CS4243

Computer Vision & Pattern Recognition

AY 2023/24

Lab Session 8

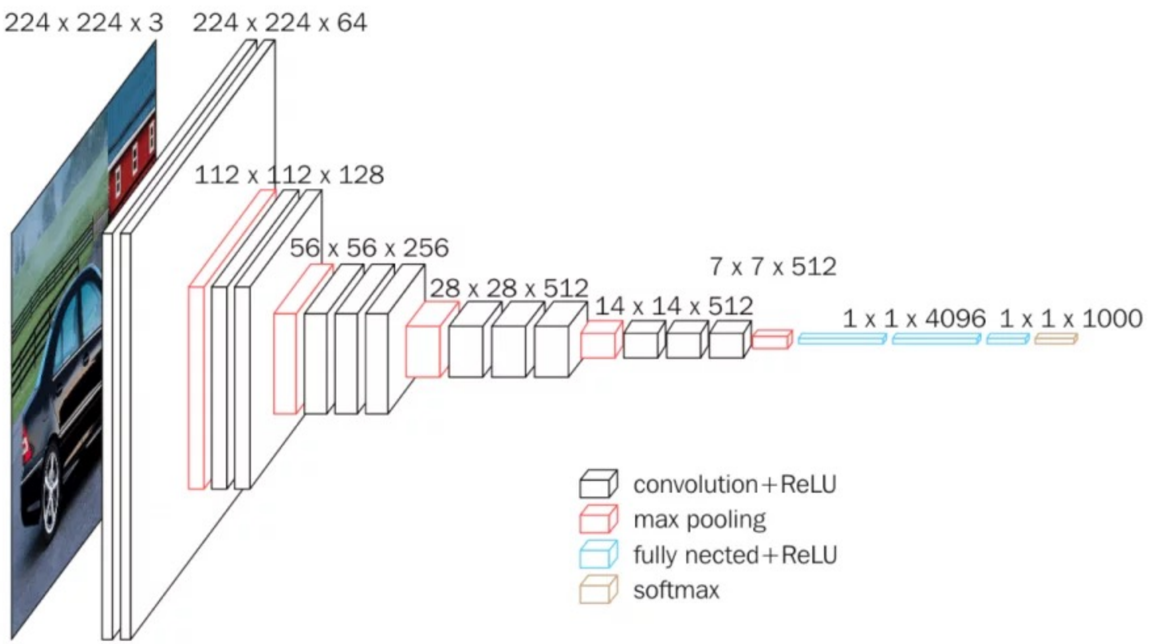
National University of Singapore | School of Computing

Arrangement

- Part 1 – Quick Recap from the Lecture (~20 min)
- Part 2 – Lab Tutorial (~30 min)
- Break (10 min)
- Part 3 – Lab Solution (~30 min)

Lab Materials

- GitHub Repo: https://github.com/ldkong1205/cs4243_lab
- Slides
- Notebook & Solution
- Other Materials (image, media, etc.)



Lesson 6

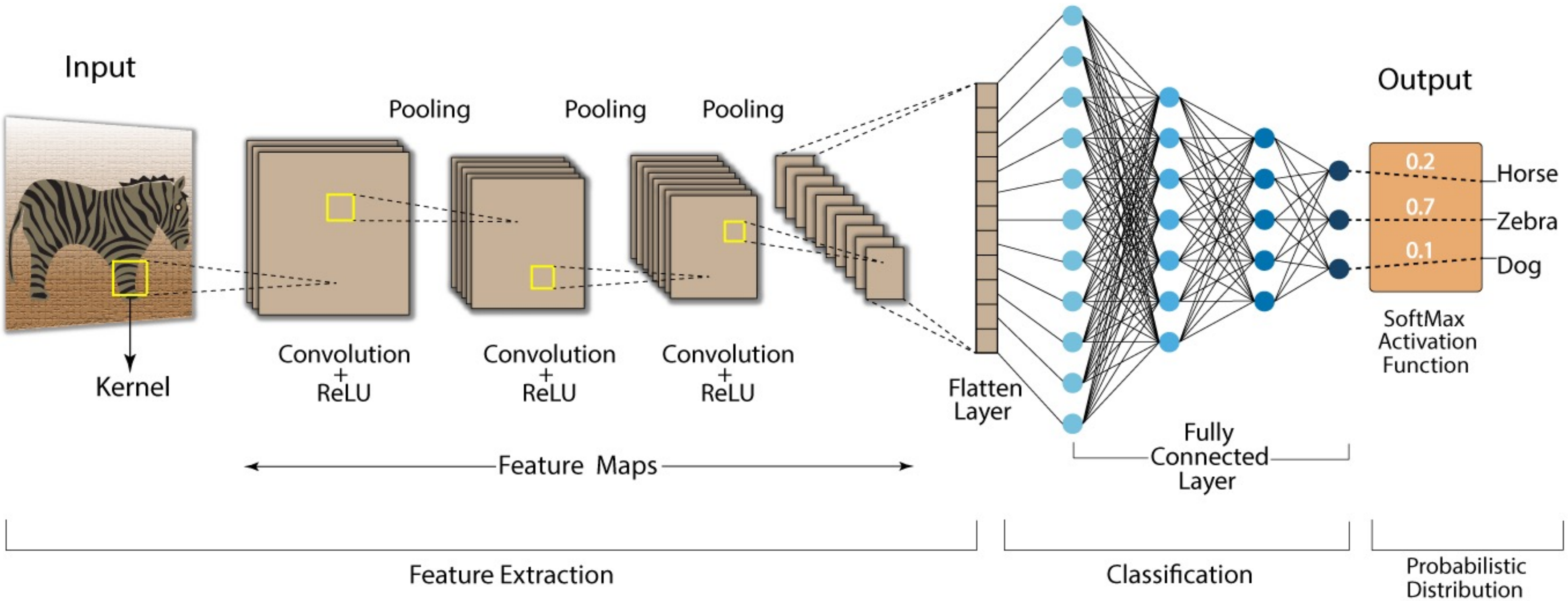
Computer Vision and Deep Learning

Convolutional Neural Networks

CNNs or ConvNets, are a type of feed-forward artificial neural network whose connectivity structure is inspired by the organization of the animal visual cortex.

Convolutional Neural Networks

CNNs or ConvNets, are a type of feed-forward artificial neural network whose connectivity structure is inspired by the organization of the animal visual cortex.



Convolutional Neural Networks

CNNs have an input layer, an output layer, numerous hidden layers, and millions of parameters, allowing them to learn complicated objects and patterns.

Convolutional Neural Networks

CNNs have an input layer, an output layer, numerous hidden layers, and millions of parameters, allowing them to learn complicated objects and patterns.

Convolution is the process of combining two functions to produce the output of the other function. The input image is convoluted with the application of filters in CNNs, resulting in a Feature Map.

Convolutional Neural Networks

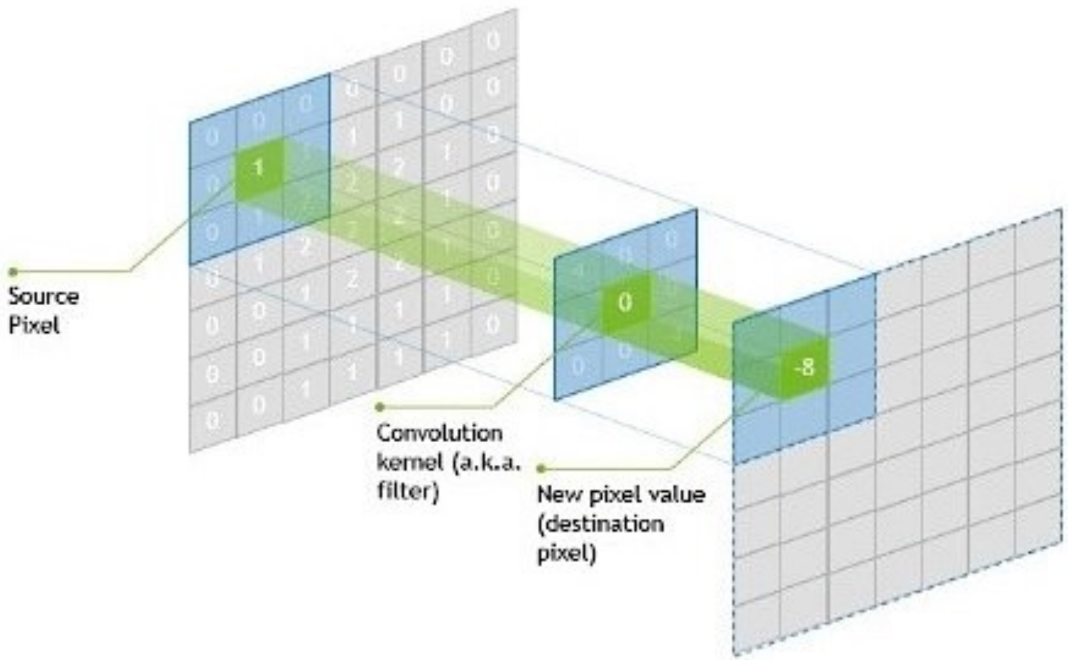
CNNs have an input layer, an output layer, numerous hidden layers, and millions of parameters, allowing them to learn complicated objects and patterns.

Convolution is the process of combining two functions to produce the output of the other function. The input image is convoluted with the application of filters in CNNs, resulting in a Feature Map.

Filters are weights and biases that are randomly generated vectors in the network. Instead of having individual weights and biases for each neuron, CNN uses the same weights and biases for all neurons.

Convolutional Layer

A convolution is a grouping function in mathematics. Convolution occurs in CNNs when two matrices (rectangular arrays of numbers arranged in columns and rows) are combined to generate a third matrix.



Padding & Stride

Padding and stride have an impact on how the convolution procedure is carried out.

Padding and stride can be used to increase or decrease the dimensions (height and width) of input/output vectors.

Padding & Stride

Padding is a term used in convolutional neural networks to describe how many pixels are added to an image when it is processed by the CNN kernel.

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

Padding & Stride

If the padding in a CNN is set to zero, every pixel value-added will have the value zero. If the padding is set to one, a one-pixel border with a pixel value of zero will be added.

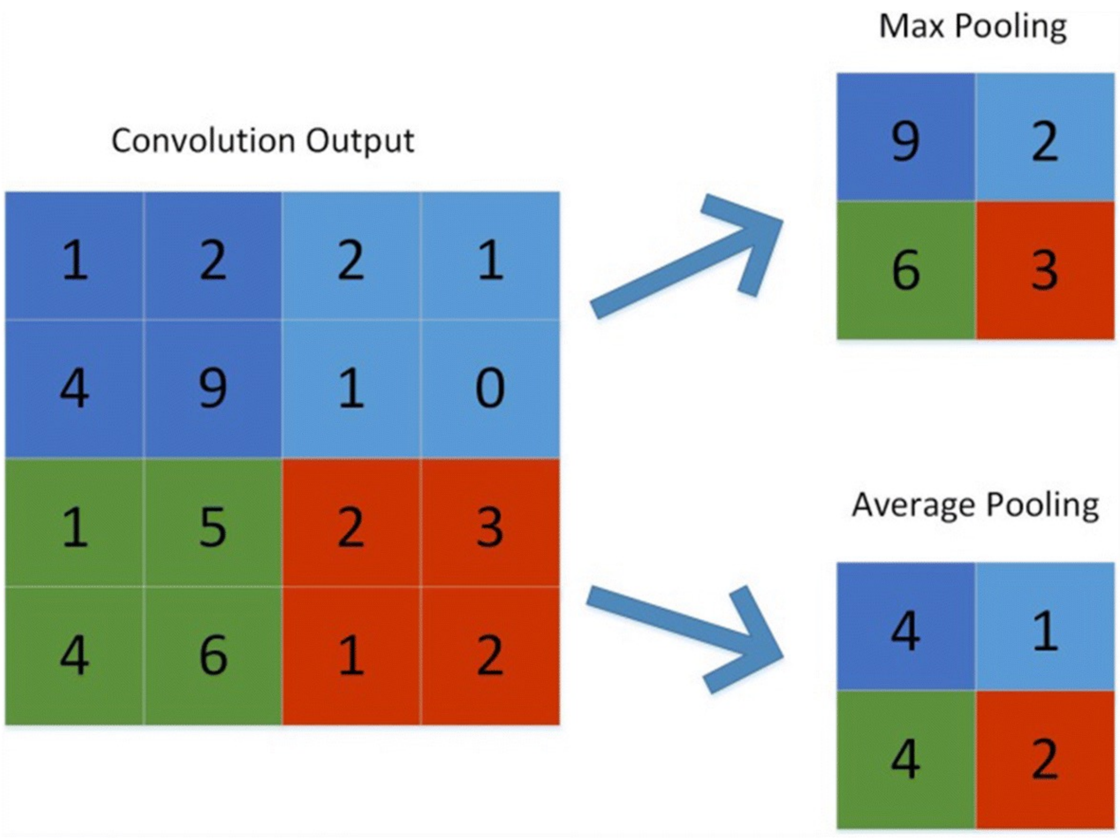
0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

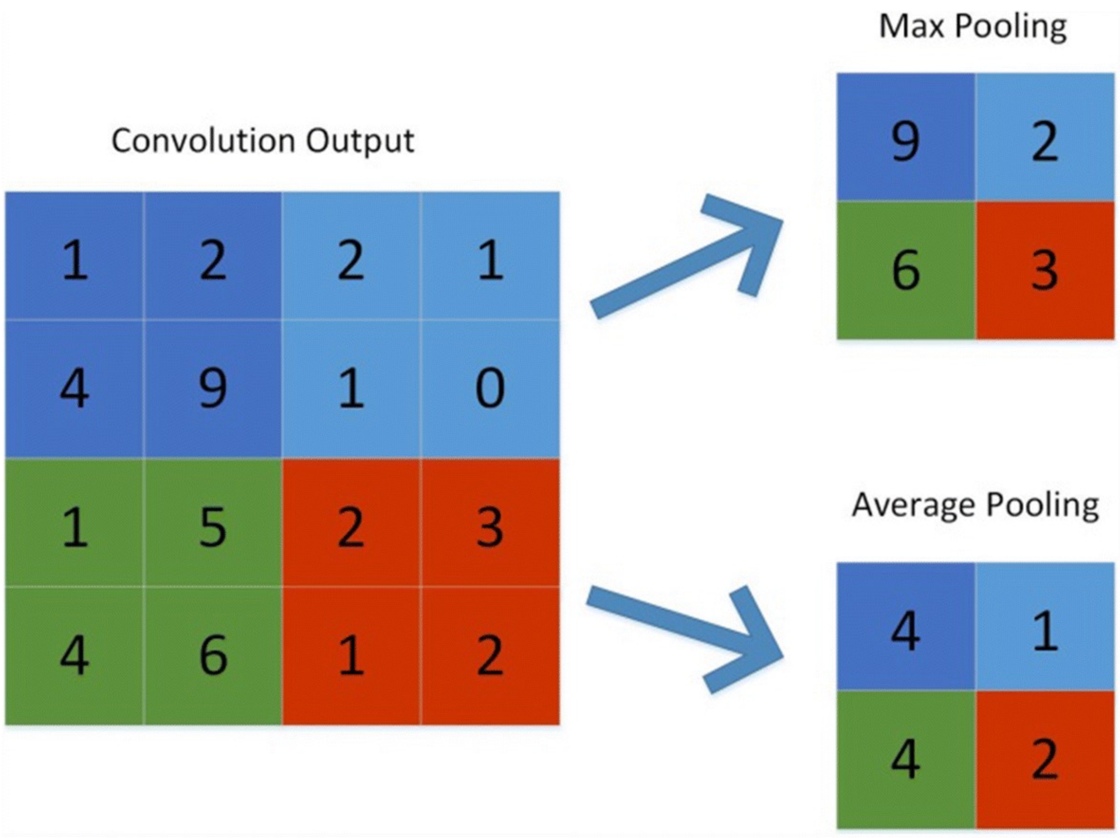
Pooling



Its purpose is to gradually shrink the representation's spatial size to reduce the number of parameters and computations in the network.

The pooling layer treats each feature map separately.

Pooling

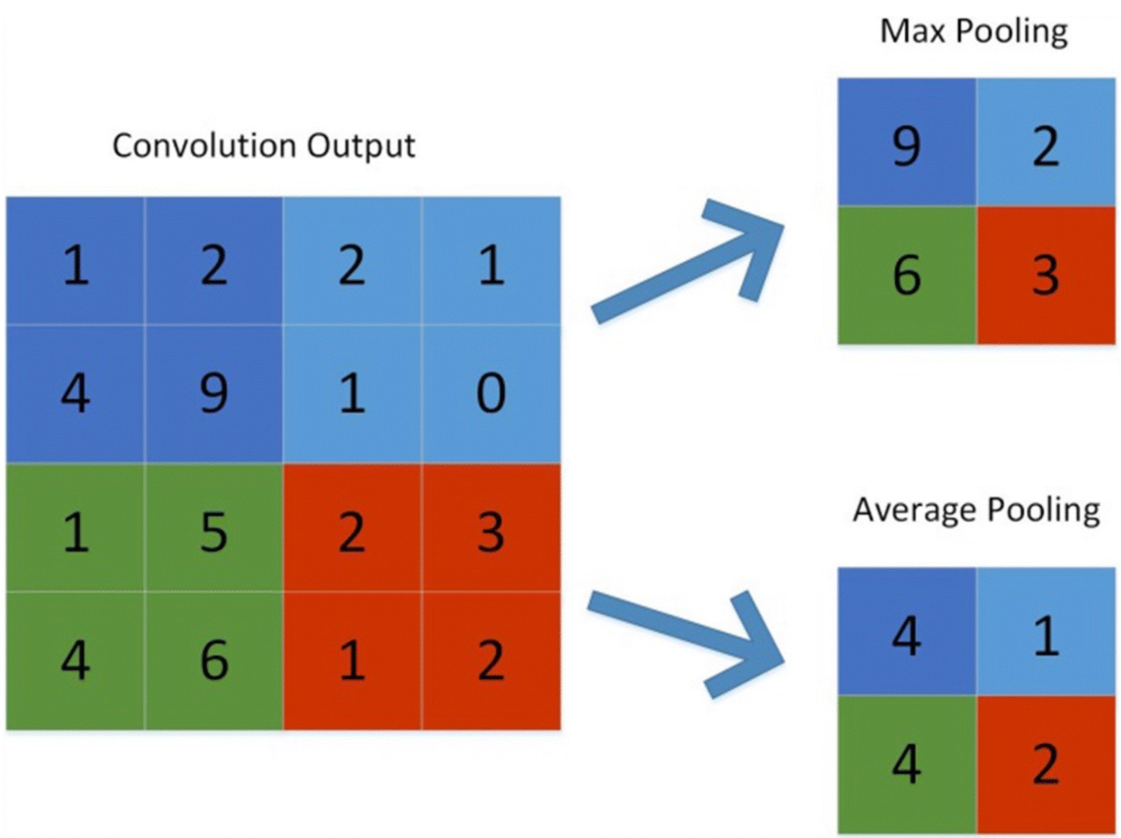


Max-pooling:

It chooses the most significant element from the feature map.

It is the most popular method since it produces the best outcomes.

Pooling

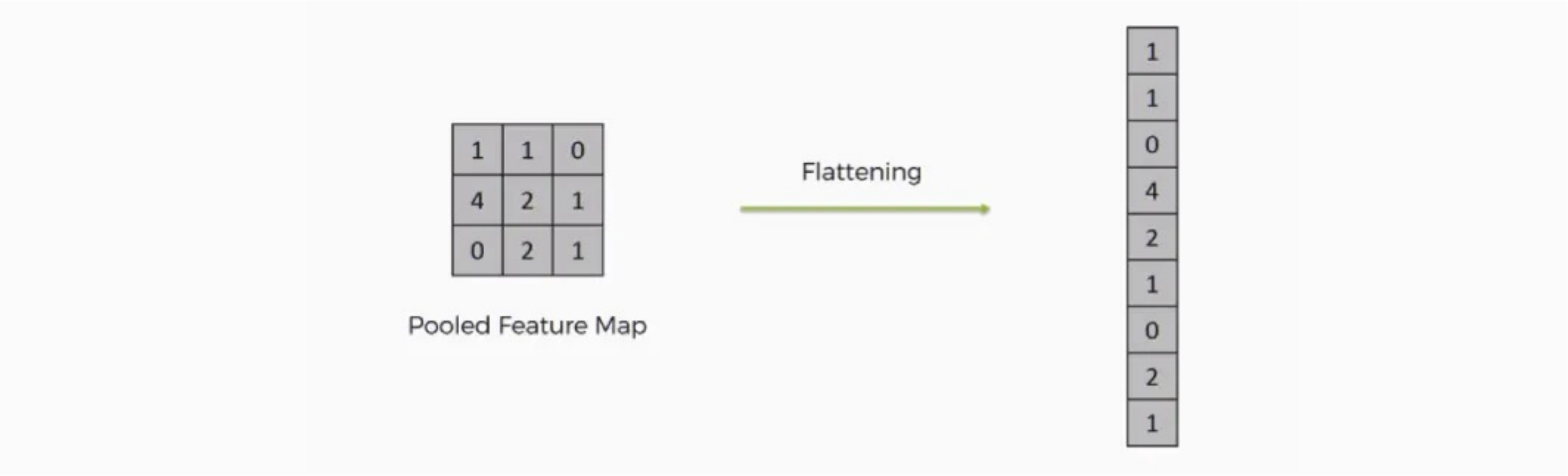


Average pooling:

It entails calculating the average for each region of the feature map.

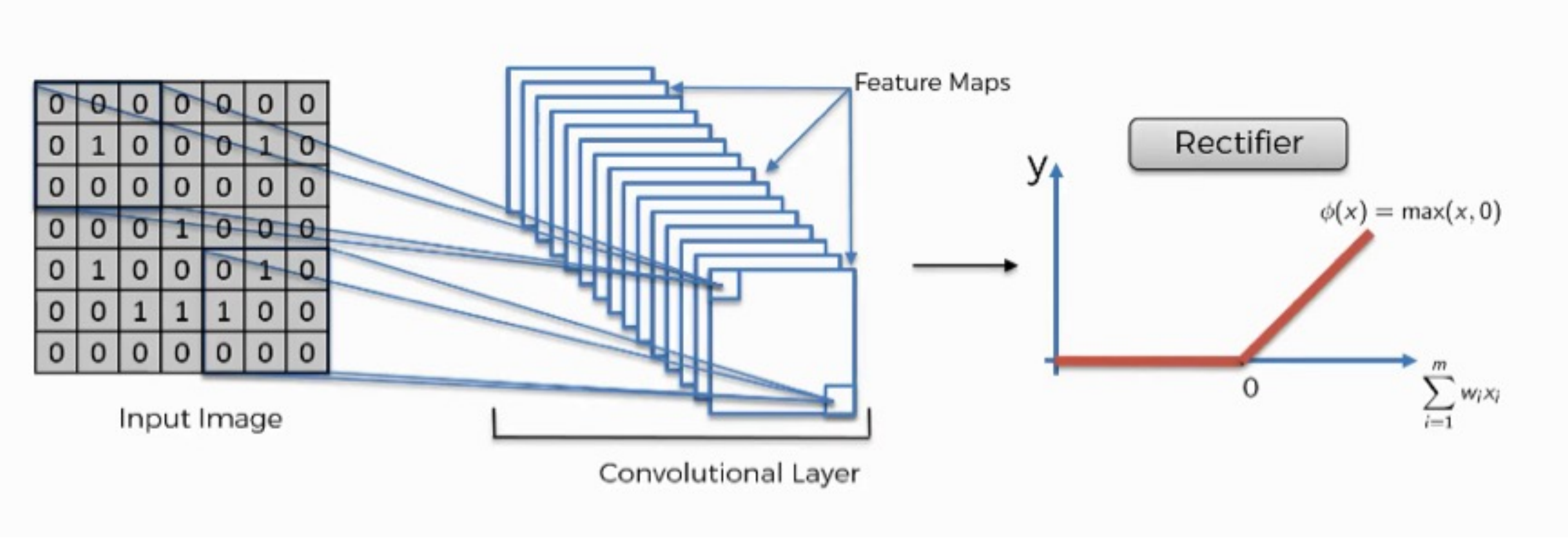
Flattening

Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fed as input to the fully connected layer to classify the image.



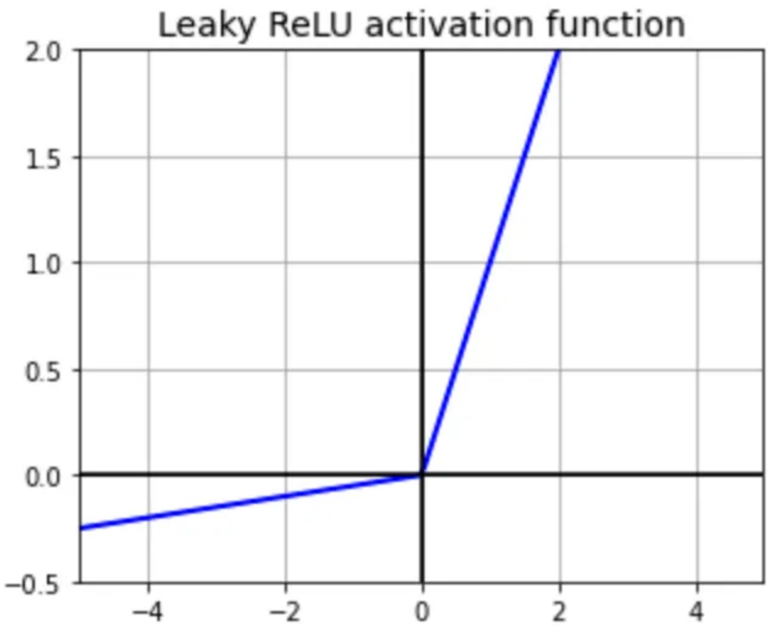
ReLU

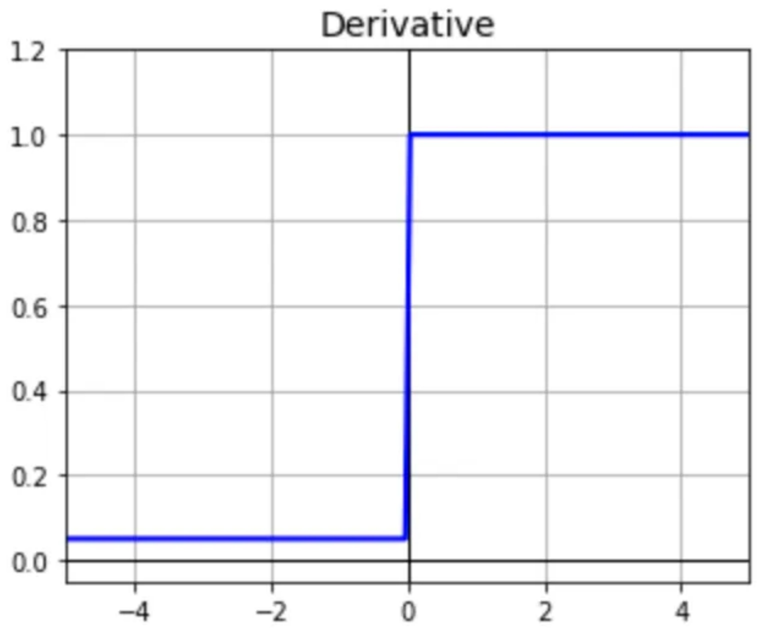
Because a model that utilizes it is quicker to train and generally produces higher performance, it has become the default activation function for many types of neural networks.

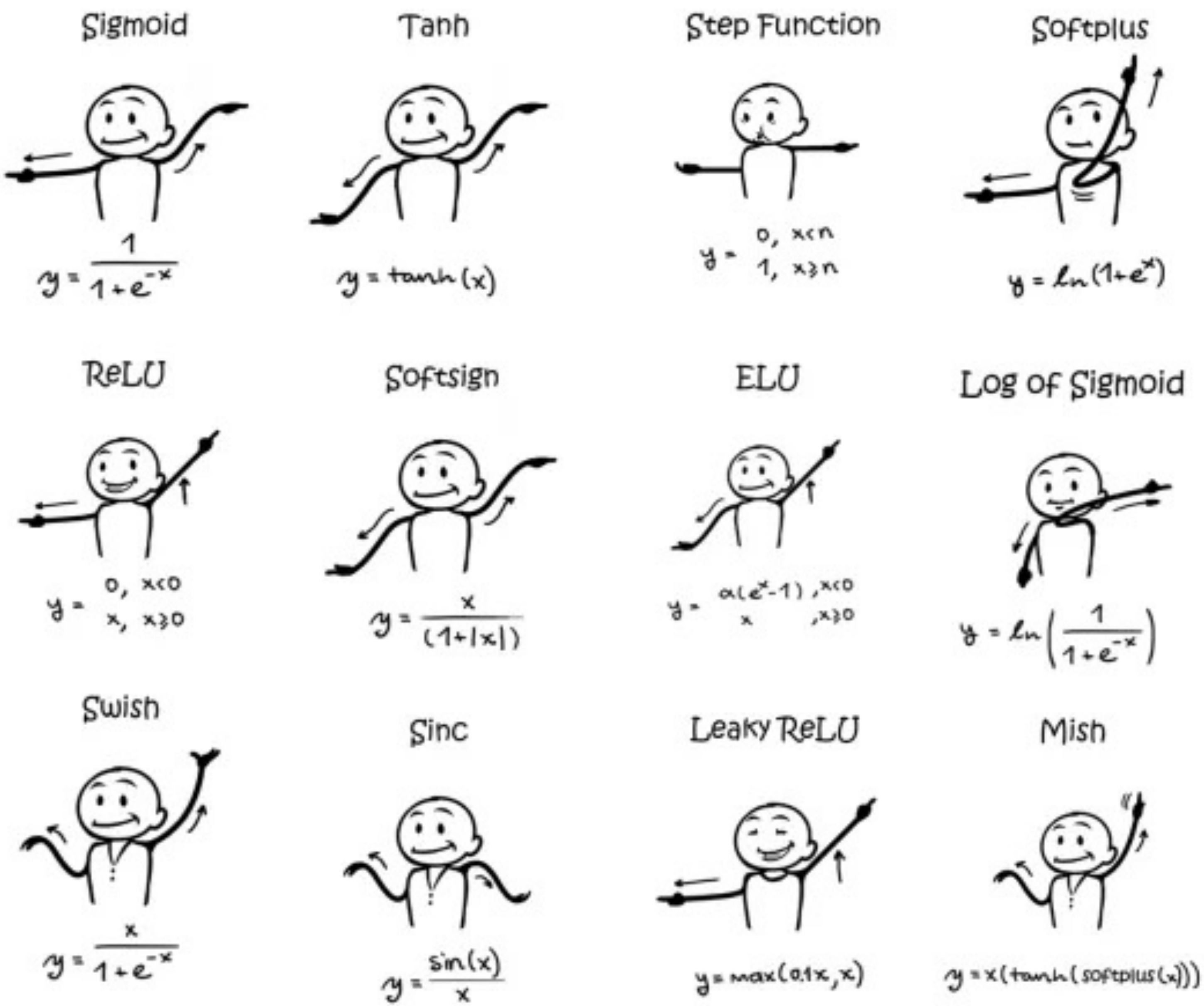


Leaky ReLU

We give some small positive alpha value so that whole activation will not becomes zero. The hyperparameter alpha (typically set to 0.01) defines how much the function leaks.







Other Activation Functions

Soft-Max

Soft-max is an activation layer that is typically applied to the network's last layer, which serves as a classifier.

This layer is responsible for categorizing provided input into distinct types.

A network's non-normalized output is mapped to a probability distribution using the soft-max function.

Batch Normalization

Normalization of any data is about finding the mean and variance of the data and normalizing the data so that the the data has zero mean and unit variance.

Batch Normalization

Normalization of any data is about finding the mean and variance of the data and normalizing the data so that the the data has zero mean and unit variance.

Consider we have d number of hidden units in a hidden layer of any deep neural network. We can represent the activation values of this layer as x = [x1,x2,………xd].

Now we can normalize the kth hidden unit activation using the formula bellow:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization

Here x^ is the normalized value of the kth hidden unit.

E(x^k) is the expectation of the kth units' values also called the mean value.

Var(x^k) is the variance of the kth hidden unit.

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization

After normalization each hidden unit will have zero mean and unit variance but we typically do not want 0 mean and variance of 1. Instead we want the network to learn and adapt these mean and variance values.

For this we introduce 2 new variable, one for learning the mean and other for variance.

The final normalized scaled and shifted version of the hidden activation for the kth hidden unit is given bellow:

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}.$$

Mini-Batch Batch Normalization

Typically, when we train an NN, we don't feed the entire data in one shot.

we use mini-batch of size 32, 64, 128, etc.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

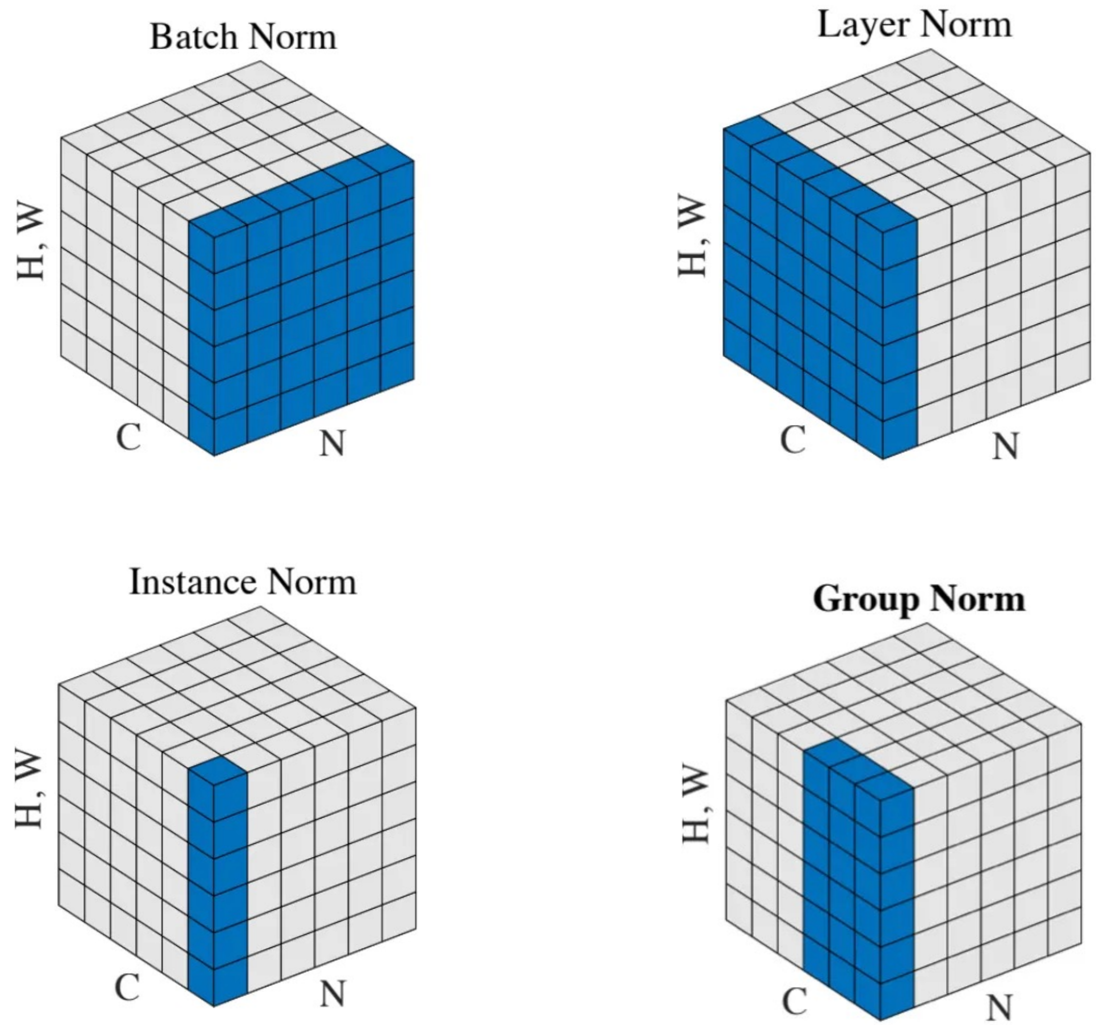
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Other Normalization Techniques

Spatial size of H x W.

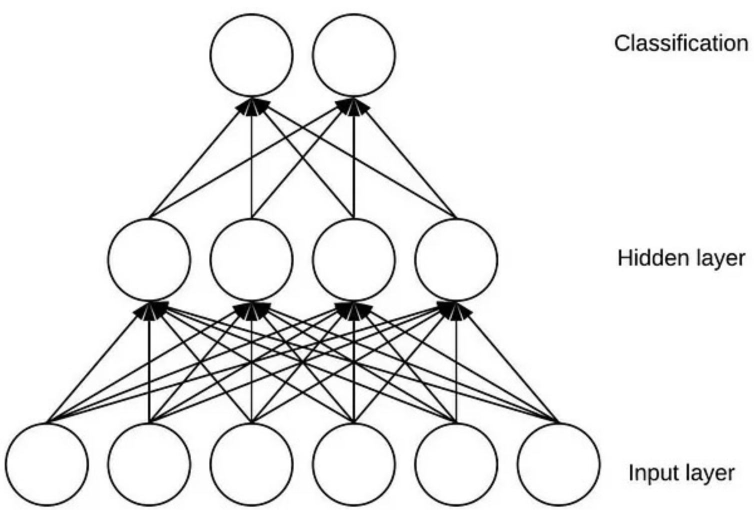
Channels size of C.

Batch size of N.

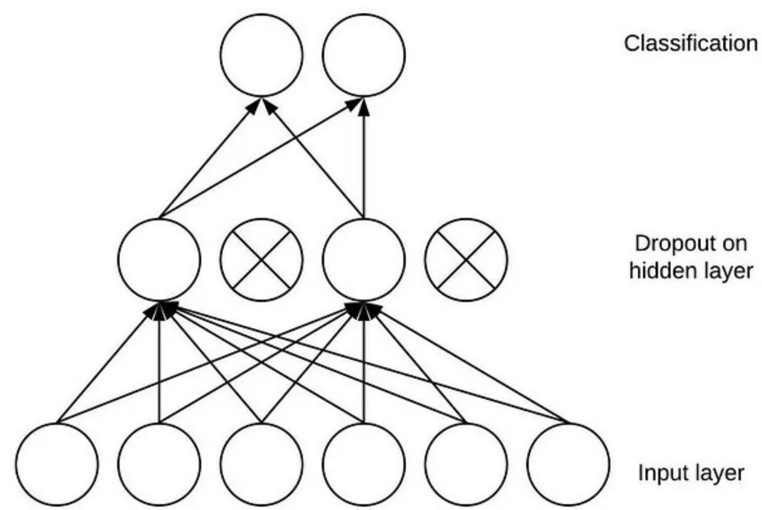


Dropout

To avoid overfitting (when a model performs well on training data but not on new data), a dropout layer is utilized, in which a few neurons are removed from the neural network during the training phase, resulting in a smaller model.



Without Dropout



With Dropout

Lab Session 8

Image Classification

NUS National University of Singapore | School of Computing