Convolutional Neural Network

Jungtaek Kim (jtkim@postech.ac.kr)

Machine Learning Group,
Department of Computer Science and Engineering, POSTECH,
77 Cheongam-ro, Nam-gu, Pohang 37673,
Gyeongsangbuk-do, Republic of Korea

March 6, 2019

Table of Contents

Convolutional Neural Network

Convolutional Neural Network

- Convolutional neural network (CNN) is one of feed-forward neural networks.
- It usually solves a classification task.
- It can be composed of convolutional layers, pooling layers, and fully-connected layers.
- An activation function, which is a non-linear transformation can be applied in a layer.
- Usually, three-dimensional data (for multi-channel image case) is fed into an input layer, and the last layer produces an output as one-hot representation (for classification task).

Convolutional Neural Network

- A regular three-layer neural network.

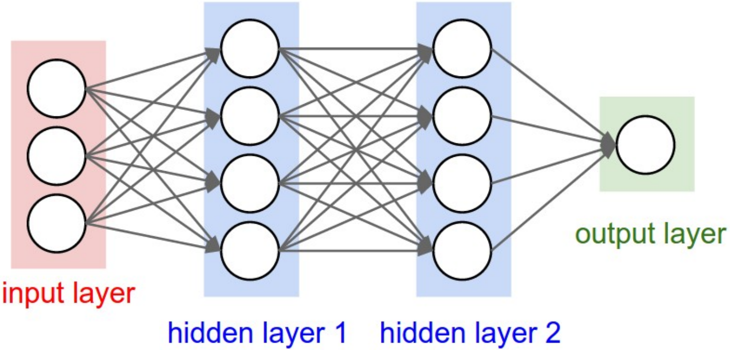

Figure 1: http://cs231n.github.io/convolutional-networks/

Convolutional Neural Network

- CNN is built with many neurons in three dimensions.

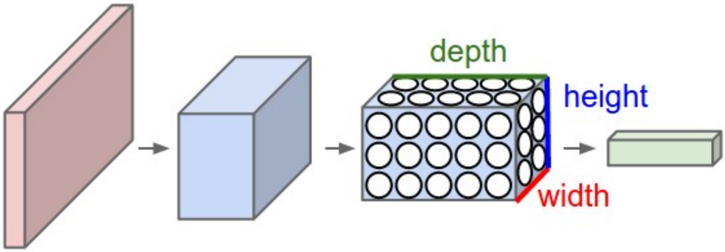

Figure 2: `http://cs231n.github.io/convolutional-networks/`

Architecture of Convolutional Neural Network

- From now, the components of CNN is introduced in detail.
- Components of CNN for classification task:
  - input layer
  - convolutional layer
  - pooling layer
  - fully-connected layer
  - output layer
  - activation fuction
  - cross entropy.

Input Layer

- In this slides, we use an image dataset as training and test datasets.

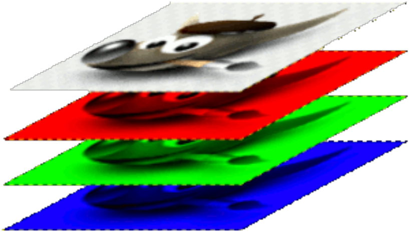

Figure 3: `https://docs.gimp.org/2.4/en/gimp-images-in.html`

Convolutional Layer

- Convolutional layer consists of a set of learnable filters (or kernels).
- It usually has parameters as four-dimensional tensor, `(kernel width, kernel height, input channel, output channel)`.
- It is known to capture local features in images.

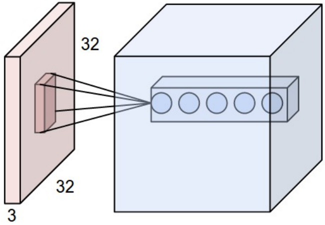

Figure 4: `http://cs231n.github.io/convolutional-networks/`

Convolutional Layer

- It reduces the number of parameters, connecting neurons locally and sharing their parameters.
- To apply the filters, `stride` and `zero-padding` should be set.

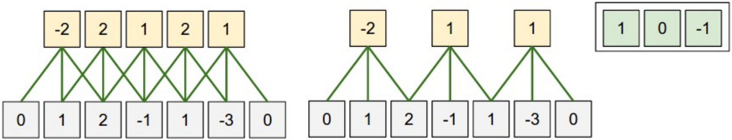

Figure 5: `http://cs231n.github.io/convolutional-networks/`

Convolutional Layer

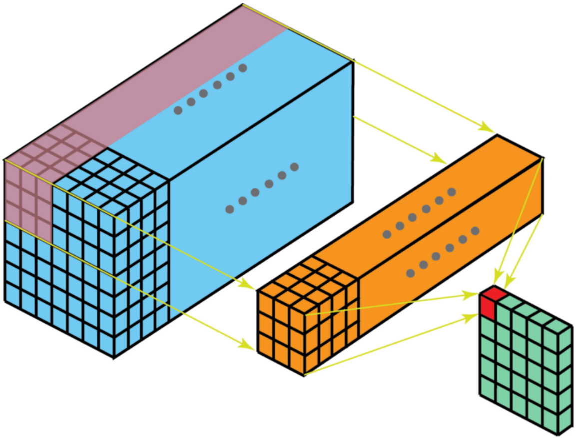

Figure 6: https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215.
$k \times k \times d$ convolutional filter.

Convolutional Layer

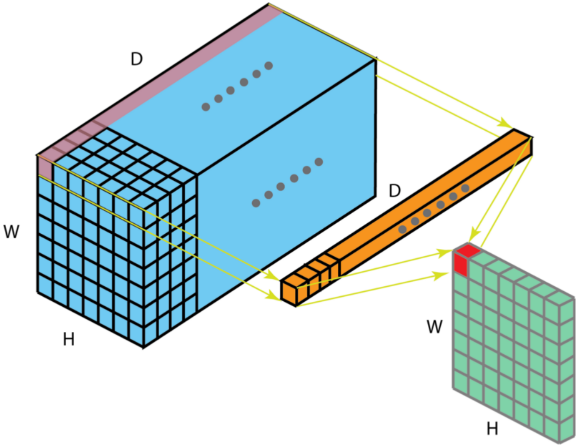

Figure 7: https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215.
$1 \times 1 \times d$ convolutional filter.

Convolutional Layer

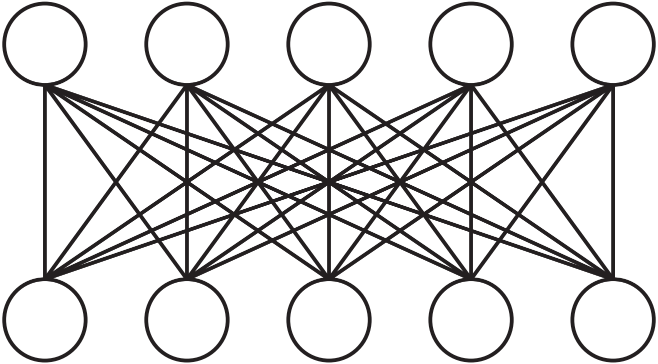

Figure 8: Fully-connected layer. All lines indicate different weights.

Convolutional Layer

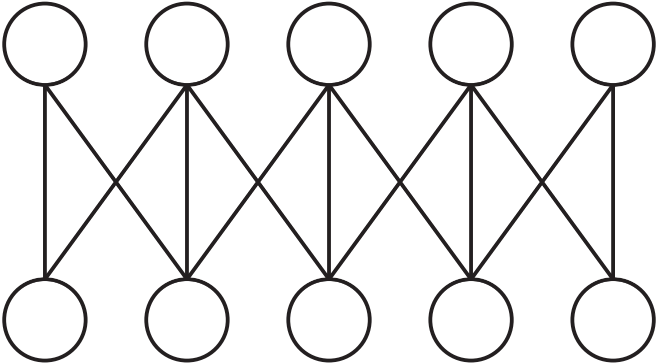

Figure 9: Locally-connected layer. All lines indicate different weights.

Convolutional Layer

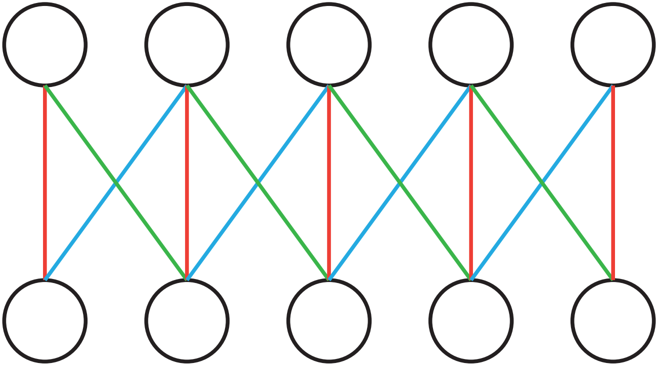

Figure 10: Convolutional layer. Same color indicates same weight.

Convolutional Layer

- If 1000×1000 image is given and the number of neurons is 1000, we need $\mathbf{10^9}$ **parameters** for a **fully-connected layer** case.
- If 1000×1000 image is given, the number of filters is 1000, and a filter size is 10×10, we need $\mathbf{10^5}$ **parameters** for a **convolutional layer** case.

Pooling Layer

- Pooling layer downsamples an input tensor with respect to width and height.
- It can reduce the number of parameters and control overfitting.

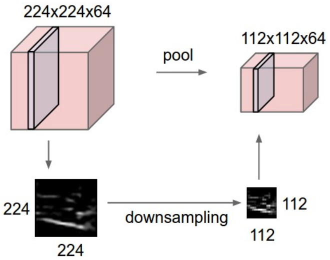

Figure 11: http://cs231n.github.io/convolutional-networks/

Pooling Layer

- There are several methods to downsample, such as max pooling and average pooling.
- For back-propagation, the indices of downsampled nodes are kept during forward-propagation.

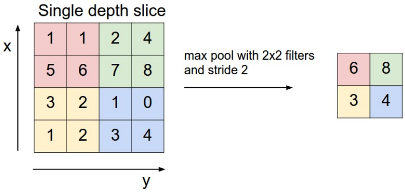

Figure 12: `http://cs231n.github.io/convolutional-networks/`

Fully-Connected Layer

- Fully-connected layer connects all nodes between two layers.
- It can be written as

$$y = \mathbf{w}^\top \mathbf{x} + b.$$

Output Layer

- For classification task, the dimension of output layer is the number of classes.
- Class probabilities of each data can be computed by softmax function:

$$p(\mathbf{z}) = [p(z_1) \cdots p(z_k)]^\top$$

where

$$p(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)}$$

for $1 \leq i \leq k$.

Activation Function

- It is a function to express the switch which has two outputs, ON and OFF.
- In a neural network field, it is non-linear and its shape is usually sigmoid.
- There are several activations such as logistic function, hyperbolic tangent function, and rectified linear unit (ReLU).

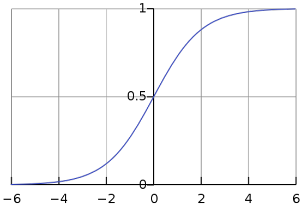

Figure 13: From Wikipedia

Type of Activation Functions

- Sigmoid.
- Logistic: $\sigma(x) = \frac{1}{1+\exp(-x)}$.
- Hyperbolic tangent: $\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$.
- ReLU: $f(x) = \max(0, x)$.
- Leaky ReLU: $f(x) = \mathbf{1}(x < 0)(ax) + \mathbf{1}(x \geq 0)(x)$ where a is a constant.

Logistic Function

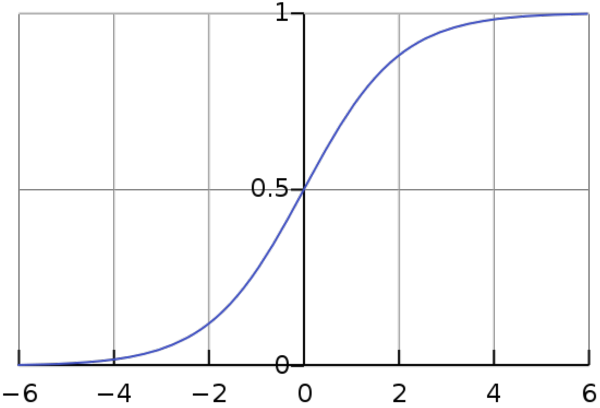

Figure 14: From Wikipedia

Hyperbolic Tangent Function

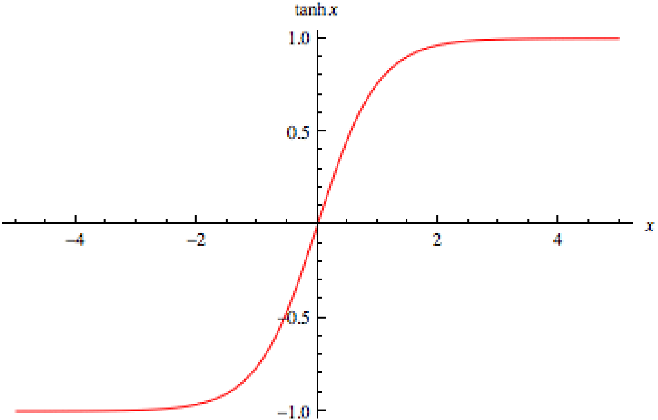

Figure 15: From http://mathworld.wolfram.com/HyperbolicTangent.html

ReLU

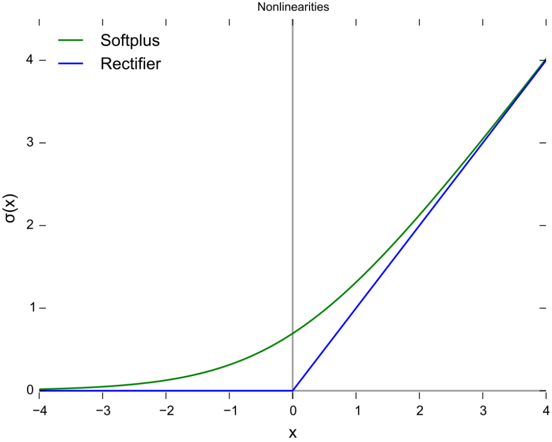

Figure 16: From Wikipedia

Leaky ReLU

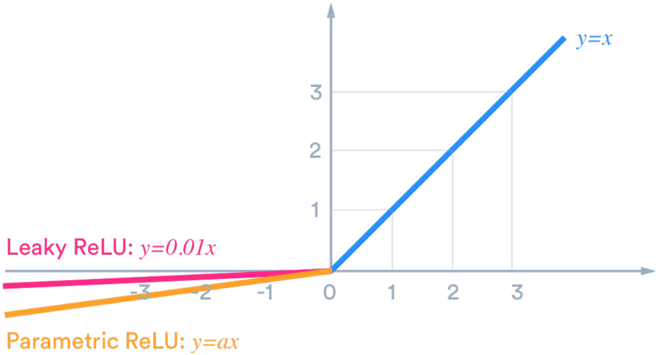

Figure 17: From https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7

Cross Entropy

- Cross entropy is usually used as a loss function for classification task.
- The cross entropy between two probability distributions p and q is defined as

$$H(p,q) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log q(\mathbf{x})$$

for discrete p and q.

Cross Entropy

- Minimizing the cross entropy is equivalent to maximizing log-likelihood:

$$\frac{1}{N}\log\prod_{i=1}^{k} q_i^{Np_i} = \sum_{i=1}^{k} p_i \log q_i = -H(p,q)$$

where N is the number of training data and k is the number of classes.

Famous Architecture of Convolutional Neural Networks

- LeNet-5
- AlexNet
- VGGNet
- GoogLeNet
- Inception-v2, Inception-v3
- ResNet
- Inception-v4

- http://slazebni.cs.illinois.edu/spring17/lec01_cnn_architectures.pdf

LeNet-5

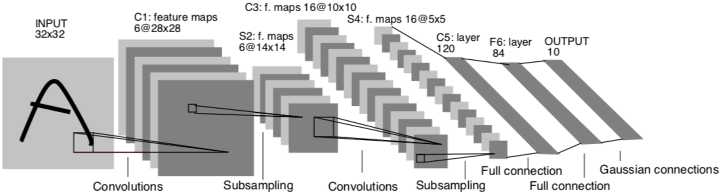

Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Figure 18: From Y. LeCun et al., Gradient-based Learning Applied to Document Recognition, 1998.

AlexNet

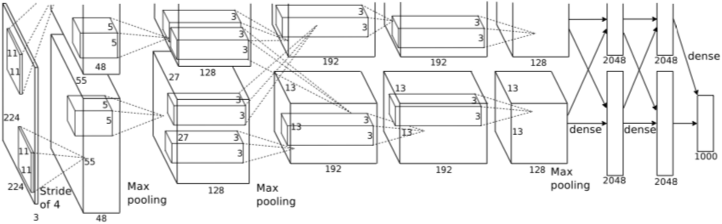

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Figure 19: From A. Krizhevsky et al., ImageNet Classification with Deep Convolutional Neural Networks, 2012.

VGGNet

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 20: From K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, 2015.

GoogLeNet

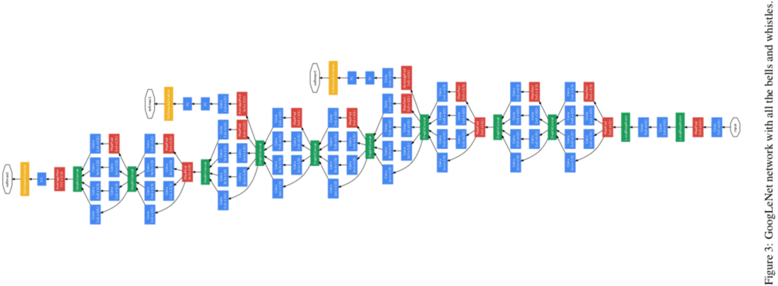

Figure 21: From C. Szegedy et al., Going Deeper with Convolutions, 2014.

GoogLeNet

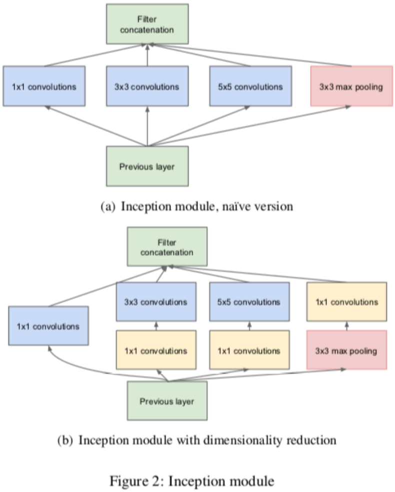

Figure 22: From C. Szegedy et al., Going Deeper with Convolutions, 2014.

ResNet

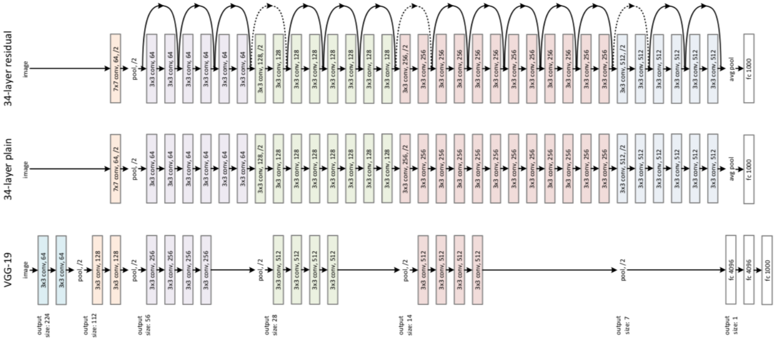

Figure 23: From K. He et al., Deep Residual Learning for Image Recognition, 2015.

ResNet

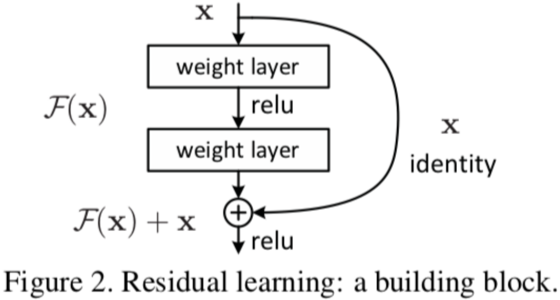

Figure 24: From K. He et al., Deep Residual Learning for Image Recognition, 2015.

References

- Professor Seungjin Choi's materials
- http://cs231n.github.io/convolutional-networks/
- Wikipedia

Thank you.