

Image Classification III

Dr Wenjia Bai

Department of Computing & Brain Sciences

Outline

- Convolutional neural networks
 - Building blocks
- Examples
 - LeNet-5 (1998)
 - AlexNet (2012)
 - VGG (2014)
 - ...

Convolutional neural networks

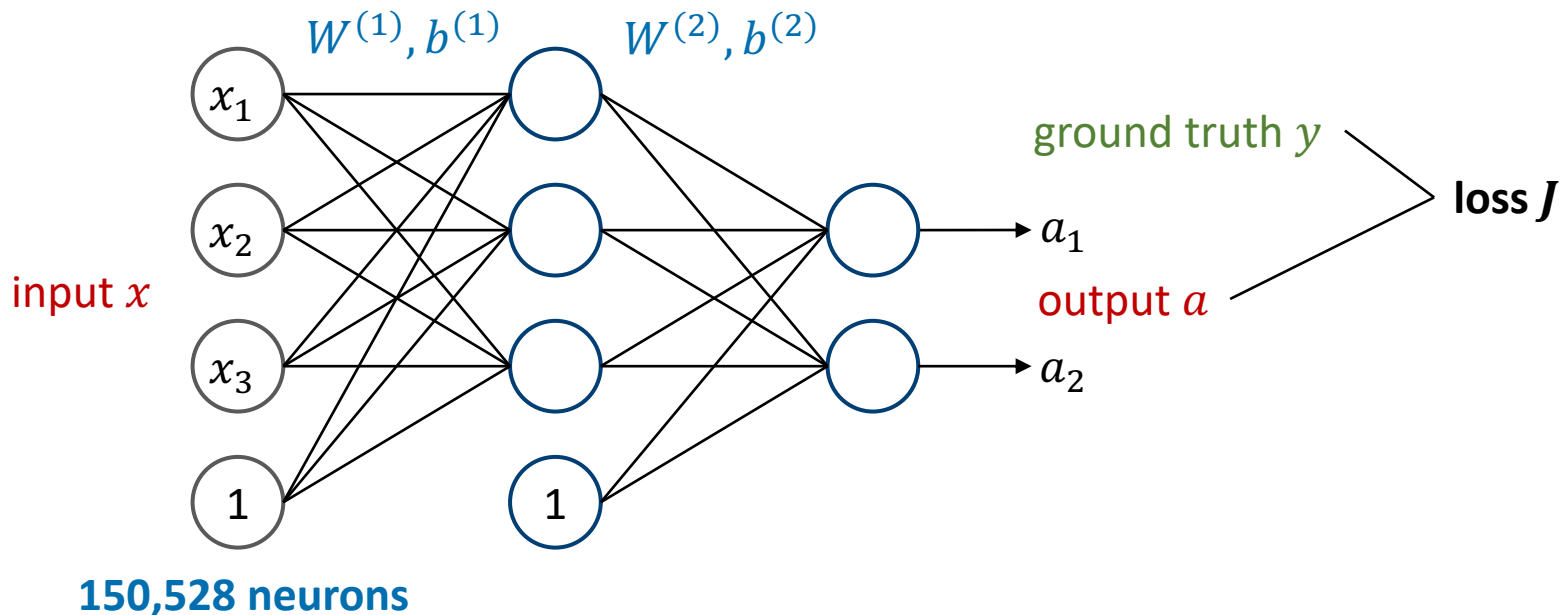
- Convolutional neural networks (CNNs) are similar to the multi-layer perceptron (MLP) in the previous lecture.
- However, CNNs assume that the inputs are images and encode certain properties (local connectivity, weight sharing etc.) into the architecture, which will make the computation more efficient and substantially reduce the number of parameters.

Limitations with MLP

- It may not scale up to bigger images. For example,
 - For a 224 x 224 RGB images, $224 \times 224 \times 3$ channels = 150,528 neurons for Layer 1.
 - A single neuron in Layer 2 has 150,528 parameters, not to mention all neurons on this layer and following layers.

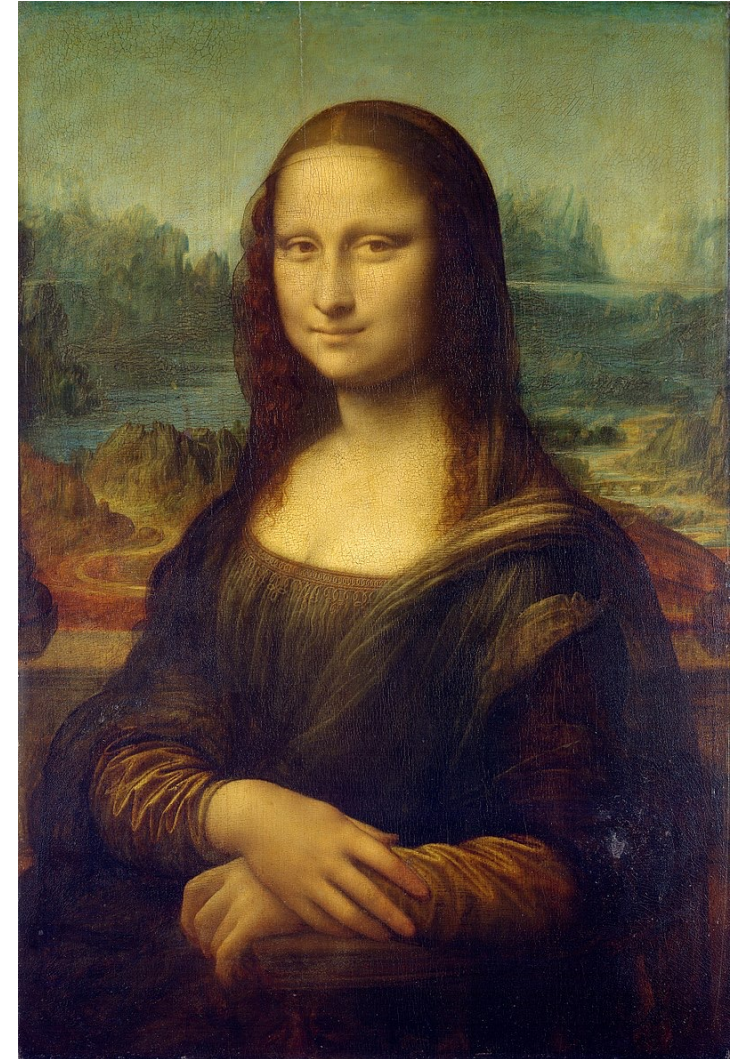


224 x 224 pixels by 3 channels

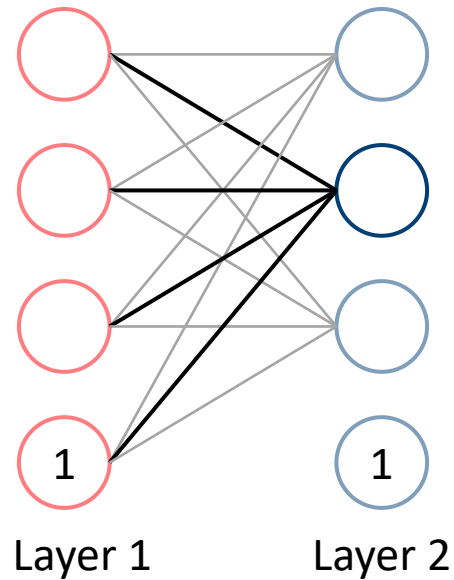


Convolutional neural networks

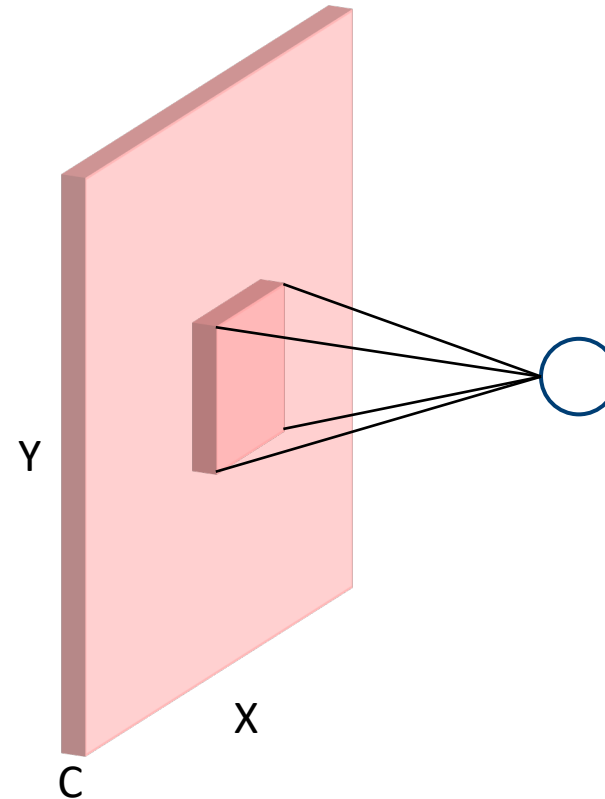
- When we look at images, we look at each local region, process information and combine the local information from different regions to form a global understanding.
- In convolutional neural networks (CNNs), each neuron only see a small local region in the layer before it. The region it sees is called the receptive field.
- The local connectivity substantially reduces the number of parameters.



Convolutional neural networks

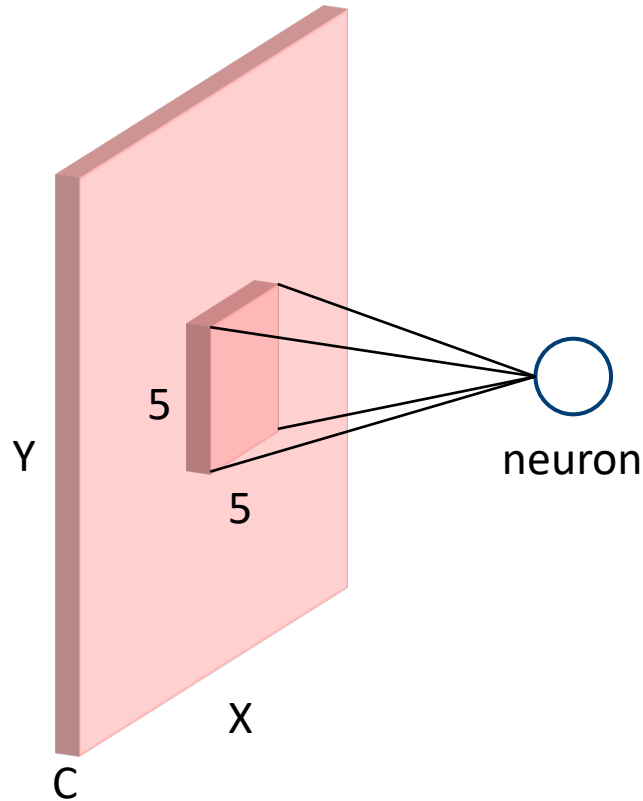


A fully connected layer in MLP.
A neuron in Layer 2 depends on all
the neurons in the previous layer.



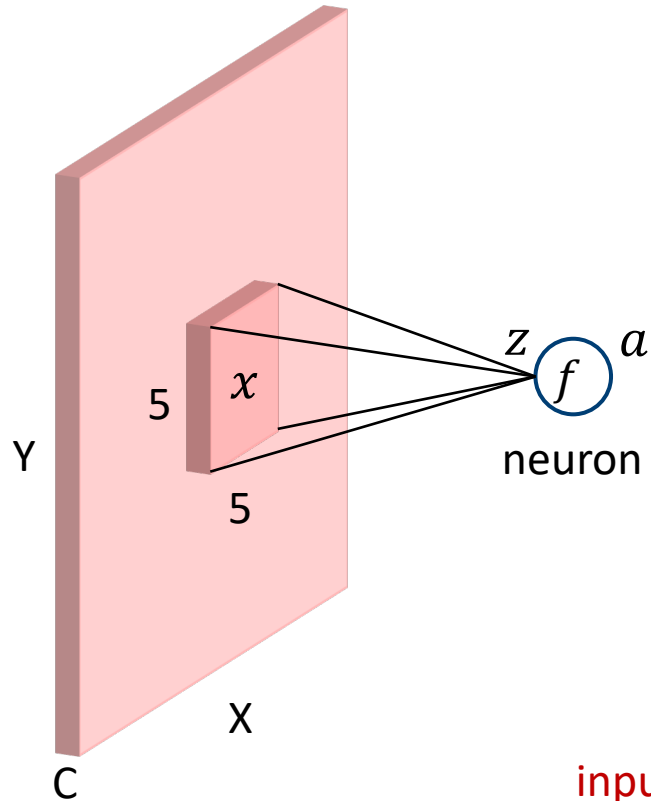
A convolutional layer in CNN.
A neuron in Layer 2 only depends on a
small local region in the previous layer.

Convolutional layer



- Input: $X \times Y \times C$
- C: channel.
 - $C = 3$ for RGB image.
 - $C = 1$ for grayscale image.
- This neuron depends a $5 \times 5 \times C$ cube of the input.
- The number of parameters is $5 \times 5 \times C$ for connection weights and 1 for bias.

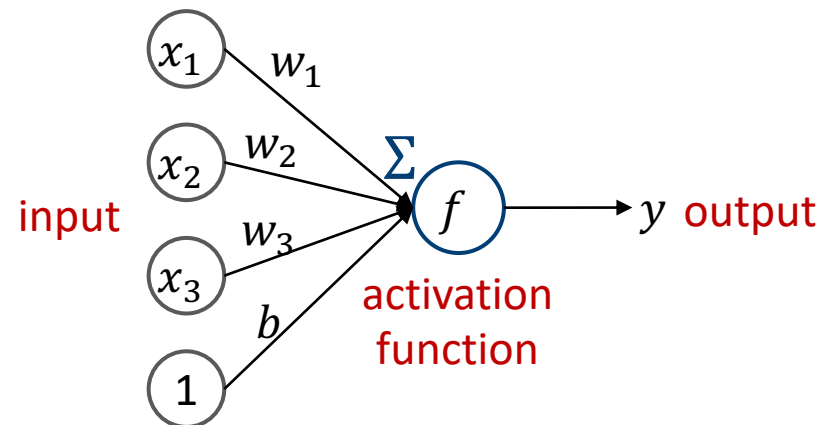
Convolutional layer



- The input and output of the neuron are defined by,

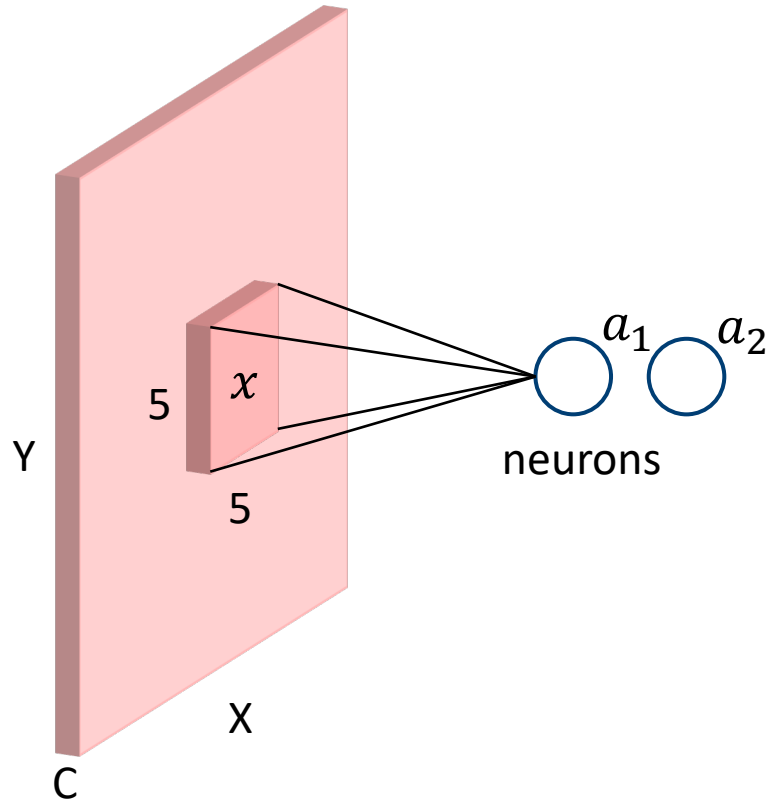
$$z = \sum_{ijk} \overset{\text{weights}}{W_{ijk}} \overset{\text{sum over 3 axes}}{x_{ijk}} + \overset{\text{bias}}{b}$$

$$a = \underset{\text{activation function}}{f}(z)$$



Analogy to a single neuron in MLP

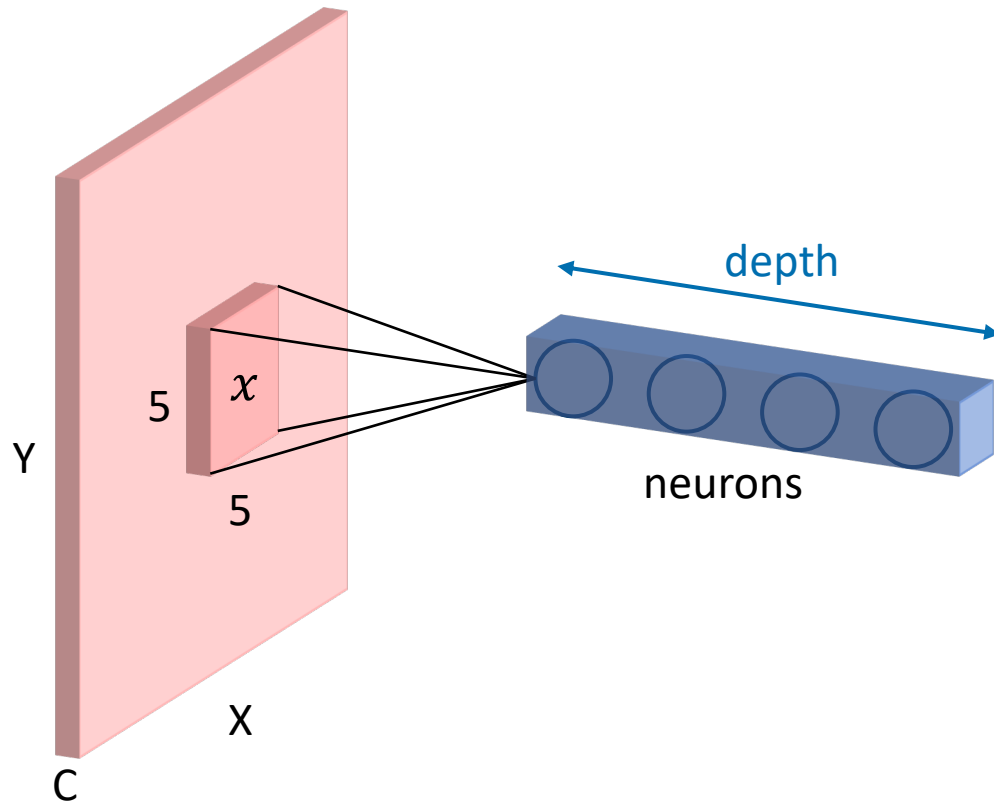
Convolutional layer



- With 2 neurons, the outputs are

$$a_1 = f\left(\sum_{ijk} W_{1ijk} x_{ijk} + b_1\right)$$
$$a_2 = f\left(\sum_{ijk} W_{2ijk} x_{ijk} + b_2\right)$$

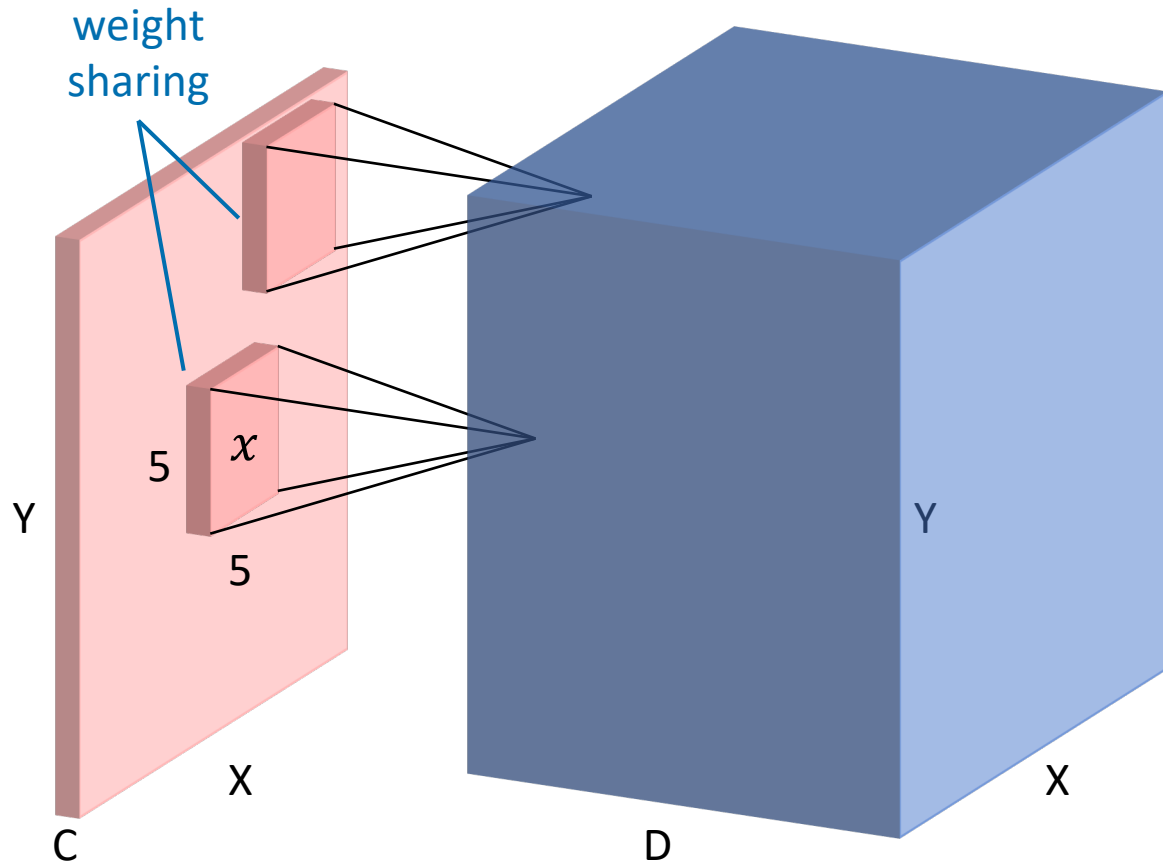
Convolutional layer



- We can add more neurons, which form a $D \times 1 \times 1$ cube of output, where D denotes depth.
- Each of the D neurons has its own weights W and bias b .

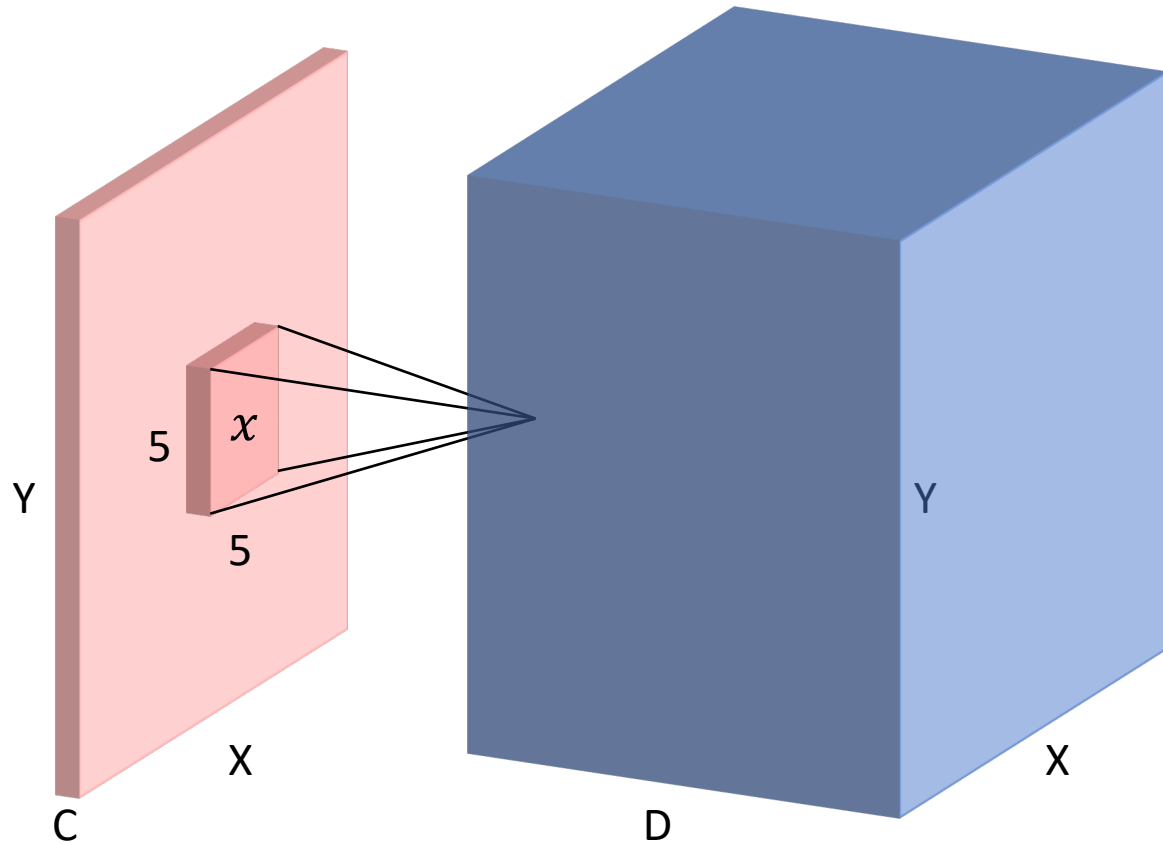
$$a_n = f\left(\sum_{ijk} W_{nijk} x_{ijk} + b_n\right)$$

Convolutional layer



- We can move the small window across the input image and get an output cube of size $X \times Y \times D$.
- Each of the small window shares the same weights (weight sharing).
- Therefore, for this example, the number of parameters is only $5 \times 5 \times C \times D$ for connection weights and D for bias.

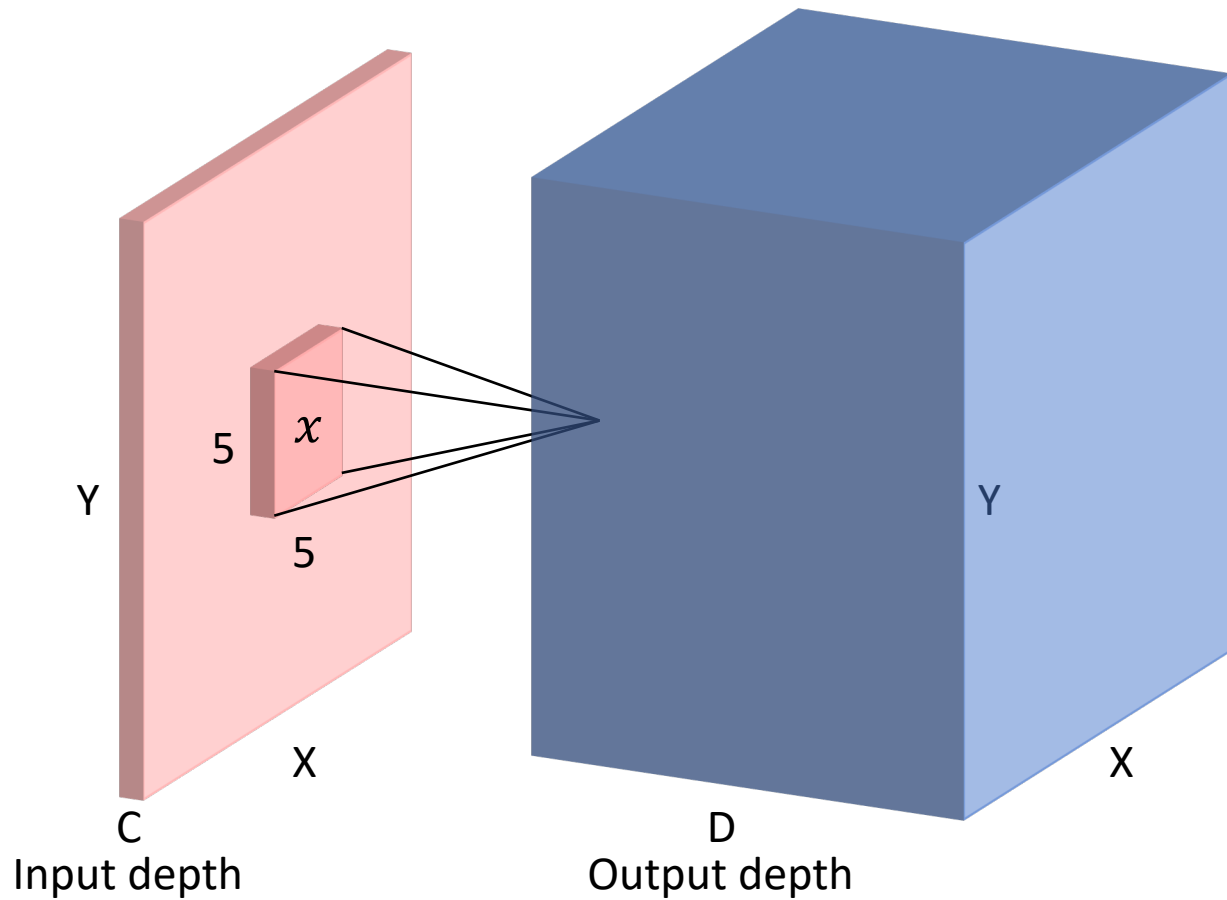
Convolutional layer



- This is called a convolutional layer.

$$a_d = f\left(\sum_{ijk} \overset{\text{convolutional}}{\overset{\text{kernel}}{W_{dijk}}} x_{ijk} + b_d\right)$$

Convolutional layer



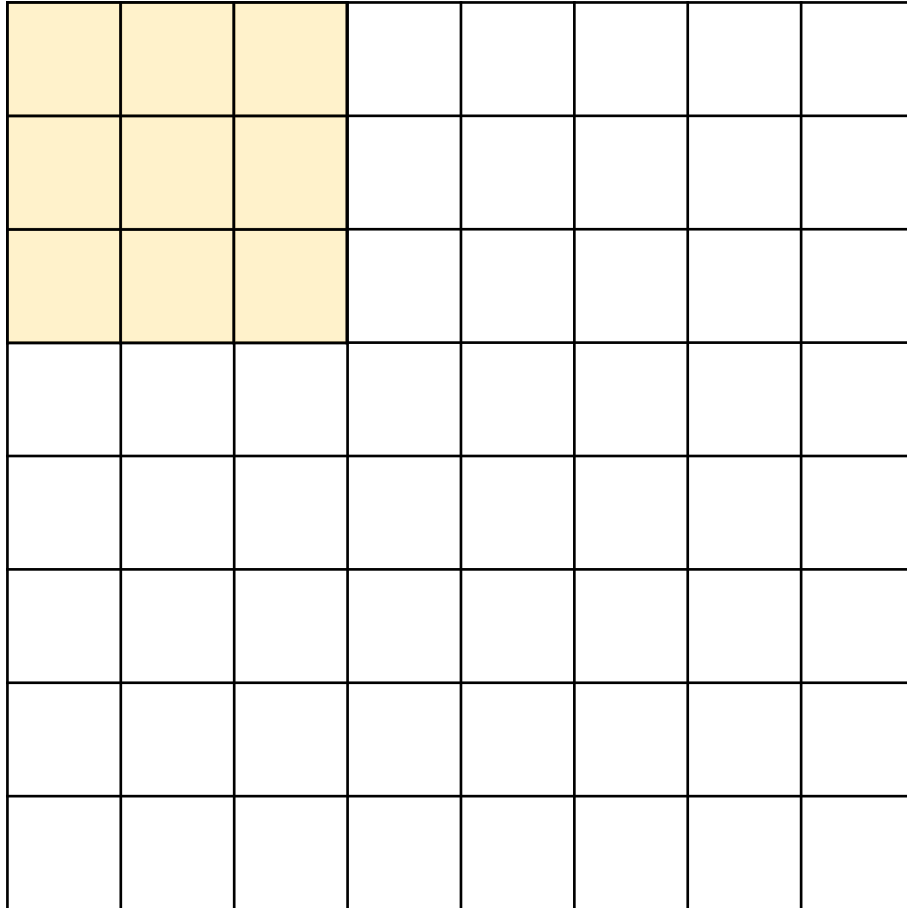
- All together, the convolutional kernel W has four dimensions:

$$a_d = f\left(\sum_{ijk} W_{dijk} x_{ijk} + b_d\right)$$

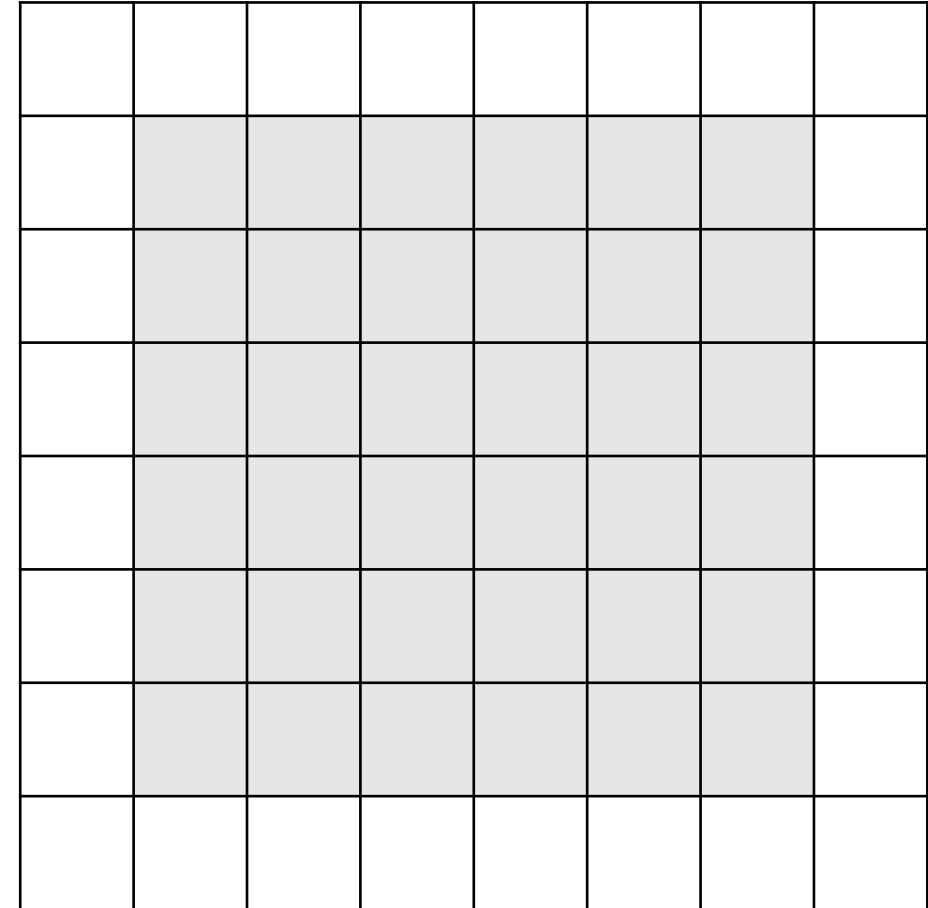
- Output depth
 - Kernel width
 - Kernel height
 - Input depth
- The output feature map has three dimensions:
 - Output depth
 - Width
 - Height

Convolutional layer

- Some operations during convolution
 - Padding
 - Stride
 - Dilation



3x3 convolutional kernel applied
to an image of shape [X, Y]



output shape = $[X - 2, Y - 2]$

0	0	0	0	0	0	0	0	0	0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0	0	0	0	0	0	0	0	0	0

3x3 convolutional kernel applied
to the image with padding $p = 1$

output shape = $[X, Y]$

Stride

- We do not need to move the window across all the pixels.
- We can use a stride, e.g. $s=2$, to move the window faster and look at a downsampled grid.

0	0	0	0	0	0	0	0	0	0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0	0	0	0	0	0	0	0	0	0

padding $p = 1$, stride $s = 2$

output shape = $[X/2, Y/2]$

0	0	0	0	0	0	0	0	0	0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0	0	0	0	0	0	0	0	0	0

padding $p = 1$, stride $s = 2$

output shape = $[X/2, Y/2]$

0	0	0	0	0	0	0	0	0	0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0	0	0	0	0	0	0	0	0	0

padding $p = 1$, stride $s = 2$

output shape = $[X/2, Y/2]$

0	0	0	0	0	0	0	0	0	0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0									0
0	0	0	0	0	0	0	0	0	0

padding $p = 1$, stride $s = 2$

output shape = $[X/2, Y/2]$

Dilation

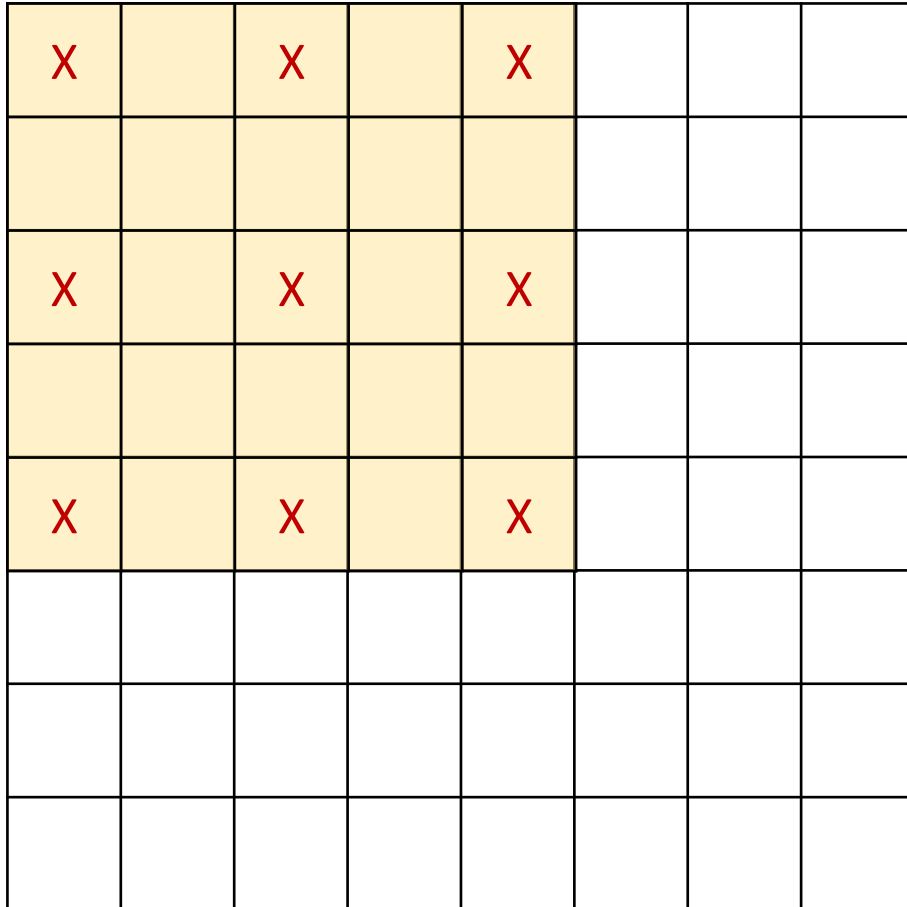
- If we want a neuron to look at a large region (receptive field) in the image, there are two ways to achieve this:
 - Keep the convolutional kernel size unchanged, e.g. still 3x3. But downsample the image and apply the kernel to the downsampled image.
 - Increase the kernel size, e.g. 5x5 or 7x7. However, this will increase the number of parameters.
- Dilation aims to increase the receptive field on the original image (not downsampled) without increasing the number of parameters.

X	X	X					
X	X	X					
X	X	X					

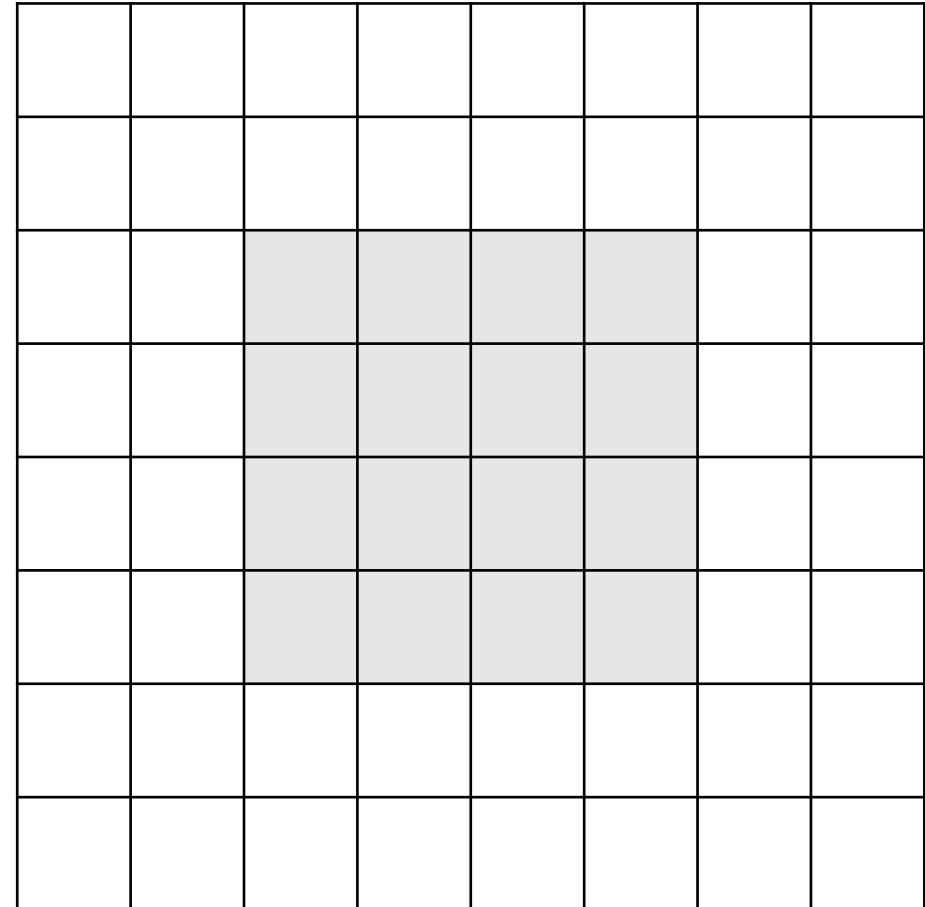
3x3 convolutional kernel,
dilation = 1

X	X	X	X	X			
X	X	X	X	X			
X	X	X	X	X			
X	X	X	X	X			
X	X	X	X	X			

5x5 convolutional kernel,
dilation = 1



3x3 convolutional kernel,
dilation = 2



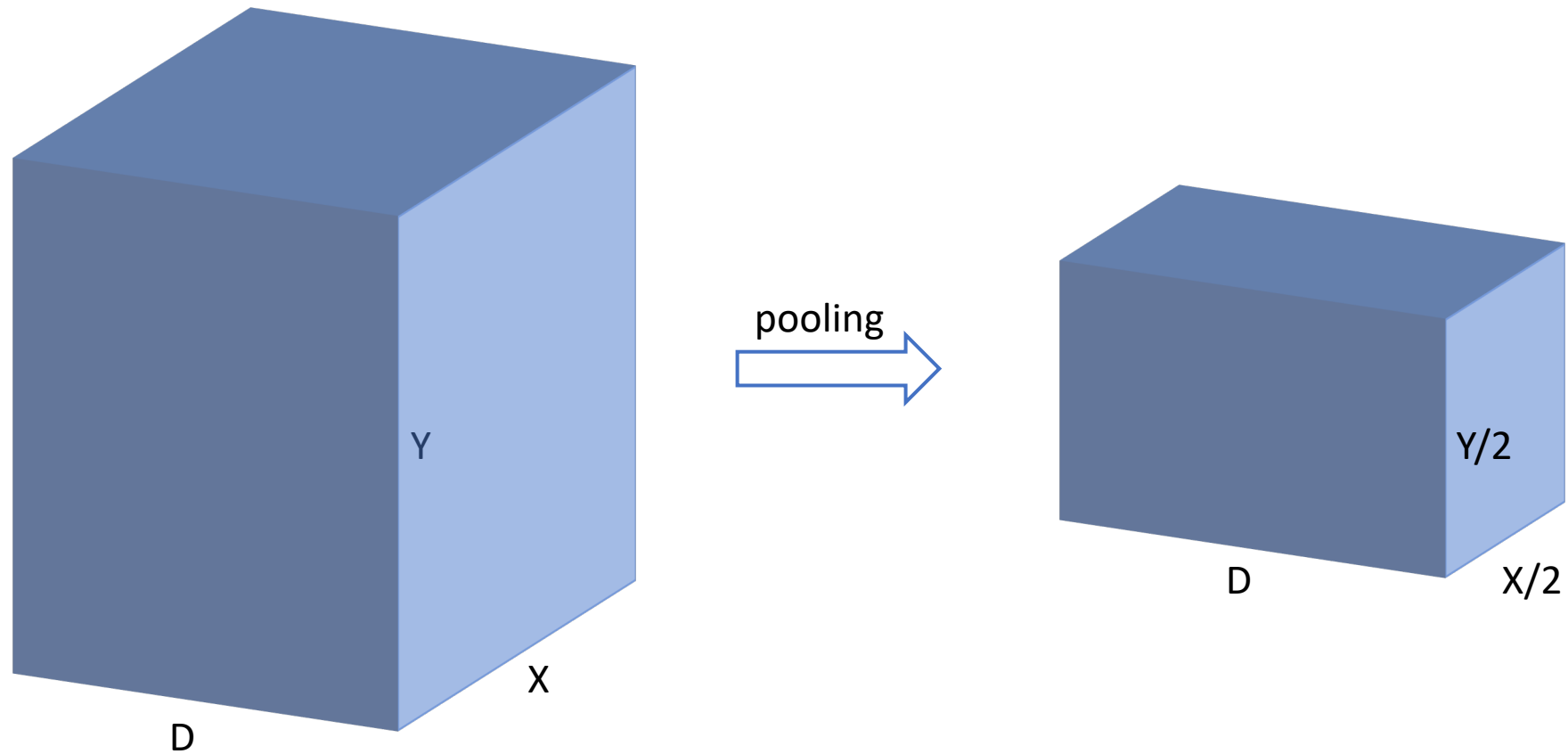
Convolutional layer

- When we perform convolution, we can set these parameters
 - Padding
 - Stride
 - Dilation

Pooling layer

- Apart from convolutional layer, pooling layer is another building block for convolutional neural networks.
- It is an operation to make feature maps or representations smaller.
- It is similar to image downsampling.
- After pooling, neurons in the next layer can see a larger region of image.


Pooling layer



Max pooling

1	2	3	4
5	6	7	8
4	3	2	1
0	1	2	3

max pooling
with 2x2 window
and stride=2



6	8
4	3

Pooling layer

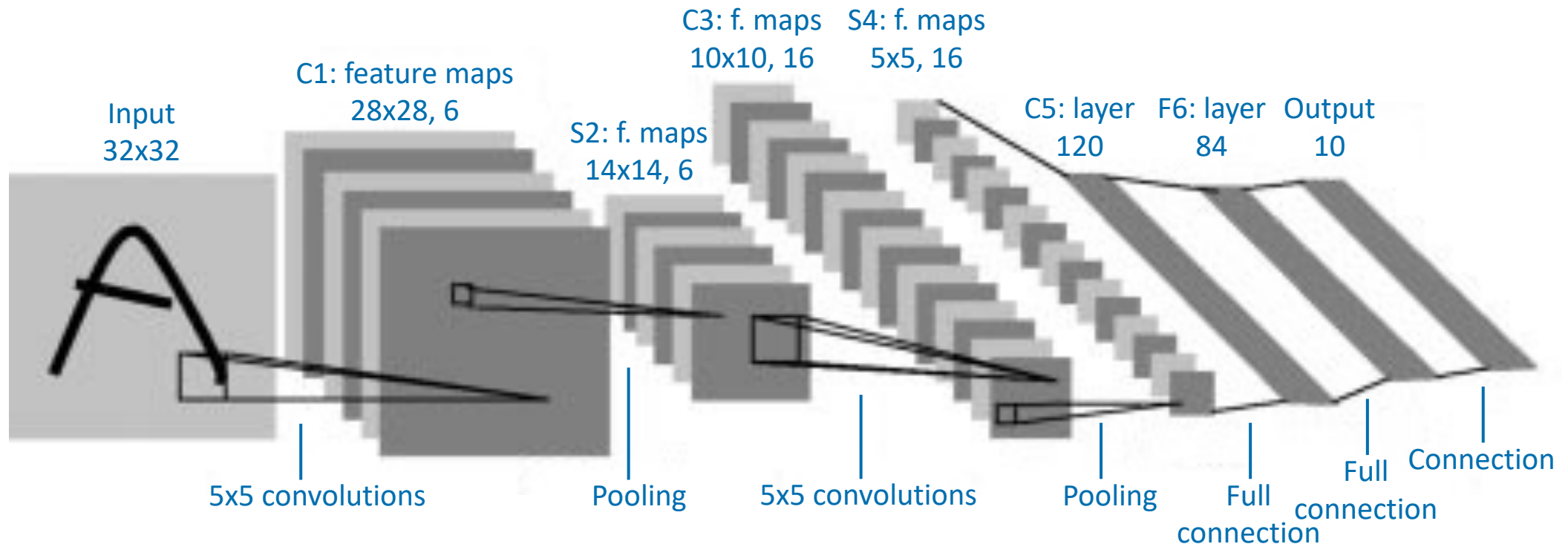
- Apart from max pooling, there is also average pooling, which takes the average response in each window.

Convolutional neural networks

- After introducing the convolution operation and other operations, we can use them to build a convolutional neural network.
- We will show a classic example here.

LeNet-5

- 7 layers in total, not counting the input layer.
 - C: convolutional layer
 - S: pooling layer (subsampling)
 - F: fully connected layer



LeNet-5

Small number of parameters here.

C1: $5 \times 5 \times 6$ (weight) + 6 (bias) = 156 parameters

Large feature map.

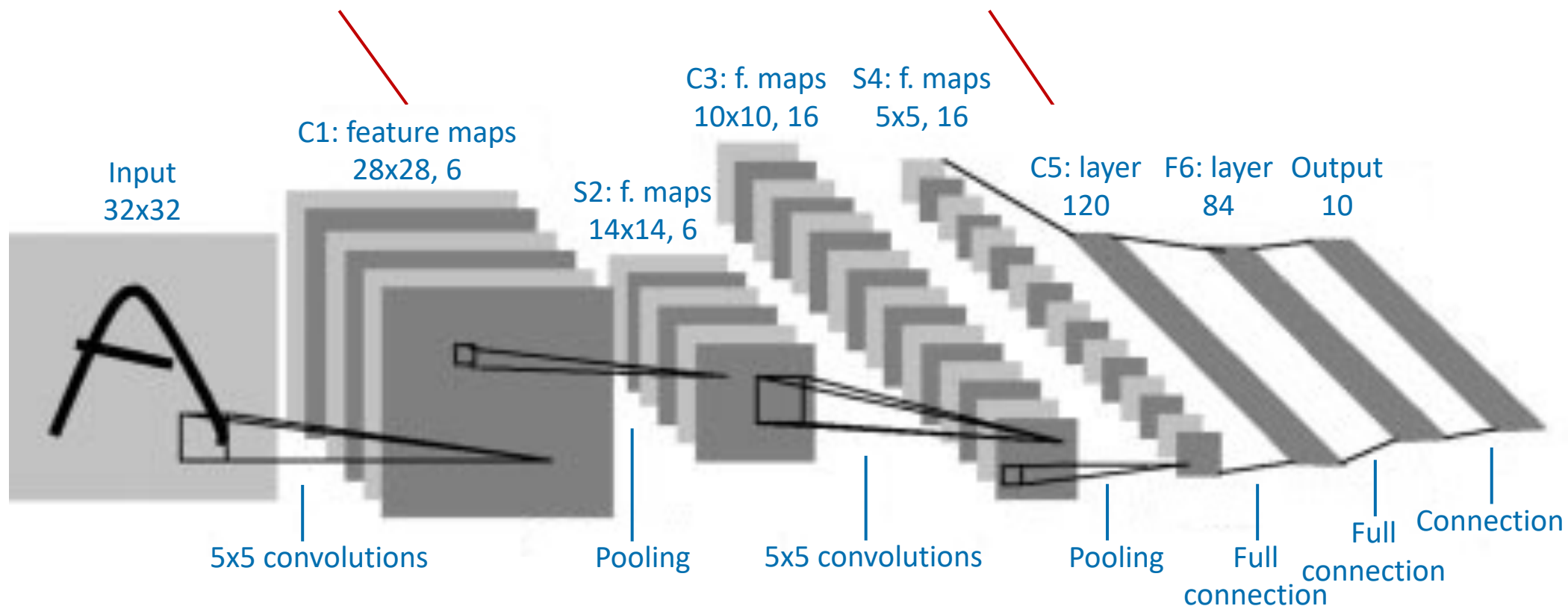
C1: 28×28 pixel \times 6 channel = 4,704 floating-point num.

Large number of parameters here.

C5: $5 \times 5 \times 16 \times 120$ (weight) + 120 (bias) = 48120 parameters

Small feature map, abstract representation of the image.

C5: 120 floating-point number



LeNet-5

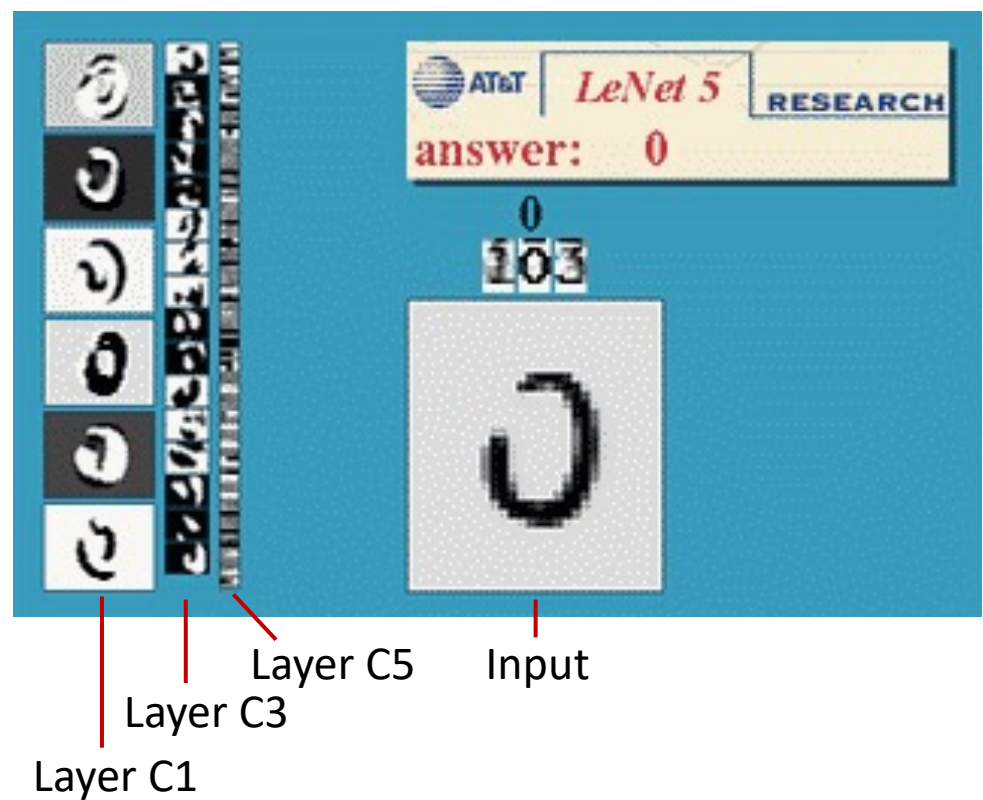
- We can define a loss function, e.g. cross-entropy, for classification.
- Each convolutional layer is mathematically similar to a layer in MLP,

$$z = \sum_{ijk} W_{ijk} x_{ijk} + b$$
$$a = f(z)$$

- The network can be trained similarly as MLP, using backpropagation and stochastic gradient descent.
- After training, we can evaluate its performance on test data.

LeNet-5

- Now performance is comparable to SVM.



Demo from <http://yann.lecun.com/exdb/lenet/>.

Methods	Error Rate
KNN	5.0%
KNN (deslanted)	2.4%
SVM (polynomial, d=4)	1.1%
V-SVM (polynomial, d=9, augment.)	0.8%
MLP (784-300-10)	4.7%
MLP (784-300-10, augment.)	3.6%
MLP (784-1000-10)	4.5%
MLP (784-1000-10, augment.)	3.8%
MLP (784-500-150-10)	2.95%
MLP (784-500-150-10, augment.)	2.45%
LeNet-5	0.95%
LeNet-5 (augment.)	0.8%

MNIST classification performance

Deep neural networks

- In the 1990s and 2000s, it was technically challenging to develop and train a deep neural network.
- Instead, many people used SVM or random forests for classification or regression problems.
- When did deep neural networks become popular again?
 - In ICASSP 2011, a method based on deep belief networks (DBN) increased the speech recognition accuracy by a large margin [1].
 - In NIPS 2012, a deep convolutional neural network substantially increased the image classification accuracy in the ImageNet challenge [2].

[1] G.E. Dahl et al. Large vocabulary continuous speech recognition with context-dependent DBN-HMMs. ICASSP 2011.

[2] A. Krizhevsky et al. ImageNet classification with deep convolutional neural networks. NIPS 2012.

Deep neural networks

- Why did it become successful?
 - Hardware
 - Efficient use of graphical processing units (GPUs)
 - Data
 - Large datasets (e.g. ImageNet)
 - Algorithm
 - Improvement of optimisation: ReLU
 - Network architectures: AlexNet, VGG, ResNet, Transformer ...

Hardware

- Graphical processing units (GPU)
 - Developed for computer graphics and games.
 - Ideal for training neural networks, performing convolutional operations in parallel using thousands of cores on a GPU card.



Large datasets

- ImageNet Challenge
 - 256 x 256 x 3 RGB image
 - 1.2 million images
 - 1,000 classes



<http://www.image-net.org>

Improvement in optimisation

- Stochastic gradient descent

- For each batch of samples, we perform gradient descent

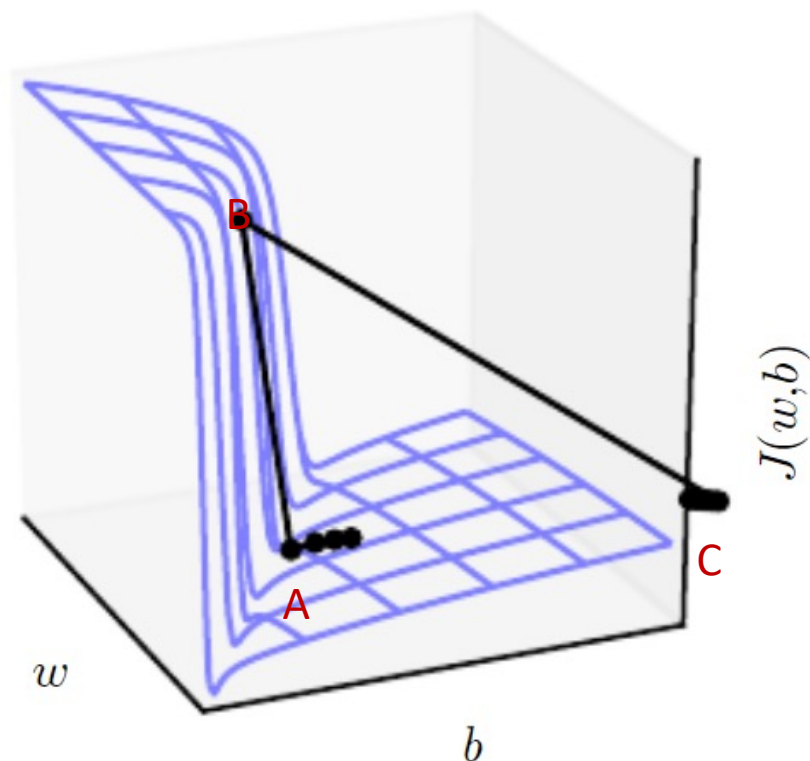
$$W = W - \alpha \frac{\partial L_B}{\partial W}$$
$$b = b - \alpha \frac{\partial L_B}{\partial b}$$

- Problems in optimisation

- Exploding gradient: The gradient becomes very large, preventing the algorithm from converging.
 - Vanishing gradient: The gradient becomes vanishingly small, preventing the weights from changing their values.

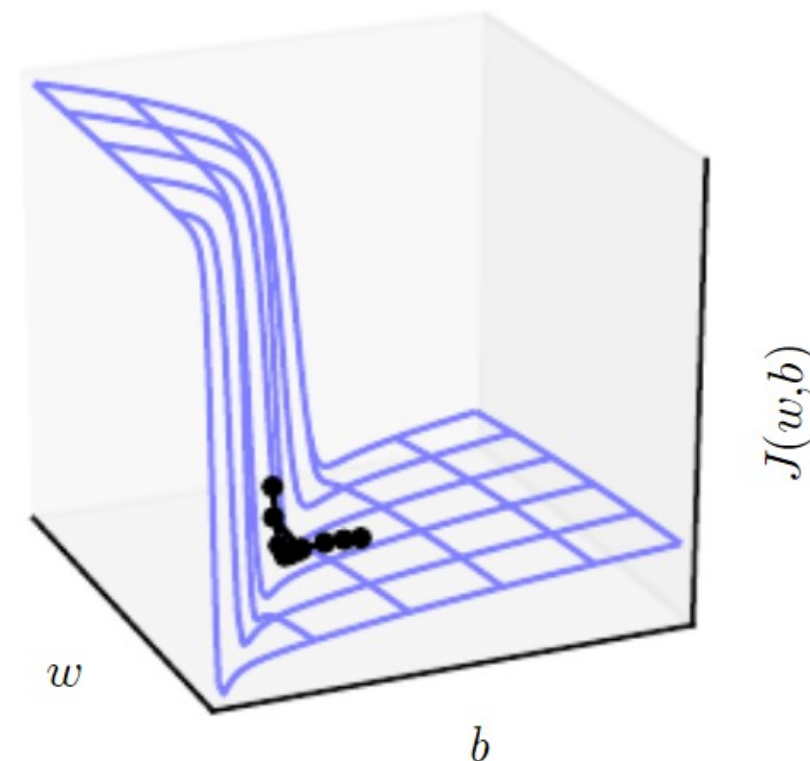
Exploding gradient problem

Without clipping



The gradient becomes very large in the vicinity of the cliff. Gradient descent will overshoot from A to B, then to C. It can not converge.

With clipping



After clipping the gradients, gradient descent converges.

Gradient clipping

- Clip by value

- For each element of the gradient g , clip by a minimal value and a maximal value,

$$g_i = \begin{cases} v_{min} , & \text{if } g_i < v_{min} \\ v_{max} , & \text{if } g_i > v_{max} \\ g_i , & \text{otherwise} \end{cases}$$

- Clip by norm

- Clip by the L2-norm of the gradient g ,

$$g = \begin{cases} \frac{g}{||g||} v , & \text{if } ||g|| > v \\ g , & \text{otherwise} \end{cases}$$

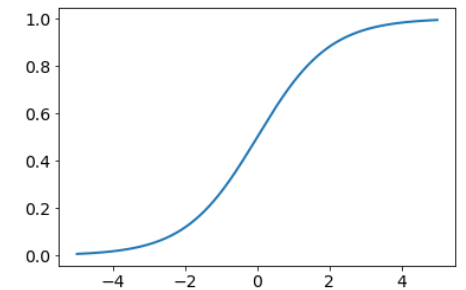
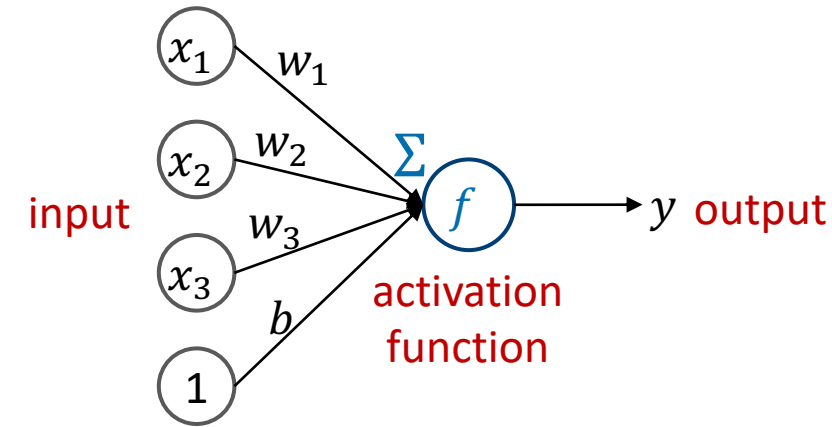
Vanishing gradient problem

- Let us revisit what a single neuron looks like.
- The neuron is a computational unit that takes an input, applies an activation function f and generates an output.

$$y = f\left(\sum_{i=1}^3 w_i x_i + b\right)$$

- For many years, people use the sigmoid function for f .

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



Sigmoid activation function

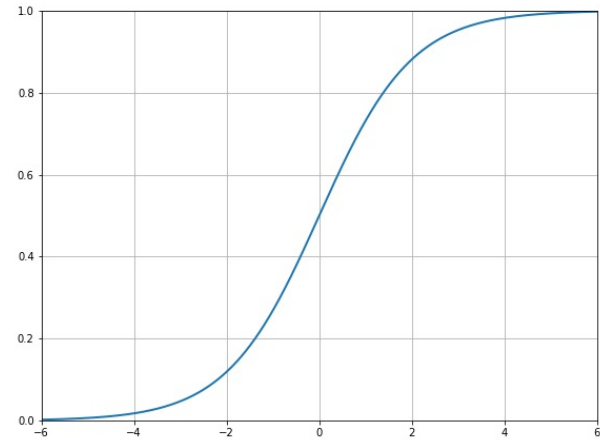
Vanishing gradient problem

- Sigmoid function

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

$$f'(z) = \frac{e^{-z}}{(1 + e^{-z})^2} = f(z)(1 - f(z))$$

- If $f(z)$ saturates at 0 or 1, then $f'(z)$ becomes almost 0.
- This causes a problem in backpropagating the gradient.



Sigmoid function

Activation function

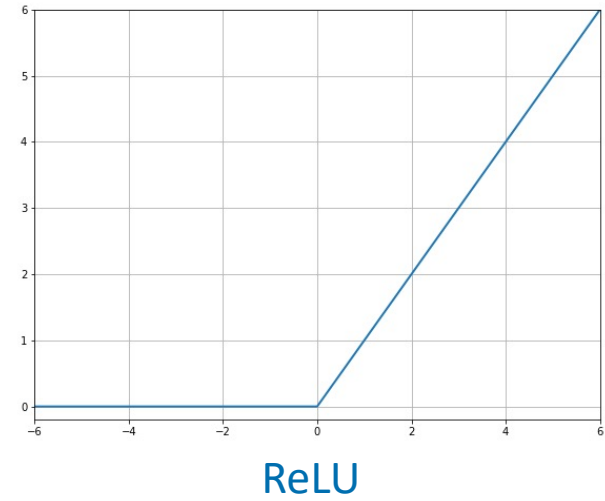
- Rectified linear unit (ReLU)

$$f(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases}$$

- Its subgradient is

$$f'(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$$

- The gradient will not vanish when z is very large.
- However, the gradient becomes 0 when z becomes negative.
- For deep neural networks, it has been found to outperform the sigmoid activation function and thus it has been widely used.



Activation function

- Leaky ReLU

$$f(z) = \begin{cases} 0.01z, & z < 0 \\ z, & z \geq 0 \end{cases}$$

- Parameteric ReLU (PReLU): similar to leaky ReLU, but learning the parameter a in training.

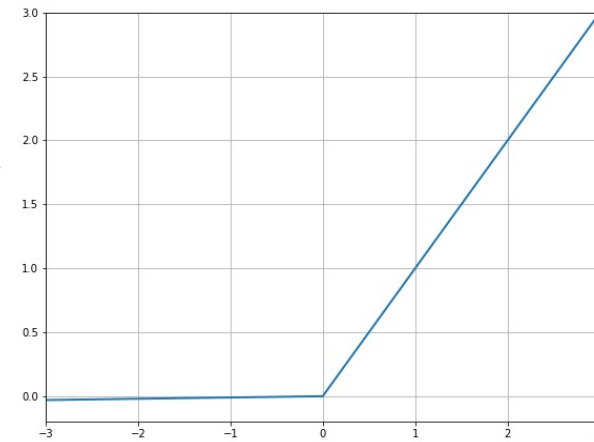
$$f(z) = \begin{cases} az, & z < 0 \\ z, & z \geq 0 \end{cases}$$

- Exponential linear unit (ELU)

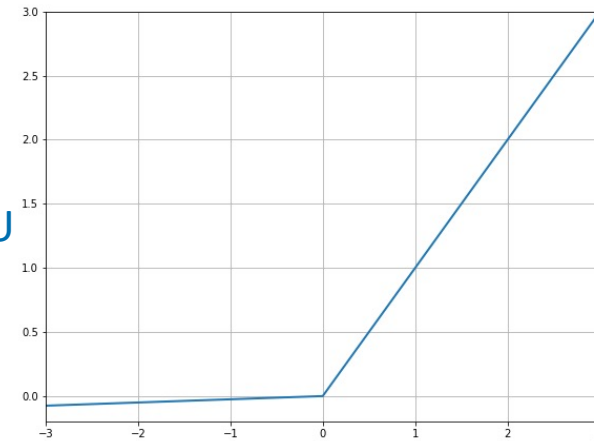
$$f(z) = \begin{cases} a(e^z - 1), & z < 0 \\ z, & z \geq 0 \end{cases}$$

- They allow a small gradient when z is negative and may perform better than ReLU.

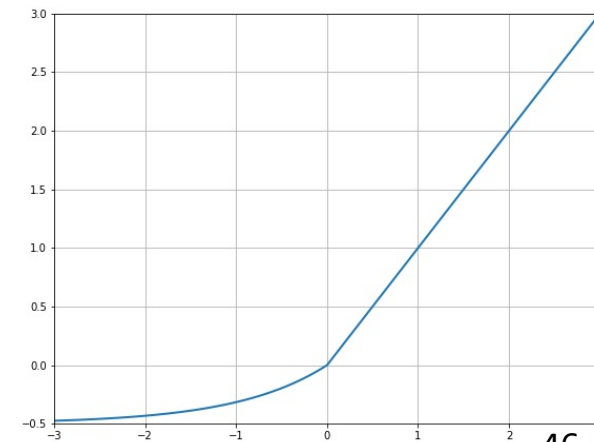
Leaky
ReLU



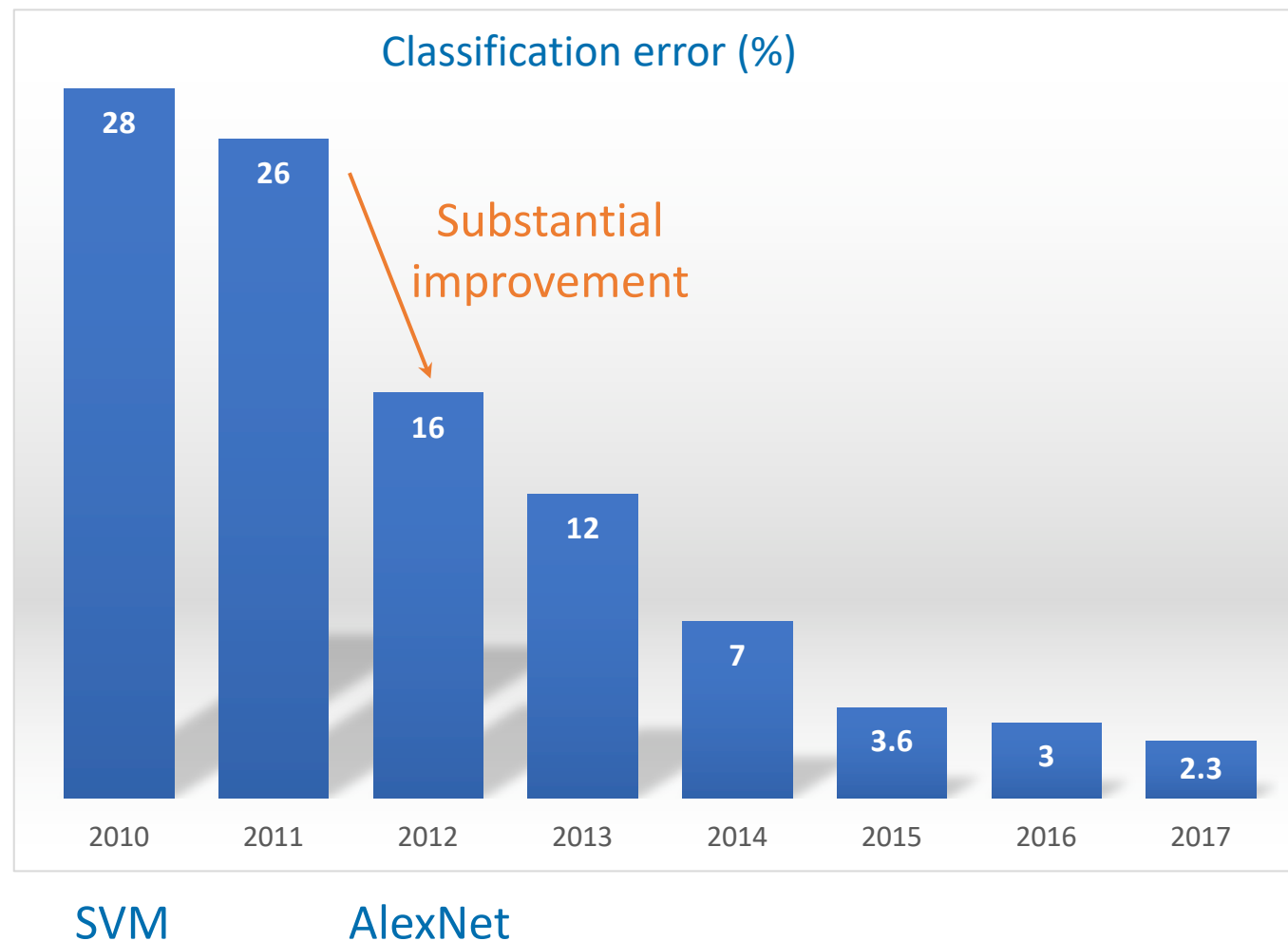
PReLU



ELU



ImageNet classification challenge



Architecture

- AlexNet

- Input

- 224 x 224 x 3

- Layer 1 (Conv 1)

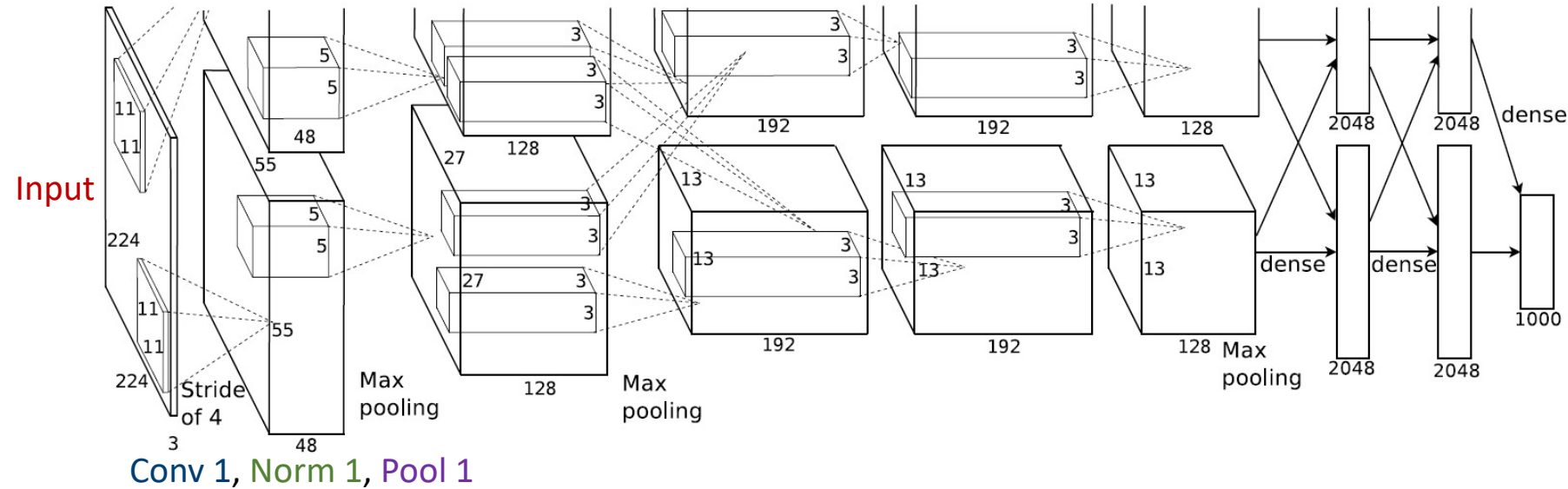
- 11 x 11 convolution, stride of 4, 96 filters
 - Output feature map: 55 x 55 x 96

- Layer 2 (Norm 1)

- Local response normalisation and using ReLU as activation function

- Layer 3 (Pool 1)

- 3 x 3 max pooling window, stride of 2
 - Output feature map: 27 x 27 x 96



AlexNet

[224x224x3] **Input**

[55x55x96] **Conv1**, 11x11, 96, s=4

[55x55x96] **Norm1**

[27x27x96] **Pool1**

[27x27x256] **Conv2**, 5x5, 256

[27x27x256] **Norm2**

[13x13x256] **Pool2**

[13x13x384] **Conv3**, 3x3, 384

[13x13x384] **Conv4**, 3x3, 384

[13x13x256] **Conv5**, 3x3, 256

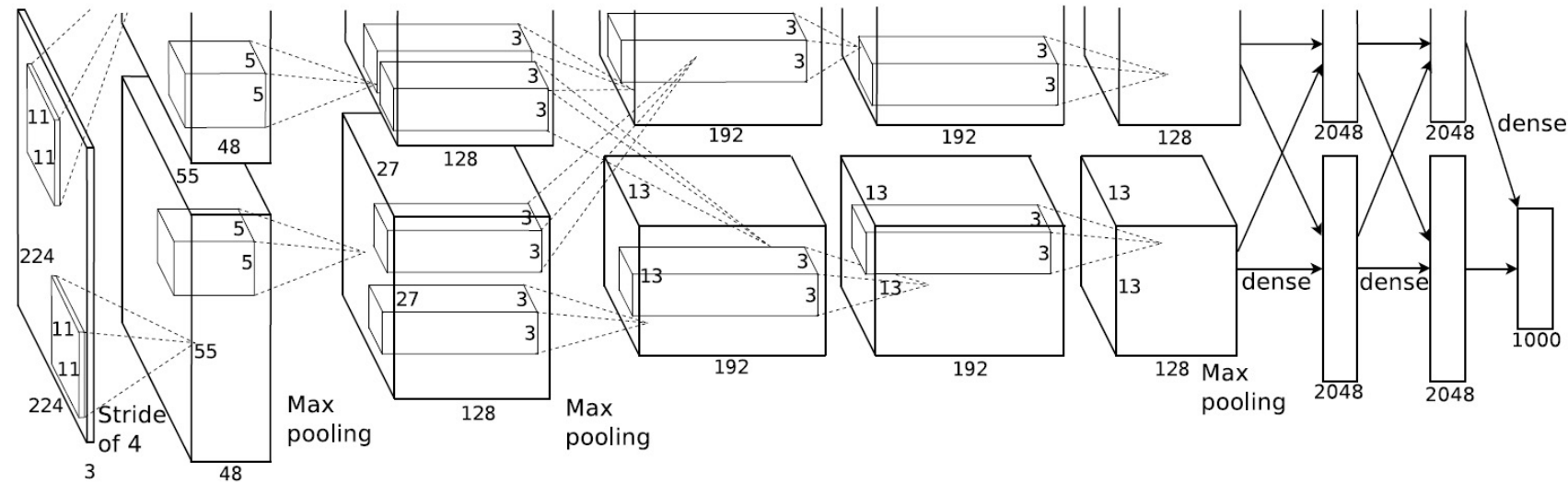
[13x13x256] **Norm3**

[6x6x256] **Pool3**

[4096] **FC1**

[4096] **FC2**

[1000] **FC3** (class score)



AlexNet

- ReLU activation function
 - Previously, most people use sigmoid or tanh functions.
- Data augmentation
 - Randomly crop 224x224 regions from 256x256 original images
 - Horizontal reflection
 - Perturb RGB intensities
- Training on multiple GPUs
 - 2 GTX 580 GPUs, each with only 3GB of memory.
 - Spread the feature maps onto 2 GPUs.

AlexNet

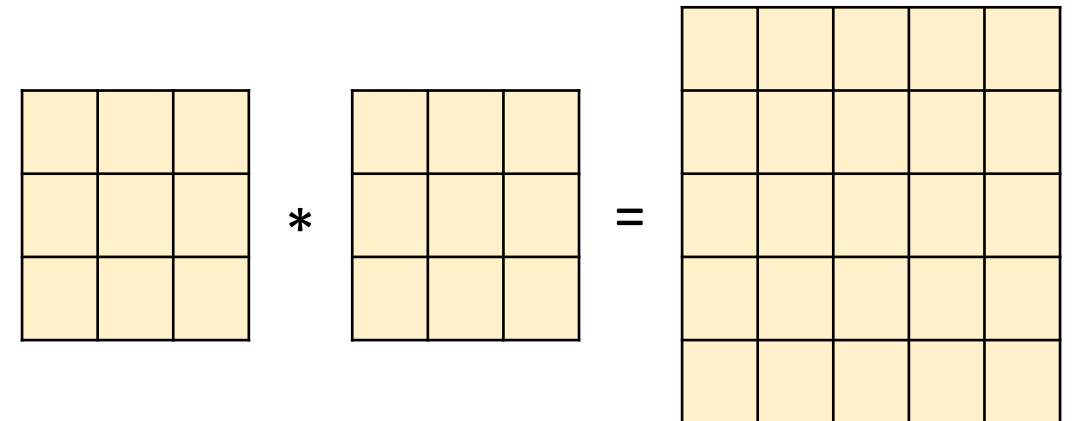
- The main architecture is similar to LeNet.
 - It consists of a number of convolutional layers and pooling layers, followed by fully connected layers at the end.
- But it is deeper.
 - 8 weight layers (with trainable parameters): 5 Conv layers + 3 FC layers
- Research in the following years explores deeper networks and different network architectures.

VGG

- Visual Geometry Group (VGG) at Oxford
- Let us go deeper.
- But how? How do we design the network architecture?

VGG

- Previous research uses 11x11, 7x7 or 5x5 convolutional kernels.
- VGG only uses 3x3 very small kernels.
 - Convolution of several 3x3 kernels is equivalent to a large kernel.
 - For example, convolution of two 3x3 kernels is equivalent to a 5x5 kernel.
 - Convolution of three 3x3 kernels is equivalent to a 7x7 kernel ...
 - It is deeper and saves the number of parameters, e.g. $3 \times (3 \times 3) = 27$ vs $7 \times 7 = 49$.
 - After each 3x3 convolutional layer, we add non-linearity (ReLU). Using three 3x3 convolutional layers adds more non-linearity than a 7x7 layer.



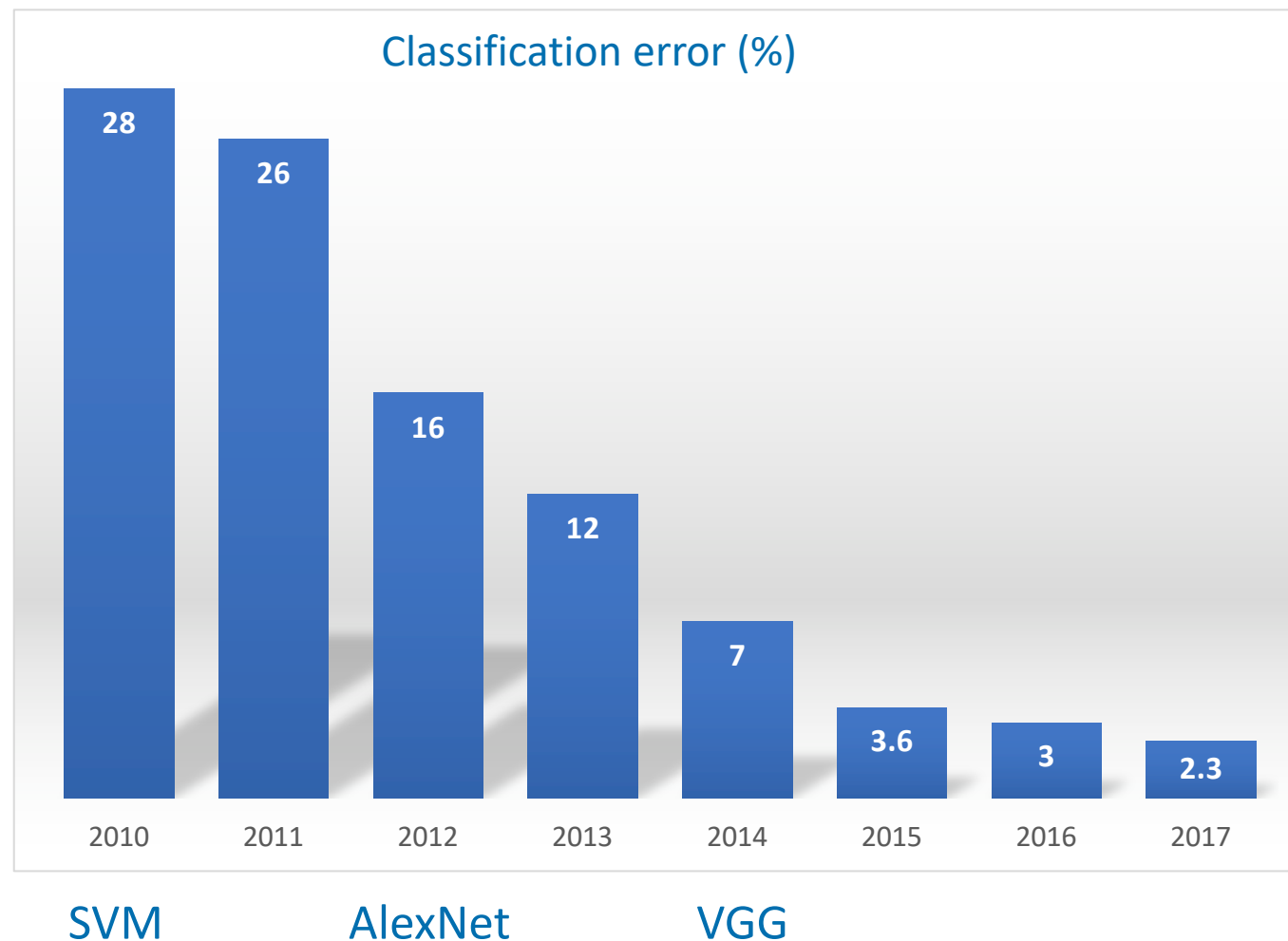
VGG-16

[224x224x3] **Input**
[224x224x64] 3x3 conv1, 64
[224x224x64] 3x3 conv1, 64
[112x112x64] **Pool**
[112x112x128] 3x3 conv2, 128
[112x112x128] 3x3 conv2, 128
[56x56x128] **Pool**
[56x56x256] 3x3 conv3, 256
[56x56x256] 3x3 conv3, 256
[56x56x256] 3x3 conv3, 256
[28x28x256] **Pool**
[28x28x512] 3x3 conv4, 512
[28x28x512] 3x3 conv4, 512
[28x28x512] 3x3 conv4, 512
[14x14x512] **Pool**
[14x14x512] 3x3 conv5, 512
[14x14x512] 3x3 conv5, 512
[14x14x512] 3x3 conv5, 512
[7x7x512] **Pool**
[4096] **FC**, 4096
[4096] **FC**, 4096
[1000] **FC**, 1000

VGG-19

[224x224x3] **Input**
[224x224x64] 3x3 conv1, 64
[224x224x64] 3x3 conv1, 64
[112x112x64] **Pool**
[112x112x128] 3x3 conv2, 128
[112x112x128] 3x3 conv2, 128
[56x56x128] **Pool**
[56x56x256] 3x3 conv3, 256
[56x56x256] 3x3 conv3, 256
[56x56x256] 3x3 conv3, 256
[56x56x256] 3x3 conv3, 256
[28x28x256] **Pool**
[28x28x512] 3x3 conv4, 512
[28x28x512] 3x3 conv4, 512
[28x28x512] 3x3 conv4, 512
[28x28x512] 3x3 conv4, 512
[14x14x512] **Pool**
[14x14x512] 3x3 conv5, 512
[14x14x512] 3x3 conv5, 512
[14x14x512] 3x3 conv5, 512
[14x14x512] 3x3 conv5, 512
[7x7x512] **Pool**
[4096] **FC**, 4096
[4096] **FC**, 4096
[1000] **FC**, 1000

ImageNet classification challenge

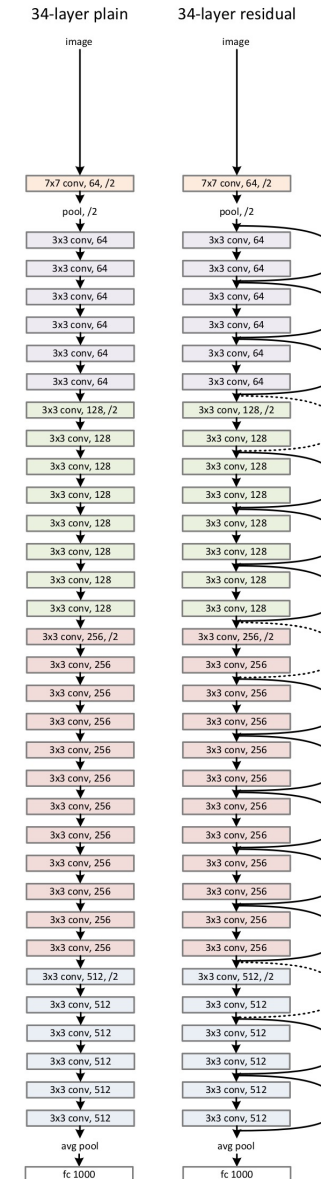


Some other networks

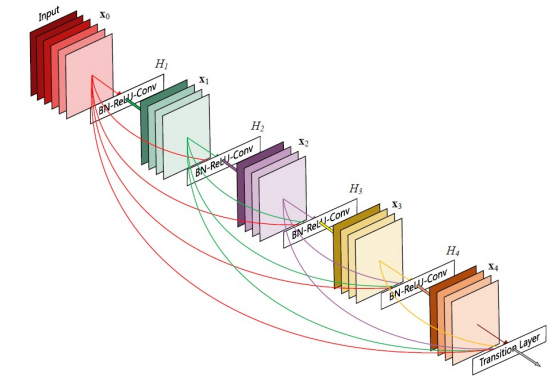
- GoogLeNet (Szegedy, CVPR 2015)
- ResNet (He, CVPR 2016)
- DenseNet (Huang, CVPR 2017)
- SENet (Hu, CVPR 2018)



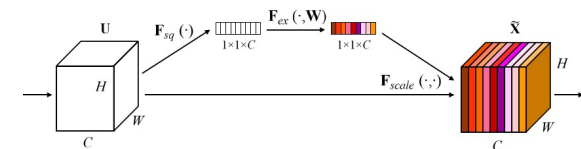
GoogLeNet



ResNet



DenseNet



SENet

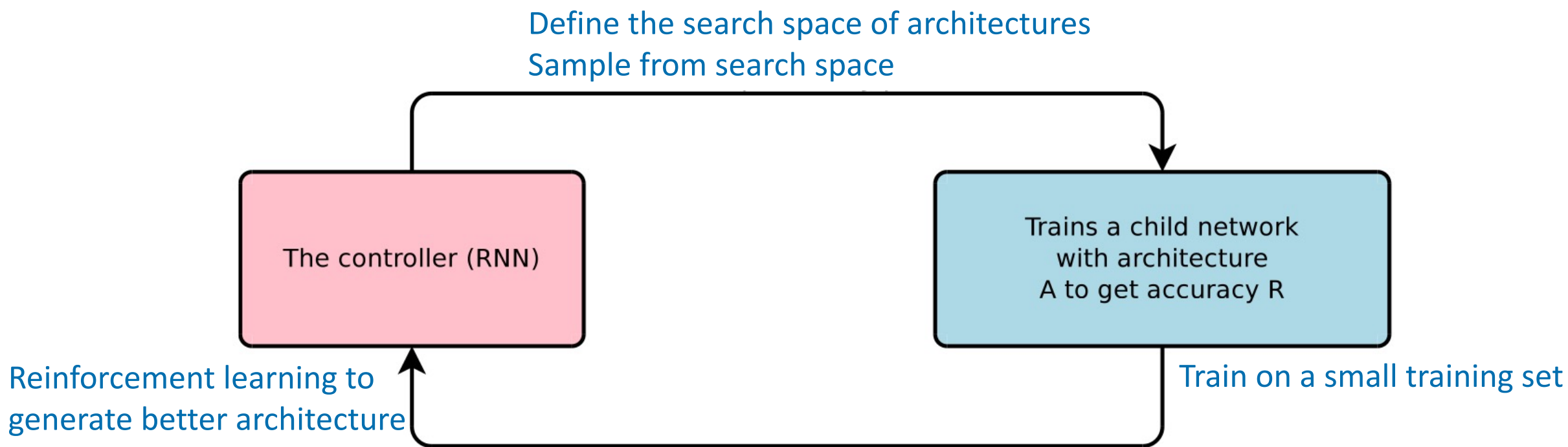
Recent trends

- Network architecture search (NAS) [1]
 - Automate the design of architecture
 - Regard this as an optimisation problem
- Borrowing ideas from natural language processing (NLP) [2]
 - Regard image patches as words
 - Use the Transformer architecture for image classification, which was originally designed for sequence-to-sequence learning in machine translation

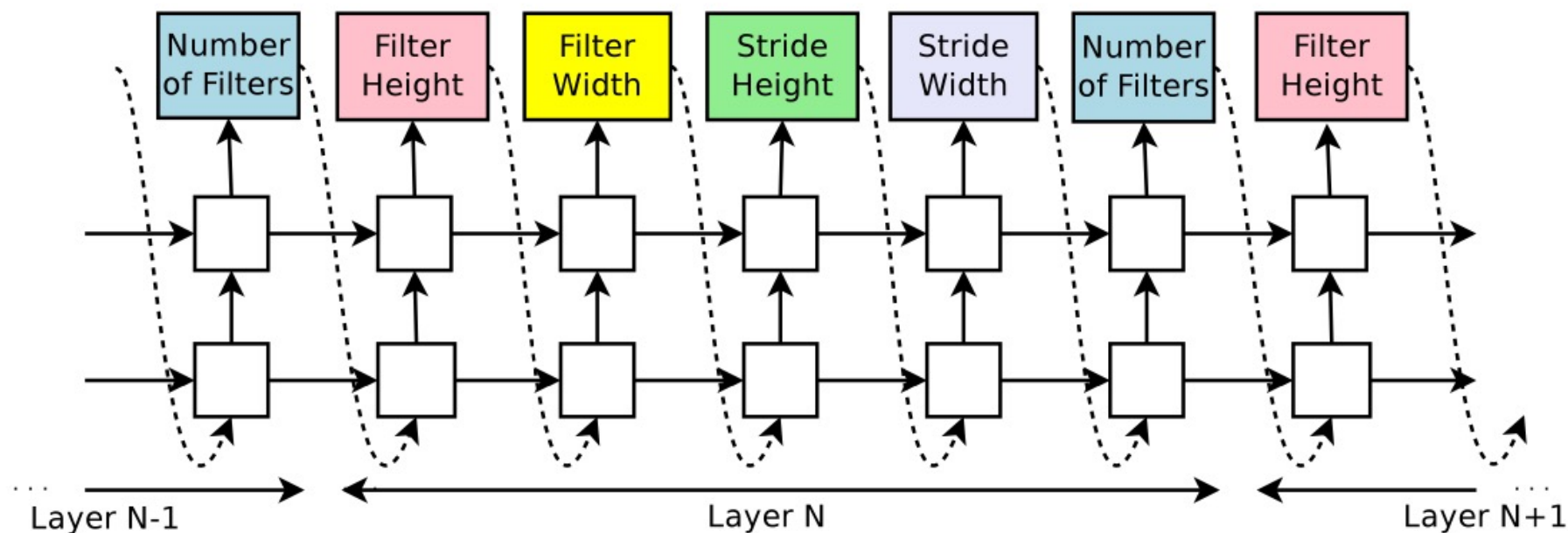
[1] B. Zoph et al. Learning Transferable Architectures for Scalable Image Recognition. CVPR 2018.

[2] A. Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021.

Neural architecture search (NAS)

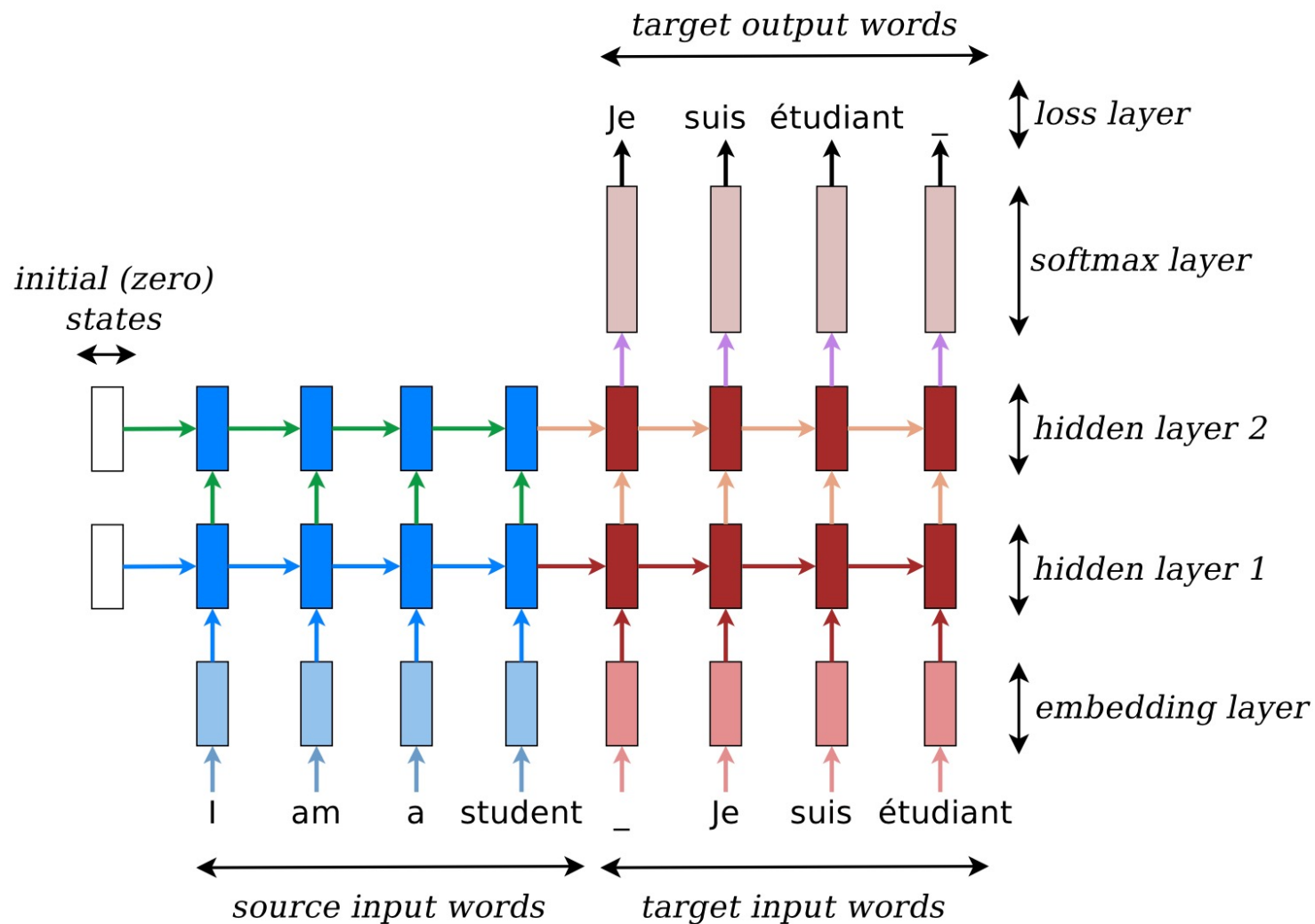


Neural architecture search (NAS)



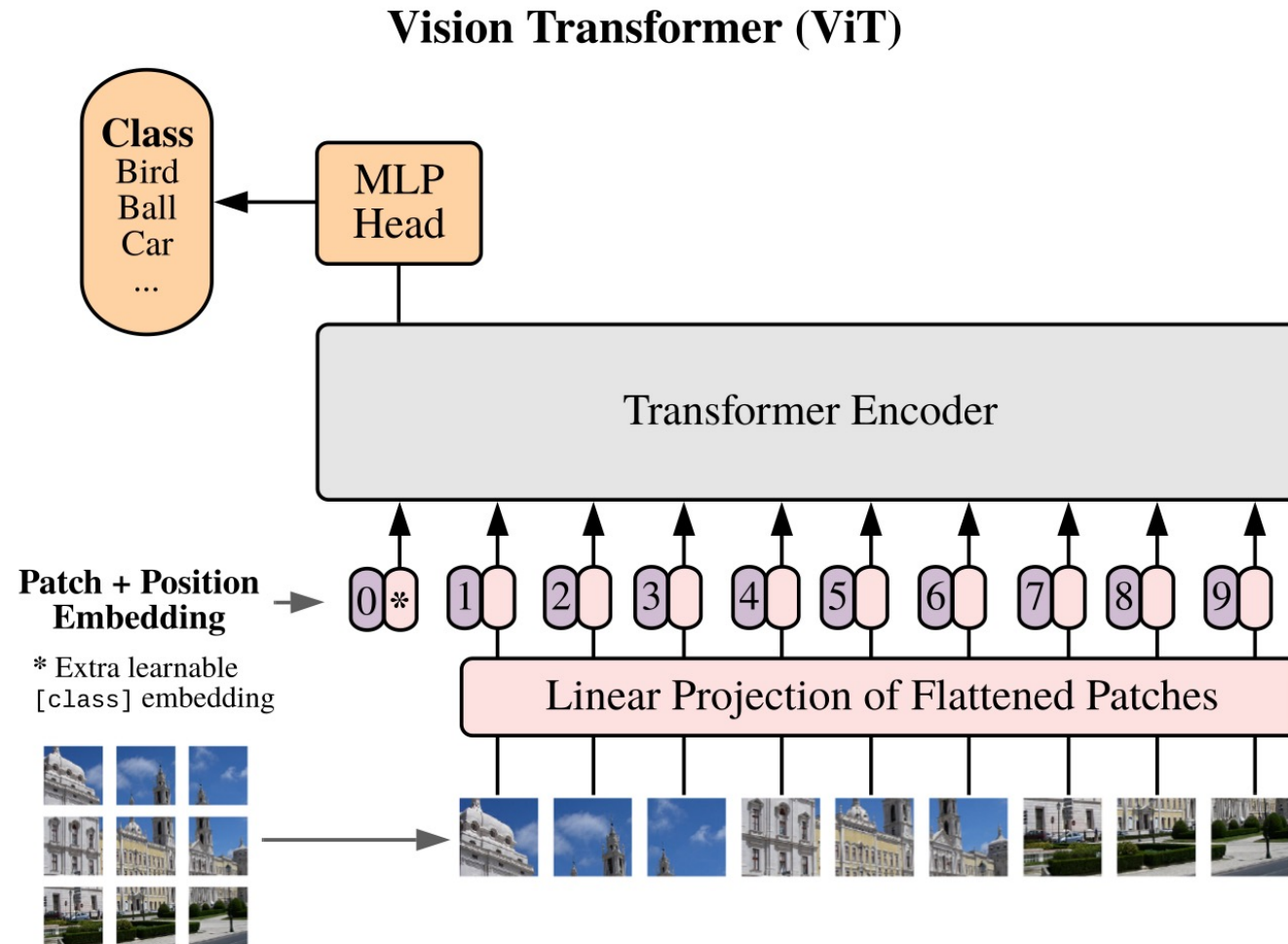
Use a neural network to sample from the search space of network architectures.

Natural language processing



Neural machine translation. Each word is represented as a vector and input to the network.

Vision Transformer (ViT)



An image is split into patches. Each image patch is represented as a vector and input to the network.

Summary

- Convolutional neural networks
 - Building blocks
- Examples
 - LeNet-5 (1998)
 - AlexNet (2012)
 - VGG (2014)
 - ...

References

- Ch. 9, Convolutional Networks. Ian Goodfellow et al. Deep Learning (<https://www.deeplearningbook.org/>).