# Convolutional Neural Networks

# Why Convolutional Neural Networks? Shortcomings of ANN!

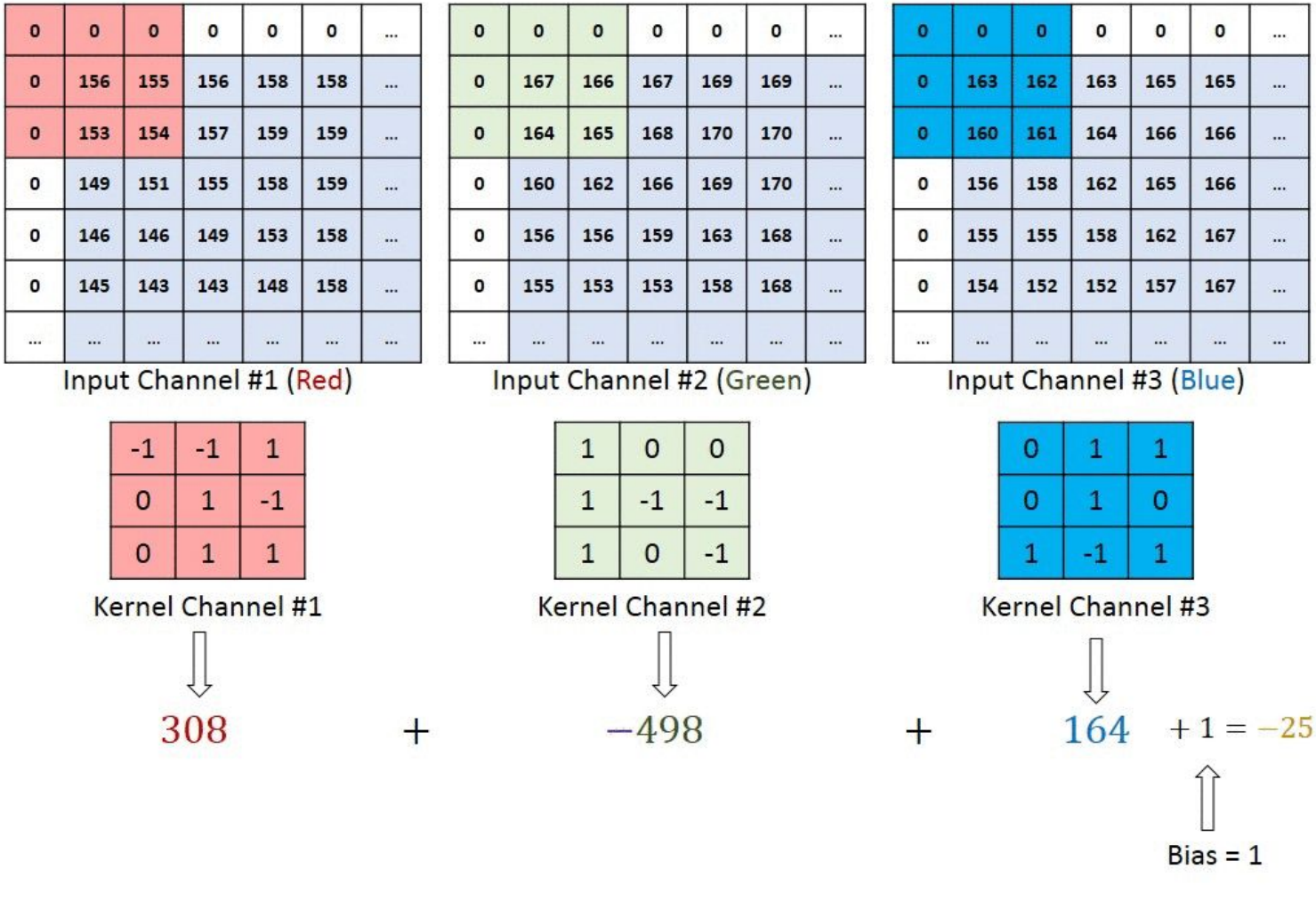
- CNNs are regularized versions of multilayer perceptrons. MLPs usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The full connectivity of these networks make them prone to overfitting data.
- CNNs take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters.
- Feedforward Neural Networks are generally impractical for larger input sizes of images such as high resolution images. It would require a very high number of neurons, even in a shallow architecture, since each pixel is a relevant input feature.
- Hence, convolution reduces the number of free parameters allowing the network to be deeper. How exactly does it do this? We will see.

# What exactly is a CNN?

- Class of ANN to analyse visual imagery, like image recognition, image classification, image segmentation, video recognition, etc.
- CNNs are also known as Shift Invariant or Shift Invariant Artificial Neural Networks, based on the shared-weight architecture of the convolution kernels or filters that slide along the input features and provide translation-equivariant responses known as feature maps.
- A CNN consists of an input layer, hidden layers and an output layer. Hidden layers include layers which perform dot product of the convolutional kernel with the layer's input matrix. This is generally followed by the activation function, mostly ReLU.
- There are other layers as well, Pooling layers, fully connected layers, normalization layers, etc.

# Convolutional Layers

- In a CNN, the input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) x (feature map height) x (feature map width) x (feature map channels).
- Using regularized weights over fewer parameters avoids the vanishing gradients and exploding gradients problems seen during backpropagation in traditional neural networks. Furthermore, convolutional neural networks are ideal for data with a grid-like topology (such as images) as spatial relations between separate features are taken into account during convolution and/or pooling.


0 0 0 0 0 0 ...
0 156 155 156 158 158 ...
0 153 154 157 159 159 ...
0 149 151 155 158 159 ...
0 146 146 149 153 158 ...
0 145 143 143 148 158 ...
... ... ... ... ... ... ...
Input Channel #1 (Red)
0 0 0 0 0 0 ...
0 167 166 167 169 169 ...
0 164 165 168 170 170 ...
0 160 162 166 169 170 ...
0 156 156 159 163 168 ...
0 155 153 153 158 168 ...
... ... ... ... ... ... ...
Input Channel #2 (Green)
0 0 0 0 0 0 ...
0 163 162 163 165 165 ...
0 160 161 164 166 166 ...
0 156 158 162 165 166 ...
0 155 155 158 162 167 ...
0 154 152 152 157 167 ...
... ... ... ... ... ... ...
Input Channel #3 (Blue)
-1 -1 1
0 1 -1
0 1 1
Kernel Channel #1
1 0 0
1 -1 -1
1 0 -1
Kernel Channel #2
0 1 1
0 1 0
1 -1 1
Kernel Channel #3
308
+
−498
+
164
+ 1 = −25
Bias = 1
Output
-25 ... ... ... ...
... ... ... ... ...
... ... ... ... ...
... ... ... ... ...
... ... ... ... ...

# Pooling Layers

- Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as 2 x 2 are commonly used.
- There are two common types of pooling in popular use: max and average. *Max pooling* uses the maximum value of each local cluster of neurons in the feature map, while *average pooling* takes the average value.
- Intuitively, the exact location of a feature is less important than its rough location relative to other features. This is the idea behind the use of pooling in convolutional neural networks. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters, memory footprint and amount of computation in the network, and hence to also control overfitting. This is known as down-sampling.

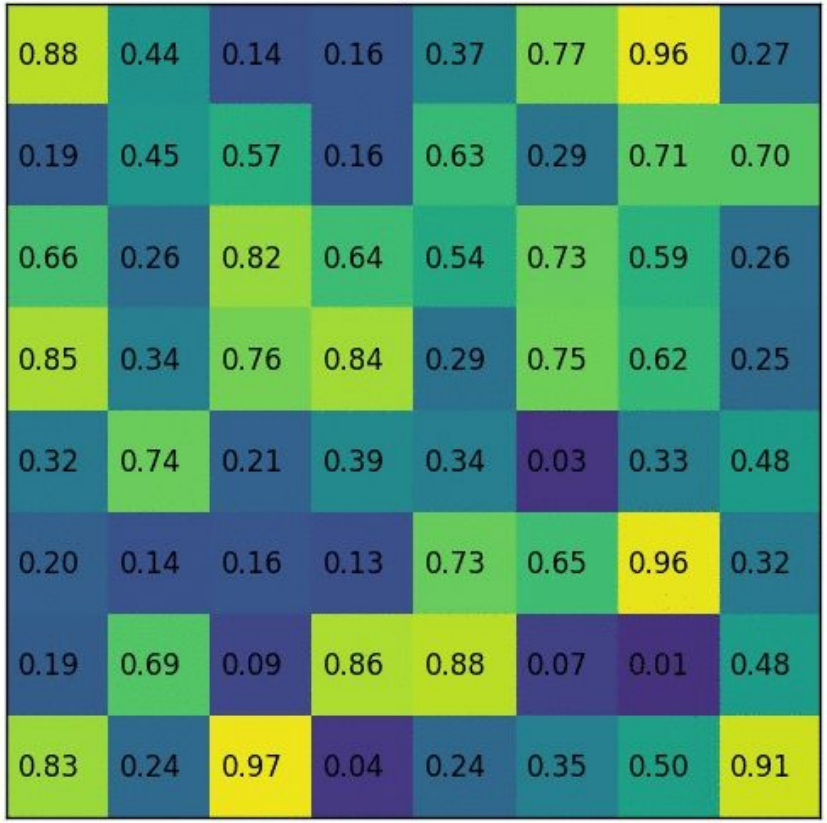
# Pooling Layers

- A very common form of max pooling is a layer with filters of size 2×2, applied with a stride of 2, which subsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations.
- In this case, every max operation is over 4 numbers. The depth dimension remains unchanged (this is true for other forms of pooling as well).
- "Region of Interest" pooling (also known as RoI pooling) is a variant of max pooling, in which output size is fixed and input rectangle is a parameter.

| | | |
|---|---|---|
| 3.0 | 3.0 | 3.0 |
| 3.0 | 3.0 | 3.0 |
| 3.0 | 2.0 | 3.0 |

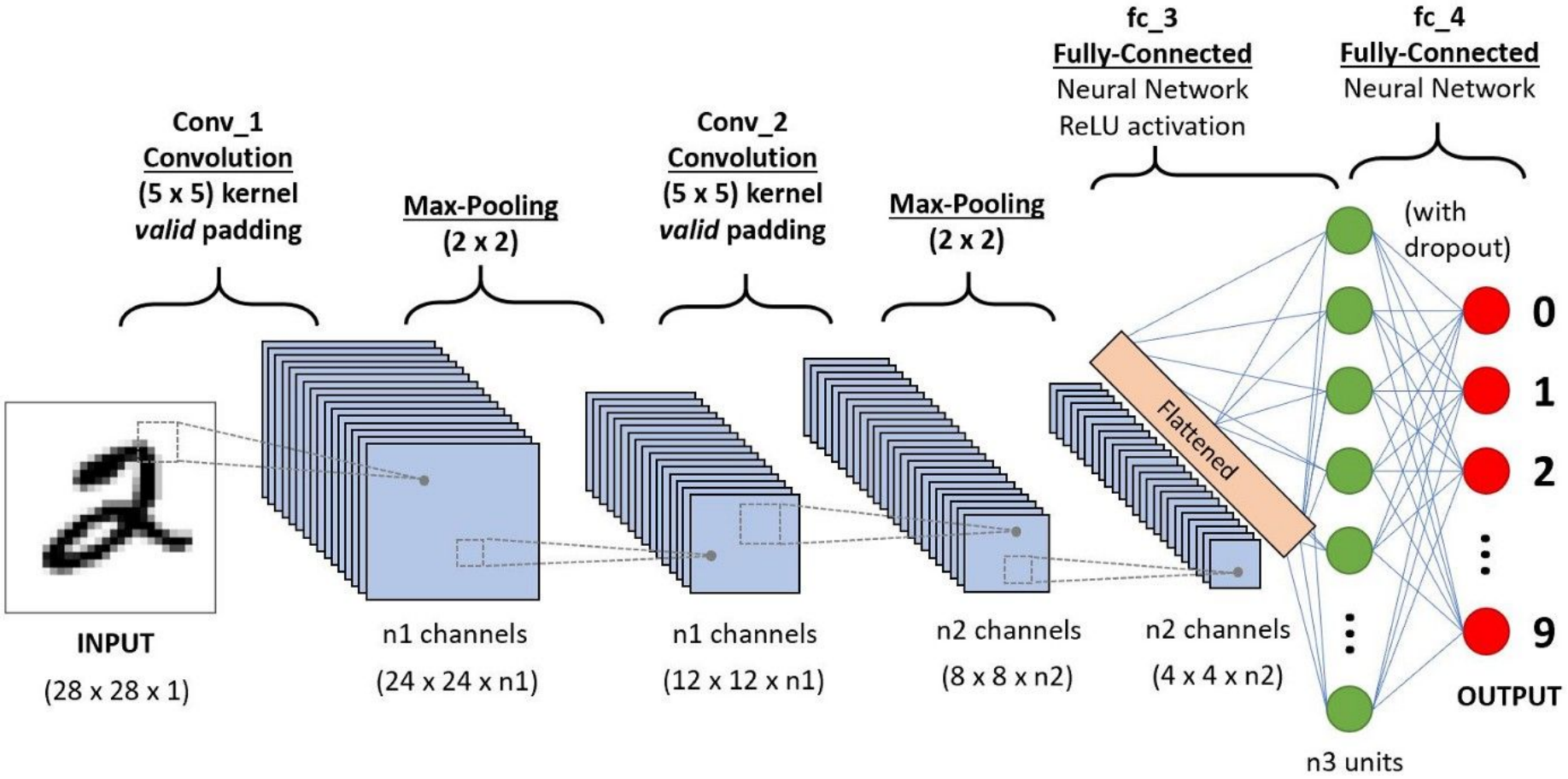
| | | | | |
|---|---|---|---|---|
| 3 | 3 | 2 | 1 | 0 |
| 0 | 0 | 1 | 3 | 1 |
| 3 | 1 | 2 | 2 | 3 |
| 2 | 0 | 0 | 2 | 2 |
| 2 | 0 | 0 | 0 | 1 |

input


0.88 0.44 0.14 0.16 0.37 0.77 0.96 0.27
0.19 0.45 0.57 0.16 0.63 0.29 0.71 0.70
0.66 0.26 0.82 0.64 0.54 0.73 0.59 0.26
0.85 0.34 0.76 0.84 0.29 0.75 0.62 0.25
0.32 0.74 0.21 0.39 0.34 0.03 0.33 0.48
0.20 0.14 0.16 0.13 0.73 0.65 0.96 0.32
0.19 0.69 0.09 0.86 0.88 0.07 0.01 0.48
0.83 0.24 0.97 0.04 0.24 0.35 0.50 0.91

# Fully Connected Layer

- After several convolutional and max pooling layers, the final classification is done via fully connected layers.
- Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks.
- Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset (vector addition of a learned or fixed bias term).


Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
0
1
2
9
OUTPUT


LeNet
Image: 28 (height) × 28 (width) × 1 (channel)
Convolution with 5×5 kernel+2padding:28×28×6
sigmoid
Pool with 2×2 average kernel+2 stride:14×14×6
Convolution with 5×5 kernel (no pad):10×10×16
sigmoid
Pool with 2×2 average kernel+2 stride: 5×5×16
flatten
Dense: 120 fully connected neurons
sigmoid
Dense: 84 fully connected neurons
sigmoid
Dense: 10 fully connected neurons
Output: 1 of 10 classes
AlexNet
Image: 224 (height) × 224 (width) × 3 (channels)
Convolution with 11×11 kernel+4 stride:54×54×96
ReLu
Pool with 3×3 max. kernel+2 stride: 26×26×96
Convolution with 5×5 kernel+2 pad:26×26×256
ReLu
Pool with 3×3 max.kernel+2stride:12×12×256
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×256
ReLu
Pool with 3×3 max.kernel+2stride:5×5×256
flatten
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 1000 fully connected neurons
Output: 1 of 1000 classes

# Parameter Sharing

- A parameter sharing scheme is used in convolutional layers to control the number of free parameters. It relies on the assumption that if a patch feature is useful to compute at some spatial position, then it should also be useful to compute at other positions. Denoting a single 2-dimensional slice of depth as a *depth slice*, the neurons in each depth slice are constrained to use the same weights and bias.
- Sometimes, the parameter sharing assumption may not make sense. This is especially the case when the input images to a CNN have some specific centered structure; for which we expect completely different features to be learned on different spatial locations. One practical example is when the inputs are faces that have been centered in the image: we might expect different eye-specific or hair-specific features to be learned in different parts of the image. In that case it is common to relax the parameter sharing scheme, and instead simply call the layer a "locally connected layer".

# Loss Layer

- The "loss layer", or "loss function", specifies how training penalizes the deviation between the predicted output of the network, and the true data labels (during supervised learning). Various loss functions can be used, depending on the specific task.
- The softmax loss function is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for for predicting K independent probability values in [0,1]. Euclidean loss is used for regressing to real-valued labels in (-inf, +inf).

# A Note on Hyperparameters

- The kernel size is the number of pixels processed together. It is typically expressed as the kernel's dimensions, e.g., 2x2, or 3x3.
- Padding is the addition of (typically) 0-valued pixels on the borders of an image. This is done so that the border pixels are not undervalued (lost) from the output because they would ordinarily participate in only a single receptive field instance. The padding applied is typically one less than the corresponding kernel dimension. For example, a convolutional layer using 3x3 kernels would receive a 2-pixel pad, that is 1 pixel on each side of the image.
- The stride is the number of pixels that the analysis window moves on each iteration. A stride of 2 means that each kernel is offset by 2 pixels from its predecessor.
- Since feature map size decreases with depth, layers near the input layer tend to have fewer filters while higher layers can have more. To equalize computation at each layer, the product of feature values with pixel position is kept roughly constant across layers. Preserving more information about the input would require keeping the total number of activations (number of feature maps times number of pixel positions) non-decreasing from one layer to the next.

# A Note on Hyperparameters

- Number of filters to be used is also a hyperparameter. Common filter sizes found in the literature vary greatly, and are usually chosen based on the data set.
- Pooling type and size is also a hyperparameter. Max pooling is typically used, often with a 2x2 dimension. This implies that the input is drastically downsampled, reducing processing cost.
- CNNs can also have dilations. Dilation involves ignoring pixels within a kernel. This reduces processing/memory potentially without significant signal loss. A dilation of 2 on a 3x3 kernel expands the kernel to 5x5, while still processing 9 (evenly spaced) pixels. Accordingly, dilation of 4 expands the kernel to 9x9

# Regularization in a CNN

https://en.wikipedia.org/wiki/Convolutional_neural_network

# Thank You!