Convolutional Neural Network

Tran Giang Son, tran-giang.son@usth.edu.vn

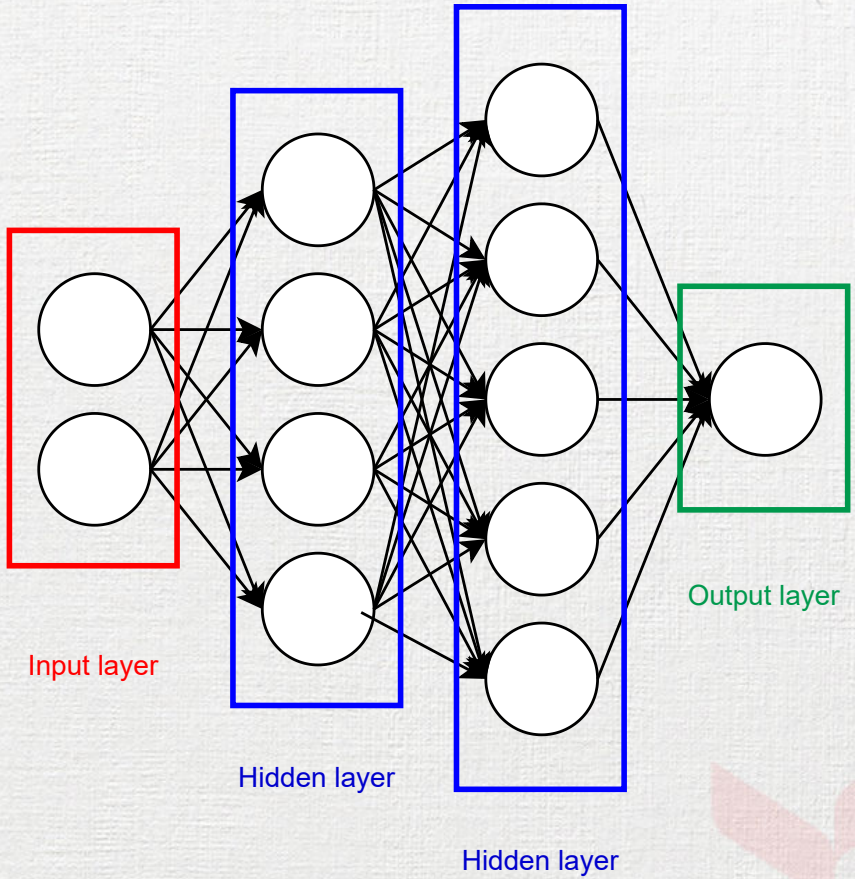
ICT Department, USTH

Layers

General Neural Network Model

Each node in hidden layer and output layer:

- Each hidden layer is called a fully connected layer (or Dense layer)
- Each node in hidden layer is connected to all nodes in the previous layer

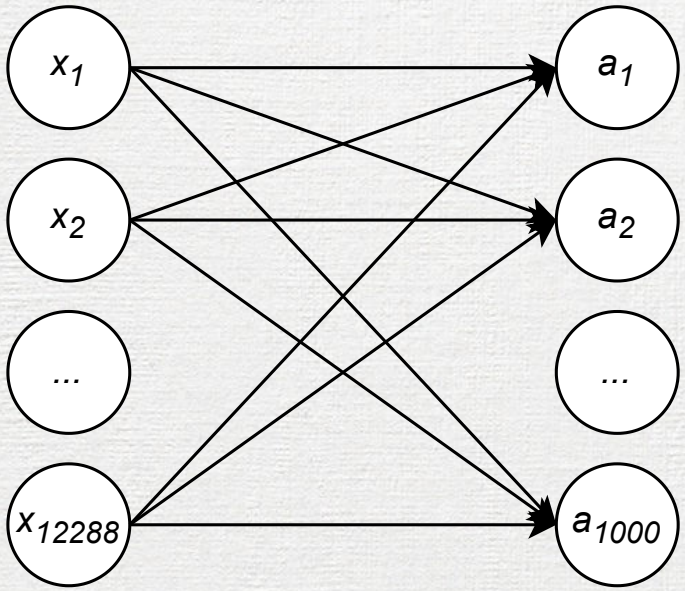


General architecture

Simple Problem

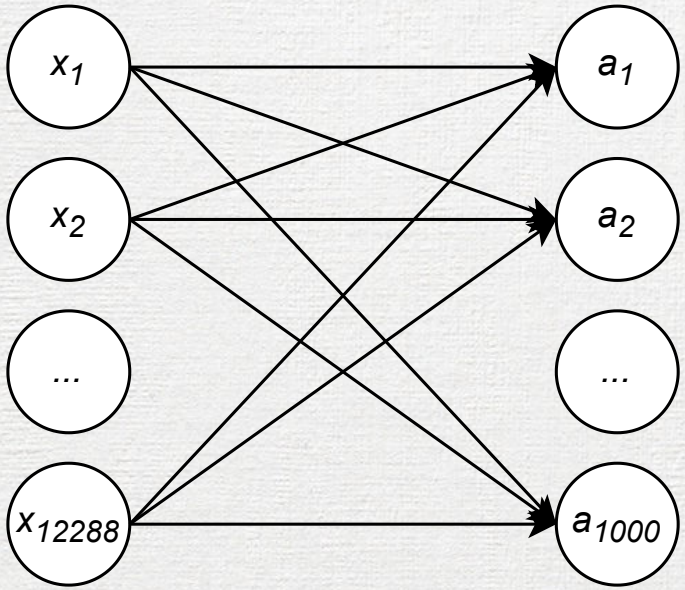
- Input: color image of size 64 * 64
- Output: image contains human face or not
- Neural network model?

Simple Problem



- Color image size 64 * 64 needs 64 * 64 * 3 pixels
- Input layer has 12288 values
- Hidden layer 1 has 1000 nodes
- # of weights is 12,288,000 + # of bias is 1000

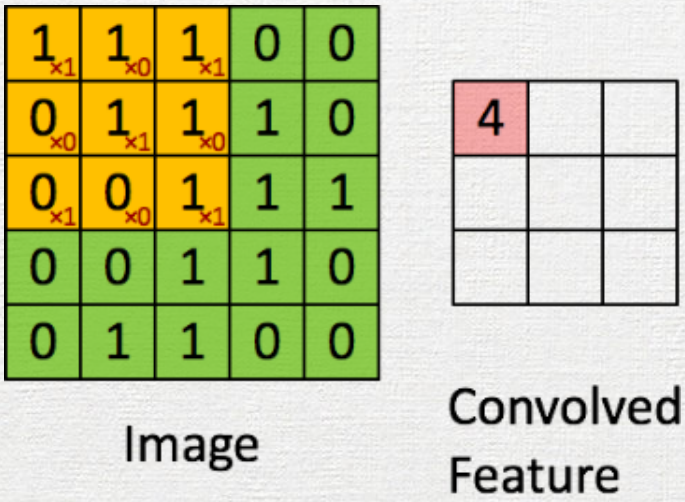
Simple Problem



- What if
  - Image size is 512×512
  - 10 hidden layers
  - 1000 neurons each?

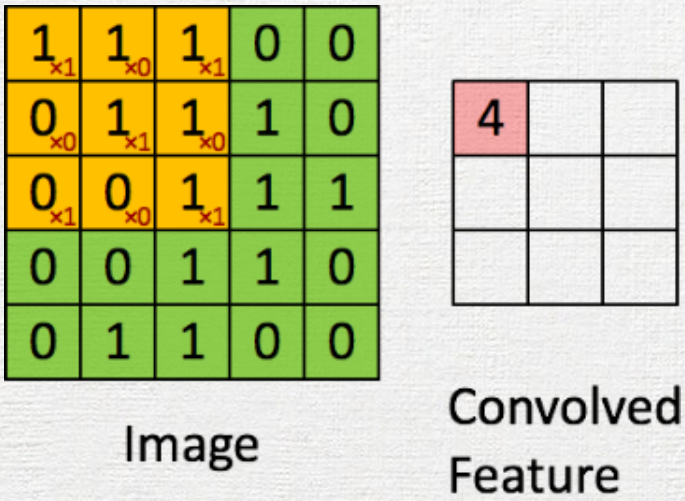
$$512 \times 512 \times 1000^{10} \times 1$$

Convolution



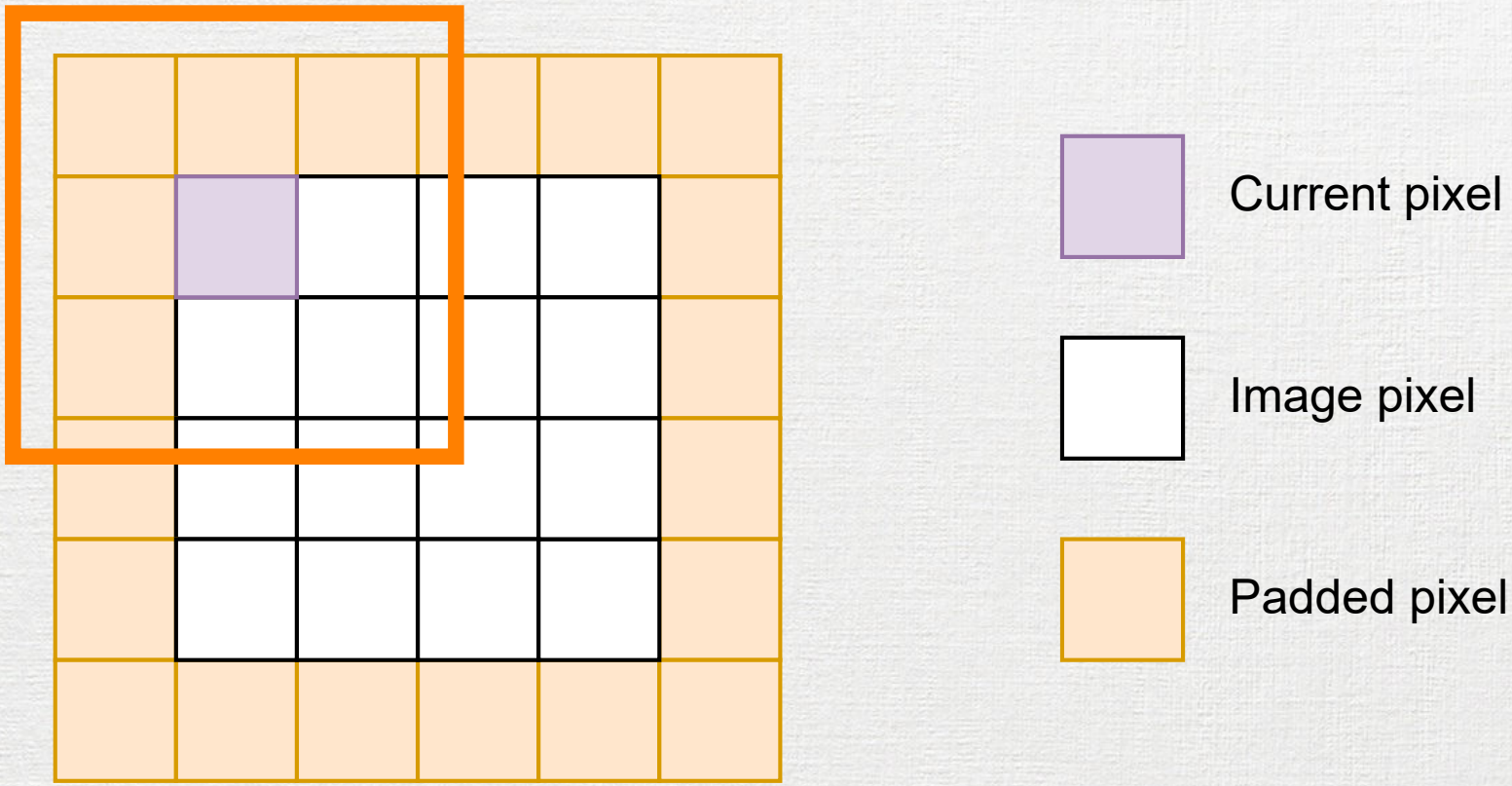
- Neuron depends only on a few local input neurons
- Similarly to the local connectivity of visual features in images

Convolution



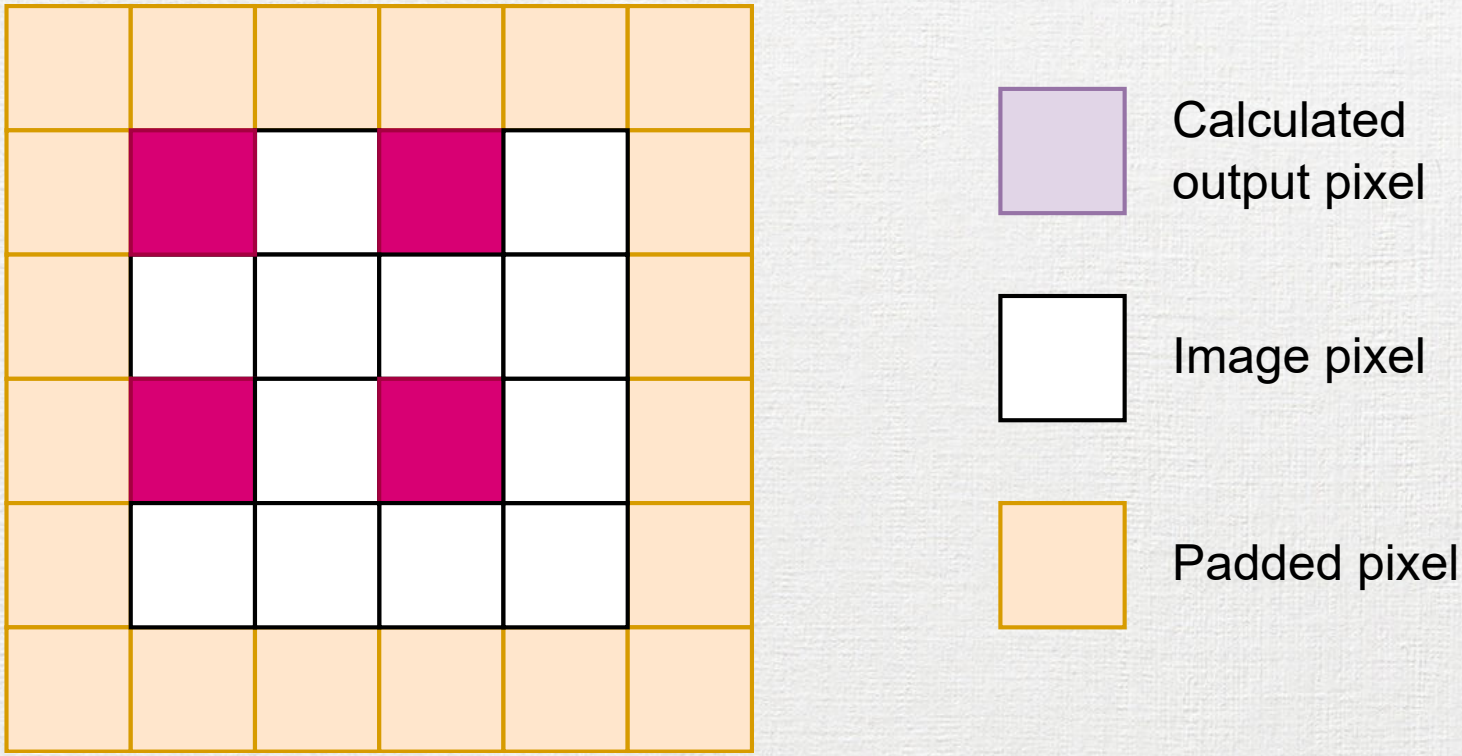
- x is a 3×3 chunk (yellow area) of the image (green area)
- Each output neuron is parametrized with the 3×3 weight matrix w (small red numbers in yellow area)
- Output image contains convolved features (in pink)
- The process is performed by sliding the 3×3 window through the image

Padding



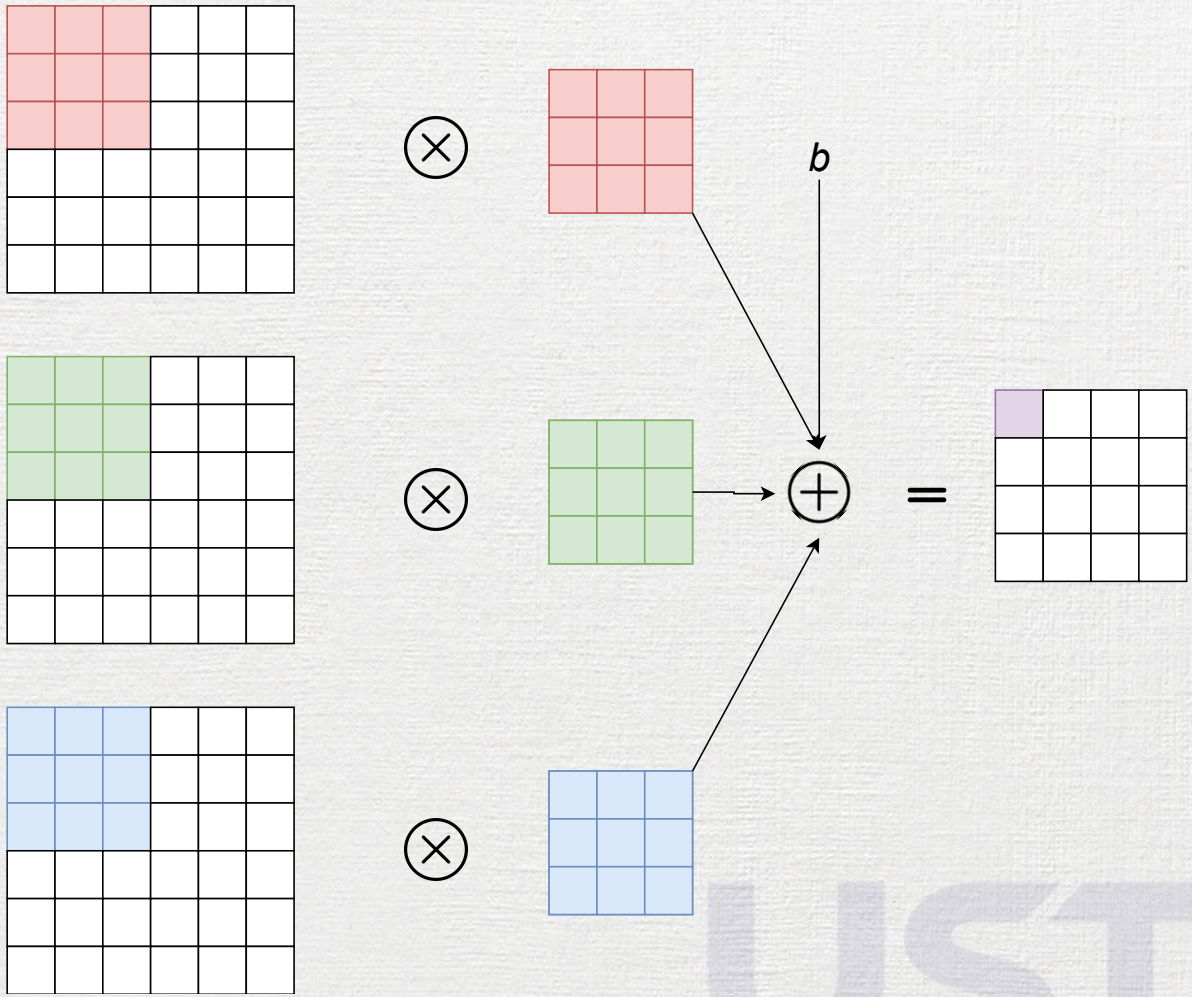
Border of the image is padded with zero values

Stride

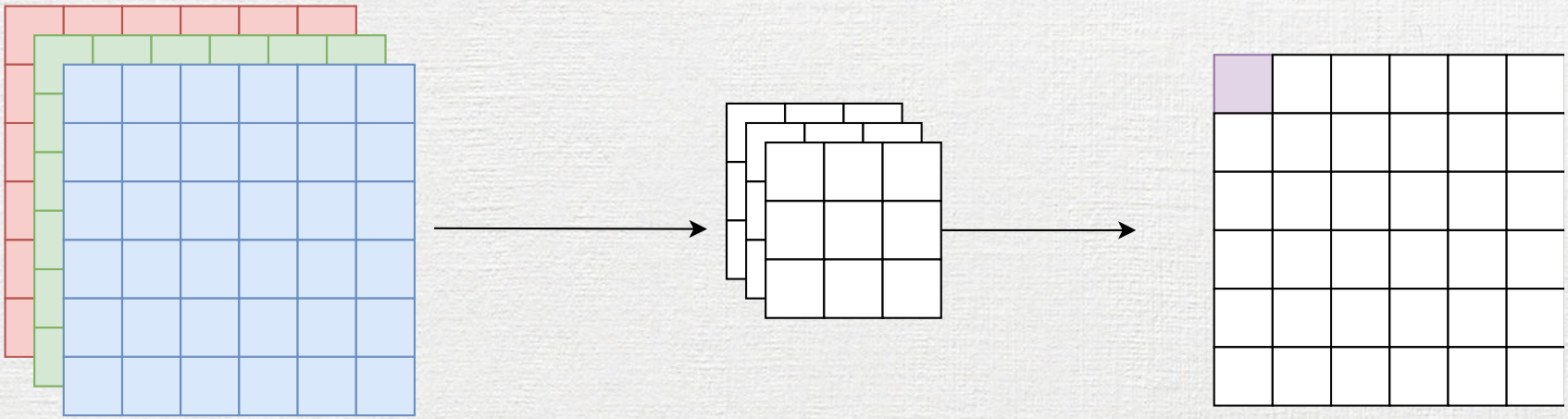


Stride=2

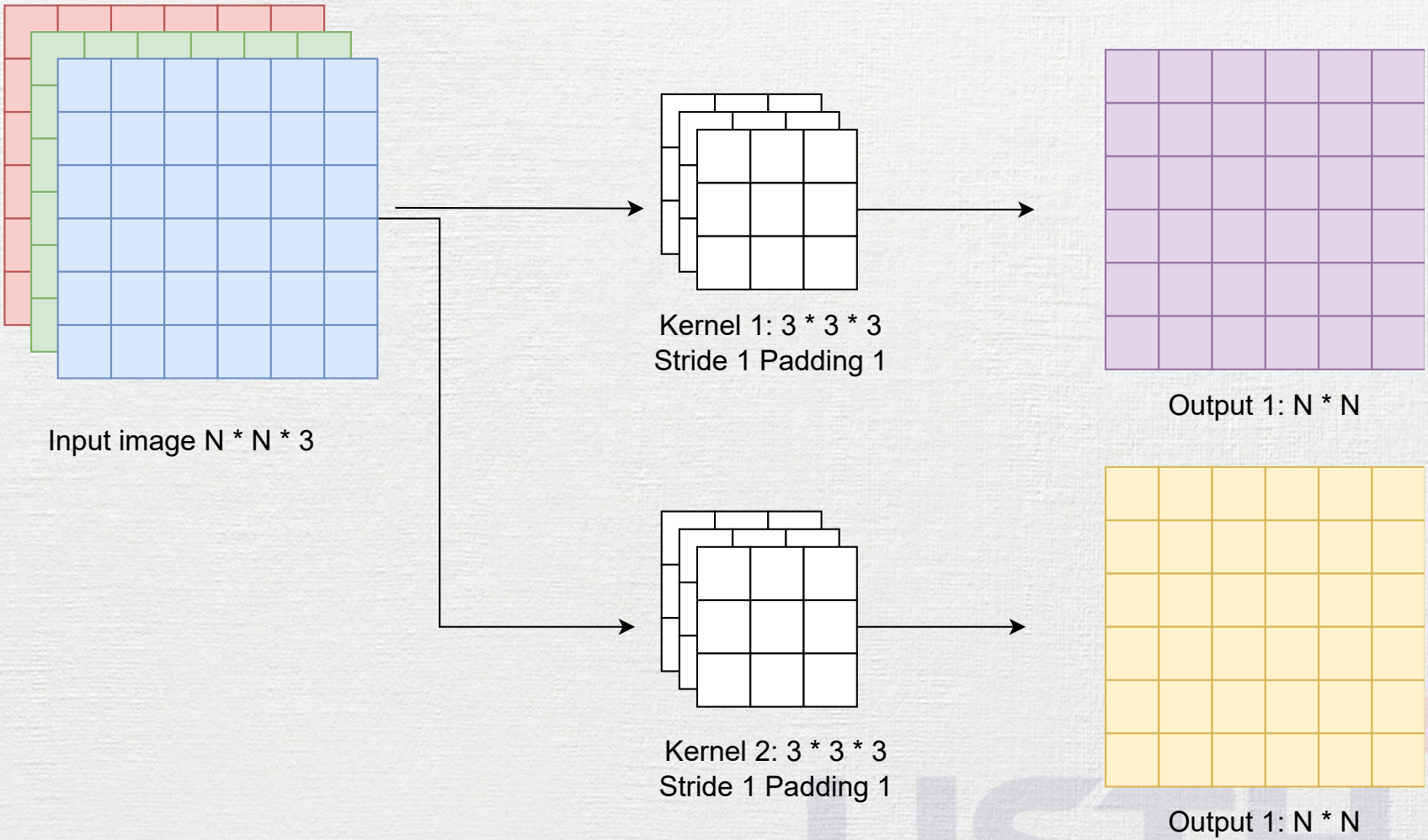
Multiple input, single output



Multiple input, single output

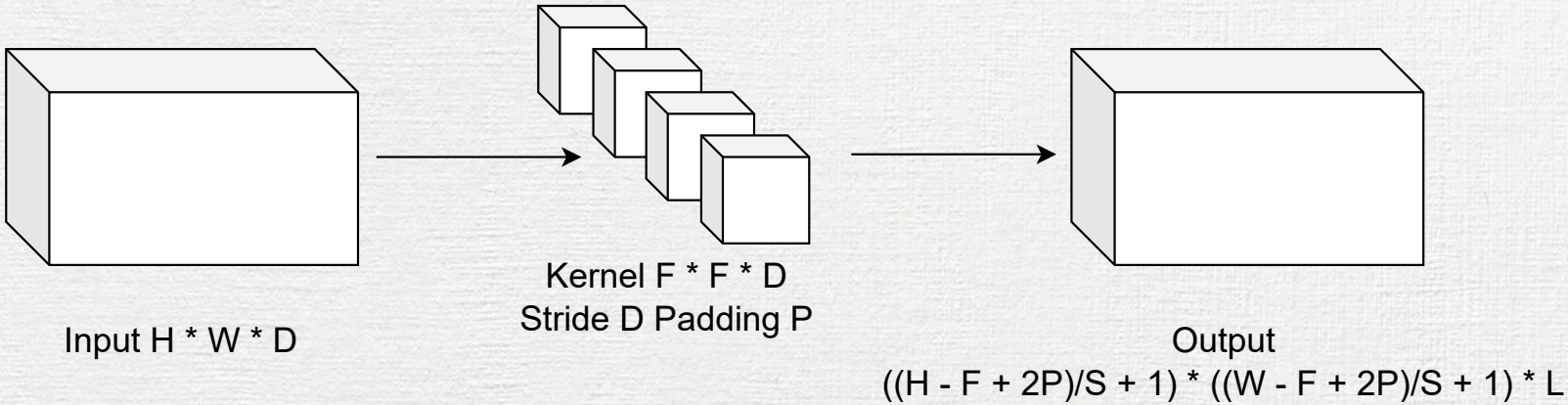


Multiple input, multiple output



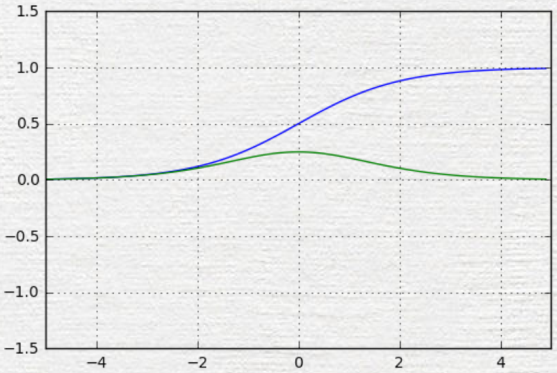
Output of the first convolutional layer will be input of the next convolutional layer

General Convolutional Layer



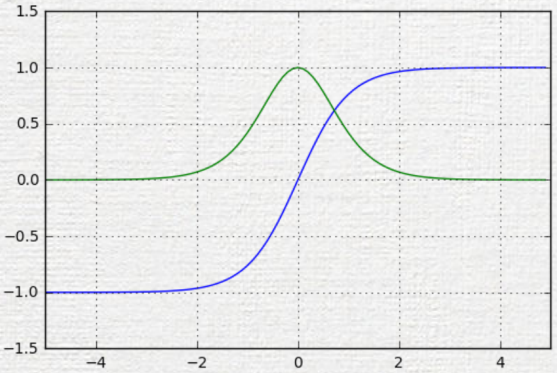
- Number of parameters of each kernel is $F * F * D + 1$ (for bias)
- Number of parameters of layer is $K * (F * F * D + 1)$
- Output of the convolutional layer will be applied with a non-linear activation function before being the input of the next convolutional layer

Element-wise activation functions



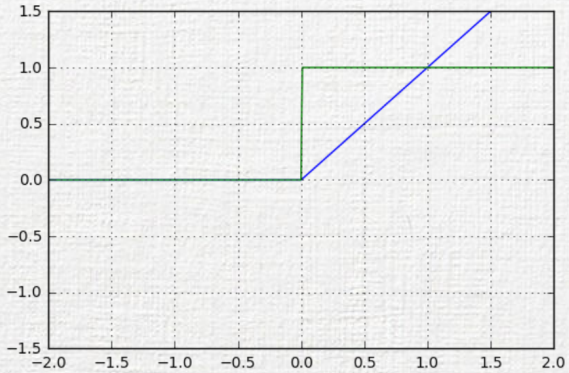
$$\text{sigm}(x) = \frac{1}{1+e^{-x}}$$

$$\text{sigm}'(x) = \text{sigm}(x)(1-\text{sigm}(x))$$



$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

$$\tanh'(x) = 1-\tanh(x)^2$$

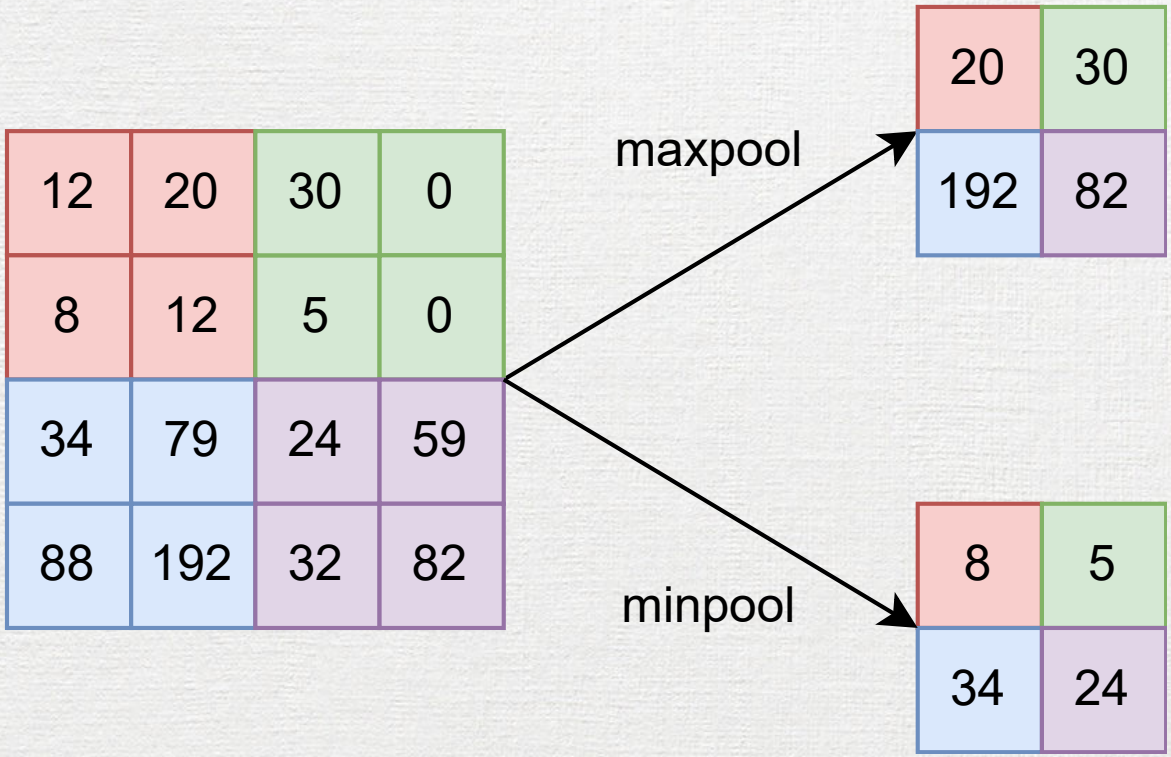


$$\text{relu}(x) = \max(0,x)$$

$$\text{relu}'(x) = 1_{x>0}$$

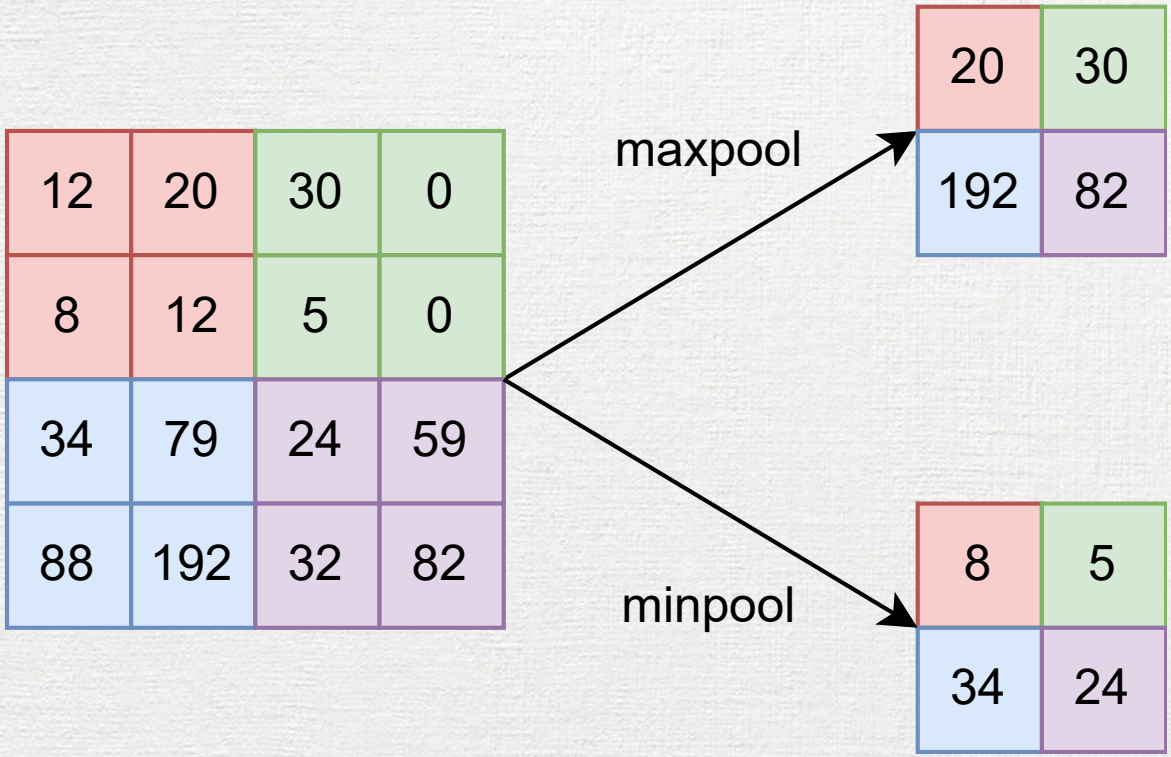
- Blue line: activation function
- Green line: derivative
- Relu activation function is often used after each convolutional layer since it is an efficient activation function without heavy computation

Pooling



- Pooling layer is placed between two convolutional layers to reduce sizes of output data and still preserve the important features of images

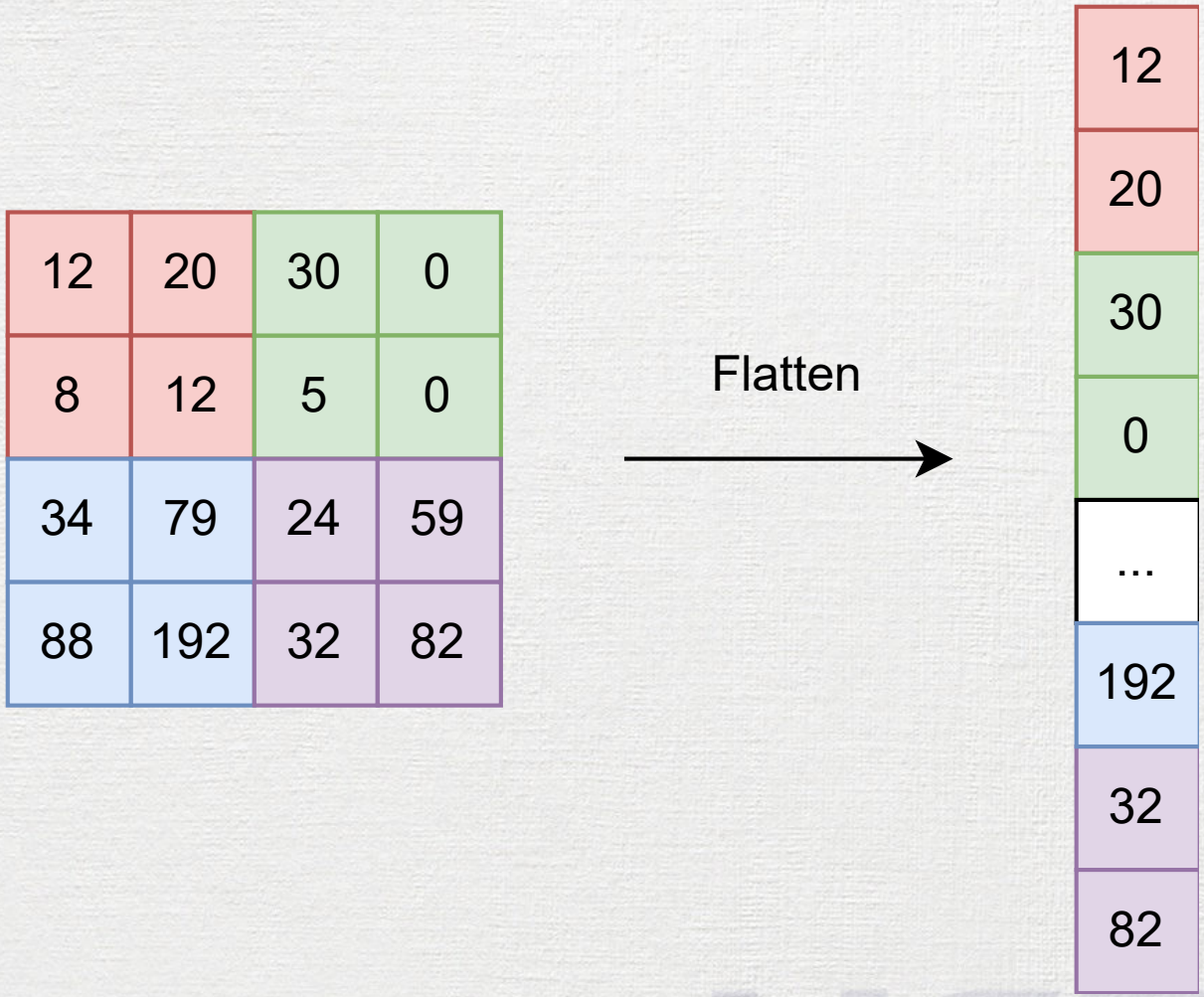
Pooling



- In practice, pooling layer with size = (2,2), stride = 2 and padding = 0 is often used so that output width and height of data are reduced half while depth is unchanged

Note: in some models, convolutional layer with stride > 1 is used to reduce data sizes instead of pooling layer

Flatten



- Tensor of output of last layer with size $(H * W * D)$ is flatten to the vector with size $(H * W * D, 1)$

Softmax

- Apply the standard exponential function to each element z_i of the input vector $\mathbf{z}$.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{1}$$

With

$$\sum_{i=1}^{K} \sigma(\mathbf{z})_i = 1 \tag{2}$$

- Each value in the output of the softmax function is interpreted as the probability of membership for each class

Softmax

- Softmax activation is used to normalize the outputs of the last dense layer, converting them from weighted sum values into probabilities that sum to 1
- Specifically, softmax activation outputs one value for each node in the output layer.

Classical Networks

Classical Architectures

Input

Conv blocks:

- Convolution + activation (relu)
- Convolution + activaton (relu)
- ...
- Maxpooling 2x2

Output

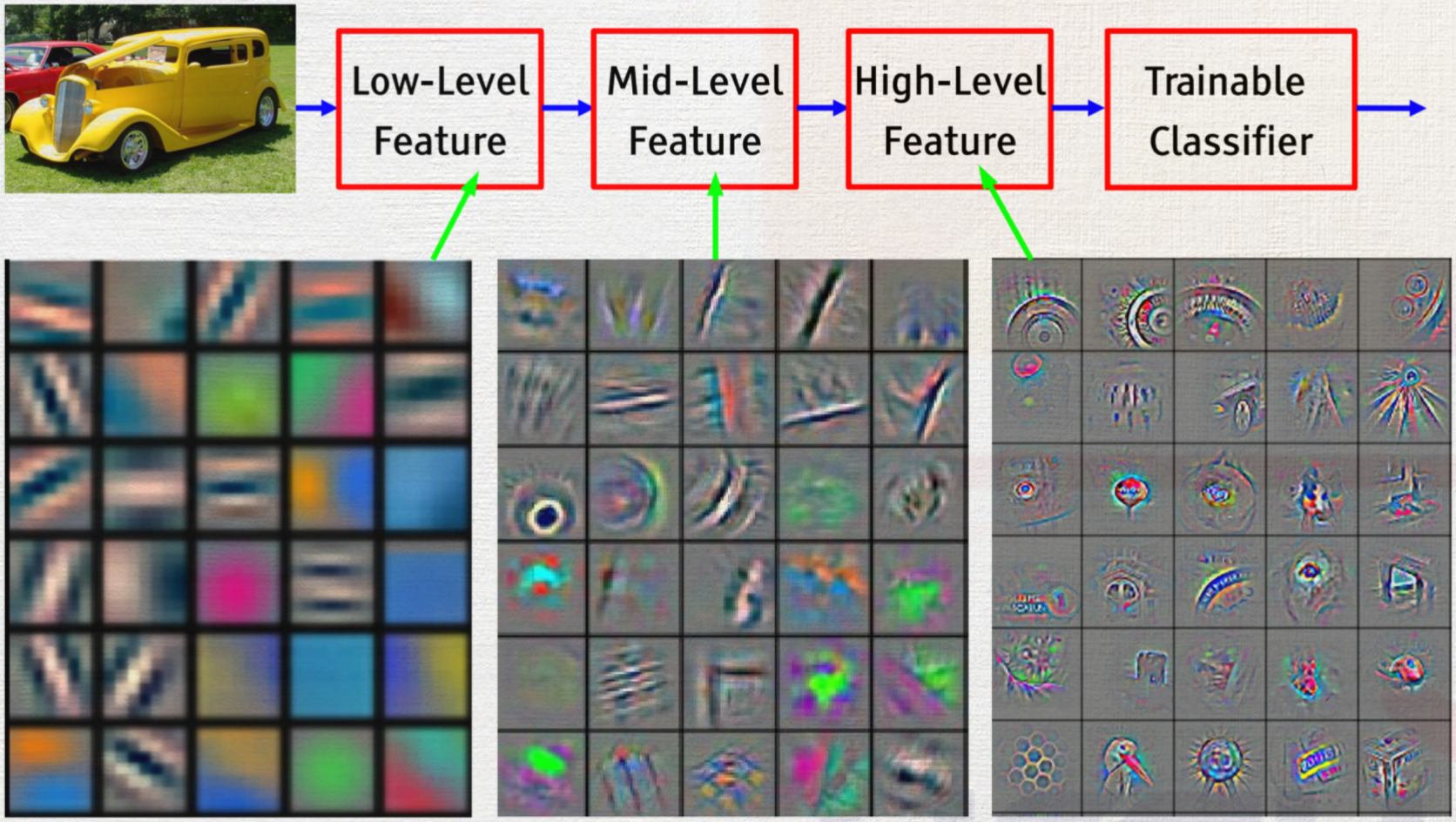
- Fully connected layers
- Softmax / Sigmoid activation function

Classical Architectures

Last output fully connected layers

- If only one neuron: Sigmoid activation function
- If multiple neurons: Softmax activation function

Feature Extraction



Visualization of image features learned automatically by convolutional layers

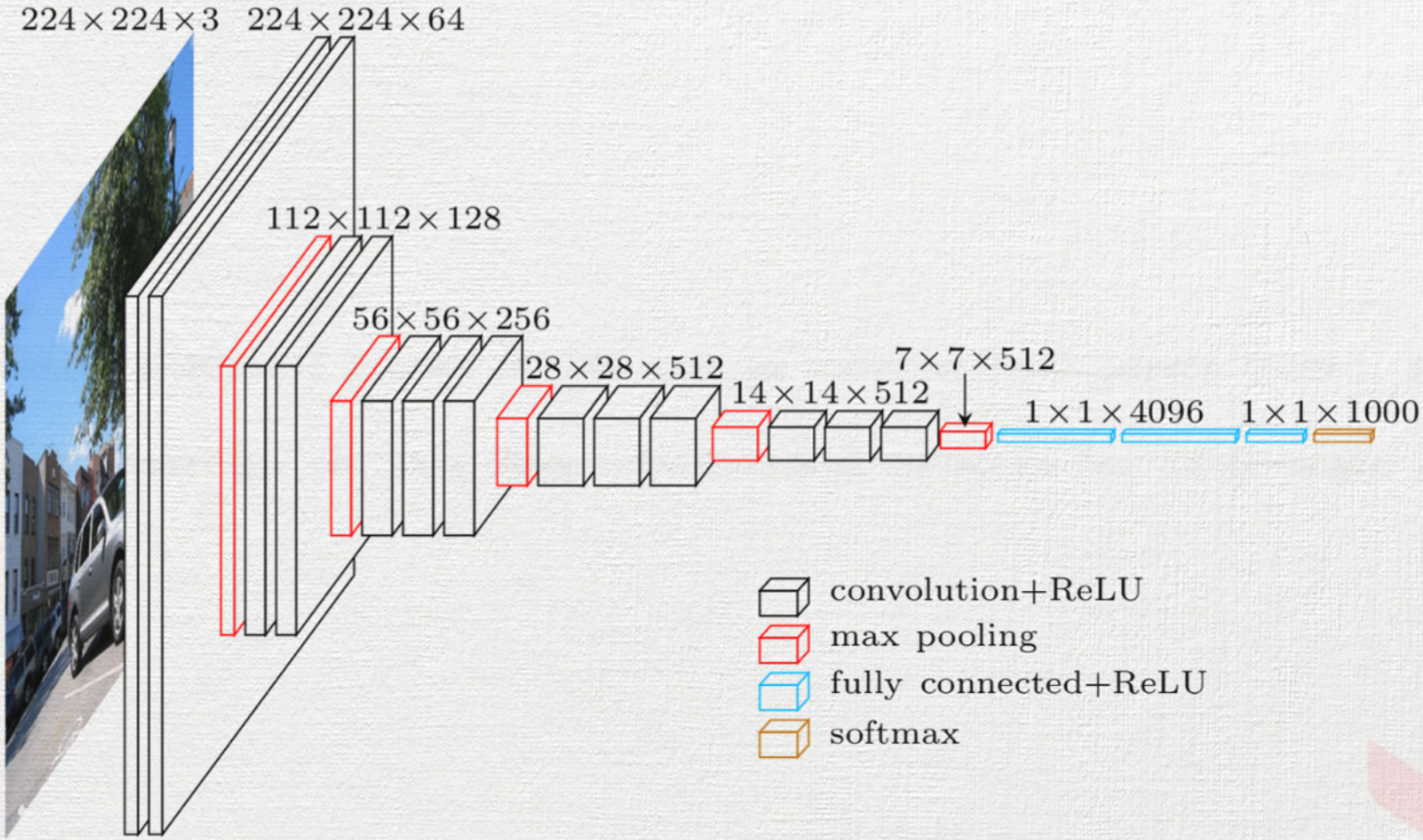
Popular Networks

- VGG (Visual Geometry Group)
- ResNet (Residual Network)

VGG Architecture

- VGG is a deep CNN architecture containing classical blocks of CNN such as convolutional layers (conv), pooling layers (pool) and fully connected layers (fc)
- Network architectures: VGG16, VGG19
- VGG is proposed by Simonyan, Karen, and Zisserman in "Very deep convolutional networks for large-scale image recognition." (2014)

VGG16

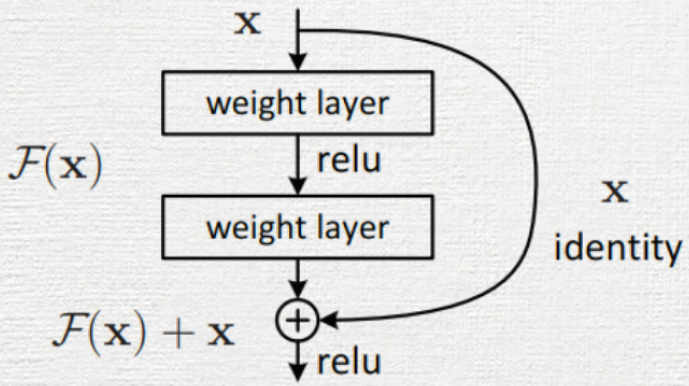


- From left to right: size of output features decreases, but depth increases

VGG16

- Conv: size 3x3, padding = 1, stride = 1, # of kernels = 64 or dept of output layer
- Pool/2: max pooling layer with size = 2x2, stride = 2
- fc 4096: fully connected layer with 4096 nodes
- After passing through all conv layers and pooling layers, data are flattened and fed into the fc layers

ResNet



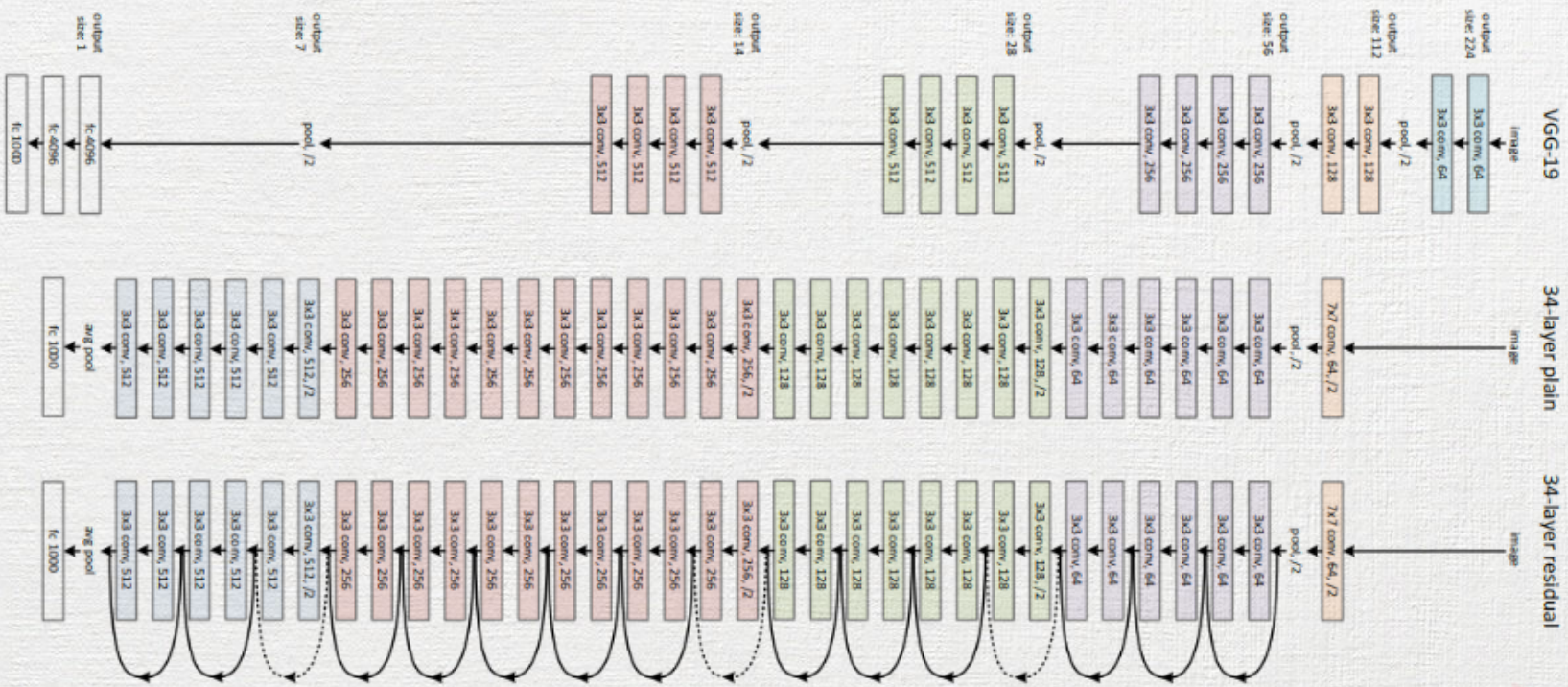
Residual learning: a building block

- ResNet introduces the concepts called residual block using skip (shortcut) connection
- A ResNet architecture is created by stacking a set of residual blocks together

ResNet

- ResNet solves the problem of vanishing or exploding gradient
- ResNet is able to support hundreds or thousands of convolutional layers
- Proposed by He, Kaiming, et al. "Deep residual learning for image recognition." CVPR. 2016.

ResNet



ResNet34 vs VGG19 vs 34-layer plain

VGG19 vs ResNet50

Factor	VGG19	ResNet50
Accuracy	5.25% top-5	7.1%
Parameters	25M	138M
Complexity	3.8B FLOPS	15.3B FLOPS
Convolution	Fully conv	Several fully connected layers

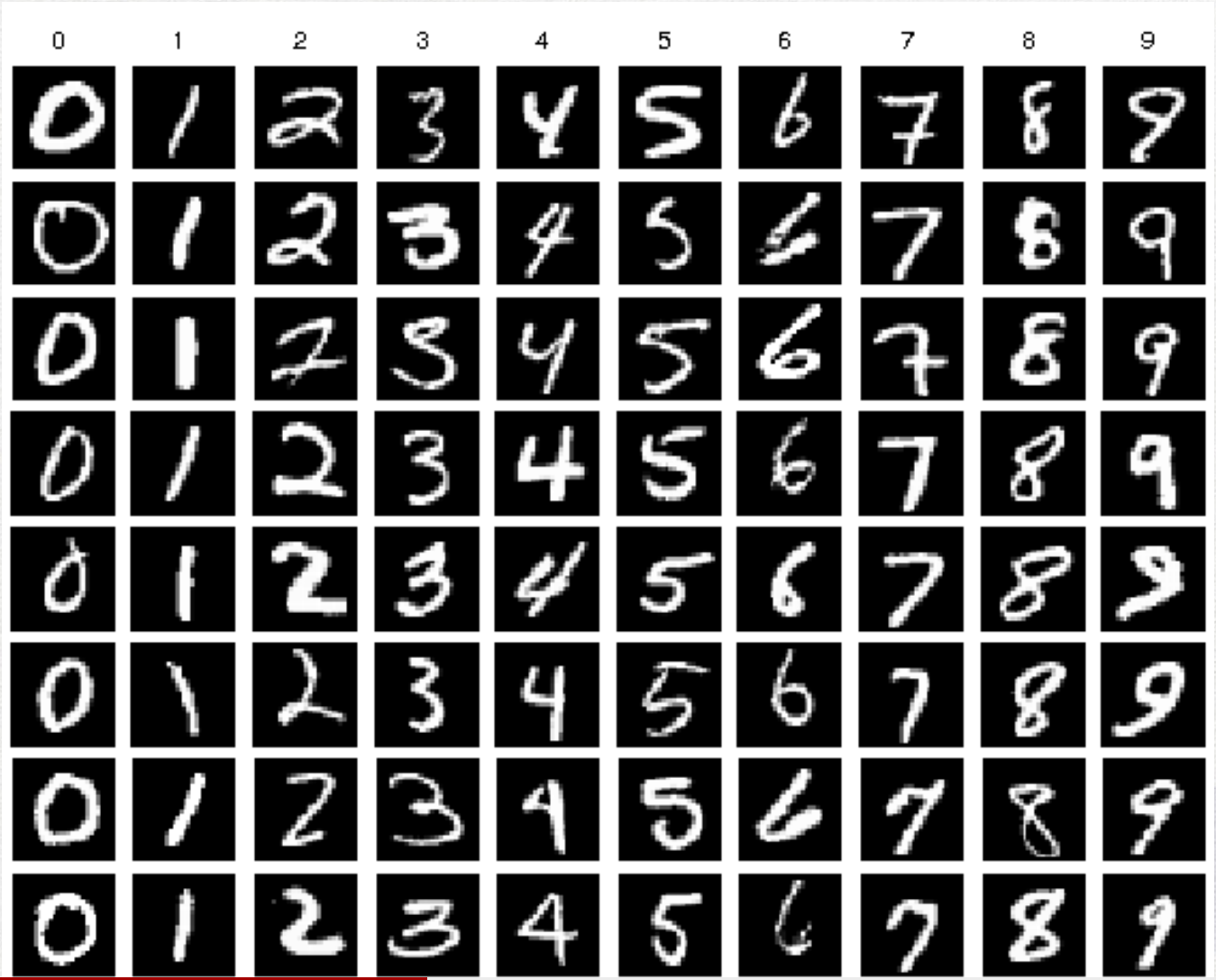
Image Classification

What?

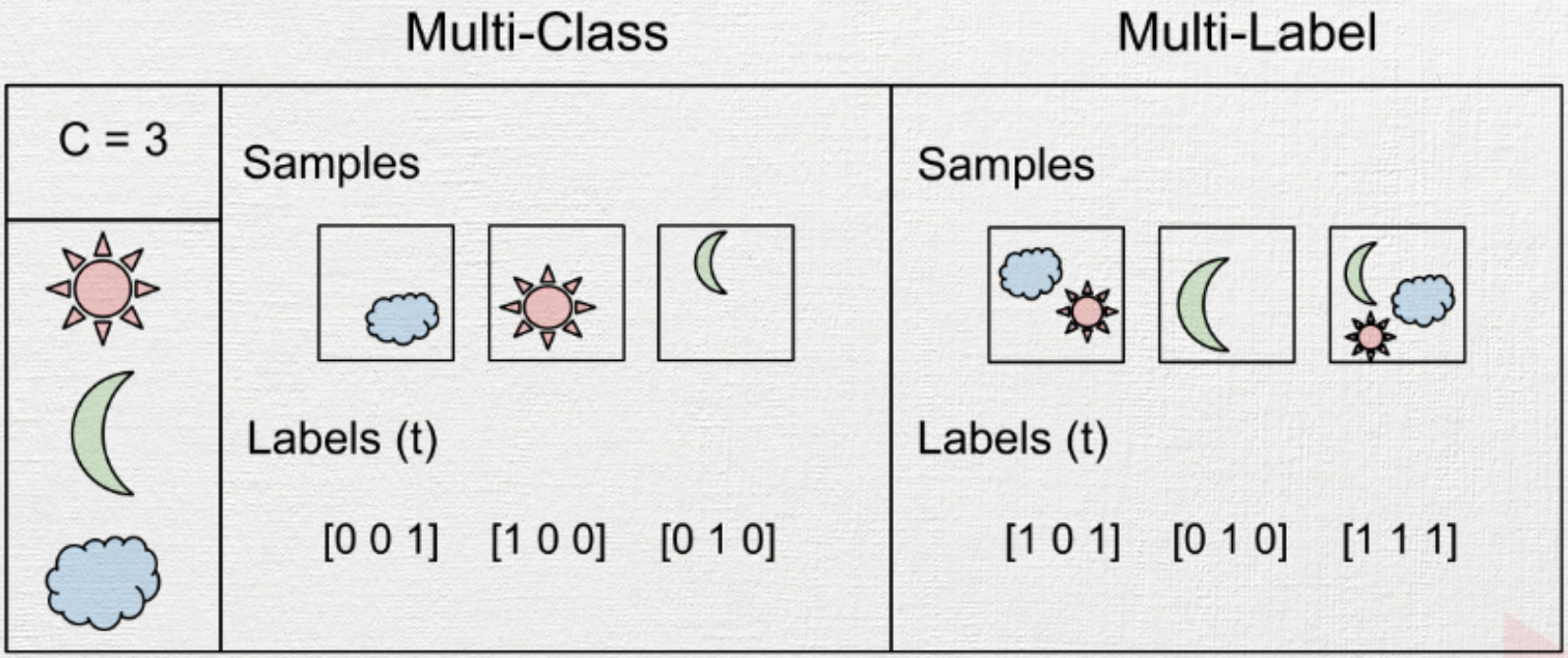
- Image classification is the process of predicting the class of an image
- Images are expected to have only one class for each image
- "Dog vs. cat": binary classification of images

What?

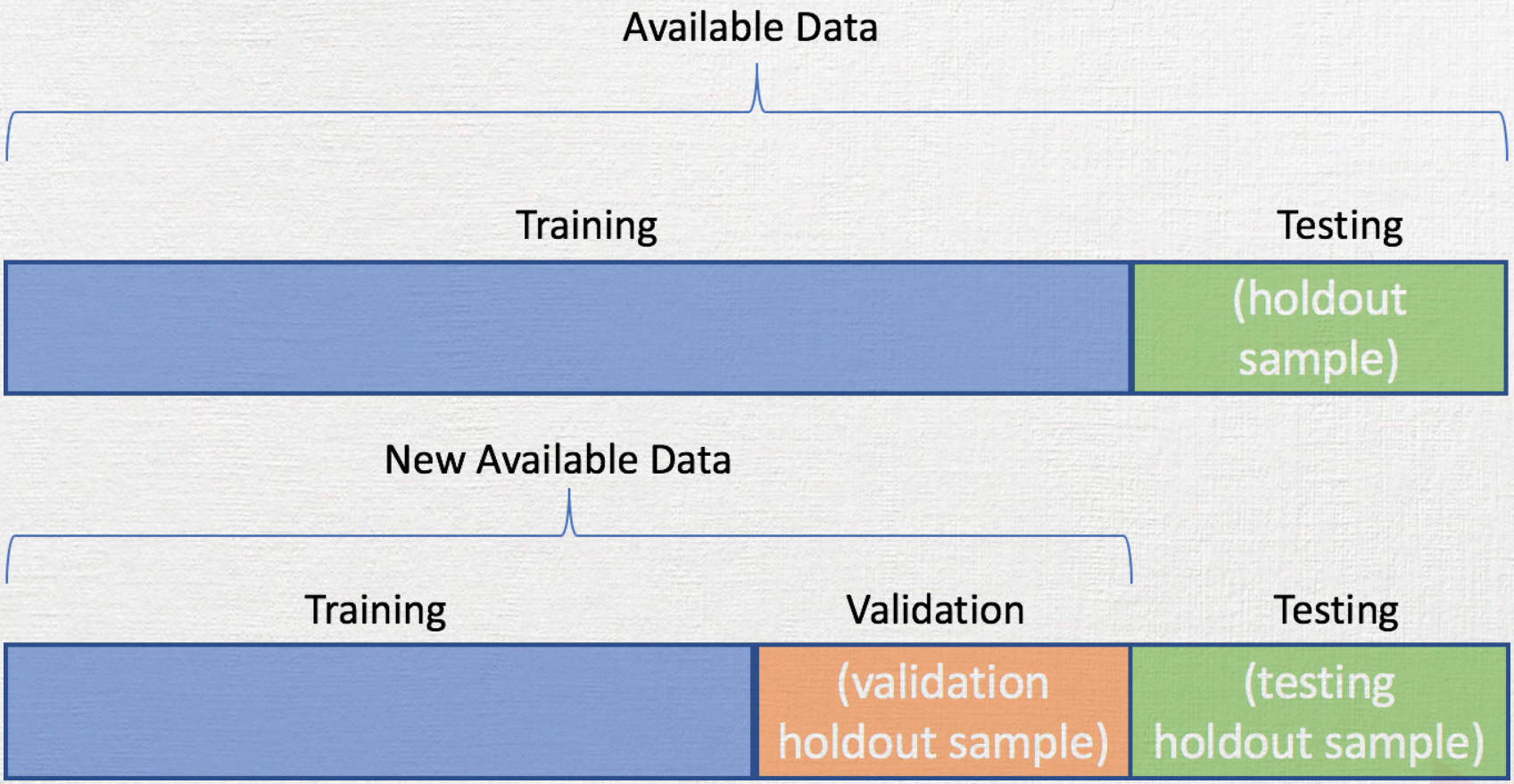
- Digit classification: multi-class classification



What?

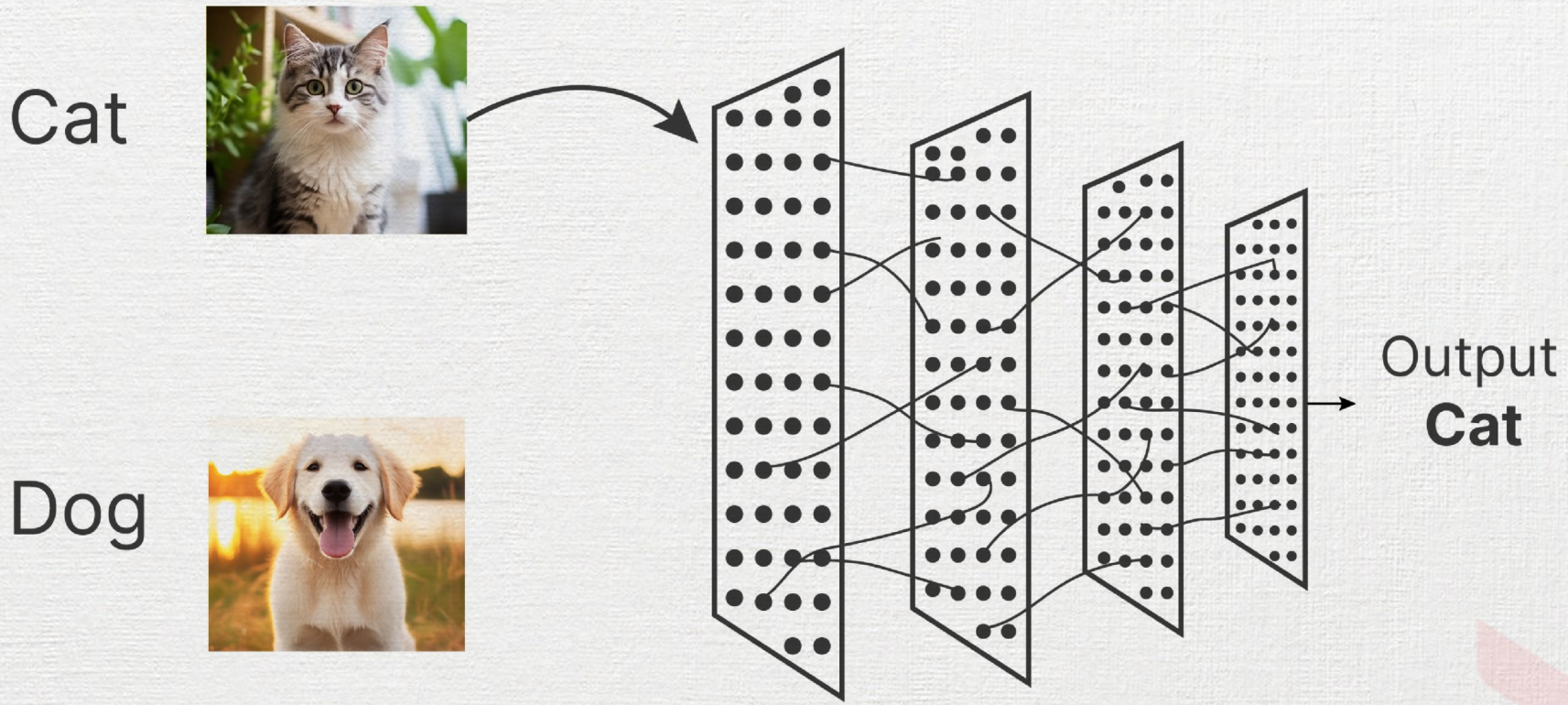

Multi-Class
Multi-Label
C = 3
Samples
Labels (t)
[0 0 1]
[1 0 0]
[0 1 0]
Samples
Labels (t)
[1 0 1]
[0 1 0]
[1 1 1]

Dataset

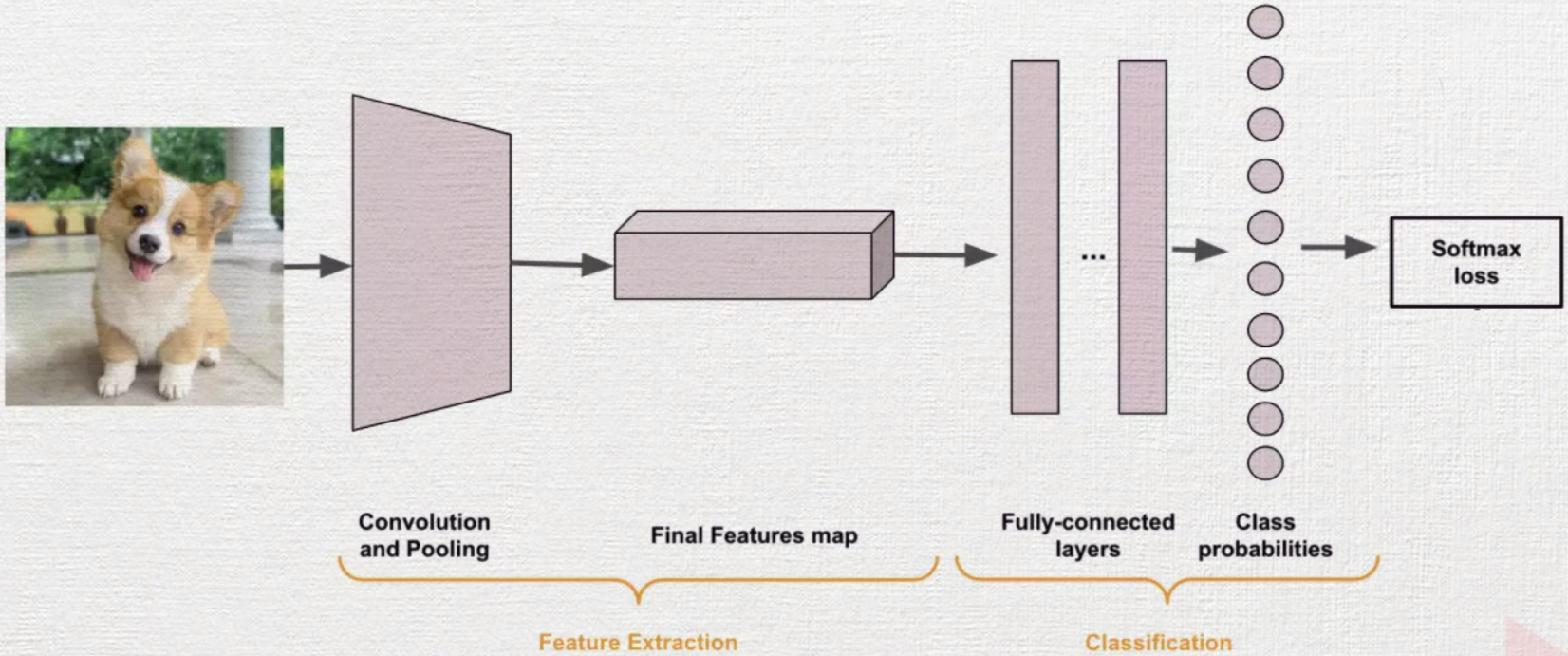


- Standardize directories for training set, validation set and test set
- Standardize images prior to the model requirement

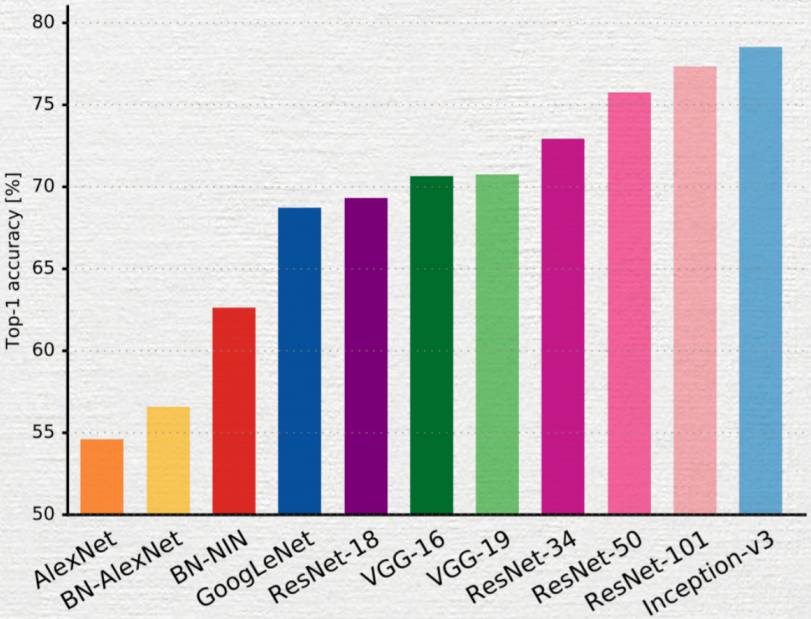
Sample Model

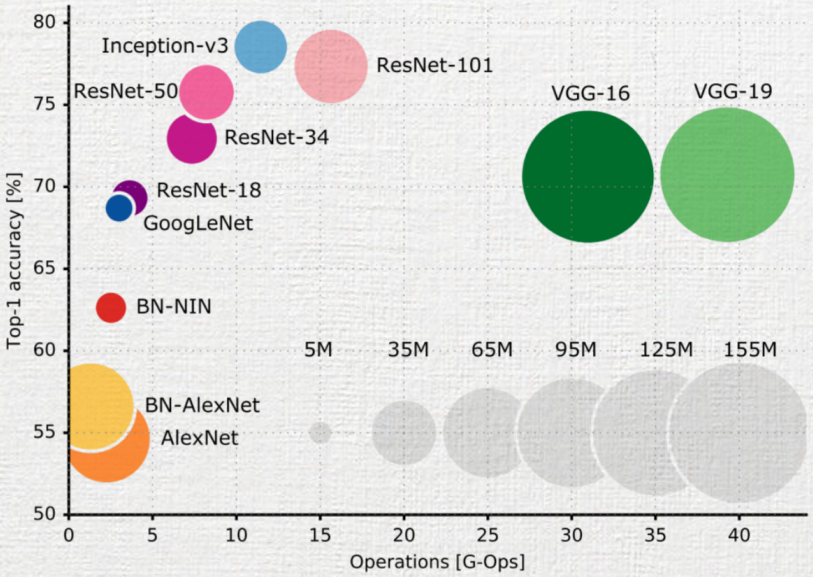


Sample Model



Performance





Other Techniques

Activation Function

- Softmax activation function

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3}$$

- Sigmoid activation function

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4}$$

Loss Function

- Binary cross entrophy

$$J = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i)) \tag{5}$$

- Categorical cross entrophy: softmax + cross entrophy

$$J = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right) \tag{6}$$

In which s_j is the prediction for the j^{th} class, s_p is the prediction of the model for the **positive** class, C is the total number of classes.

Practice!

Labwork 6: Convolutional Neural Network

- Implement VGG19 using a deep learning framework
  - Problem: Image classification
  - Input: image
  - Output: class of the input image
- Train and test the implemented network on a dataset of your choice
- Note: don't load a pretrained model!

Labwork 6: Neural Network

- Write a report (in LaTeX):
  - Name it « Report.6.CNN.tex »
  - How you design and implement the network architecture
  - Evaluation of the network using classification metrics
    - Accuracy
    - Precision
    - Recall
    - F1-score
  - Extra: comparison with ResNet19 if you are fast enough :)
- Push your code and report to your forked repository