Νευρο-Ασαφής Υπολογιστική
Neuro-Fuzzy Computing

Διδάσκων –
Δημήτριος Κατσαρός

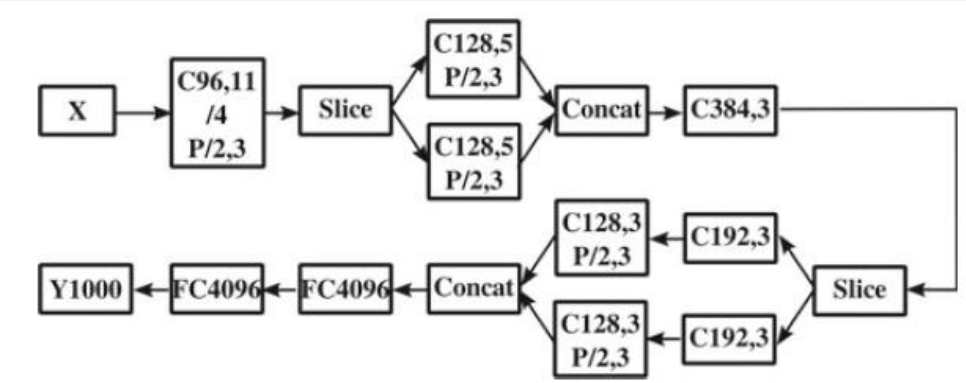
@ Τμ. ΗΜΜΥ
Πανεπιστήμιο Θεσσαλίας

Διάλεξη 12η

Convolutional Neural Networks architecture(s)

CNN architecture: DAG representation

- Although there is no golden rule in designing a ConvNet, there are a few rule-of thumbs that can be found in many successful architectures
- A ConvNet typically consists of several convolution-pooling layers followed by a few fully connected layers
- Also, the last layer is always the output layer
- From another perspective, a ConvNet is a directed acyclic graph (DAG) with one leaf node
  - In this DAG, each node represents a layer and edges show the connection between layers
- A convenient way of designing a ConvNet is to use a DAG diagram like the one shown below:



CNN architecture: DAG representation

- One can define other nodes or combine several nodes into one node. You can design any DAG to represent a ConvNet
- However, two rules have to be followed
  - First, there is always one leaf node in a ConvNet which represents the classification layer or the loss function. That does not make sense to have more than one classification layer in a ConvNet
  - Second, inputs of a node must have the same spatial dimension. The exception could be the concatenation node where you can also concatenate the inputs spatially
  - As long as these two rules are observed in the DAG, the architecture is valid and correct
  - Also, all operations represented by nodes in the DAG must be differentiable so that the backpropagation algorithm can be applied on the graph

CNN architecture: Rules of thumb

- As the first rule of thumb, remember to always compute the size of feature maps for each node in the DAG
  - Usually, nodes that are connected to fully connected layers have spatial size less than 8×8. Common sizes are 2×2, 3×3, and 4×4
  - However, the channels (third dimension) of the nodes connecting to fully connected layer could be any arbitrary size
- The second rule of thumb is that the number of feature maps, usually, has a direct relation with depth of each node in the DAG
  - That means we start with small number of feature maps in early layers and the number of feature maps increases as they the depth of nodes increases
  - However, some flat architectures have been also proposed in literature where all layers have the same number of feature maps or they have a repetitive pattern

CNN architecture: Rules of thumb

- The third rule of thumb is that state-of-the-art ConvNets commonly use convolution filters of size 3×3, 5×5, and 7×7
  - AlexNet is the only one that has utilized 11×11 convolution filters
- The fourth rule of thumb is activation functions usually come immediately after a convolution layer
  - A few works put the activation function after the pooling layer
- As the fifth rule of thumb remember that while putting several convolution layers consecutively makes sense, it is not common to add two or more consecutive activation function layers
- The sixth rule of thumb is to use an activation function from the family of ReLU function (ReLU, Leaky ReLU, PReLU, ELU or Noisy ReLU)
  - Also, always compute the number of trainable parameters of your ConvNet. If you do not have plenty of data and you design a ConvNet with millions of parameters, the ConvNet might not generalize well on the test set

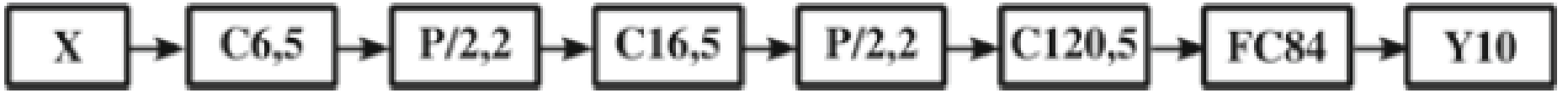
CNN architecture: Software libraries

- Software libraries
  - Theano (deeplearning.net/software/theano/)
  - Lasagne (lasagne.readthedocs.io/en/latest/)
  - TensorFlow (www.tensorflow.org/)
  - Keras (keras.io/)
  - Torch (torch.ch/)
  - cuDNN (developer.nvidia.com/cudnn)
  - mxnet (mxnet.io)
  - Caffe (caffe.berkeleyvision.org/)
- Architectures
  - https://arxiv.org/abs/1901.06032

CNN visualization

Number of parameters calculation in Convolutional Neural Networks

LeNet architecture



LeNet (1989) - proposed for recognizing handwritten digits

- In this DAG, *Ca,b* shows a convolution layer with *a* filters of size b × b and the phrase */a* in any nodes shows the stride of that operation. *P/a,b* denotes a pooling operation with stride *a* and size *b*, *FCa* shows a fully connected layer with *a* neurons, *Ya* shows the output layer with *a* neurons
- This ConvNet that consists of four convolution-pooling layers
- The input of the ConvNet is a single-channel 32 × 32 image
- The last pooling layer (S4) is connected to the fully connected layer C5
- The convolution layer C1 contains six filters of size 5 × 5. Convolving a 32 × 32 image with these filters produces six feature maps of size 28 × 28. Since the input of the network is a single-channel image, the convolution filter in C1 are actually 5 × 5 × 1 filters

LeNet architecture

- The convolution layer C1 is followed by the pooling layer S2 with stride 2. Thus, the output of S2 is six feature maps of size 14×14 which collectively show a six-channel input
- Then, 16 filters of size 5×5 are applied on the six-channel image in the convolution layer C3. In fact, the size of convolution filters in C3 is $5 \times 5 \times 6$. As the result, the output of C3 will be 16 images of size 10×10 which, together, show a 16-channel input
- Next, the layer S4 applies a pooling operation with stride 2 and produces 16 images of size 5×5
- The layer C5 is a fully connected layer in which every neuron in C5 is connected to all the neuron in S4. In other words, every neuron in C5 is connected to $16 \times 56 \times 5 = 400$ neurons in S4. From another perspective, C5 can be seen as a convolution layer with 120 filters of size 5×5
- Likewise, S6 is also a fully connected layer that is connected to S5
- Finally, the classification layer is a radial basis function layer where the inputs of the radial basis function are 84 dimensional vectors. However, for the purpose of our discussion, we consider the classification layer a fully connected layer composed of 10 neurons (one neuron for each digit)

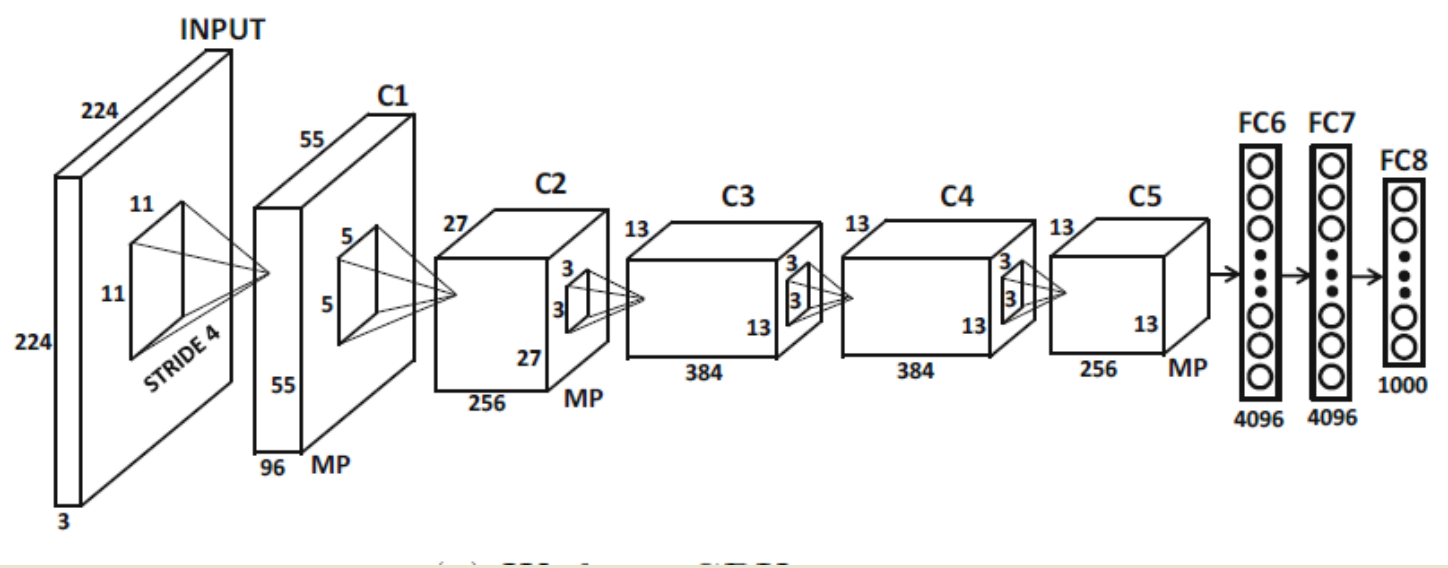
LeNet architecture

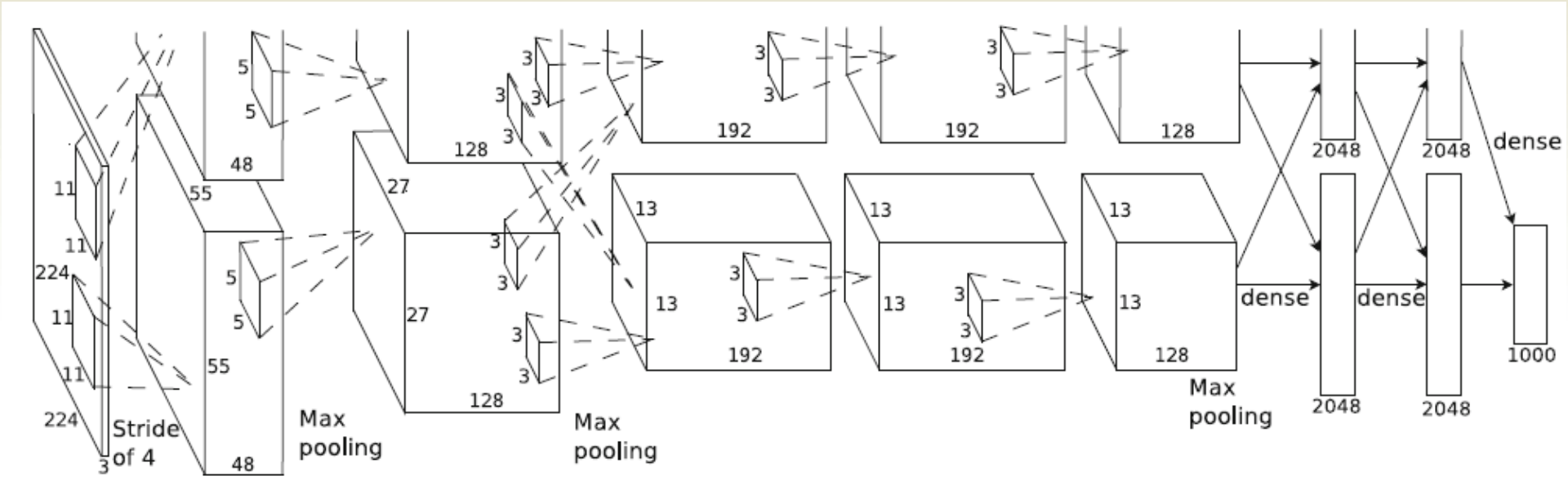
- The pooling operation in this particular ConvNet is not the max-pooling or average-pooling operations
- Instead, it sums the four inputs and divides them by the trainable parameter a and adds the trainable bias b to this result
- Also, the activation functions are applied after the pooling layer and there is no activation function after the convolution layers
- In this ConvNet, the sigmoid activation functions are used

LeNet architecture: #params calculation

- The first layer consists of six filters of size $5 \times 5 \times 1$. Assuming that each filter has also a bias term, C1 is formulated using $6 \times 5 \times 5 \times 1 + 6 = 156$ trainable parameters
- Then, (in this particular ConvNet) each pooling unit is formulated using two parameters. Hence, S2 contains 12 trainable parameters
- Then, taking into account the fact that C3 is composed of 16 filters of size $5 \times 5 \times 6$, it will contain $16 \times 5 \times 5 \times 6 + 16 = 2416$ trainable parameters
- S4 will also contain 32 parameters since each pooling unit is formulated using two parameters
- In the case of C5, it consists of $120 \times 5 \times 5 \times 16 + 120 = 48120$ parameters
- Similarly, F6 contains $84 \times 120 + 84 = 10164$ trainable parameters and the output includes $10 \times 84 + 10 = 850$ trainable parameters
- Therefore, the LeNet-5 ConvNet requires training $156 + 12 + 2416 + 32 + 48120 + 10164 + 850 = 61750$ parameters

AlexNet architecture (without/with GPU)





AlexNet architecture

- AlexNet starts with $224 \times 224 \times 3$ images and uses 96 filters of size $11 \times 11 \times 3$ in the first layer. A stride of 4 is used
- This results in a first layer of size $55 \times 55 \times 96$
- After the first layer has been computed, a max-pooling layer is used
  - The ReLU activation function was applied after each convolutional layer, which was followed by response normalization and max-pooling
- The second convolutional layer uses the response-normalized and pooled output of the first convolutional layer and filters it with 256 filters of size $5 \times 5 \times 96$

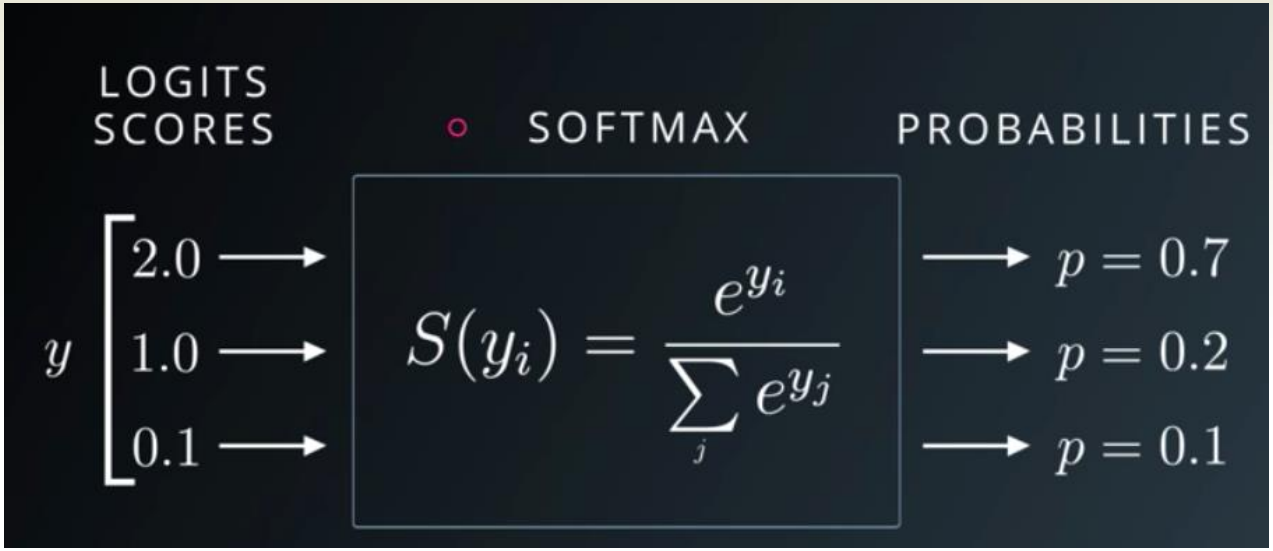
AlexNet architecture

- No intervening pooling or normalization layers are present in the third, fourth, or fifth convolutional layers
- The sizes of the filters of the third, fourth, and fifth convolutional layers are $3 \times 3 \times 256$ (with 384 filters), $3 \times 3 \times 384$ (with 384 filters), and $3 \times 3 \times 384$ (with 256 filters)
- All max-pooling layers used 3×3 filters at stride 2. Therefore, there was some overlap among the pools
- The fully connected layers have 4096 neurons
- The final set of 4096 activations can be treated as a 4096-dimensional representation of the image
- The final layer of AlexNet uses a 1000-way softmax in order to perform the classification

The Softmax function

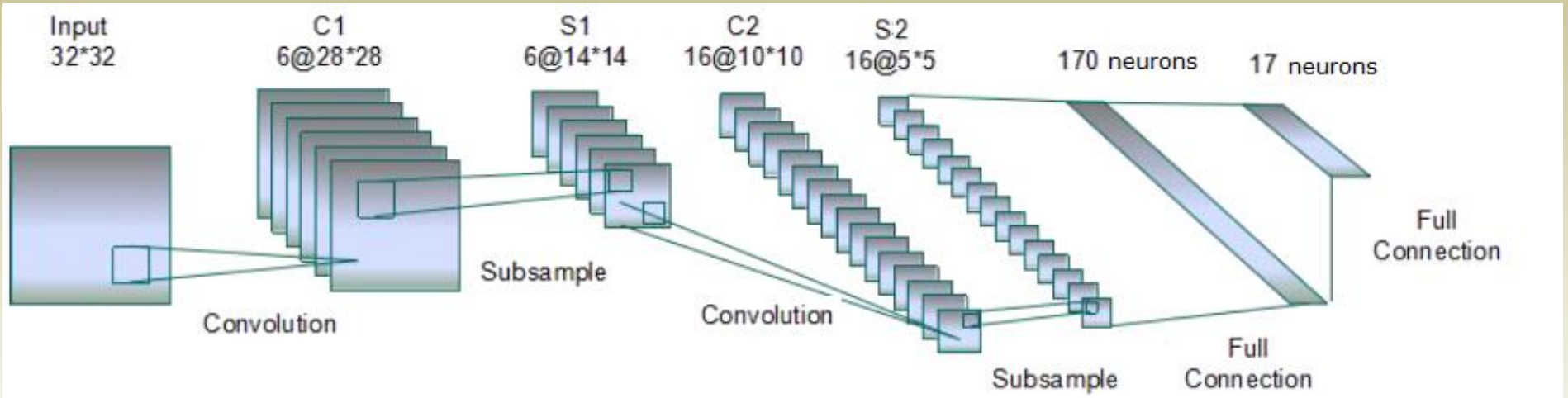
- *Softmax* function, is an activation function that turns numbers aka logits into probabilities that sum to one
- Softmax function outputs a vector that represents the probability distributions of a list of potential outcomes

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$



Exercise

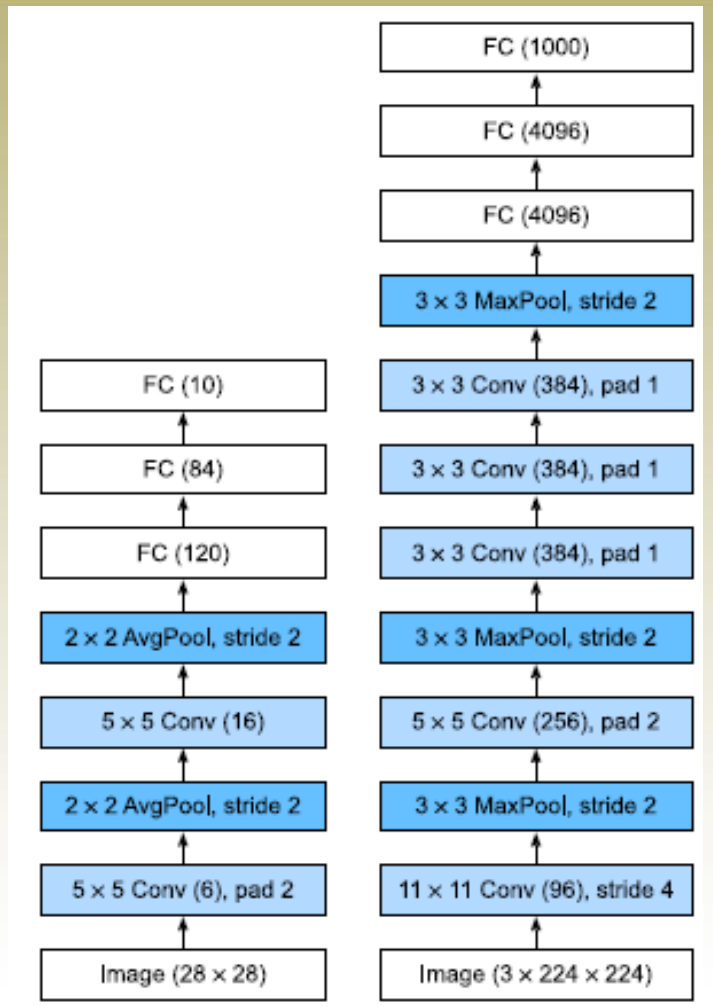
- Calculate #parameters of the following CNN
  - Assume that each pooling layer has two parameters (like AlexNet), i.e., weight and bias



Filter size= 5x5 Stride= 1 Valid padding (32-5+1)x(32-5+1) x 6 =784 x 6 = 4704 neurons [(5x5) weights +1 bias] x 6 filters= 156 trainable parameters	Pooling window= 2x2 Stride= 2 14x14x6=1176 neurons 14x14x6x(2x2+1)= 5880 connections [(1 weight +1 bias] x 6 'slices'= 12 trainable parameters	Filter size= 5x5 Stride= 1 Valid padding (14-5+1)x(14-5+1)x16=1600 neurons [(5x5x6) weights +1 bias] x 16 filters= 2416 trainable parameters	Pooling window= 2x2 Stride= 2 5x5x16=400 neurons [(1 weight +1 bias] x 16 'slices'= 32 trainable parameters	170 neurons x (400 neurons +1 bias) = 48120 trainable parameters	17 neurons x (170 neurons +1 bias) = 2907 trainable parameters

Some (not so) modern CNNs: AlexNet

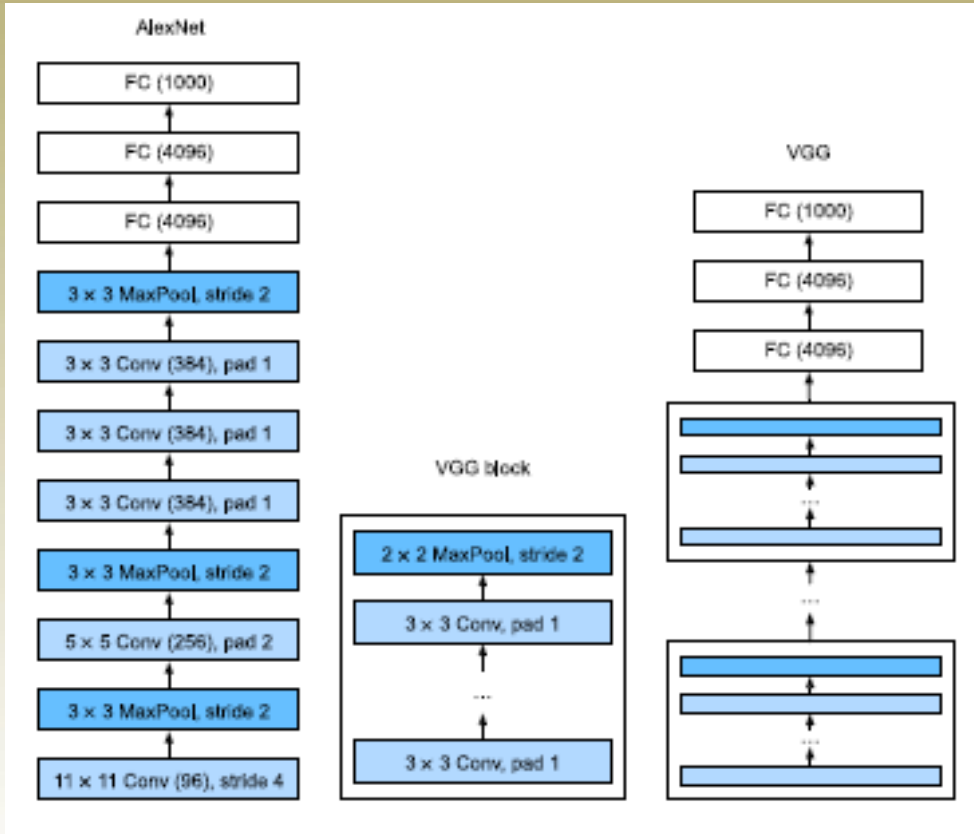
- AlexNet, which employed an 8-layer CNN, won the ImageNet Large Scale Visual Recognition Challenge 2012
- In AlexNet's first layer, the convolution window shape is 11 x 11
  - Since most images in ImageNet are more than ten times higher and wider than the MNIST images, objects in ImageNet data tend to occupy more pixels. Consequently, a larger convolution window is needed to capture the object
- AlexNet changed the sigmoid activation function to a ReLU activation function
  - The computation of the ReLU activation function is simpler. The ReLU activation function makes model training easier when using different parameter initialization methods. This is because, when the output of the sigmoid activation function is very close to 0 or 1, the gradient of these regions is almost 0, so that backpropagation cannot continue to update some of the model parameters. In contrast, the gradient of the ReLU activation function in the positive interval is always 1



LeNet (left) to AlexNet (right).

Some (not so) modern CNNs: VGG

- While AlexNet offered empirical evidence that deep CNNs can achieve good results, it did not provide a general template to guide in designing new networks. The idea of using blocks first emerged from the Visual Geometry Group96 (VGG) at Oxford University, in their eponymously-named VGG network
- The basic building block of classic CNNs is a sequence of the following: (i) a convolutional layer with padding to maintain the resolution, (ii) a nonlinearity (e.g., ReLU), (iii) a pooling layer such as a max pooling layer. One VGG block consists of a sequence of convolutional layers, followed by a max pooling layer for spatial downsampling.
  - In the original VGG paper, the authors employed onvolutions with 3 x 3 kernels with padding of 1 (keeping height and width) and 2 x 2 max pooling with stride of 2 (halving the resolution after each block)

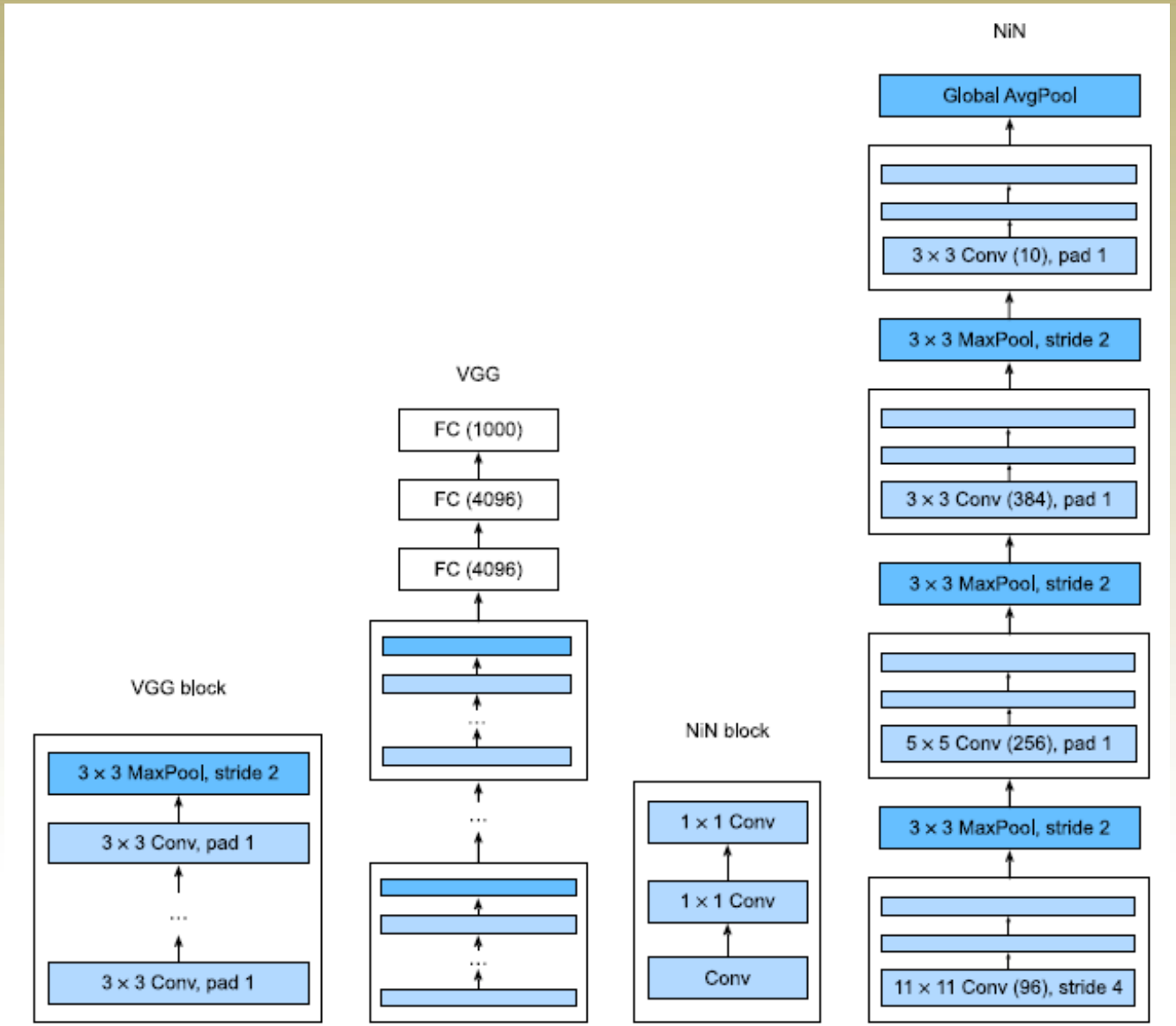


AlexNet to VGG that is designed from building blocks

Some (not so) modern CNNs: NiN

- AlexNet, and VGG all share a common design pattern:
  - extract features exploiting spatial structure via a sequence of convolution and pooling layers and then post-process the representations via fully-connected layers.
- Alternatively, one could imagine using fully-connected layers earlier in the process. However, a careless use of dense layers might give up the spatial structure of the representation entirely, network in network (NiN) blocks offer an alternative.
  - They were proposed based on a very simple insight: to use an MLP on the channels for each pixel separately
- NiN uses convolutional layers with window shapes of 11x11, 5x5, and 3x3, and the corresponding numbers of output channels are the same as in AlexNet. Each NiN block is followed by a maximum pooling layer with a stride of 2 and a window shape of 3x3
- One significant difference between NiN and AlexNet is that NiN avoids fully-connected layers altogether. Instead, NiN uses an NiN block with a number of output channels equal to the number of label classes, followed by a global average pooling layer

Some (not so) modern CNNs: NiN



Some (not so) modern CNNs: GoogLeNet

- In 2014, GoogLeNet won the ImageNet Challenge, proposing a structure that combined the strengths of NiN and paradigms of repeated blocks
  - sometimes it can be advantageous to employ a combination of variously-sized kernels
- The basic convolutional block in GoogLeNet is called an *Inception block*
  - It consists of four parallel paths. The first three paths use convolutional layers with window sizes of 1 x 1, 3 x 3, and 5 x 5 to extract information from different spatial sizes. The middle two paths perform a 1 x 1 convolution on the input to reduce the number of channels, reducing the model's complexity. The fourth path uses a 3 x 3 maximum pooling layer, followed by a 1 x 1 convolutional layer to change the number of channels. The four paths all use appropriate padding to give the input and output the same height and width. Finally, the outputs along each path are concatenated along the channel dimension and comprise the block's output.