

Convolutional Neural Networks

Weizi Li

Department of Computer Science
University of Memphis

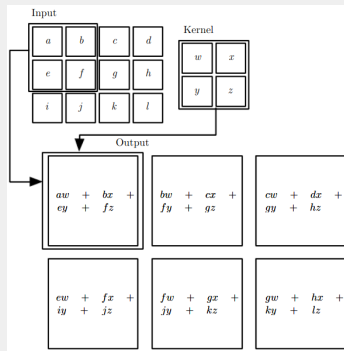


- Convolution
- Pooling
- Activation, Loss, Optimizer
- Example Networks
- Training Techniques
- Example Code

Convolution

- A type of neural network that can process **known grid-structure** data
 - ▶ Timeseries: 1-D grid
 - ▶ Image: 2-D grid
- “Convolution” represents a linear operation, which does not correspond precisely to the definition of convolution as used in other fields

- Three elements: grid-like data (input), kernel (filter), feature map (output)

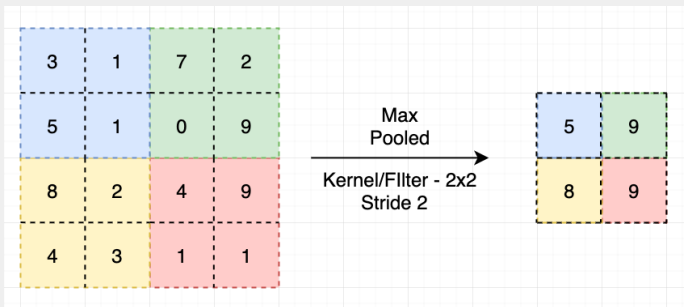


- A feature map is obtained by convolution of the input data with a kernel (image above), adding a bias term and then applying an activation function

- Convolution leverages three important ideas.
 - ▶ Sparse interactions: kernel is smaller than input
 - ▶ Parameter sharing: one set of parameters for every location
 - ▶ Equivariant representations: if the input changes, the output changes in the same way
- Convolution is dramatically more efficient than dense matrix multiplication in terms of memory requirements and statistical efficiency

Pooling

- A pooling function replaces the output of a network at a certain location with a summary statistic of the nearby outputs.
- Examples: max pooling, average pooling, L2 norm pooling



- Pooling helps to make the representation approximately invariant to small translations of the input
- Pooling layers are usually used immediately after convolutional layers

- Padding options: VALID, SAME, etc

"VALID" = without padding:

inputs: 1 2 3 4 5 6 7 8 9 10 11 (12 13) dropped

"SAME" = with zero padding:

inputs:

| | | | | | | | | | | | | | | | | | |
|-----|---|---|---|---|---|---|---|---|---|---|----|----|----|----|-----|---|---|
| pad | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | pad | 0 | 0 |
|-----|---|---|---|---|---|---|---|---|---|---|----|----|----|----|-----|---|---|

Activation, Loss, Optimizer

- Keras activation function list ([link](#))

■ Regression

- ▶ Linear: turn 1 real value to 1 real value

■ Classification

- ▶ Sigmoid: turn 1 real value to 1 real value between 0 and 1
- ▶ Softmax: turn k real values to k real values that sum to 1

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}$$

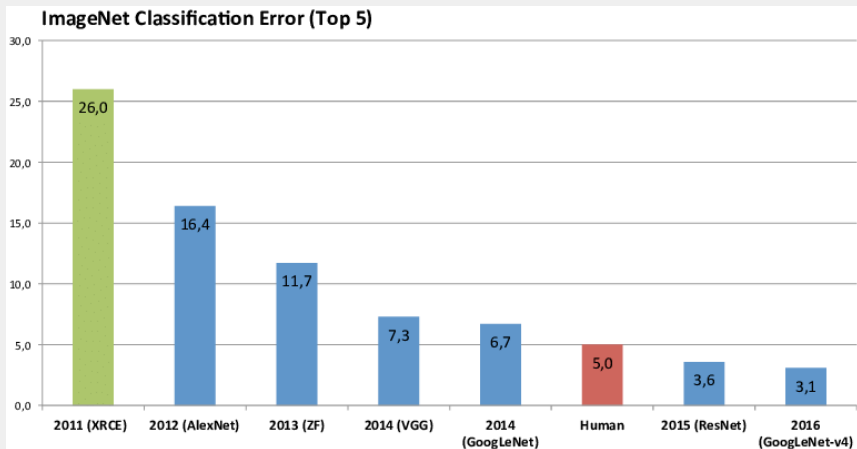
- ▶ E.g., $\vec{z} = [8, 5, 0] \rightarrow [0.9523, 0.0474, 0.0003]$

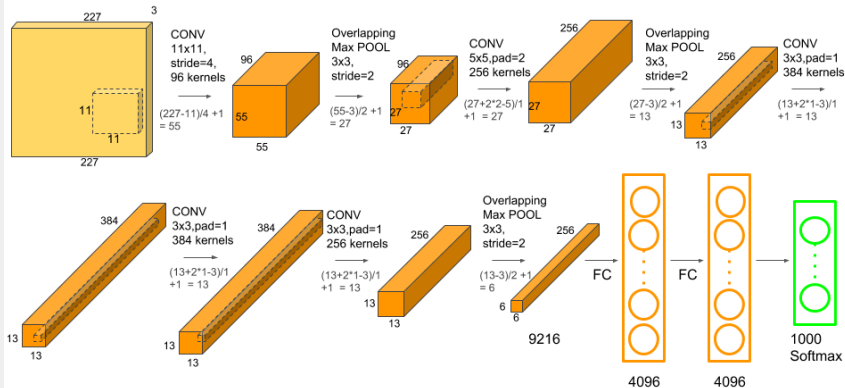
- Keras loss function list ([link](#))

- Keras optimizer list ([link](#))

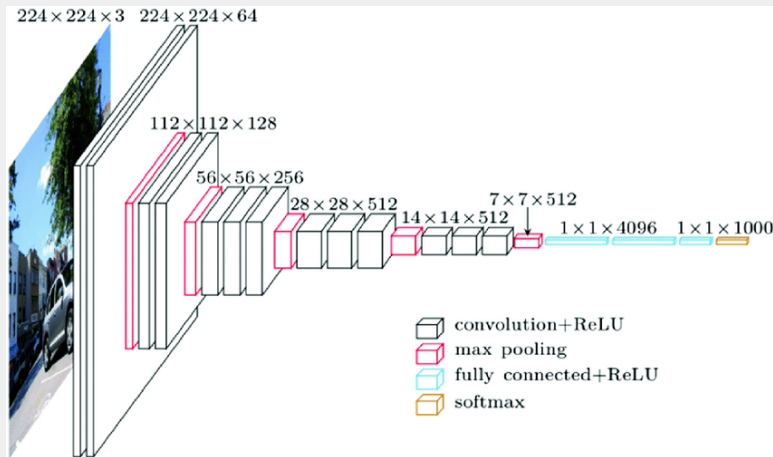
Example Networks



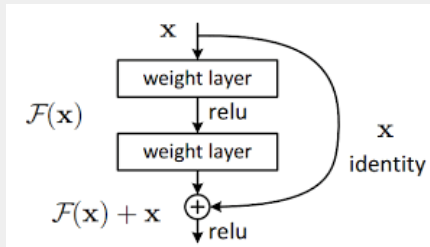


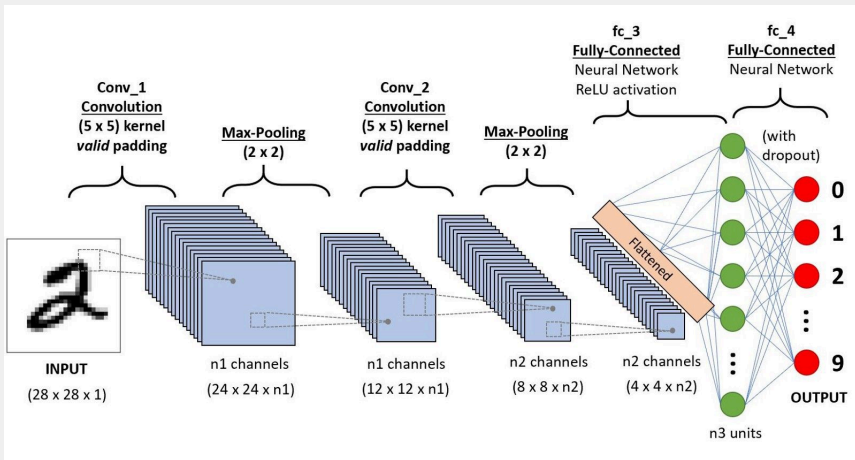


■ 2 Conv + 2 Conv + 3 Conv + 3 Conv + 3 Conv + 3 Dense



- Original version: 152 layers (first network beat human performance)
- Many variants, e.g., ResNet-18, ResNet-50
- Residual learning (skip operation) helps train very deep networks by addressing the vanishing/exploding gradient problem





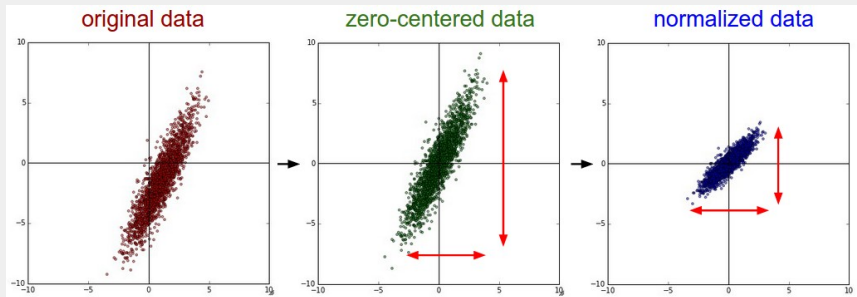
- NVIDIA's model
- Paper ([link](#))
- Video ([link](#))
- Blog ([link](#))

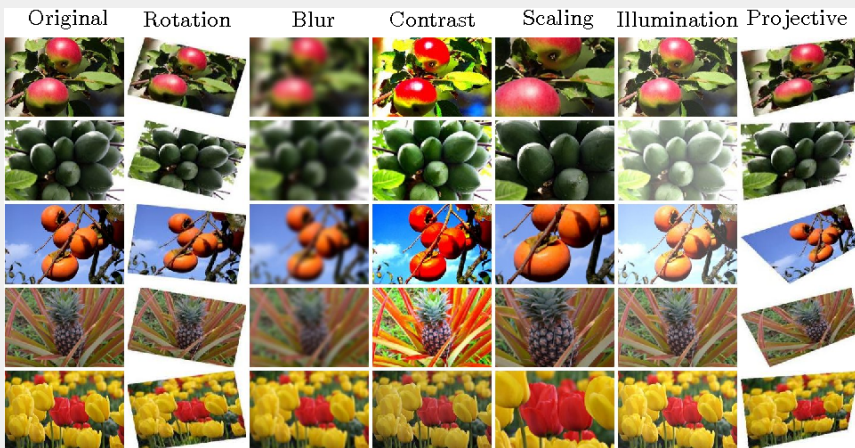
| Layer (type) | Output Shape | Param # |
|--------------------------------|--------------------|---------|
| conv2d_1 (Conv2D) | (None, 26, 26, 32) | 320 |
| conv2d_2 (Conv2D) | (None, 24, 24, 64) | 18496 |
| max_pooling2d_1 (MaxPooling2D) | (None, 12, 12, 64) | 0 |
| dropout_1 (Dropout) | (None, 12, 12, 64) | 0 |
| flatten_1 (Flatten) | (None, 9216) | 0 |
| dense_1 (Dense) | (None, 128) | 1179776 |
| dropout_2 (Dropout) | (None, 128) | 0 |
| dense_2 (Dense) | (None, 10) | 1290 |
| Total params: 1,199,882 | | |
| Trainable params: 1,199,882 | | |
| Non-trainable params: 0 | | |

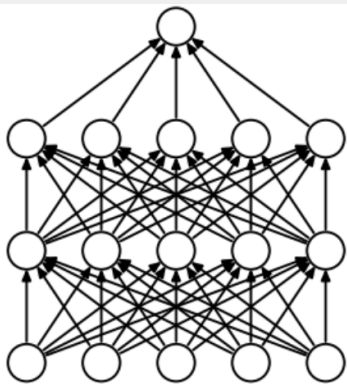
- Input shape is $28 \times 28 \times 1$ (this is data, not trainable parameters)
- conv2d_1: $32 \times 3 \times 3 + 32 = 320$ (32 2-D kernels, each is 3×3 , plus a bias term for each kernel)
- conv2d_2: $32 \times 3 \times 3 \times 64 + 64 = 18496$ (64 3-D kernels, each is $3 \times 3 \times 3$, plus a bias term for each kernel)
- dense_1: $9216 \times 128 + 128 = 1179776$
- dense_2: $128 \times 10 + 10 = 1290$
- Total number of trainable parameters: 1,199,882

Training Techniques

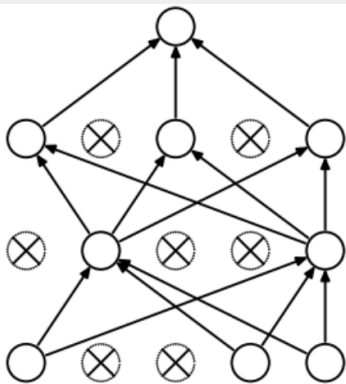
- Linear normalization (know min and max): $\frac{x - \min}{\max - \min}$, e.g., pixel values
- Z-score normalization: $\frac{x - \text{mean}(x)}{\text{std}(x)}$







(a) Standard Neural Net



(b) After applying dropout.

- grid search: simplest technique
- random search: may perform better than grid search
- coarse to fine: start with broad sweep, then zero in

Example Code

- [link](#)

- Tensorflow ([link](#))
- Keras ([link](#))
- Pytorch ([link](#))