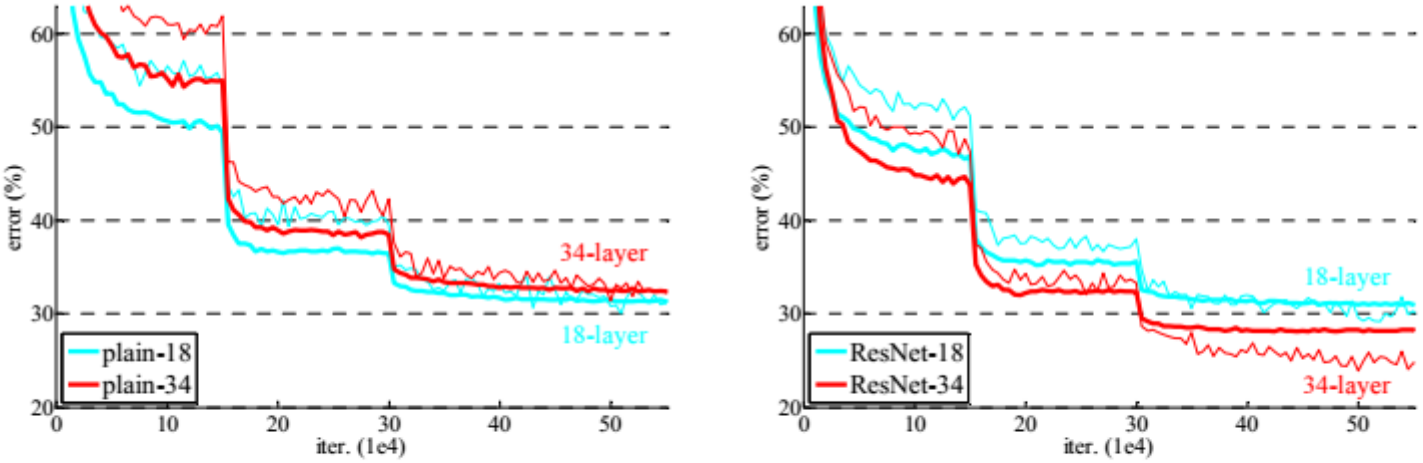

64-d
3x3, 64
relu
3x3, 64
relu


256-d
1x1, 64
relu
3x3, 64
relu
1x1, 256
relu



- قابلیت Resnet در شبکه‌هایی که تعداد لایه‌های بالایی دارد پدیدار می‌شود.
- مثلا برای شبکه‌ای با تعداد لایه‌ی ۳۴ عملکرد موفق این شبکه مشخص است.
- این در حالی است که برای شبکه‌ی ۱۸ لایه‌ای این مساله تاثیر چندانی ندارد.

نتایج



Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

. Top-1 error (%, 10-crop testing) on ImageNet validation

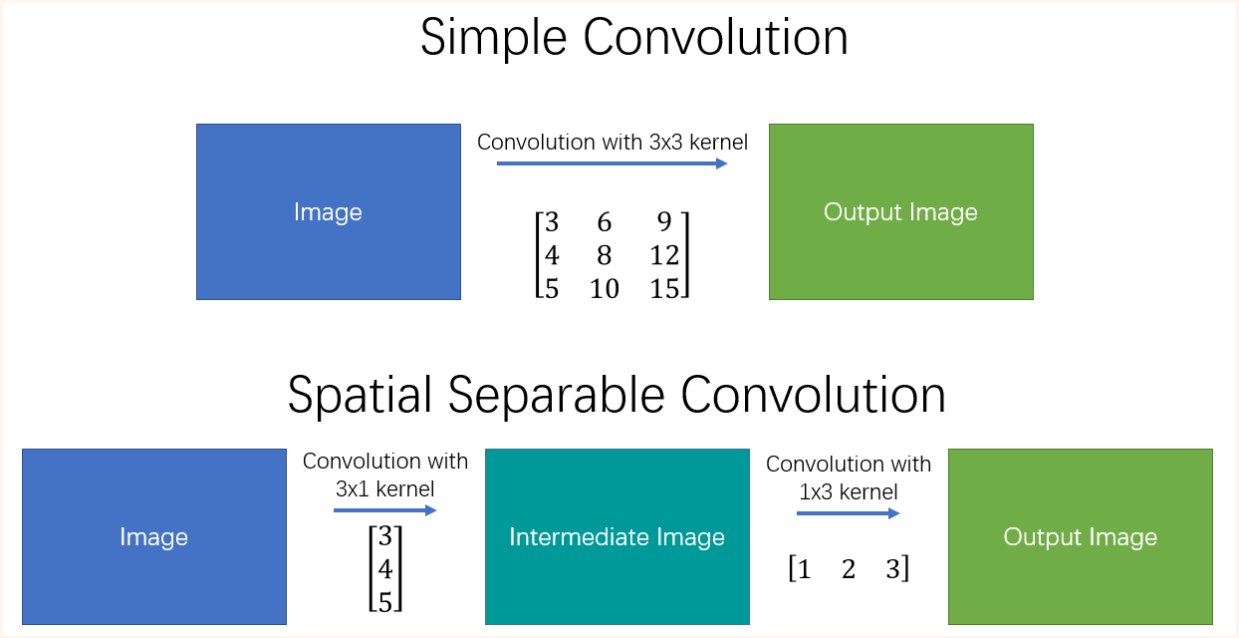
	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Year	CNN	Number of layers	Top-5 error rate	Number of parameters
1998	LeNet	5 layers	NA	60 thousand
2012	AlexNet	8 layers	15.3%	60 million
2014	VGGNet	16 layers	7.3%	138 million
2014	Inception (GoogLeNet)	22	6.67%	12 million
2015	ResNet-152	152	4.49%	

https://livebook.manning.com/book/grokking-deep-learning-for-computer-vision/chapter-5/v-3/11

Separable Convolutions

$$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

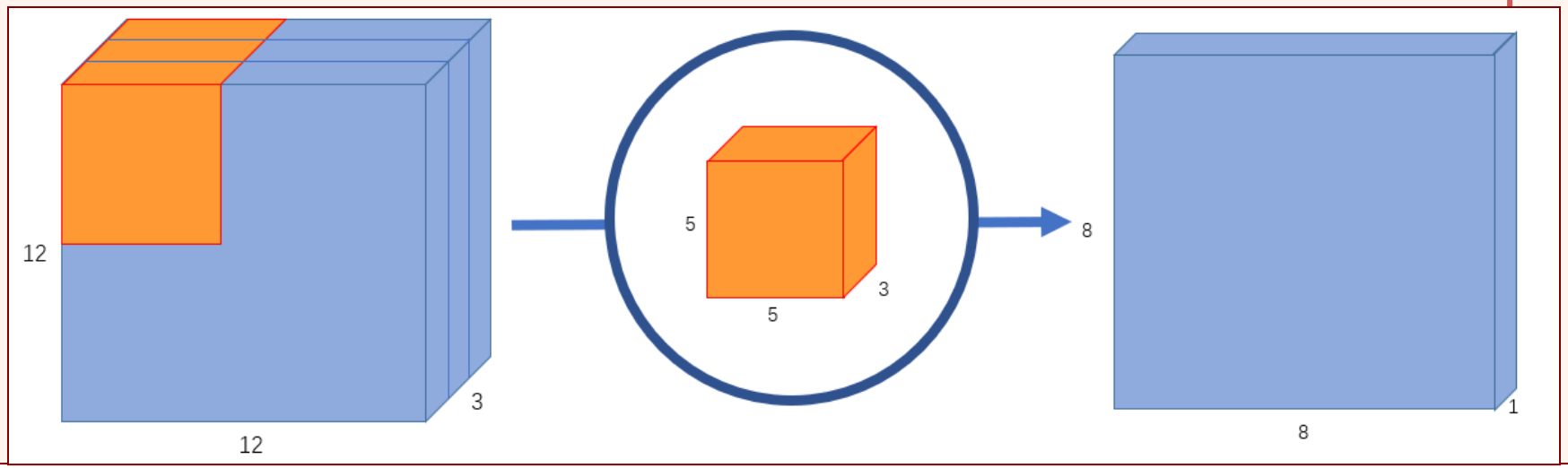


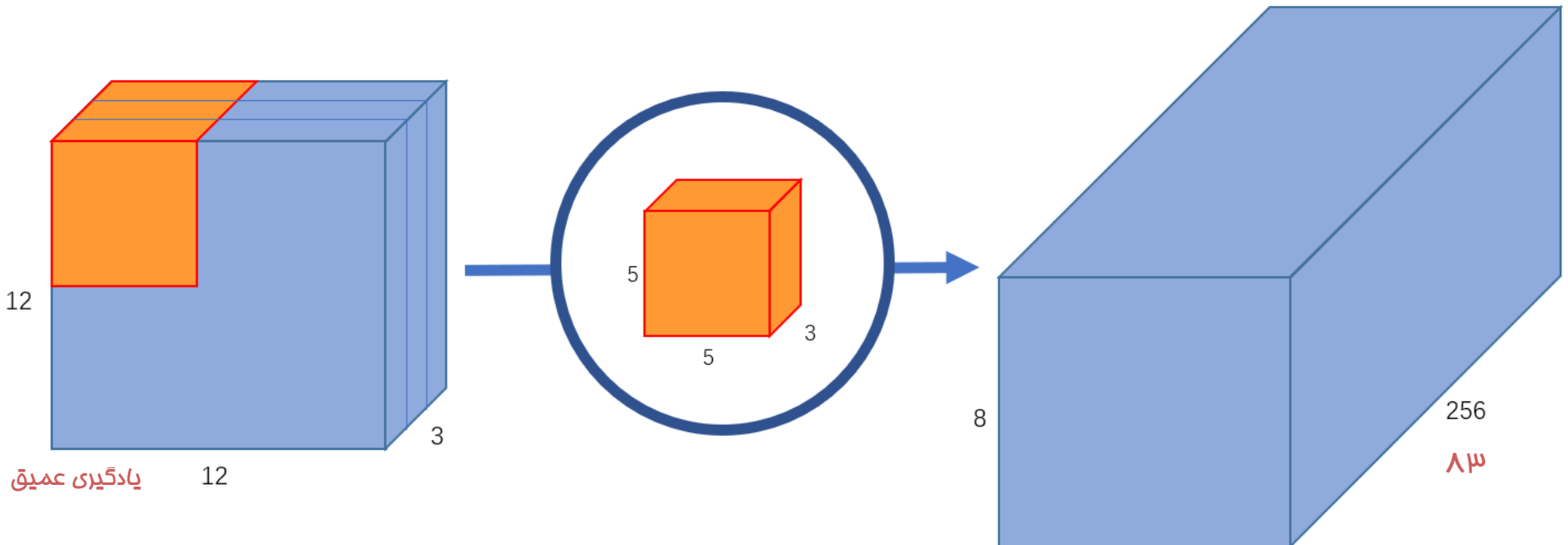
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

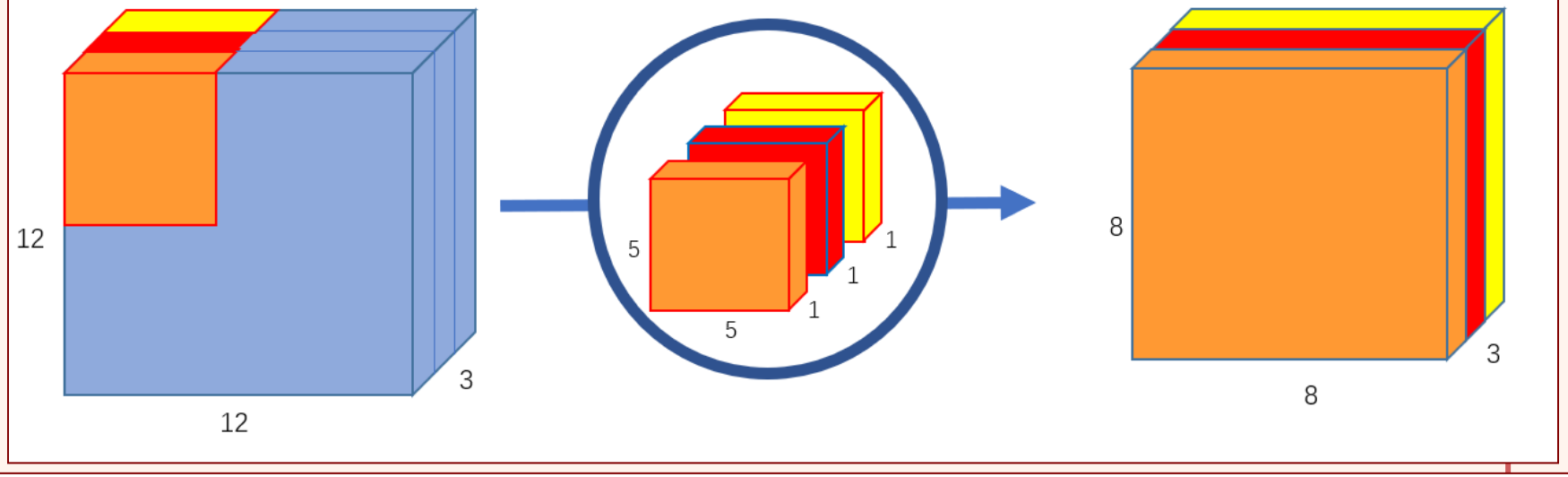
https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

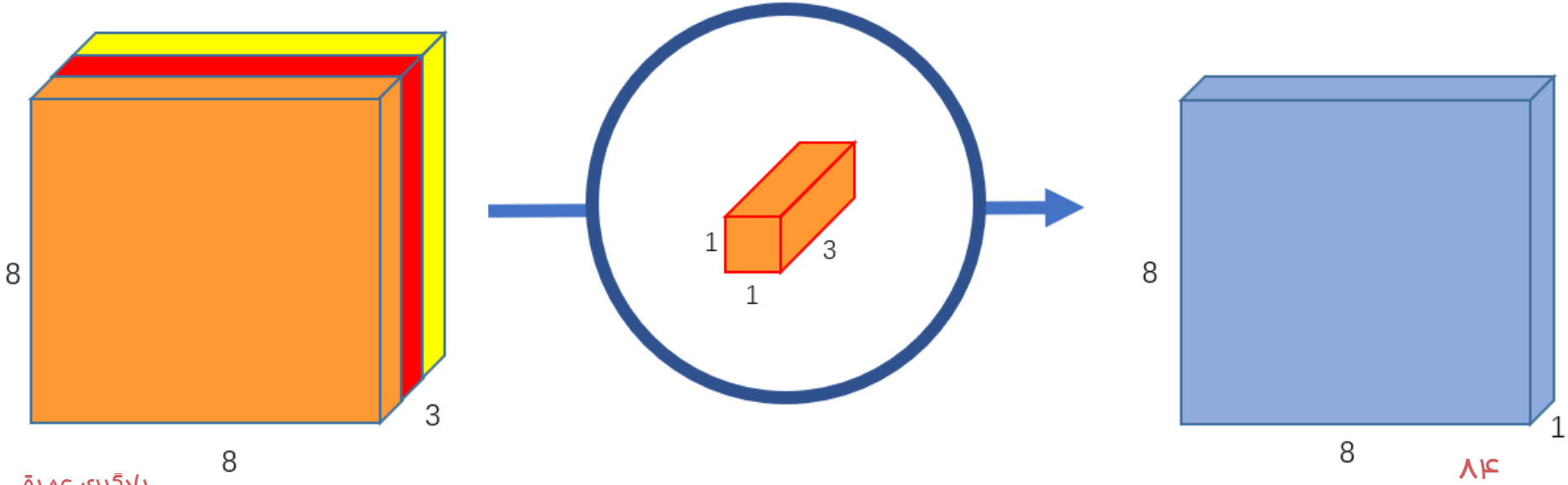
Xception

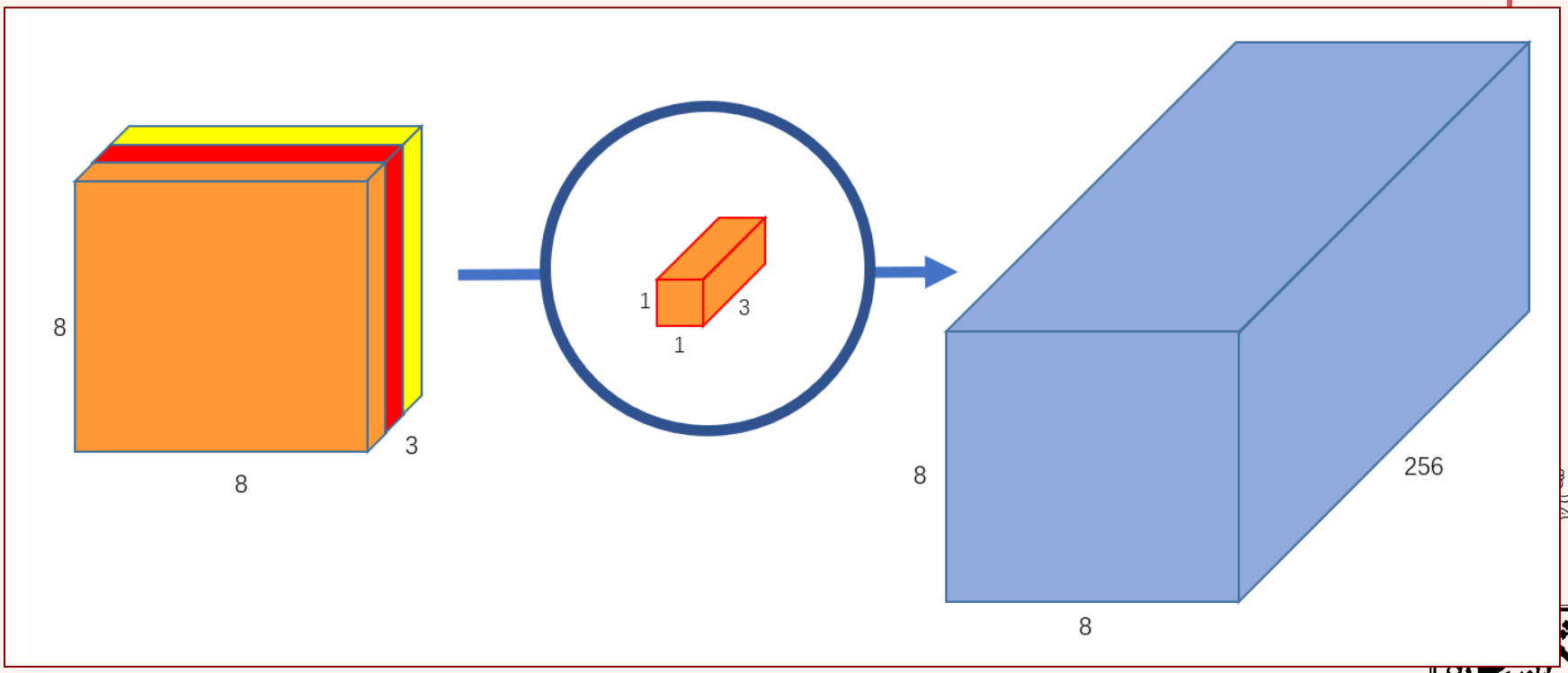
Xception: Deep Learning with Depthwise Separable Convolutions



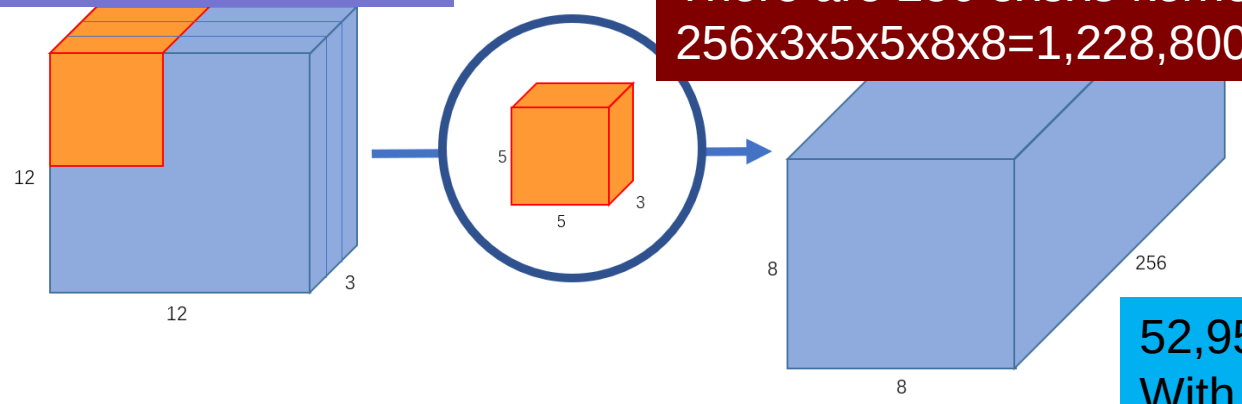



12
12
3
5
5
1
1
1
8
8
3


8
8
3
1
1
3
8
8
1


8
8
3
1
1
3
8
8
256

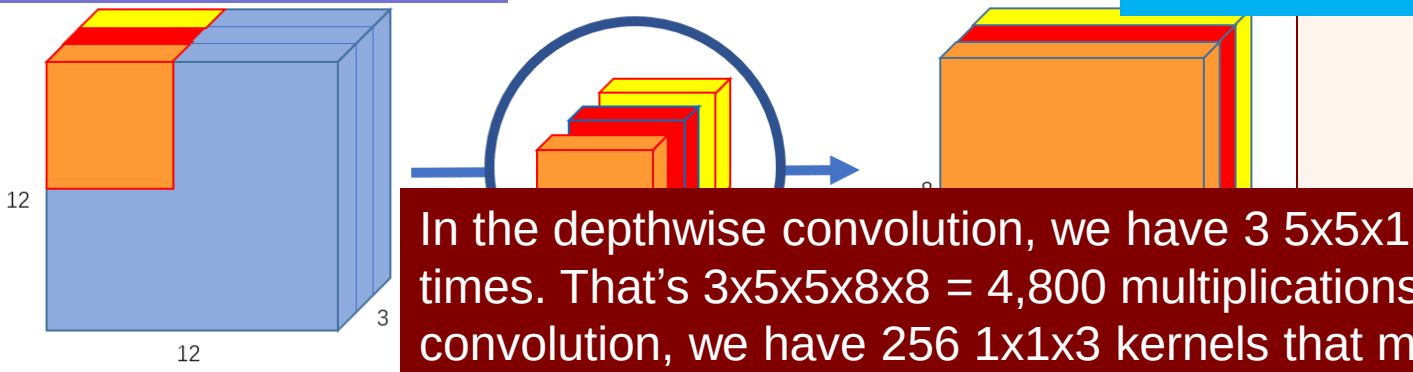
original convolution



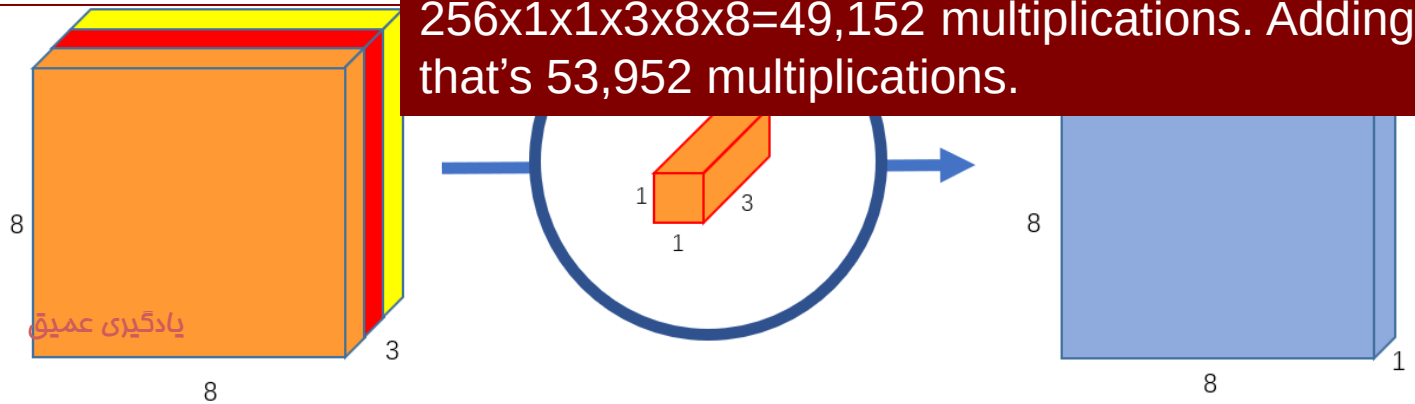
There are 256 5x5x3 kernels that move 8x8 times. That's 256x3x5x5x8x8=1,228,800 multiplications.

52,952 is a lot less than 1,228,800. With less computations, the network is able to process more in a shorter amount of time.

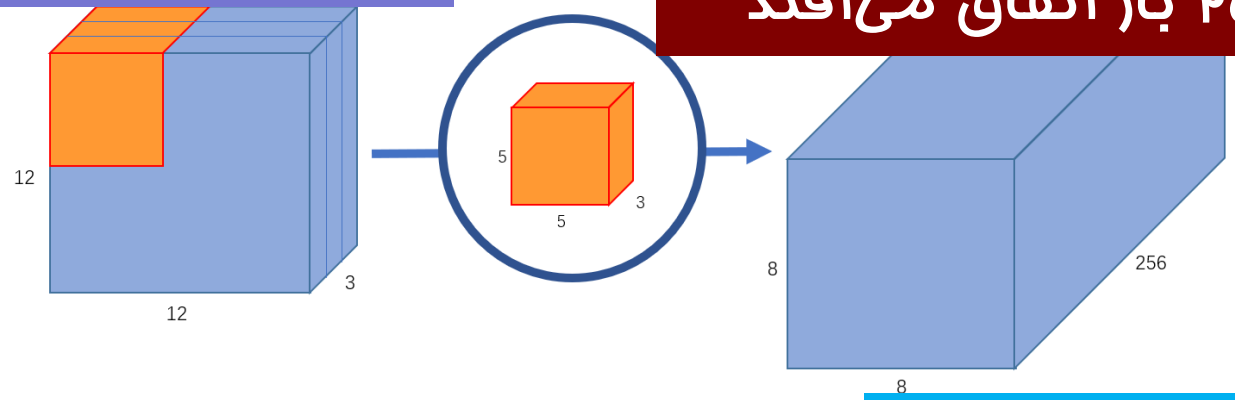
separable convolution



In the depthwise convolution, we have 3 5x5x1 kernels that move 8x8 times. That's 3x5x5x8x8 = 4,800 multiplications. In the pointwise convolution, we have 256 1x1x3 kernels that move 8x8 times. That's 256x1x1x3x8x8=49,152 multiplications. Adding them up together, that's 53,952 multiplications.



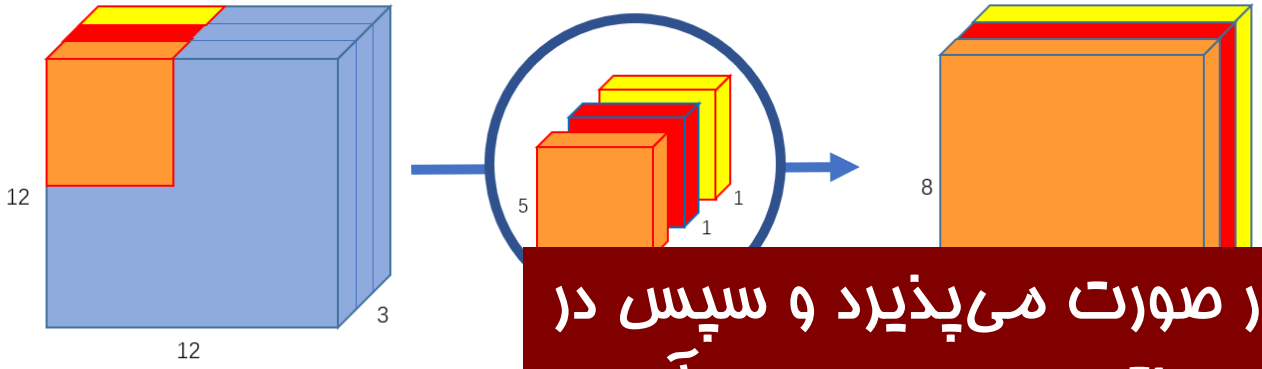
original convolution



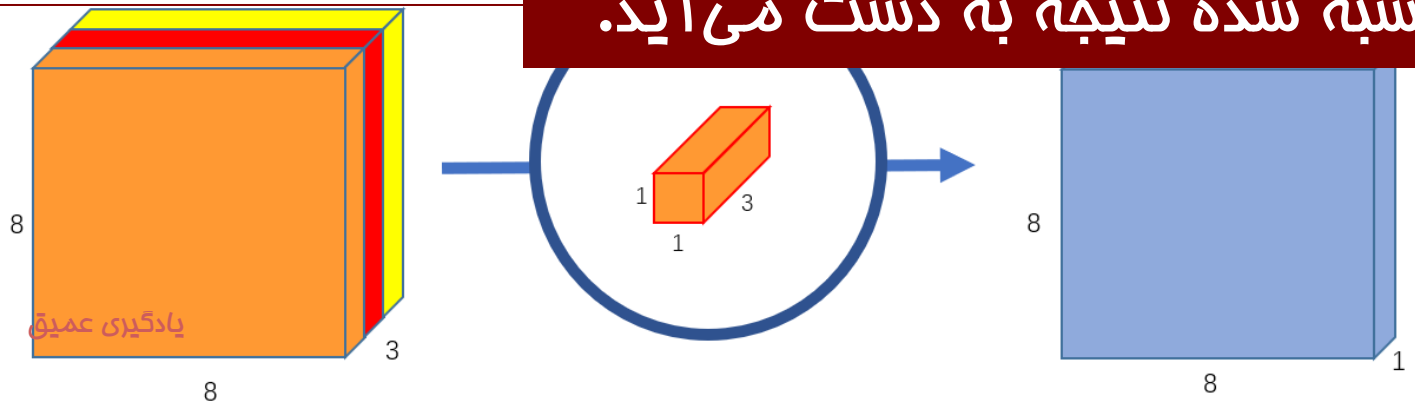
تبدیل به روی تصویر ۲۵۶ بار اتفاق می‌افتد

Prevent to transform the image over and over again, we can save up on computational power.

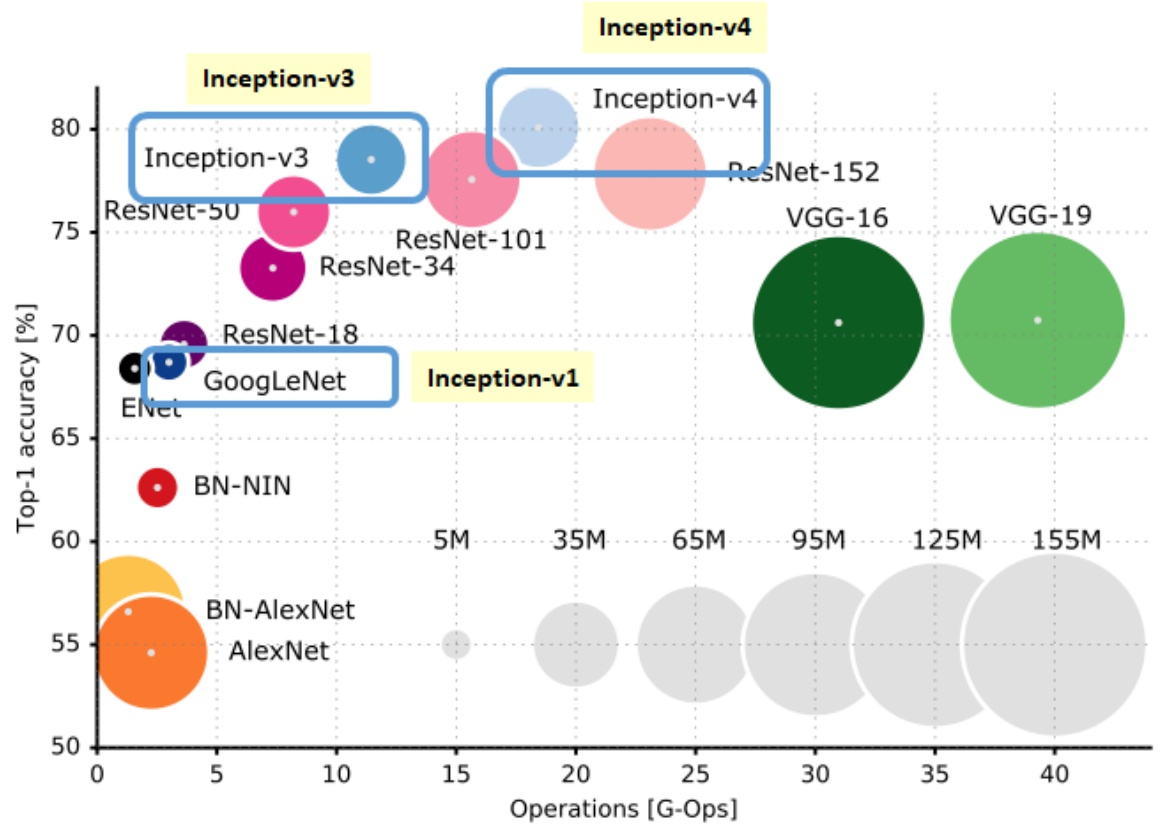
separable convolution



تبدیل به روی تصویر یک بار صورت می‌پذیرد و سپس در عمق ۲۵۶ بار محاسبه شده نتیجه به دست می‌آید.



یادگیری عمیق



https://towardsdatascience.com/review-inception-v4-evolved-from-googlenet-merged-with-resnet-idea-image-classification-5e8c339d18be