Classical networks problems

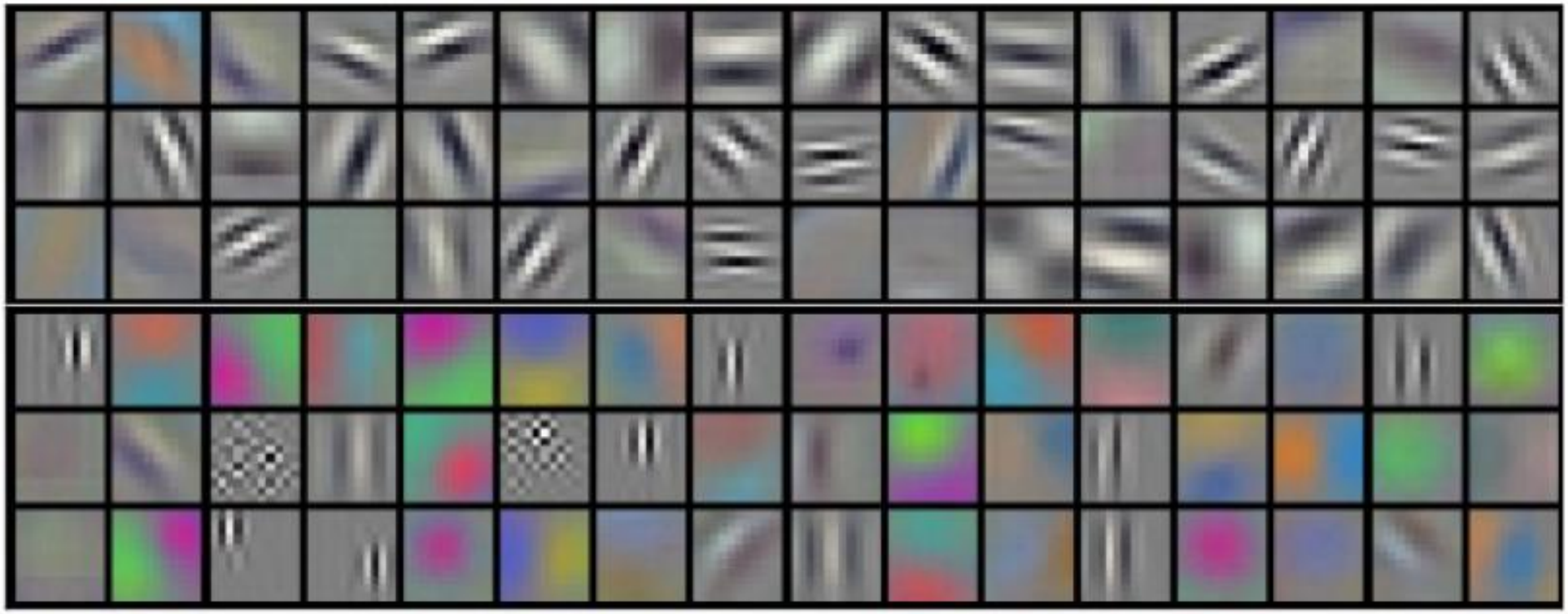
In many cases during the training of classical neural networks we observe phenomenon of vanishing gradient.

One can also wonder about the sensibility of taking into account the whole image (perhaps large) to determine the value of each neuron in each layer. If you interpret the first layers as layers for extracting features, it seems natural to take advantage of the local characteristics of the image.

Such an approach will enable the first layer to detect a set of simple image features (edges, colour blobs), from which in the subsequent layers could be composed more complex features describing the image.

Many of these features can be designed by hand, but how much more pleasant it would be to let the network to learn, which will extract them automatically and match the specific problem.

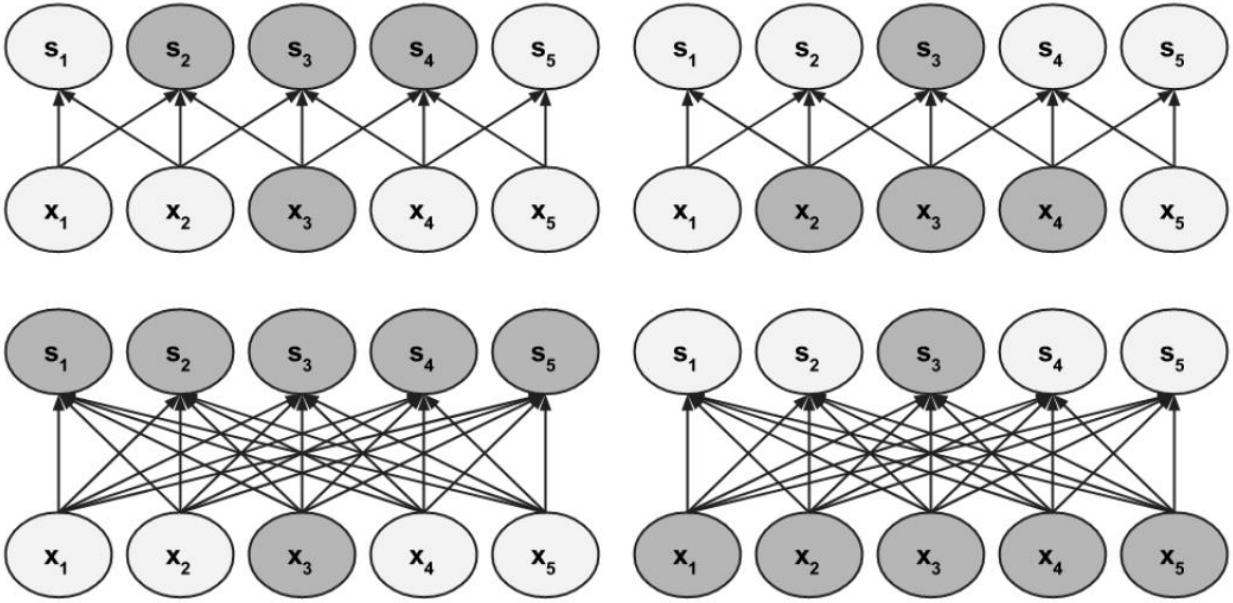
For example…



96 11x11x3 filters in the first AlexNet layer trained on ImageNet dataset.

Features and analysis locality

In convolutional networks filter has smaller size than input:



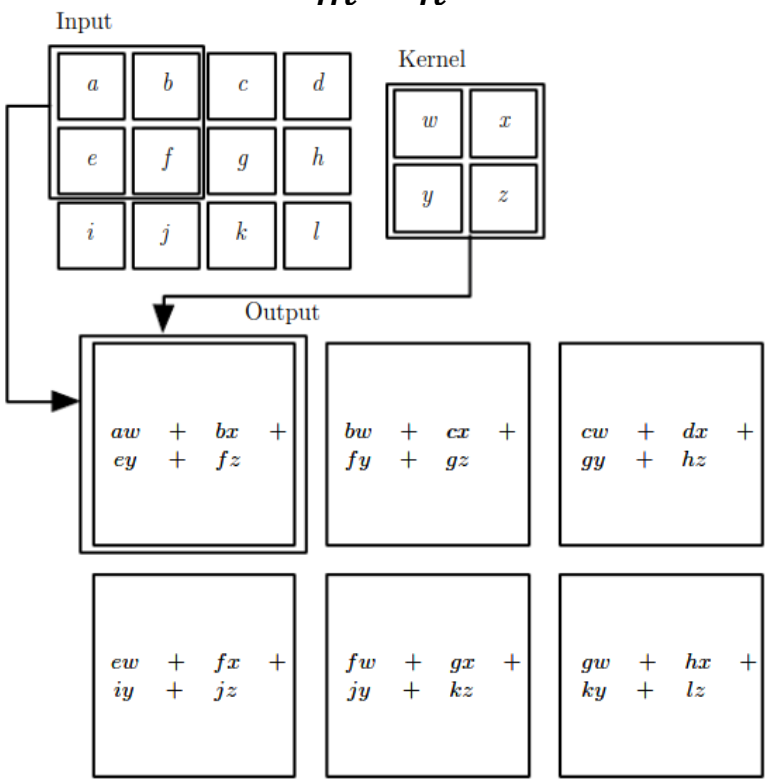
Comparing to traditional fully connected networks we can see three main characteristics:

- Locality of connections
- Parameter sharing
- Translation equivariance

Convolution

Convolution is a ***linear*** operation which computes feature map basing on input image and filter (kernel).

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$



Convolution parameters

We have a few paramters controlling convolution and size of the output feature map:

- $\boldsymbol{f}$: – *filter (kernel) size*
  You have to remember that filter covers depth of the map too (for example 11x11x3 – filter size in the first layer of AlexNet)
- $\boldsymbol{s}$: – *stride* (step length)
- $\boldsymbol{p}$: – *padding* (usually $p = \lfloor f/2 \rfloor$)
  Without extending input width and height (*valid padding*) – output of the convolution has smaller width and height ($p = 0$).
  With extending (*same padding*) – filter operates also near edges of the image and we have to decide which values to assign to „missing” pixels ($p \geq 1$).

$$output_{size} = \left\lfloor \frac{n - f \ + 2p}{s} + 1 \right\rfloor$$

Padding

Zero padding	Replicated padding

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	4	2	3	0	0
0	0	6	1	1	7	0	0
0	0	1	2	1	4	0	0
0	0	1	5	1	2	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

1	1	1	4	2	3	3	3
1	1	1	4	2	3	3	3
1	1	1	4	2	3	3	3
6	6	6	1	1	7	7	7
1	1	1	2	1	4	4	4
1	1	1	5	1	2	2	2
1	1	1	5	1	2	2	2
1	1	1	5	1	2	2	2

Padding

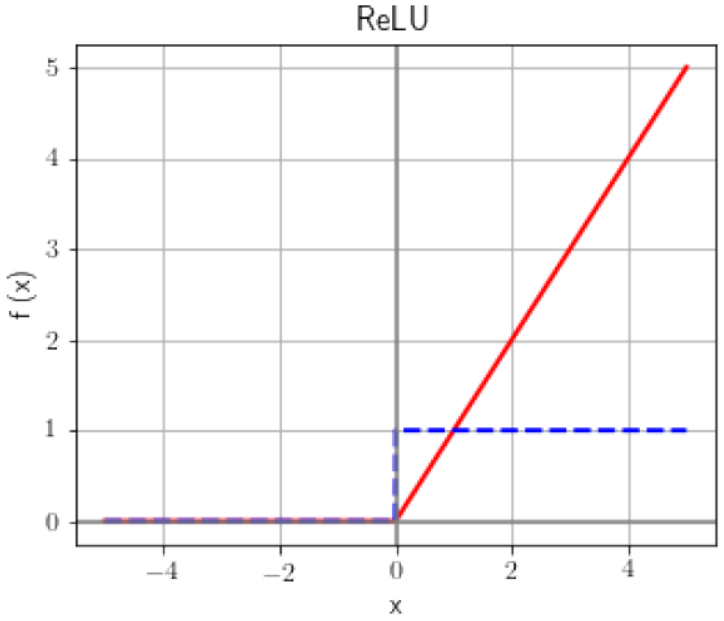
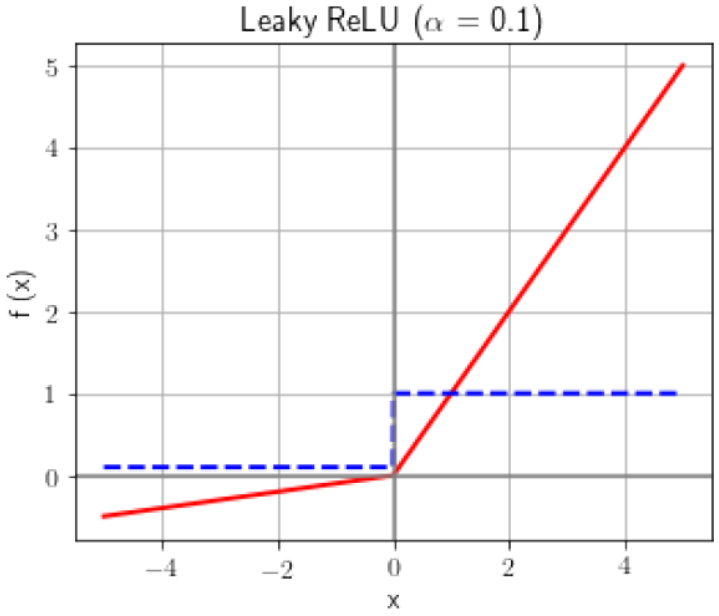
Symmetric padding	Circulated padding

1	6	6	1	1	7	7	1
4	1	1	4	2	3	3	2
4	1	1	4	2	3	3	2
1	6	6	1	1	7	7	1
2	1	1	2	1	4	4	1
5	1	1	5	1	2	2	1
5	1	1	5	1	2	2	1
2	1	1	2	1	4	4	1

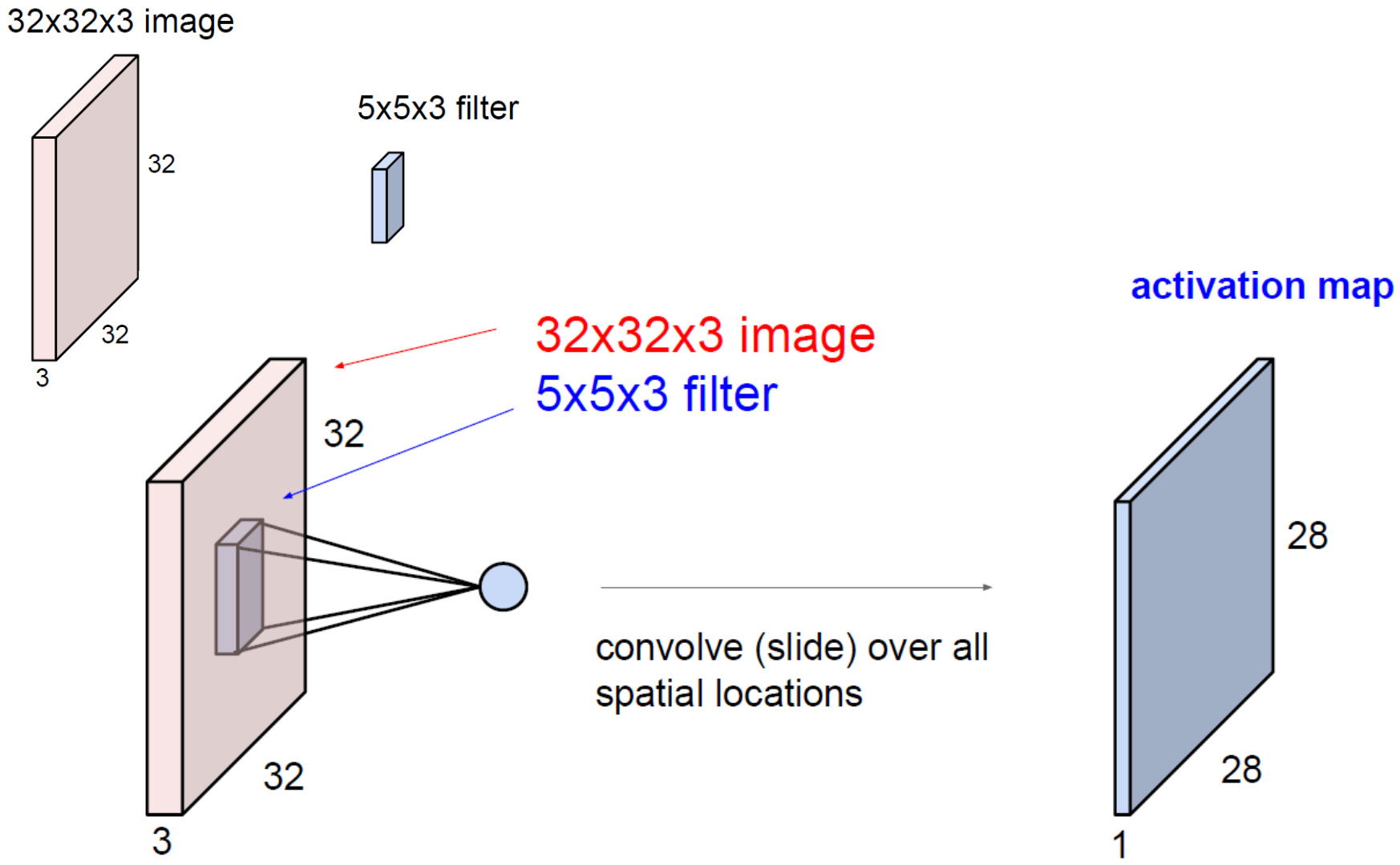
1	4	1	2	1	4	1	2
1	2	1	5	1	2	1	5
2	3	1	4	2	3	1	4
1	7	6	1	1	7	6	1
1	4	1	2	1	4	1	2
1	2	1	5	1	2	1	5
2	3	1	4	2	3	1	4
1	7	6	1	1	7	6	1

Nonlinearity

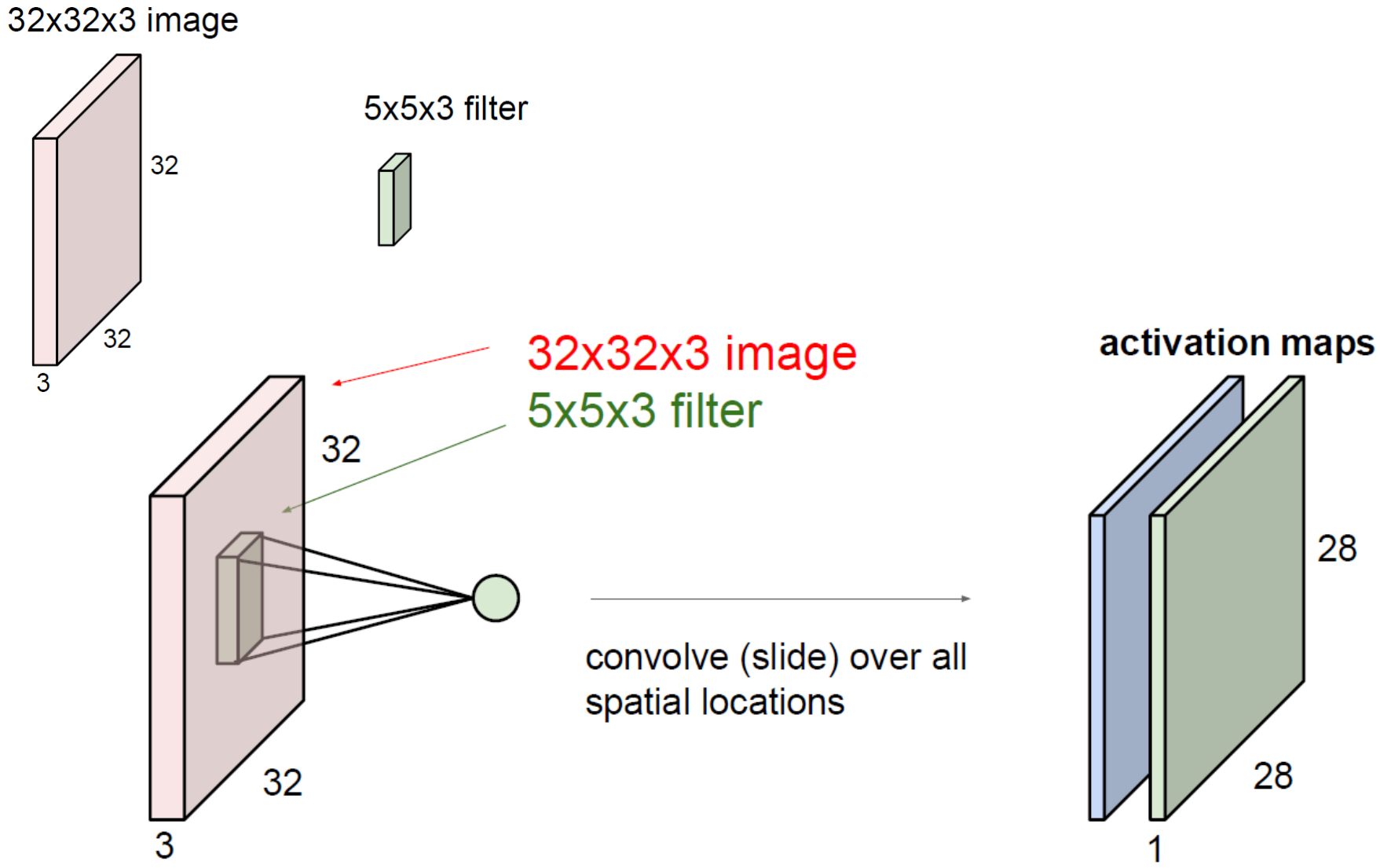
Convolution is linear operation on the input data. Introducing nonlinearity is necessary to extract more interesting features than just linear combination of input values. In convolutional networks we typically use:

ReLU Rectified Linear Unit $f(x) = \max(0, x)$	Leaky ReLU Leaky Rectified Linear Unit $f(x) = \max(\alpha x, x)$
	

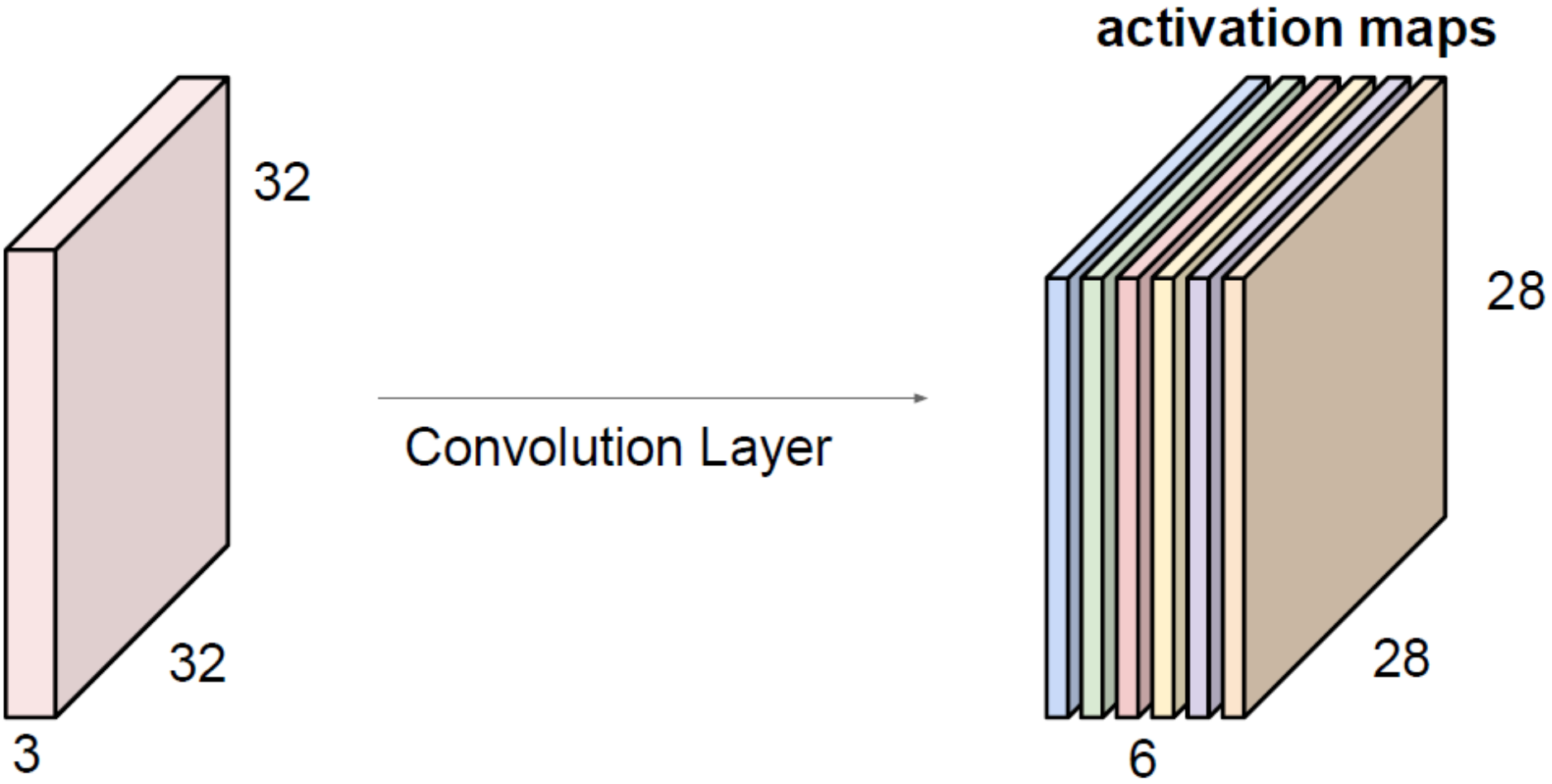
Convolutional layer (RGB input)



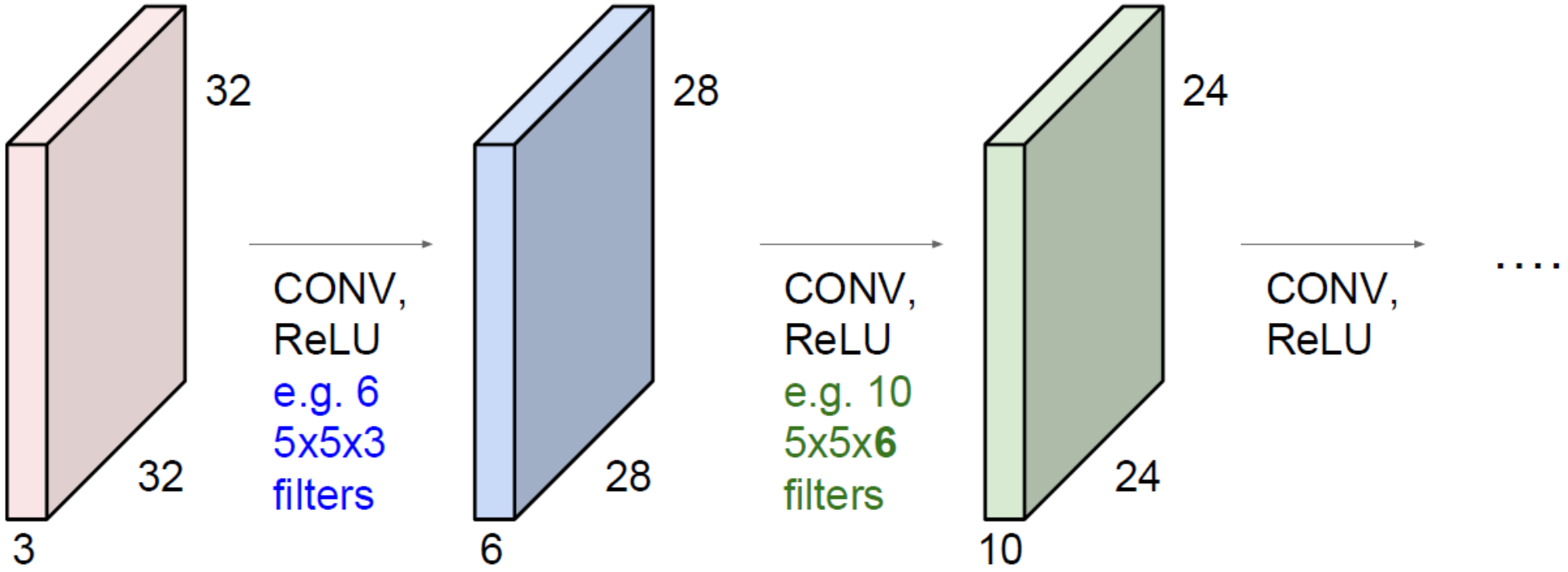
Convolutional layer (second filter)



Convolutional layer (six filters)

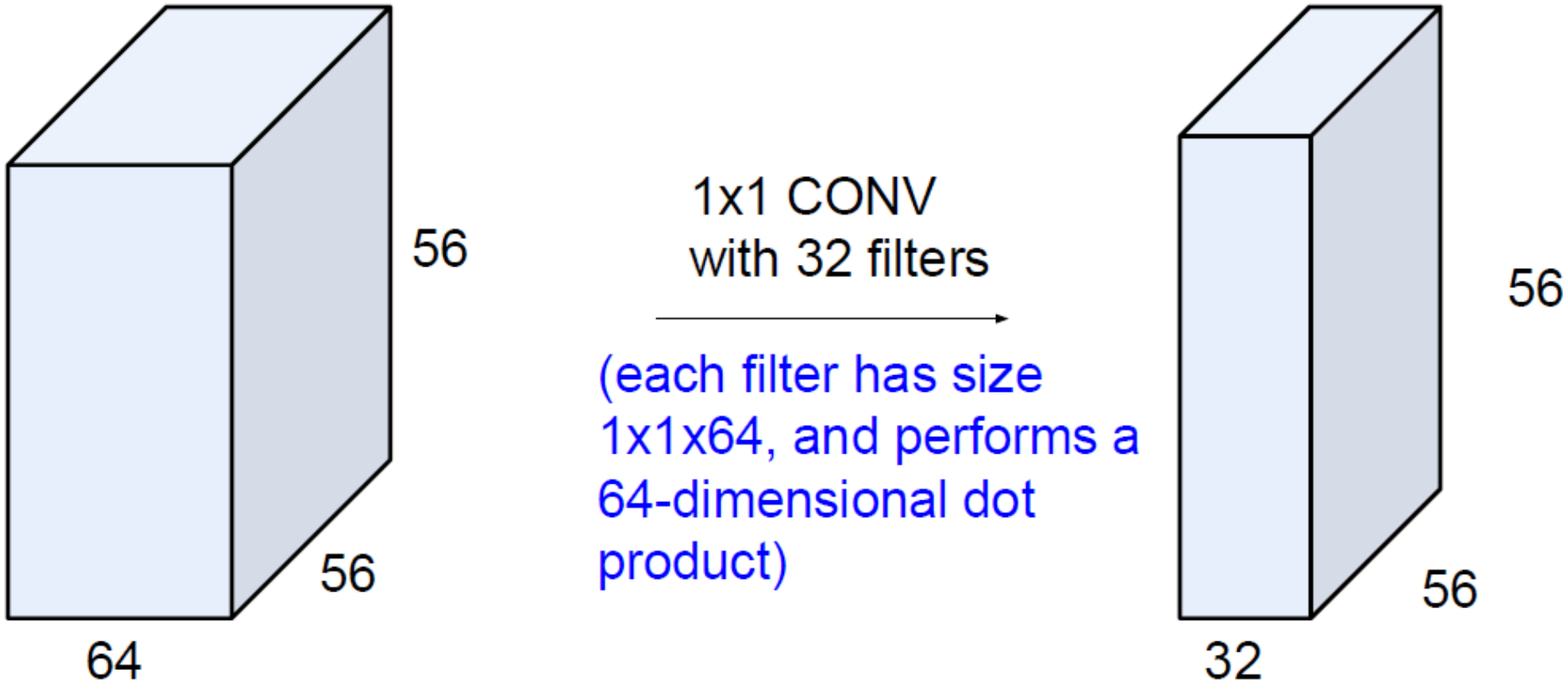


Several convolutional layers



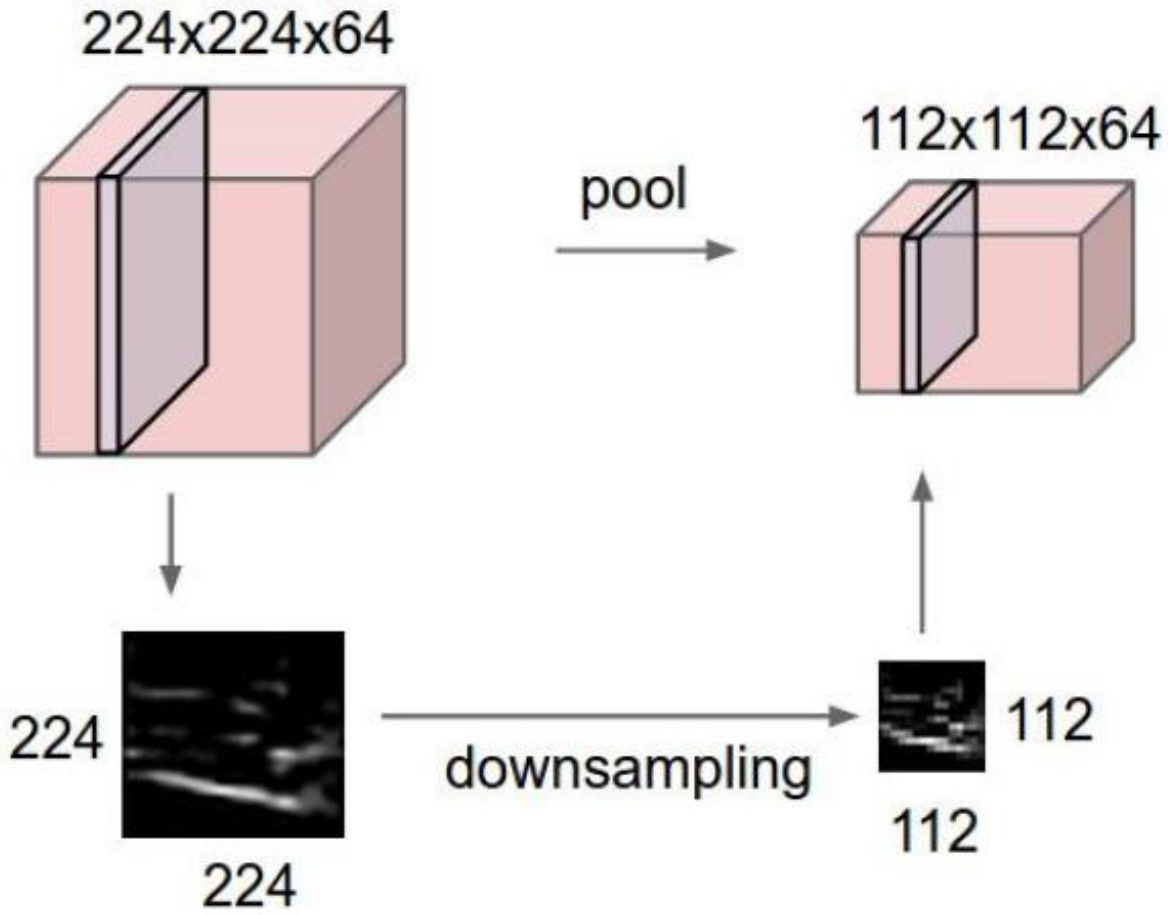
Without padding width and height of feature maps decreases; filter number increases map's depth which in consequence increases size of filters.

Map reductions



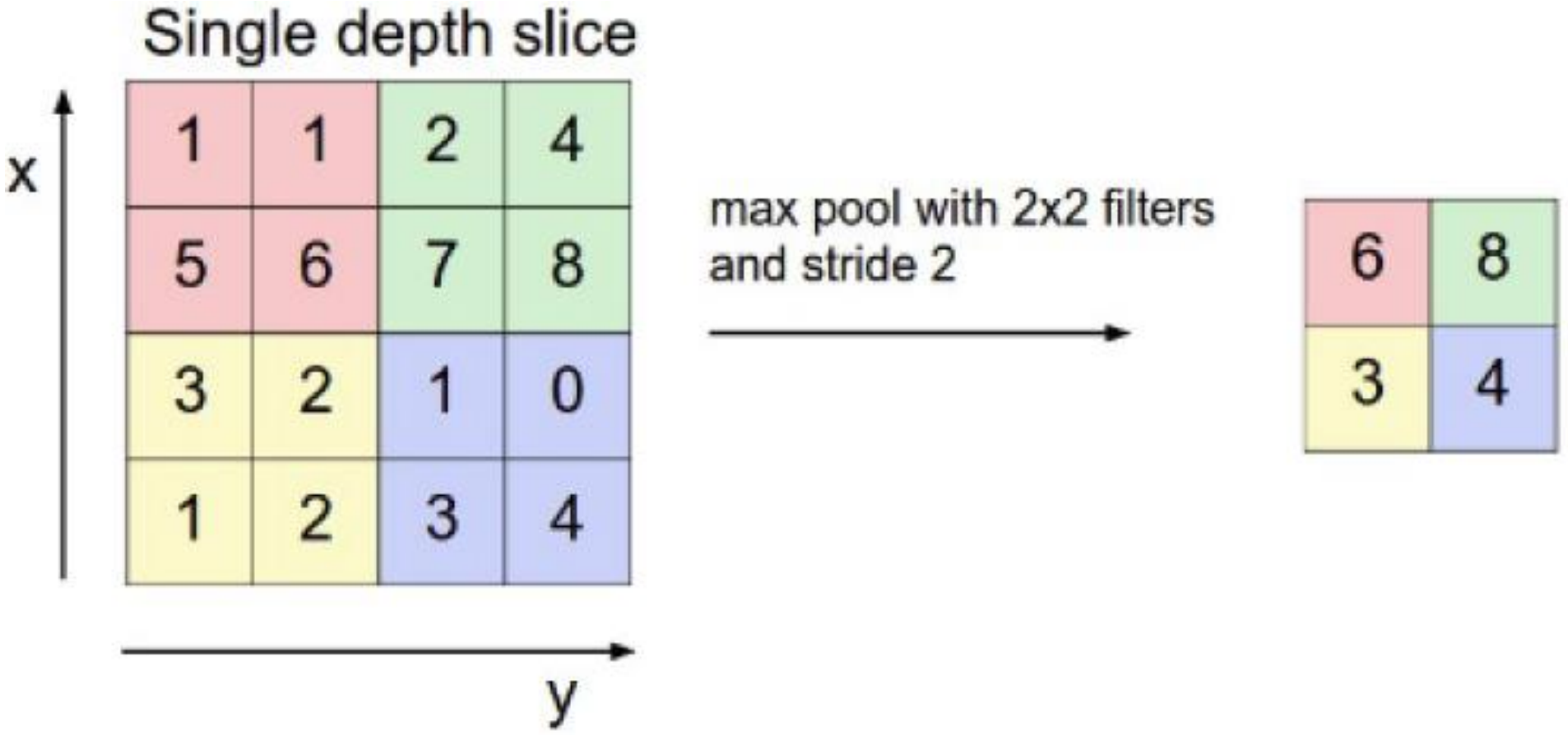
1x1 filtr is perfectly ok– it allows to reduce map's depth.

Reducing width and height – pooling



Pooling operates on each activation independetnly. We use either mean or max pooling.

Max pooling


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Normalization

There are two main approaches to normalization:

Local response normalization – generally the aim is to inhibit neighbours of the neuron with high output value (rarely being used in practice since its contribution is shown to be minimal);

Batch normalization – first *whitening* of the samples in batch is performed and the result is scaled and biased with some trained parameters.

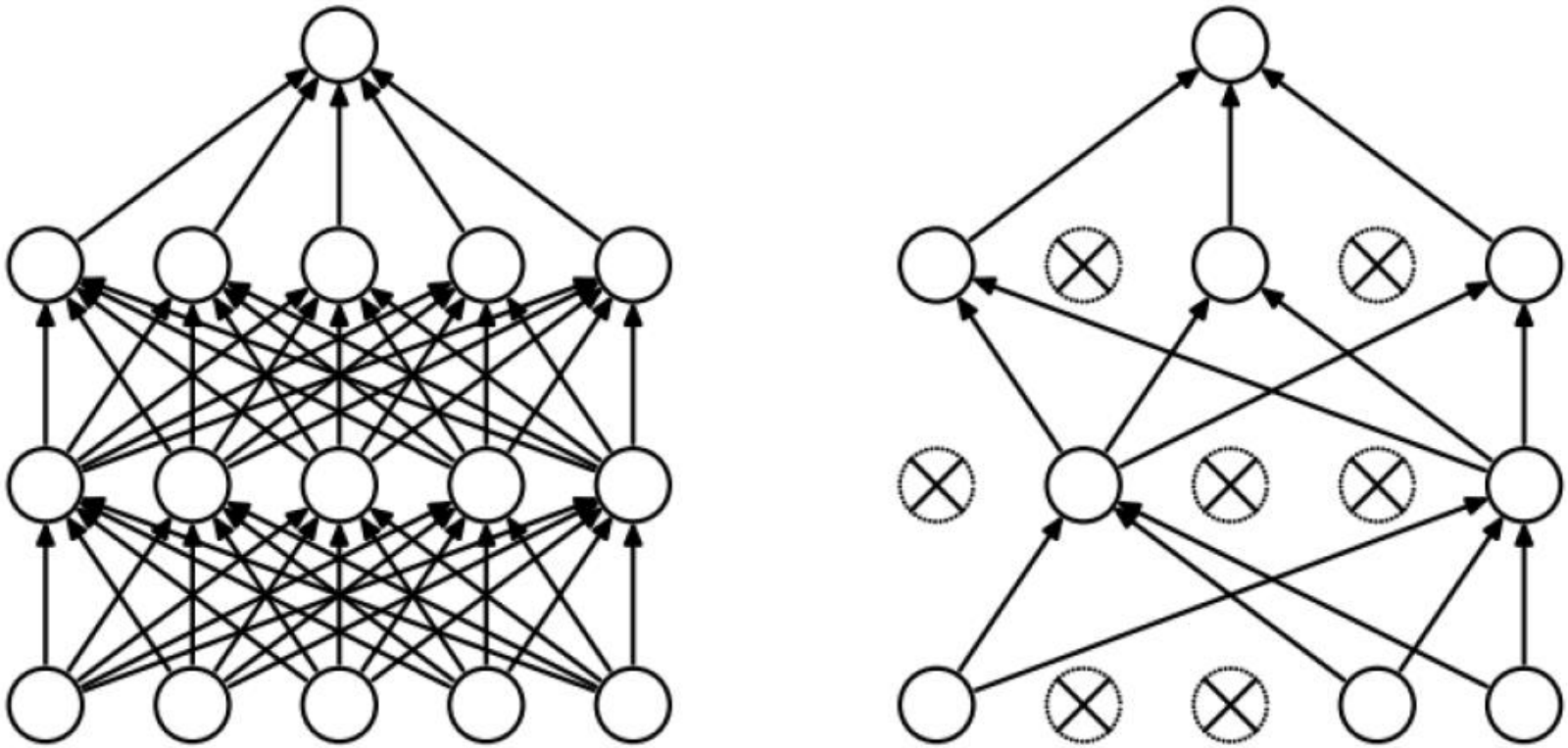
Regularization

L1 regularization defines the regularization penalty as $\lambda|w|$. It tends to drive the weights sparse: some weights are very close to zero and some have large values. In other words, L1 regularization ends up using a sparse subset of the inputs and become invariant to noisy inputs.

L2 regularization defines the regularization penalty as λw^2. It is common to see a constant 1/2 making the penalty equal to $\lambda/2w^2$ to make the gradient simply λw instead of $2\lambda w$. Since L2 regularization gives high penalty for large weight values, the weights tend to be diffused with smaller values. In practice, if explicit feature selection is not concerned, L2 regularization is superior to L1.

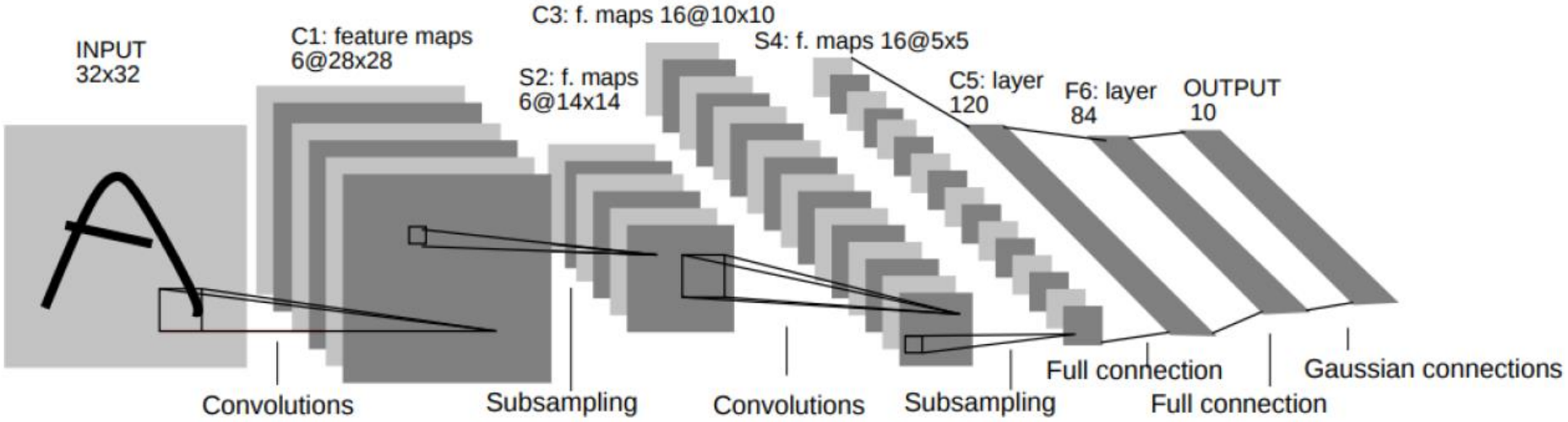
Elastic net regularization is a combination between L1 and L2 regularization with penalty equals to $\lambda|w|+ \lambda w^2$

Drop-out



LeNet-5

Y. Lecun, L. Bottou, Y. Bengio, and P. Haner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, vol. 86, pp. 2278-2324, Nov 1998

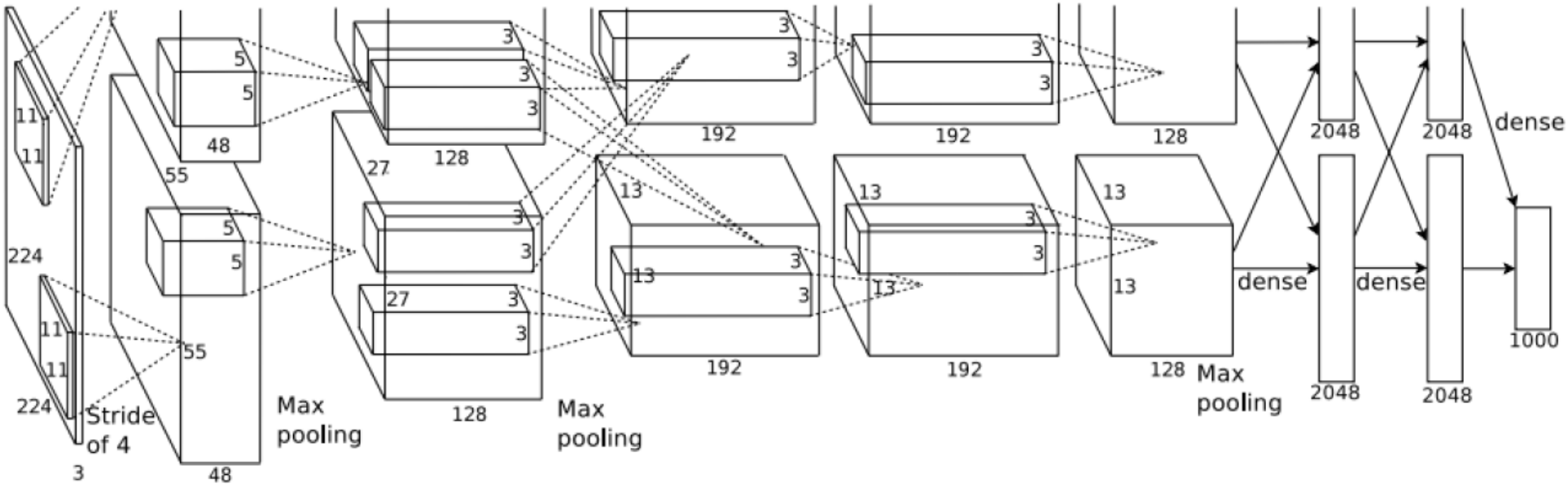


Columns show which S2 activations are used to compute C3

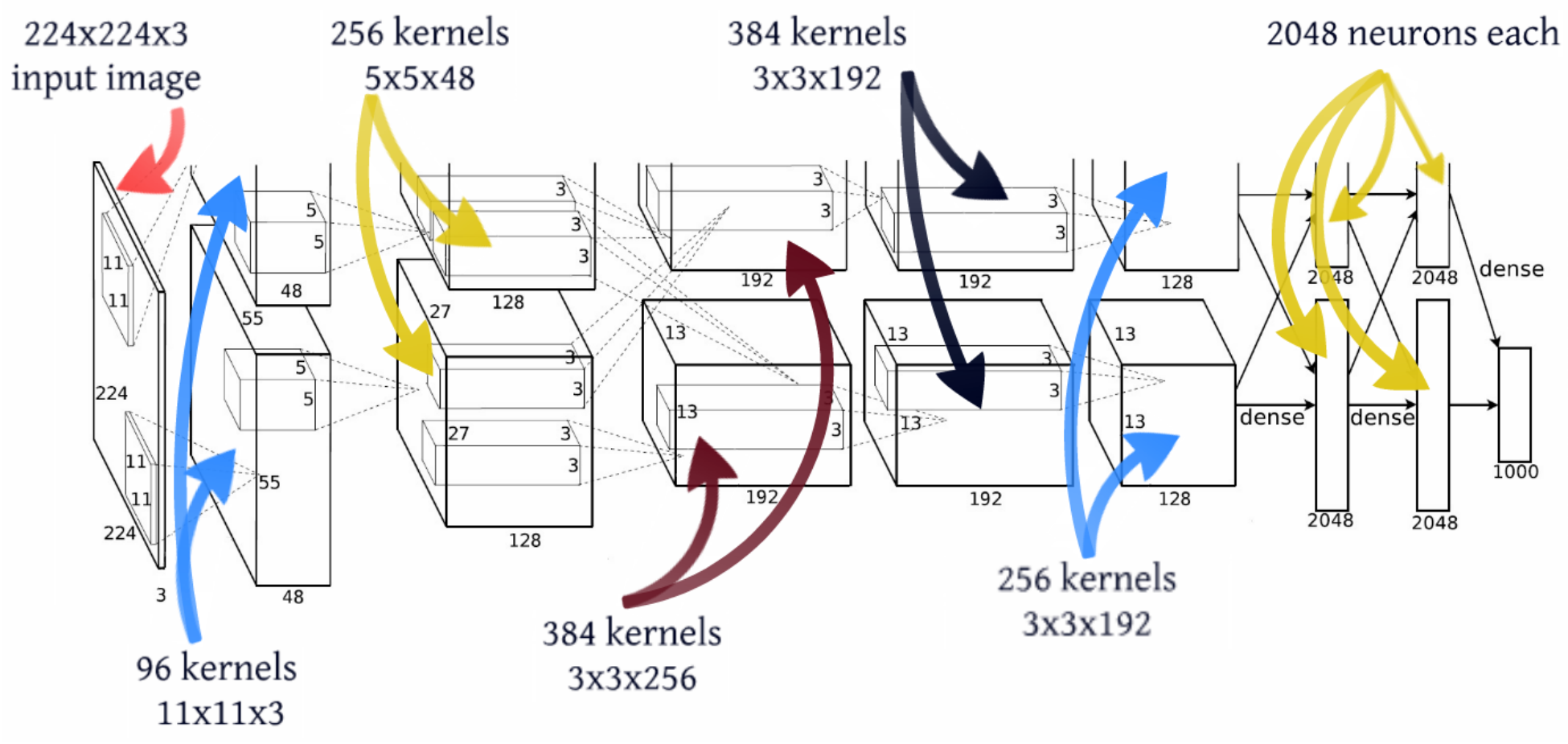
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

AlexNet

A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12, (USA), pp. 1097-1105, Curran Associates Inc., 2012



AlexNet – slightly more details

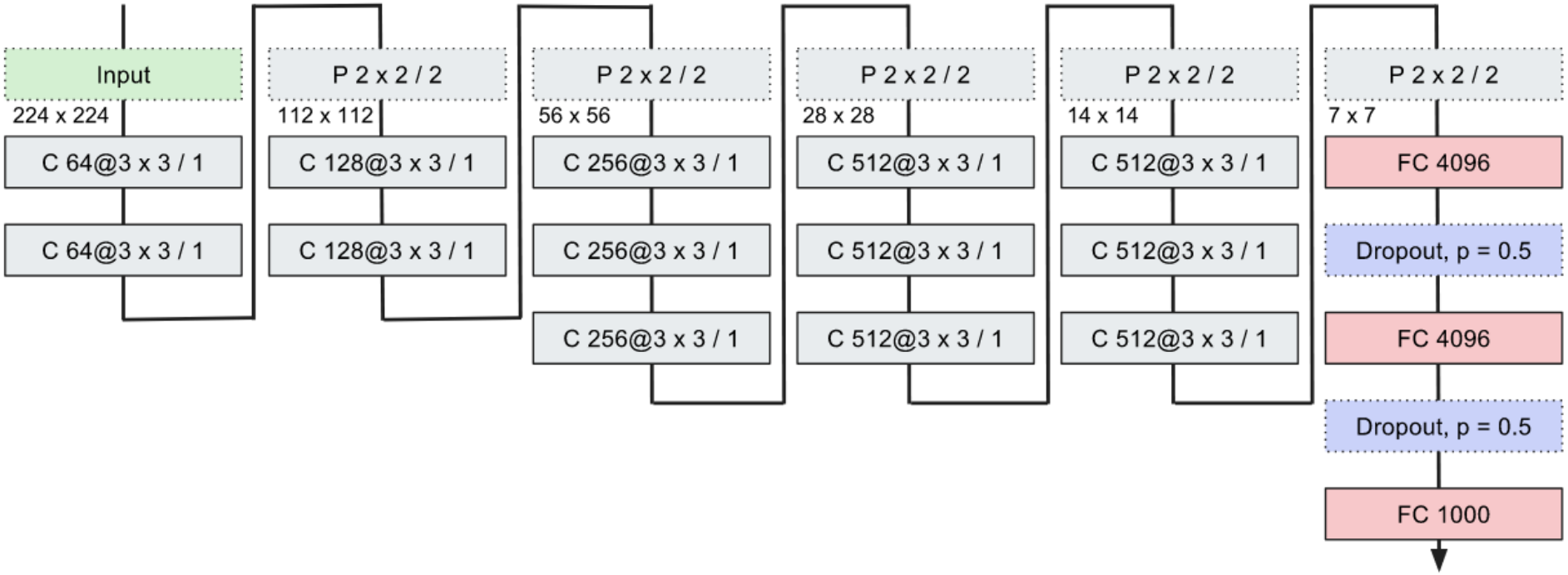


AlexNet – classification

AlexNet – retrieval

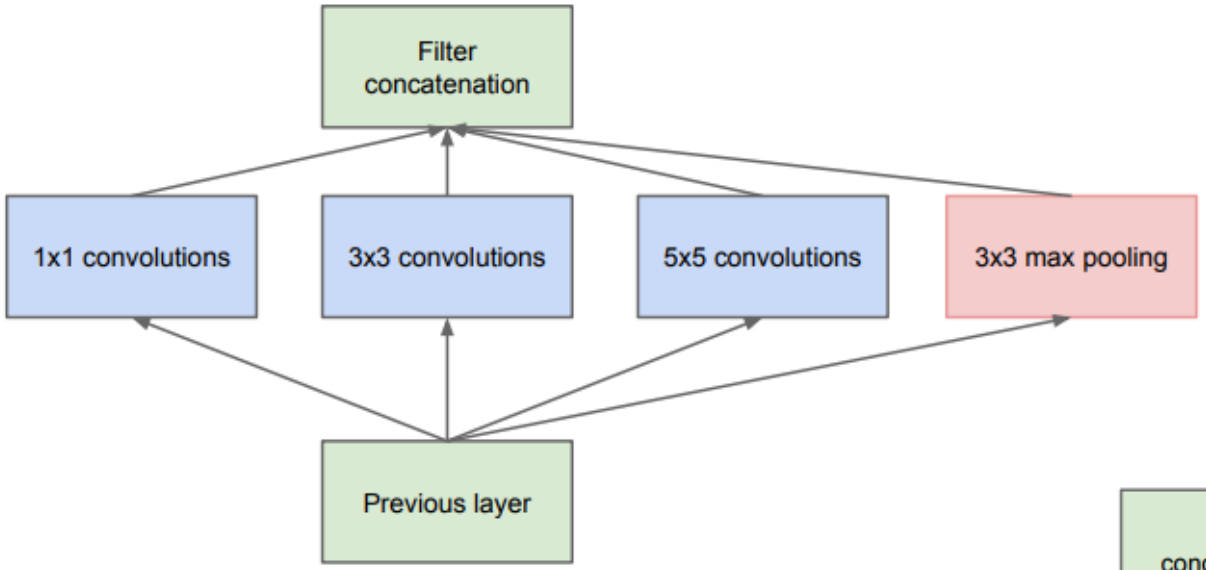
VGG16-D

K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, CoRR, arXiv:1409.1556, 2014

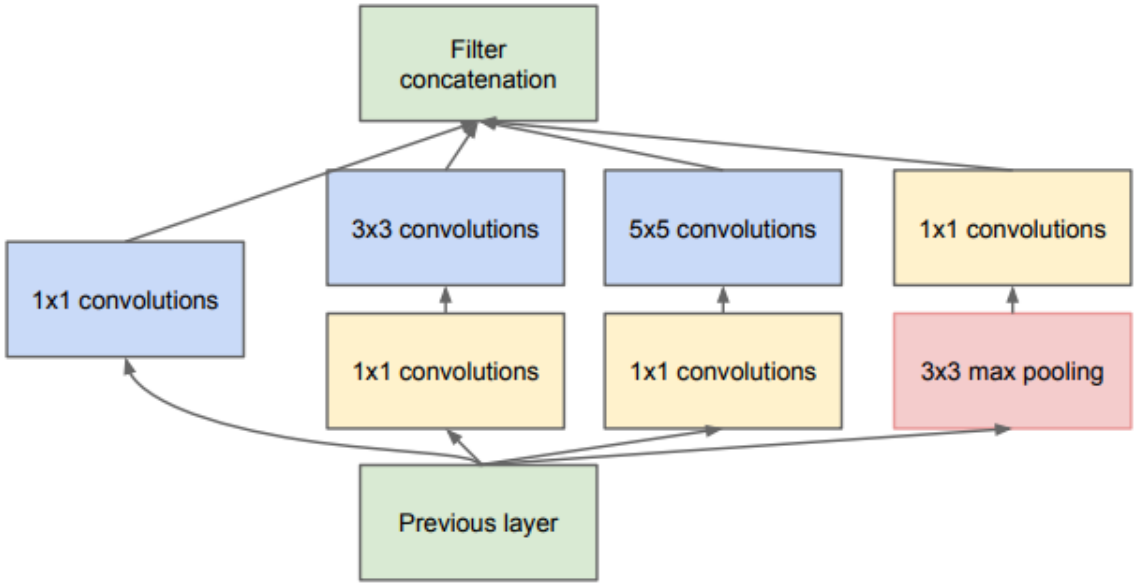


GoogLeNet / Inception

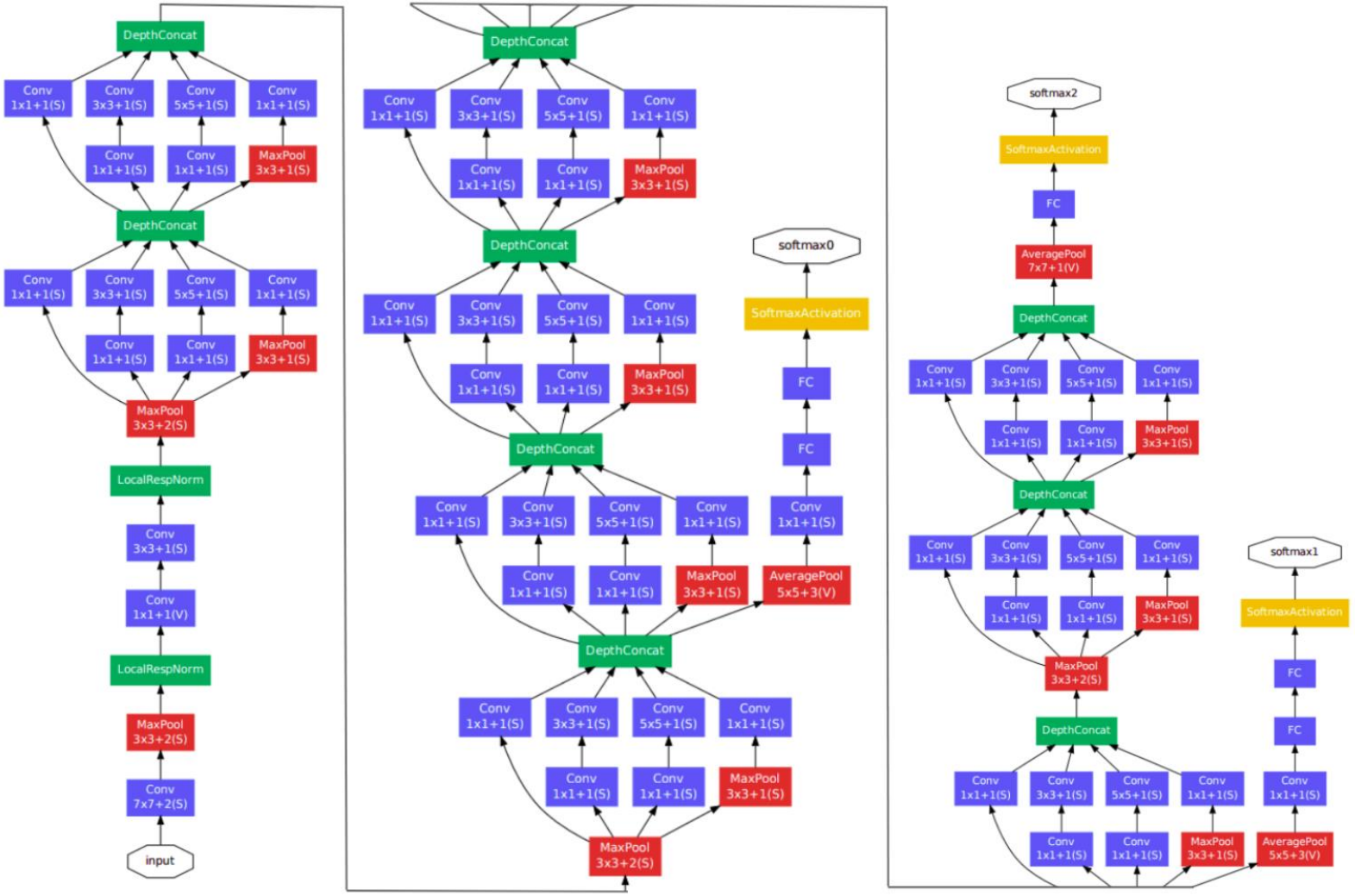
naive



With dimensions reductions

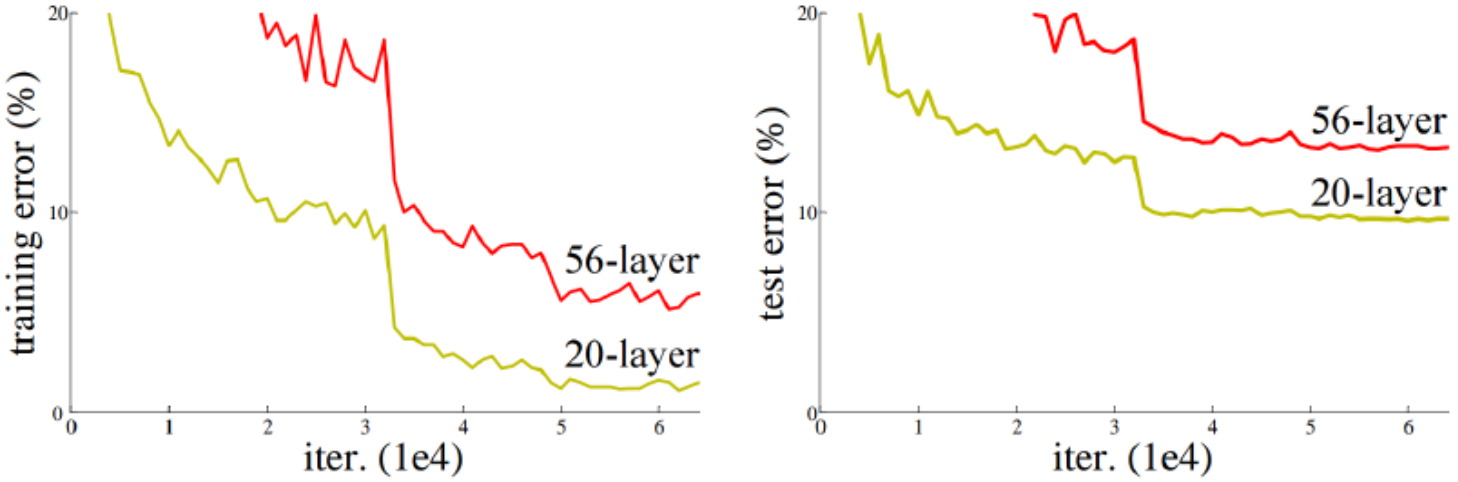


GoogleNet / Inception



Deep network problems

Learning better network is not as easy as stacking more layers. The first problem that we might encounter when having deep network is the vanishing/exploding gradient's phenomenon that prevents the network to converge.
There is a new problem known as *degradation problem*: training accuracy gets saturated with increased depth and degrades rapidly.



ResNet

K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, June 2016

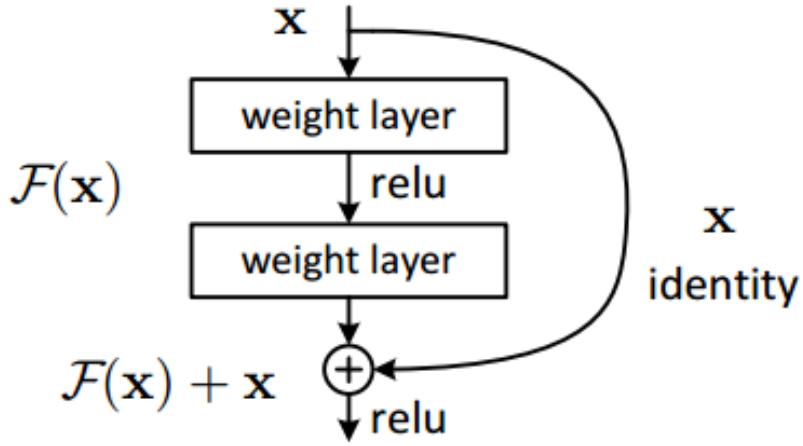
Let's give the layer a chance to do nothing - act as a transmitter replicating input on output. Such a residual block will not degrade the result of the classification, even if we increase the number of layers. This method proved to be a good remedy for the problem of network degradation.

Formally we can write residua block as:

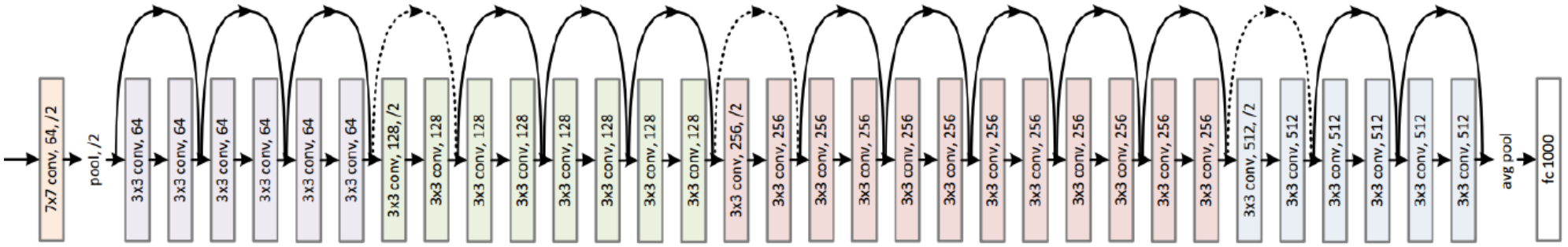
$$y = \mathcal{F}(x, \{W_i\}) + x$$

and with dimensionality fitting:

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



ResNet-34

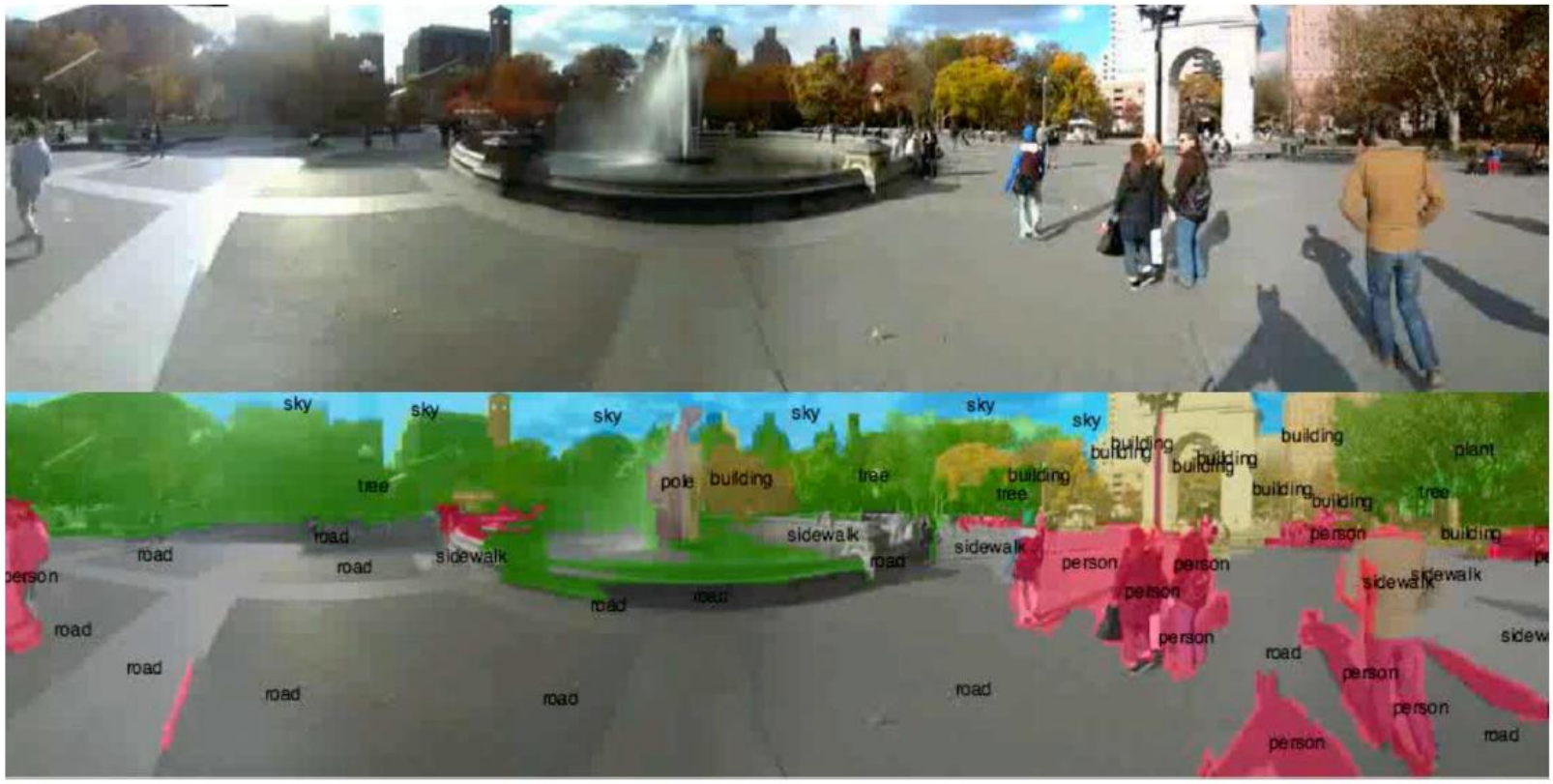

7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000

Transfer learning

Transfer learning is a technique in machine learning where a model trained on a particular task (source task) is repurposed for another task (target task) in such a way that the representation learned in source task is exploited to improve generalization in target task.
In general, there are three main questions in transfer learning: what to transfer, how to transfer, and when to transfer.
What to transfer asks which part of knowledge can be transferred across tasks or domains. Some knowledge is specific for individual domains or tasks, and some knowledge may be common between different domains. After discovering which knowledge that can be transferred, learning algorithms need to be developed which corresponds to the **how** to transfer issue. **When** to transfer asks in which situations, transferring should be done. In some situations, when the source domain and target domain are not related to each other, transfer learning may not be successful.

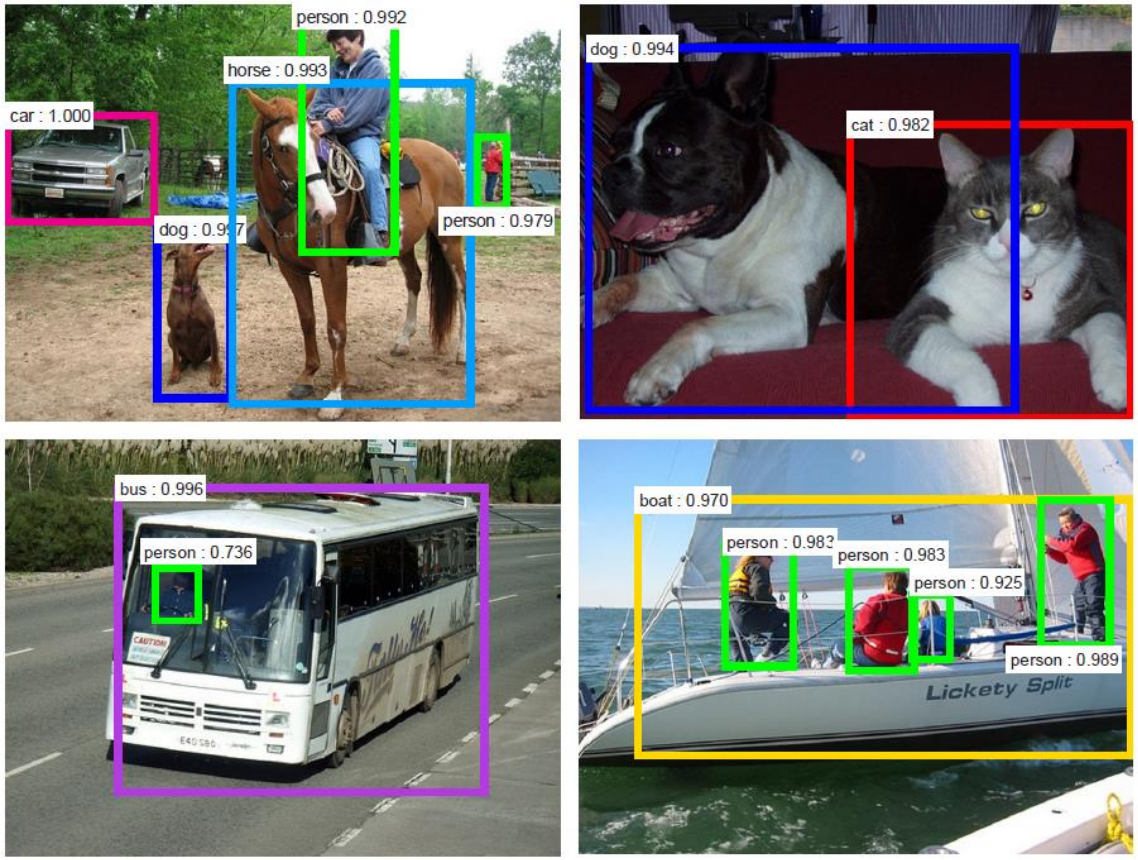
Applications – segmentation

C. Farabet, C. Couprie, L. Najman and Y. LeCun, *Learning Hierarchical Features for Scene Labeling*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1915-1929, Aug. 2013



Applications – detection

S. Ren, K. He, R. Girshick and J. Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, June 2017



Applications – image captioning

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

Minor errors

A man in a baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

Somewhat related

A woman is holding a cat in her hand

A woman standing on a beach holding a surfboard

Applications – styling