Deep Learning Revolution

- **Deep Learning** refers to deep neural networks (i.e. many hidden layers)

- The “deep” in Deep learning is NOT “learning that is deep” (i.e. meaningful or sophisticated learning!!)

- The “deep” ONLY refers to the “**depth in layers**” of the neural network.

- **Convolution Networks** are a kind of Deep Neural network.
Taking Inspiration from the Visual Cortex

- Hubel & Wiesel’s experiments with cats and primates
  - Visual cortex is a hierarchical series of layers of neurons. Layers communicate back and forth extensively.
  - Layers act as feature detectors (edges, shapes, objects, etc).
  - Each neuron receives input corresponding to a specific small region of the visual scene (neuron’s receptive field).
  - Neurons activate only if their receptive field contains a particular kind of edge/feature (e.g. horizontal edge, vertical edge, angular edge, etc).
  - Lower-level neurons feed into higher level layers of the visual cortex for detecting shapes, objects, faces, etc.

- This is still a gross simplification. The brain is much more complex!


Convolution Networks – Short History

- Taking inspiration from Hubel & Wiesel...

- From Cognitron, to NeoCognitron (Fukushima, 1970s) showed how a hierarchical network (using Relu!) could learn using unsupervised means.

- To Convolutional Neural Networks (1989) used for hand-written zipcodes. Developed by Yann LeCun (at AT&T Bell Labs). In 1995, LeCun et al developed LeNet-5 to classify handwritten digits (32x32 pixel images). Used to recognize numbers on checks by banks.
**ConvNet Architecture**


**Dense Networks versus Convolution Networks**

- Layers in dense networks learn **global patterns** in the input.
  
  E.g. MINIST Digit recognition: we flattened 28x28 images into 784 units and fed them into the hidden layer.

- Convolution Networks learn **local patterns** in input.
  
  e.g. They look for patterns in small 2D windows (using *patches/filters/kernels*) of input images.

- Convolution Networks learn spatial hierarchies of patterns (e.g., edges, larger patterns, etc.)
Convolution Networks: Core Idea

• Convolution Networks learn **local patterns** in input

  e.g. They look for patterns in small 2D windows (using *patches/filters/kernels*) of input images.

• Convolution Networks learn **spatial hierarchies of patterns** (e.g., edges, larger patterns, etc.)

Convolution Networks: Basic Elements

• **Convolution** (using filters/patches/kernels)

  Small patches of input image/map are filtered to recognize local features

• **Feature Maps**

  Starting with an input image (e.g. 28x28x1)
  Using a 3x3 filter to get K output maps (26x26xK)

• **Max Pooling**

  A way of down sampling a feature map.
ConvNet Architecture: Convolutions


Filters/Patches/Kernels

Filters/Patches/Kernels


Filters: Padding

• “valid” Padding
  No padding of input is done. Reduces the width and height of resulting map.

• “same” Padding
  Pads input in such a way to preserve the width and height of the resulting map.

Input and Output Feature Maps

ConvNet Architecture: Pooling

Max Pooling

• Pooling uses a kernel/patch/window

There two popular kinds: **Max pooling**, Average Pooling

• It is purely an arithmetic operation that helps further downsample a feature map (i.e. performs dimensionality reduction useful for reducing computational load).

• We typically use a 2x2 window.

---

ConvNet Architecture

ConvNet for MNIST Digits Recognition

**INPUT**
(28 x 28 x 1)

**Convolution** (3 x 3) Kernel valid padding
**MaxPooling** (2 x 2)

**Convolution** (3 x 3) Kernel valid padding
**MaxPooling** (2 x 2)

**Convolution** (3 x 3) Kernel valid padding
**MaxPooling** (2 x 2)

**Fully Connected** with Relu activation
**Fully Connected** with Softmax activation

**Fully Connected**
0
1
2
9
10 units

32 maps
(26 x 26 x 32)

32 maps
(13 x 13 x 32)

64 maps
(11 x 11 x 64)

64 maps
(5 x 5 x 64)

128 maps
(3 x 3 x 128)

1152 units

Parameters
(9 x 32 = 288
288 + 32 bias = 320)

Parameters
(32 x 9 x 64 = 18432
18432 + 64 bias = 18496)

Parameters
(64 x 9 x 128 = 73728
73728 + 128 bias = 73856)

Parameters
(1152 x 10 = 11520
11520 + 10 bias = 11530)

**Total Parameters**

(320 x 18496 + 73856 + 11530 = 104202)
Over to Colab

- [https://colab.research.google.com/drive/1XETuFUE9lZwBFwYjGK1lU1mT6HkyC1-h?usp=sharing](https://colab.research.google.com/drive/1XETuFUE9lZwBFwYjGK1lU1mT6HkyC1-h?usp=sharing)

ConvNet Architecture Patterns

- Overfitting/Dropout
  When models tend to overfit, a way to resolve overfitting is to use dropout. This is called a regularization technique.
  A dropout layer can be added in between layers. Typically, a dropout layer eliminates (sets to 0) 20-50% of the outputs (dropout value can be set during model/layer specification).

- Data/Batch Normalization
  Normalizing the values on inputs and outputs helps with gradient propagation and allows for deeper networks. Hence liberally used in very deep network architectures.

- Data Augmentation
  When datasets are small, “new” data can be created by transforming images in the dataset using flipping, rotation, zooming, etc. to augment the dataset with transformed images. Helps avoid overfitting and leads to better generalization.
ConvNets Applications

• Input Image (1)
• Image classification
• Image Segmentation (3, 4)
• Object Detection (3)

From: https://www.smart-interaction.com/2022/07/14/computer-vision-the-ultimate-guide-on-the-4-main-tasks/

CovNet Architectures

• AlexNet
  One of the early CNNs to use GPUs to speed up training (in 2012).

• VGGNet
  Groundbreaking DL model for Object Recognition.

• There are many more.

From: https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e7b4f9ee30
From: https://medium.com/analytics-vidhya/vggnet-convolutional-network-for-classification-and-detection-3543aaf61699
References