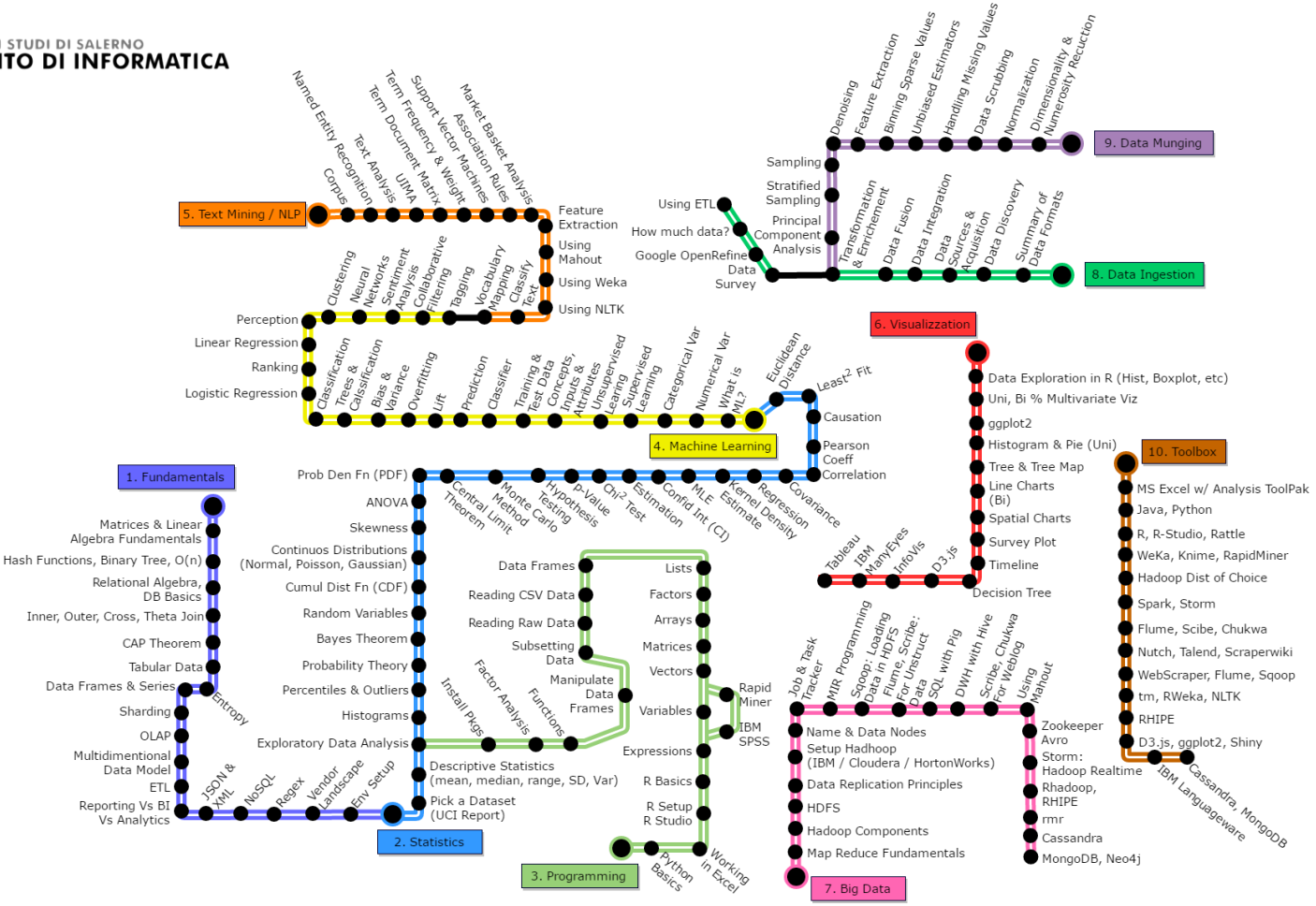
UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA



Fondamenti di Data Science e Machine Learning

Introduction to Convolutional Neural Networks

Aurelien Geron: «Hands on Machine Learning with Scikit Learn and TensorFlow, O'Reilly ed.

Prof. Giuseppe Polese, aa 2024-25

Outline

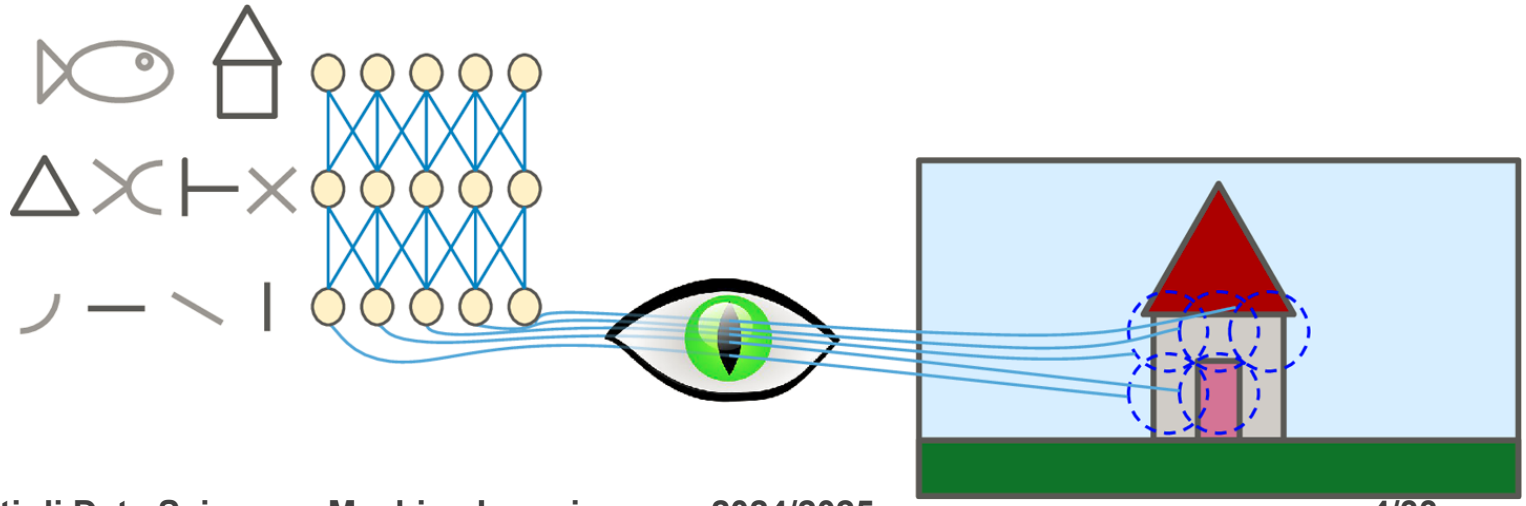
- Convolutional Neural Network
  - Convolutional Layer
  - Pooling Layer
- CNN Architectures
  - LeNet-5
  - AlexNet
  - GoogLeNet
  - ResNet

Convolutions (1)

- In maths and physics:
  - It is a mathematical operation that slides one function over another and measures the integral of their pointwise multiplication
- In neural networks:
  - a function derived from two given functions by element-wise multiplication that expresses how the value and shape of one is modified by the other

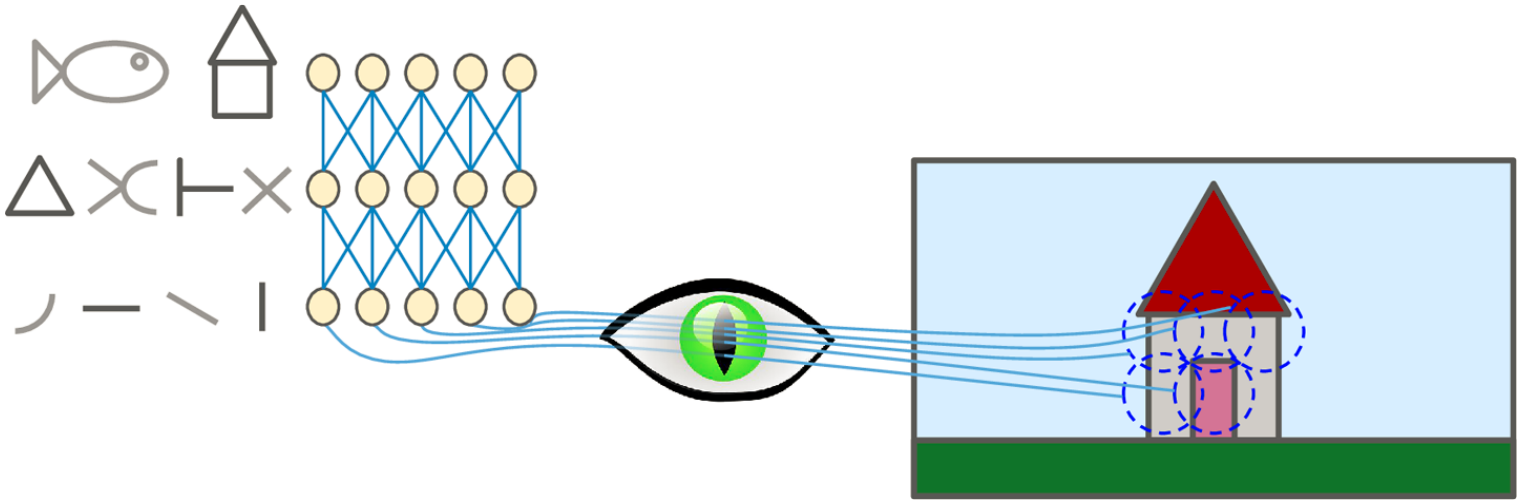
Convolutions (2)

- David H. Hubel and Torsten Wiesel performed a series of experiments on cats (later on Monkeys) in 1958 and 1959
- They showed that many neurons in the visual cortex have a small *local receptive field*, meaning they react only to visual stimuli located in a limited region of the visual field
- *The receptive fields of different neurons may overlap, and together they tile the whole visual field*



Convolutions (3)

- These observations led to the idea that the higher-level neurons are based on the outputs of neighboring lower-level neurons

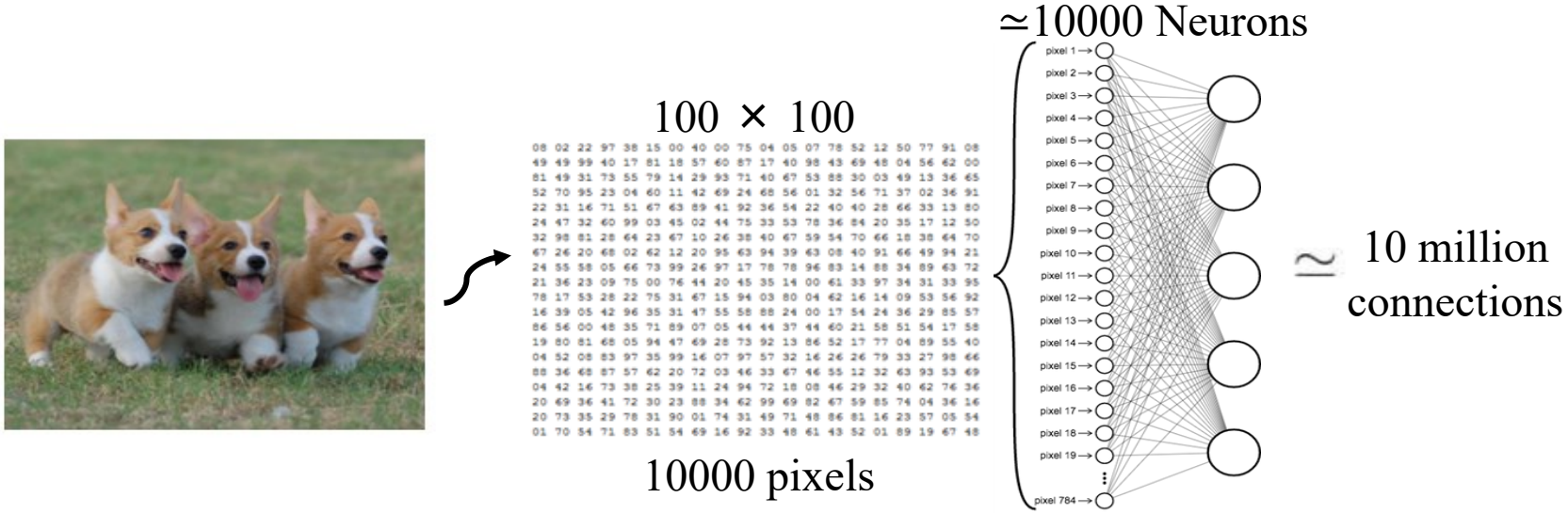


When to Use CNNs?

- Convolutional Neural Networks, or CNNs, were designed to map image data to an output variable
- Use CNNs For:
  - Image data
  - Classification prediction problems
  - Regression prediction problems
- CNNs use relatively little pre-processing compared to other image **classification** algorithms
- The network learns the filters that in traditional algorithms were hand-engineered

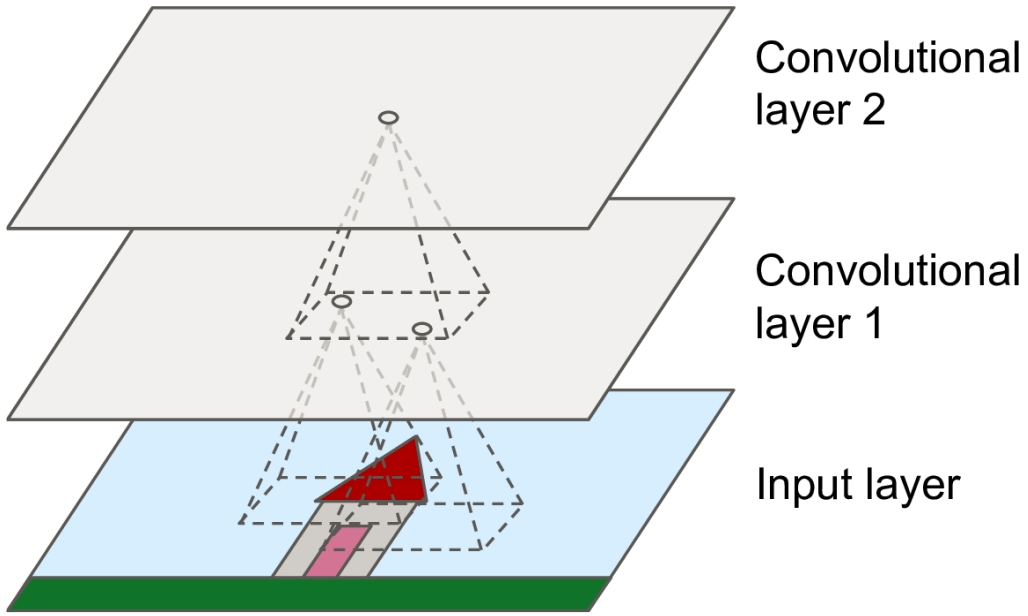
Why to Use CNNs?

- **Why not use a regular deep neural network for image recognition tasks?**
- DNNs work fine for small images
- DNNs breaks down for larger images because of the huge number of parameters it requires



Convolutional Layer (1)

- A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers
- The hidden layers of a CNN typically consist of **convolutional layers**

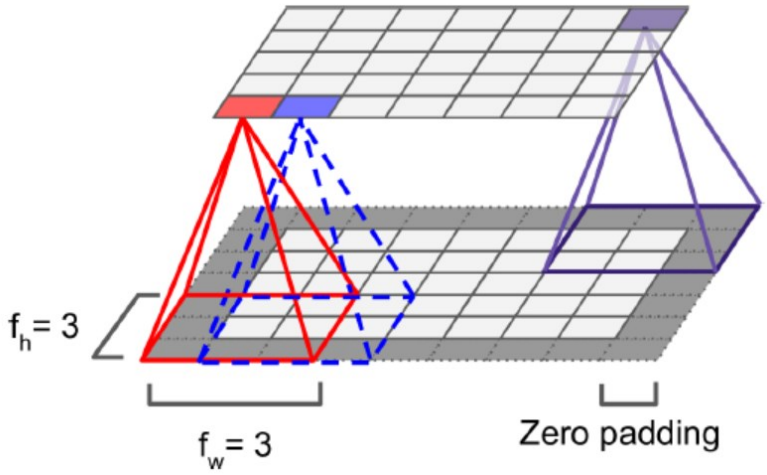


Convolutional Layer (2)

- **Neurons** in the first convolutional layer are connected only to pixels in their receptive fields
- Each layer **is represented in 2D**, which makes it easier to match neurons with their corresponding inputs
- This architecture allows the network:
  - To concentrate on low-level features in the first hidden layer
  - To assemble them into higher-level features in the next hidden layer

Convolutional Layer (3)

- A neuron at row i, column j of a layer is connected to the outputs of the neurons in the previous layer at rows i to $i + f_h - 1$, columns j to $j + f_w - 1$, where f_h and f_w are the height and width of the receptive field
- In order for a layer to have *the same height and width* as the previous one, **zeros** are added around the inputs (*Zero Padding)*

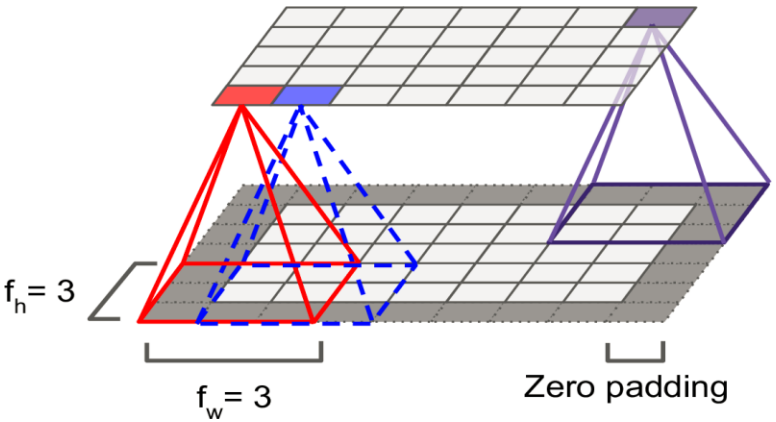


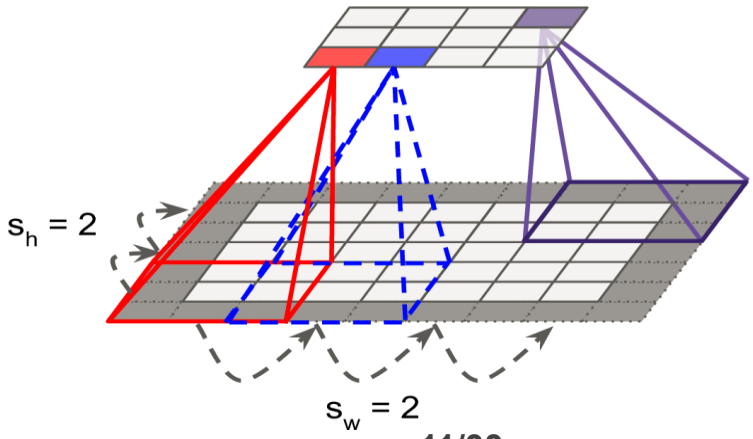
Convolutional Layer (4)

- It is possible to connect a large input layer to a much smaller layer by spacing out receptive fields
- The distance between two consecutive receptive fields is called *stride*
- In the right diagram, a 5 × 7 input layer (plus zero padding) is connected to a 3 × 4 layer, using 3 × 3 receptive fields and a stride of 2 (in the example the stride is the same in both directions, but it does not have to be so)
- A neuron at row i, column j in the upper layer is connected to the outputs of the following neurons in the previous layer:

Rows: $i \times s_h \longrightarrow i \times s_h + f_h - 1$

Columns: $j \times s_w \longrightarrow j \times s_w + f_w - 1$





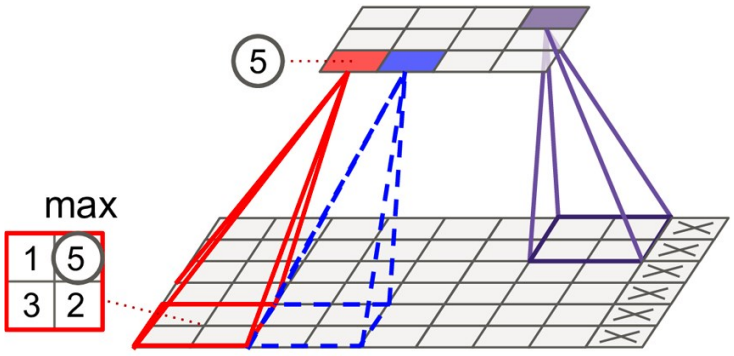
Pooling Layer (1)

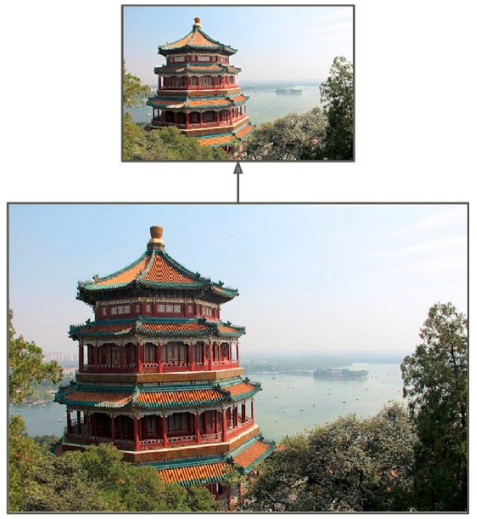
- The goal of the pooling layer is to subsample (i.e., shrink) the input image in order to reduce the computational load, the memory usage, and the number of parameters
- A neuron in a pooling layer is connected to the outputs of a limited number of neurons in the previous layer
- For each neuron, it is necessary to define:
  - Size
  - Stride
  - Padding type
- A pooling neuron has no weights

Pooling Layer (2)

- Example:
  - 2 × 2 pooling kernel
  - A stride of 2
  - No padding
- Only the max input value in each kernel makes it to the next layer, while the other inputs are dropped

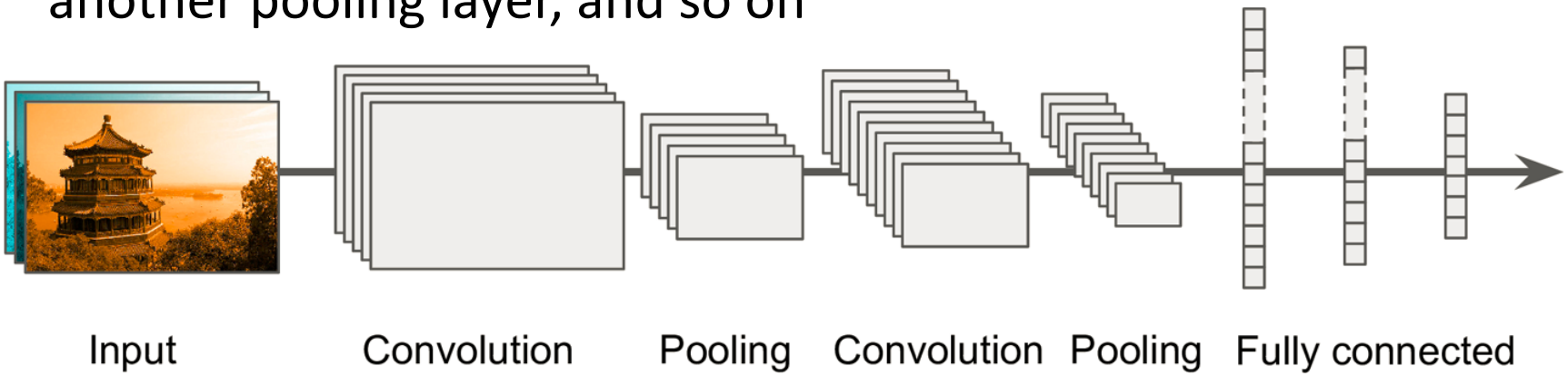
- 75% of the input values





CNN Architectures (1)

- Typical CNN architectures stack a few convolutional layers, then a pooling layer, then another few convolutional layers, then another pooling layer, and so on



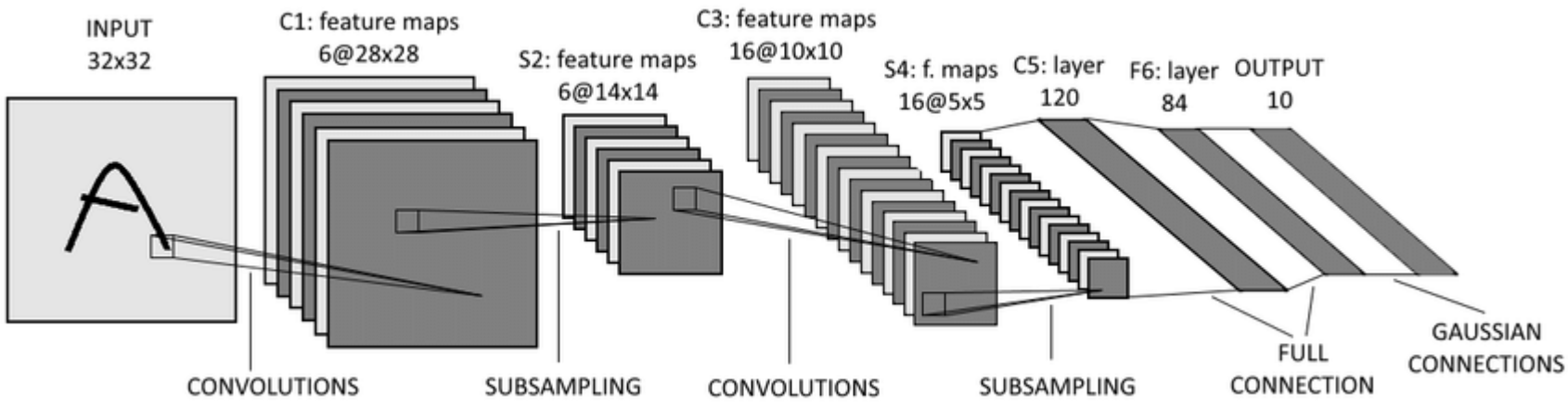
- The image gets smaller and smaller as it progresses through the network
- At the top of the stack there is a regular feedforward neural network (FNN)
- The final layer outputs the prediction

CNN Architectures (2)

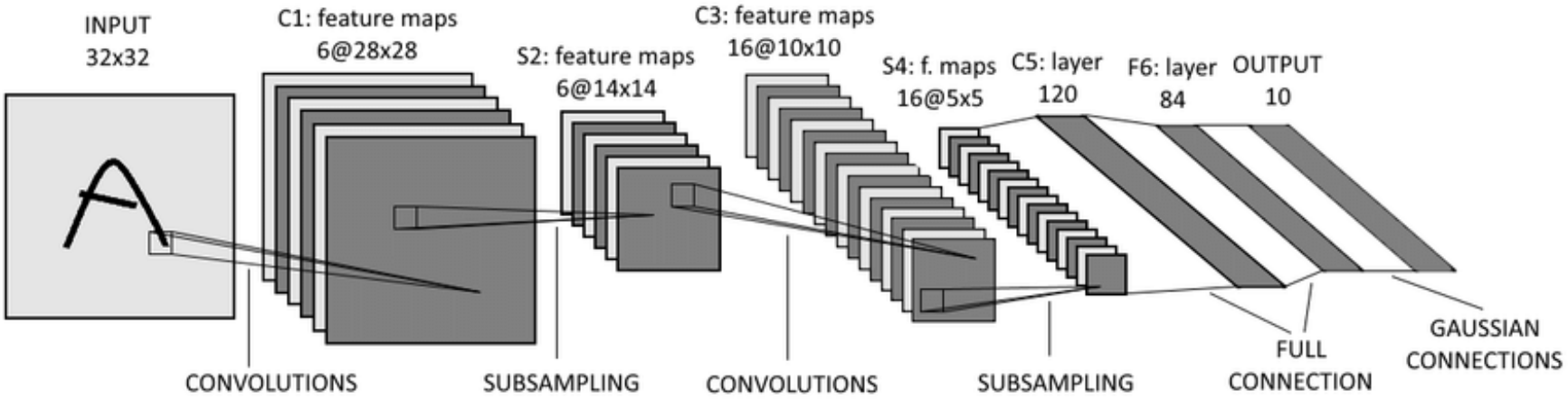
- Variants of this fundamental architecture have been developed
- A good measure to evaluate these architecture is the error rate in competitions such as the ILSVRC ImageNet challenge:
  - The top-five error rate is the number of test images for which the system's top 5 predictions did not include the correct answer
- The images are large (256 pixels high) and there are 1,000 classes, some of which are really subtle (try distinguishing 120 dog breeds)
- The best known networks are:
  - LeNet-5 architecture (1998)
  - AlexNet (2012)
  - GoogLeNet (2014)
  - ResNet (2015)

LeNet-5 (1)

- LeNet-5 is perhaps the most widely known CNN architecture
- It is widely used for handwritten digit recognition (MNIST)



LeNet-5 (2)



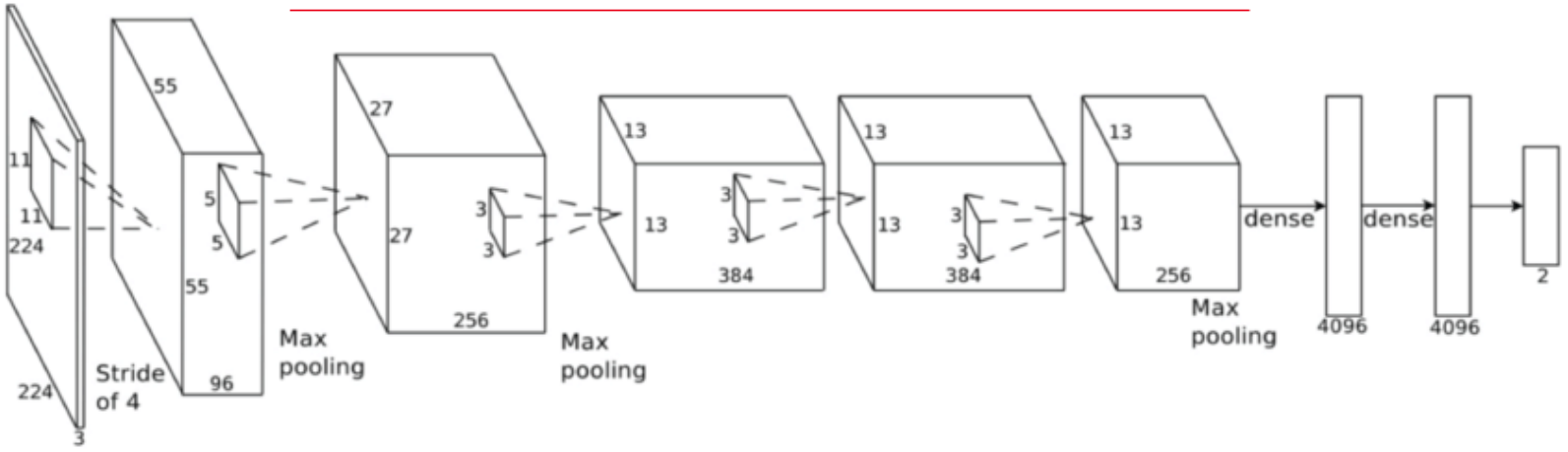
Characteristics	The Gap
• Repeat of Convolution – Pooling – Non Linearity • Average pooling • Sigmoid activation for the intermediate layer • tanh activation at F6 • 5x5 Convolution filter • 7 layers and less than 1M parameters	• Slow to train • Hard to train (Neurons dies quickly) • Lack of data

AlexNet

- The *AlexNet* CNN architecture won the 2012 ImageNet ILSVRC challenge by a large margin:
  - it achieved 17% top-5 error rate while the second best achieved only 26%!
- It is quite similar to LeNet-5:
  - Much larger
  - Much deeper
- It was the first to stack convolutional layers directly on top of each other, instead of stacking a pooling layer on top of each convolutional layer

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	SAME	ReLU
In	Input	3 (RGB)	224×224	–	–	–	–



AlexNet - Regularization

- To reduce overfitting, the authors used two regularization techniques:
  - They applied dropout (with a 50% dropout rate) during training to the outputs of layers F8 and F9;
  - They performed data augmentation by randomly shifting the training images by various offsets, flipping them horizontally, and changing the lighting conditions

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	SAME	ReLU
In	Input	3 (RGB)	224×224	–	–	–	–

AlexNet - Normalization

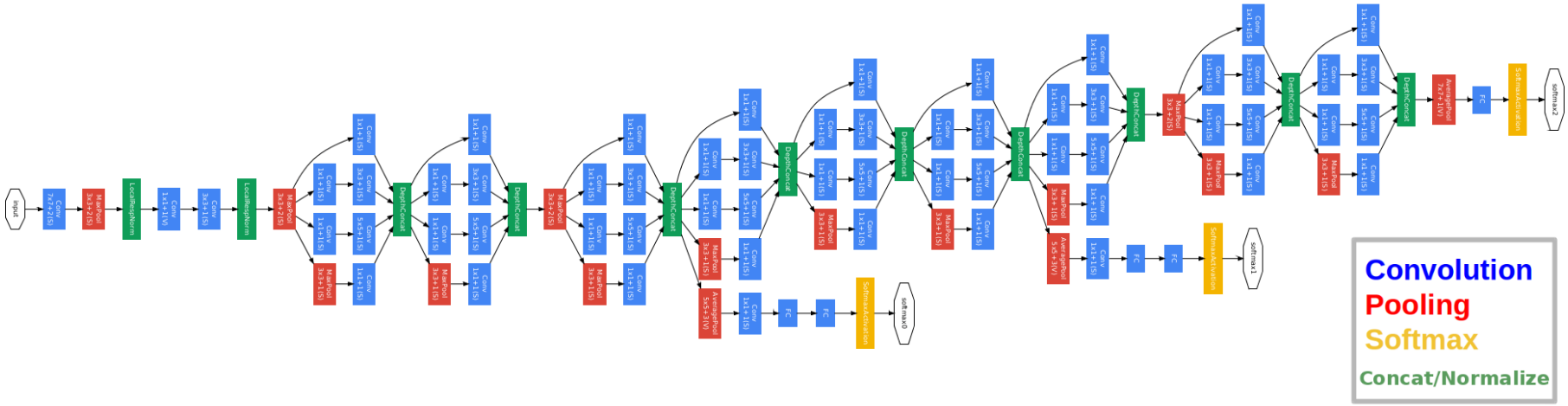
- AlexNet also uses a competitive normalization step after the ReLU step of layers C1 and C3: ***local response normalization***

$$b_i = a_i \left(k + \alpha \sum_{j=j_{\text{low}}}^{j_{\text{high}}} a_j^2 \right)^{-\beta} \quad \text{with} \quad \begin{cases} j_{\text{high}} = \min\left(i + \frac{r}{2}, f_n - 1\right) \\ j_{\text{low}} = \max\left(0, i - \frac{r}{2}\right) \end{cases}$$

  - b_i is the normalized output of the neuron located in feature map i, at some row u and column v
  - a_i is the activation of that neuron after the ReLU step
  - k, α, β, and r are hyperparameters: k is called the *bias*, and r is called the *depth radius*
  - f_n is the number of feature maps

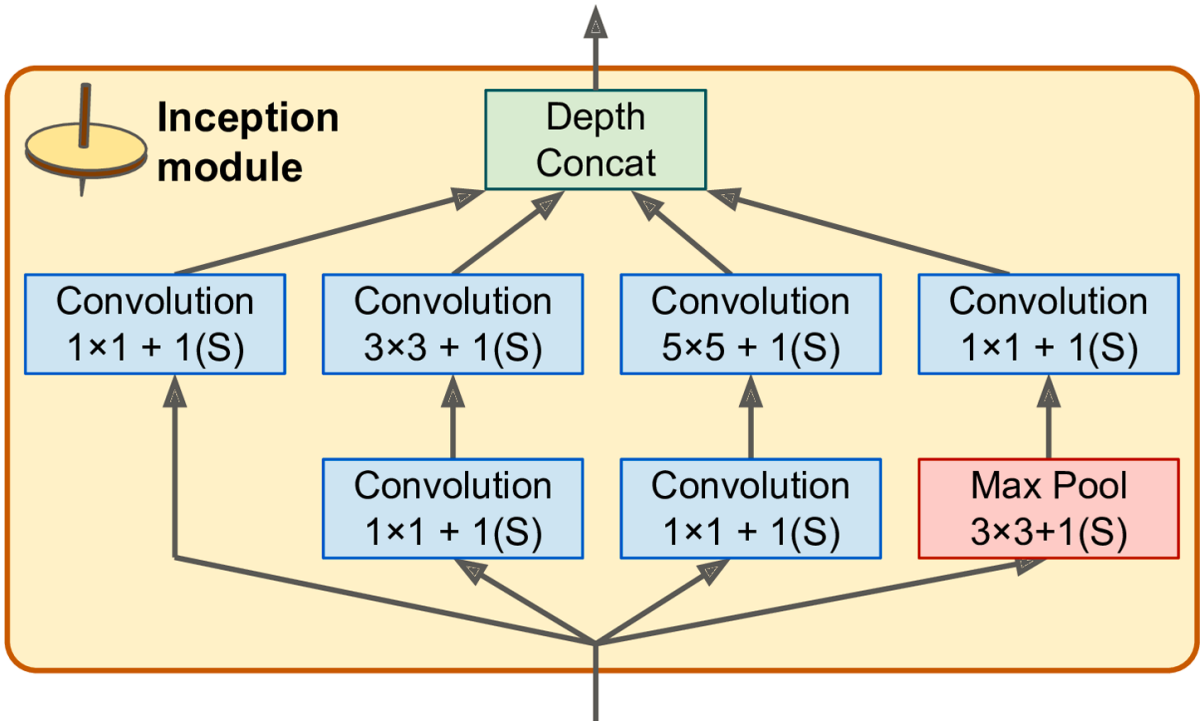
GoogLeNet

- The GoogLeNet architecture was developed from Google Research, and it won the ILSVRC 2014 challenge by pushing the top-5 **error rate below 7%**
- GoogLeNet uses sub-networks called ***inception modules*** that increase the performance and reduce the number of parameters



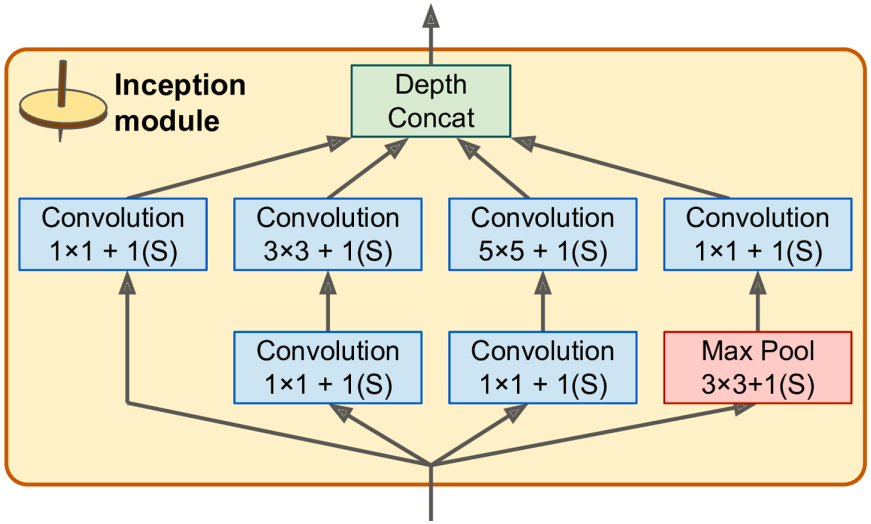
Inception module (1)

- GoogLeNet has 10 times fewer parameters than AlexNet
- Example: Layer "3 × 3 + 2(S)" means that the layer uses a 3 × 3 kernel, stride 2, and SAME padding



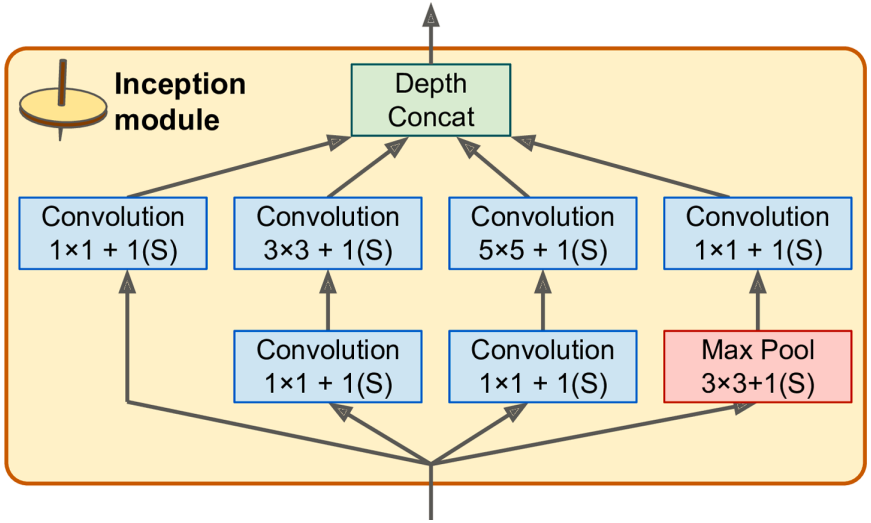
Inception module (2)

- The input signal first copied and fed to 4 different layers
- Uses different kernel sizes (1×1, 3×3, and 5×5) of the convolutional layers
- Computational expensive
  - Capture dense structure to 1x1, more spread out structure to 3x3, 5x5
  - Use 1x1 convolutional layer to reduce dimension



Inception module (3)

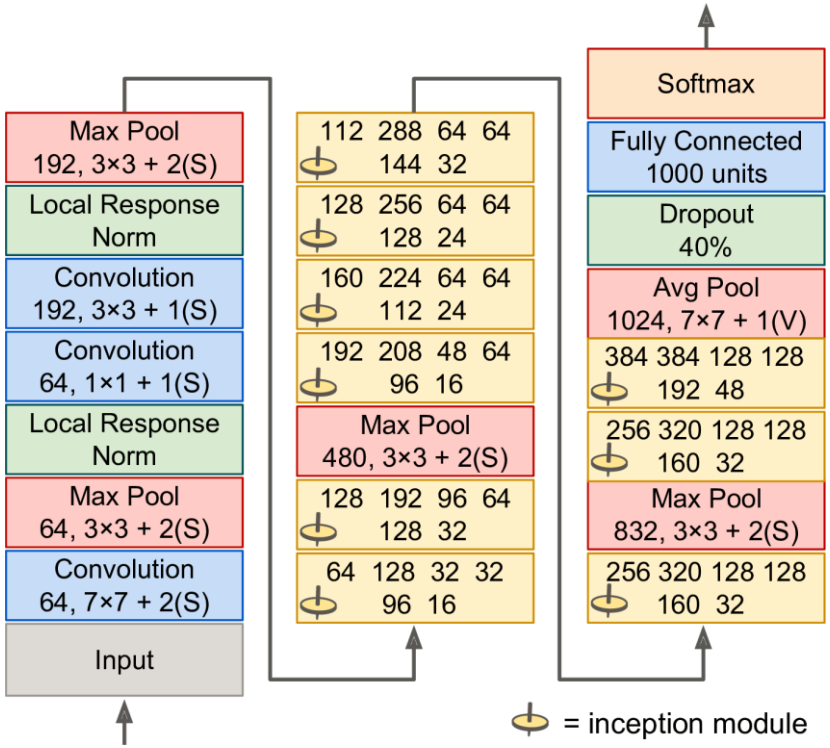
- Every single layer uses a stride of 1 and SAME padding, so their outputs all have the same height and width as their inputs
- This makes it possible to concatenate all the outputs along the depth dimension in the final *depth concat layer*



- The number of convolutional kernels for each convolutional layer is a **hyperparameter**. This means that you have six more hyperparameters to tweak for every inception layer you add

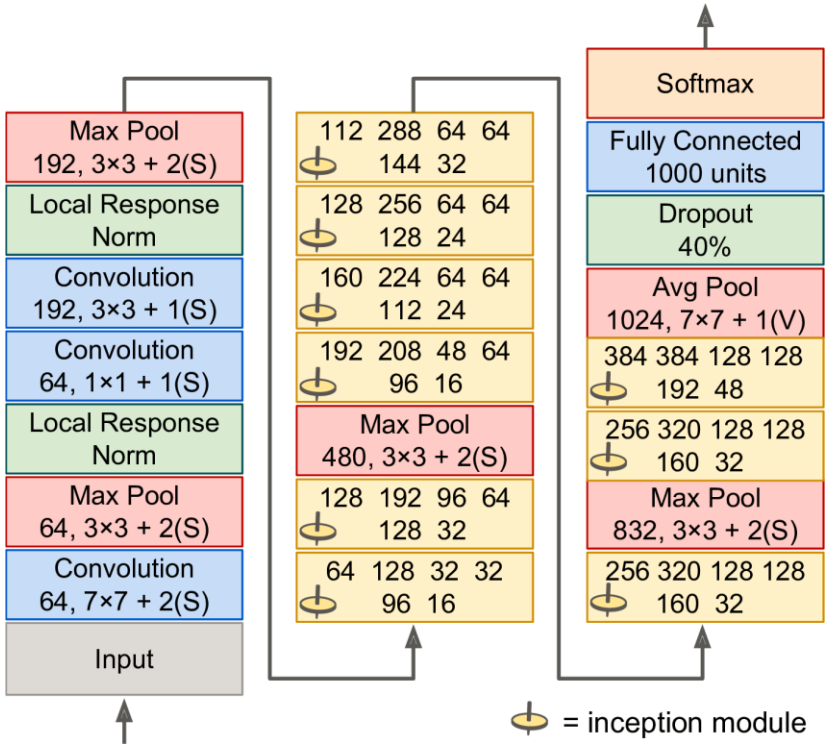
GoogLeNet Architecture (1)

- GoogLeNet architecture is one tall stack that contain three layers each
- The six numbers in the inception modules represent the number of **feature maps** output by each convolutional layer



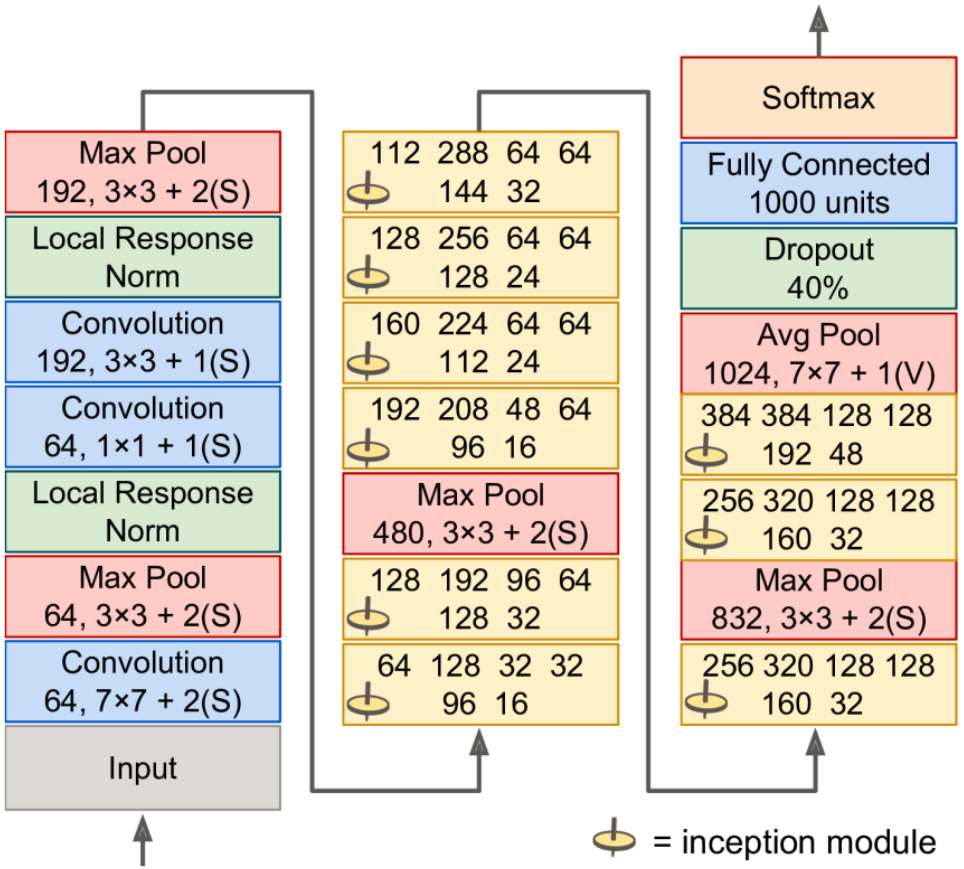
GoogLeNet Architecture (2)

- GoogLeNet architecture is one tall stack that contain three layers each
- The six numbers in the inception modules represent the number of **feature maps** output by each convolutional layer
- All the convolutional layers use the **ReLU** activation function



GoogLeNet Architecture (3)

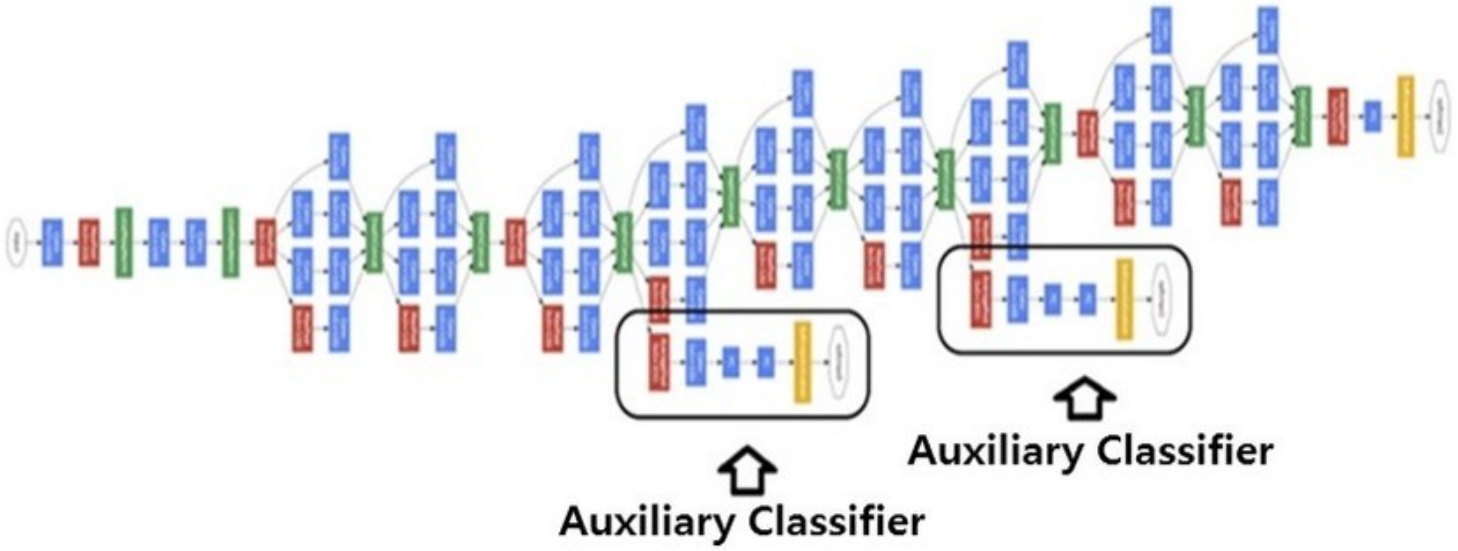
- The first two layers divide the image's height and width by 4 to reduce the computational load
- **Local Response Norm** layers ensures that the previous layers learn a wide variety of features
- **Max pooling** layer reduces the image height and width by 2, again to speed up computations
- **Average pooling layer** uses a kernel size of the feature maps with VALID padding, outputting 1 × 1 feature maps



- **Dropout layer** is used for regularization
- **Softmax layer** is used to output estimated class probabilities.

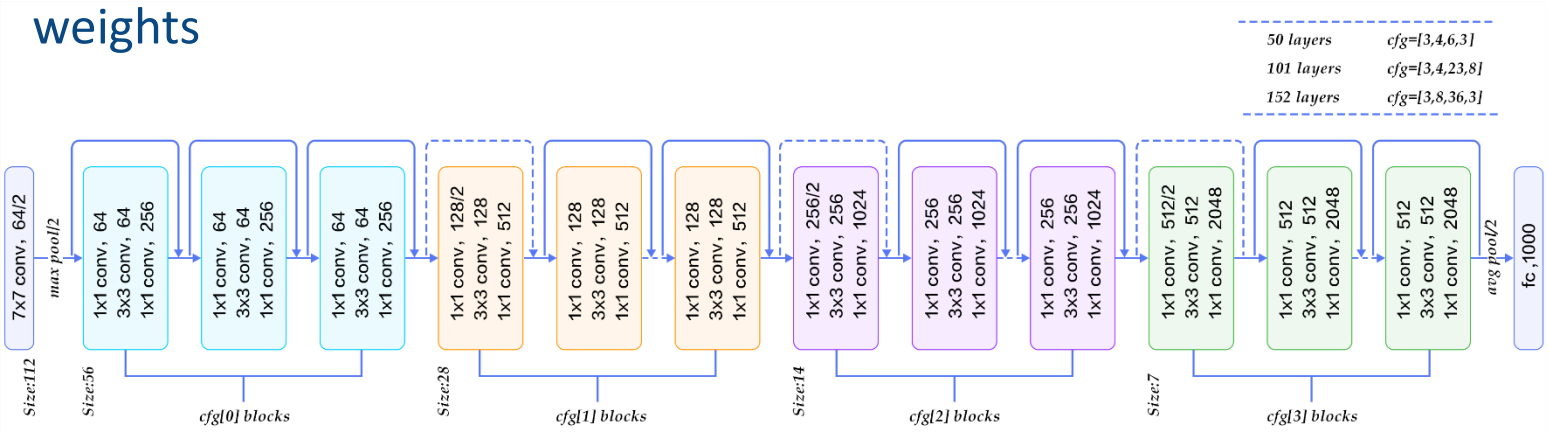
Auxiliary Classifier

- The original GoogLeNet architecture also included two auxiliary classifiers plugged on top of the third and sixth inception modules
  - To prevent the middle part of the network from "dying out", the authors of GoogLeNet have introduced two auxiliary classifiers
  - They essentially applied softmax to the outputs of two of the inception modules, and computed an auxiliary loss over the same labels



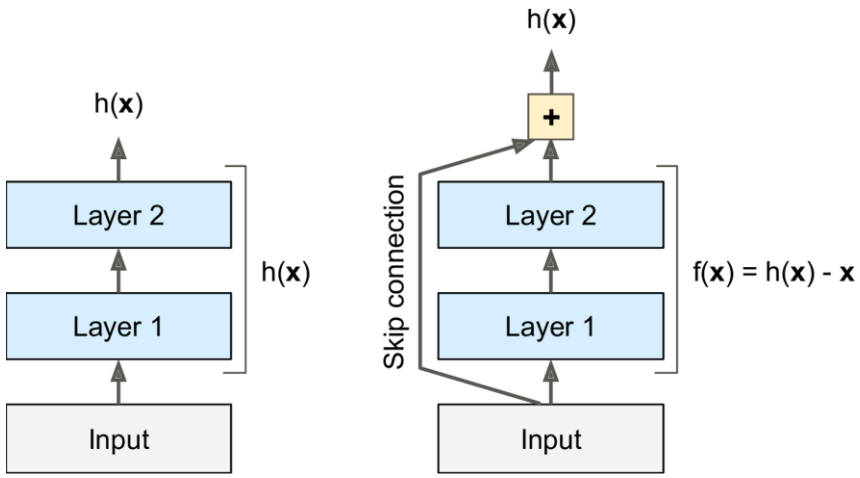
Residual Neural Network (ResNet)

- Wide network is good for memorization, but not so good for generalization
- Multiple layers can learn features at various levels of abstraction
- Deep layers can provide features with global semantic meaning and abstract details (relations of relations … of relations of objects), while using only small kernels
- Problem of going deeper?
  - **Vanishing Gradient**: Most neurons will die during back propagating the weights



Residual learning

- Introduce *shortcut connections* (exists in prior literature in various forms)
- Key invention is to *skip 2 layers*
  - Skipping single layer didn't give much improvement for some reason
- Thanks to skip connections, the signal can easily make its way across the whole network



ResNet

- The deep residual network can be seen as a stack of residual units, where each residual unit is a small neural network with a skip connection