convolutional neural network (CNN)
image classification using NN

Overall procedure:
- Collect a large amount of images
- Annotate each image with a class label
- Choose a neural network structure
- Use our training samples to adjust layers and layers of parameters (backpropagation) to minimize a chosen loss

Goal: After training, when the NN sees a new image, its output should assign the highest probability to the respective class.
Challenge: The density of connections between layers increases intractably as the size of the image increases!!
Now, all four hidden neurons
1. Share the same set of 6 weights
2. Use local connections (receptive fields)

An effective way to reduce model parameters:
• **Local connection** Each neuron only processes inputs from a local region
• **Weight sharing** Neurons within the same layer can share weights
Another view to local connection and weight sharing:

- Convolve (slide) a block of shared weights over all spatial locations
- At each spatial location, output one value (computing dot products)
• We call this block of shared weights a convolutional filter

• Convolution: Convolve a filter with the image, i.e., slide over the image spatially, computing at each position a dot product between the filter and a small chunk of the image (plus bias), $W^T X + b$

• The dot product then goes through an activation function, e.g., ReLU, to produce the output
convolution

At each spatial location, output one value

Convolve (slide) over all spatial locations to generate an image like map, referred to as a **feature map**

A **convolutional layer**: Things between an input and a feature map
During convolution, the weights “slide” along the input to generate each output.
• Multiple sets of shared weights (filters) are allowed

• Each set of shared weights (filter) give one slice in the output (feature maps)

• In practice, CNN use many filters (~64 to 1024)
visualizing convolution

How convolutional filters may look like
Convolution is often followed by **pooling**:  
- Create a smaller and more manageable representation while retaining the most important information  
- “max” is the most common operation  
- Operate over each feature map independently
Max Pooling is a pooling operation that calculates the maximum value for patches of a feature map.
convolutional neural network (CNN)

Stack layers of convolution, activation (ReLU), pooling => CNN
how to train a CNN?

Example: AlexNet [Krizhevsky 2012]

- Split the data
- Choose the network architecture
- Initialize the network weights
- Find a learning rate and regularization strength
- Minimize the loss, e.g., softmax

“max”: max pooling
“norm”: local response normalization
“full”: fully connected

Figure: [Karnowski 2015] (with corrections)
splitting the dataset

- **Train**: gradient descent and fine-tuning of parameters
- **Validation**: determining hyper-parameters (learning rate, regularization strength, etc.) and picking an architecture
- **Test**: estimate real-world performance
softmax Loss
(multinomial logistic regression)

A generalization of logistic regression for multi-class classification

\[ z_k = w_k^T x \]

\[
\begin{align*}
P(y=1|x) &= \frac{e^{z_1}}{\sum_{k=1}^{K} e^{z_k}} \\
P(y=2|x) &= \frac{e^{z_2}}{\sum_{k=1}^{K} e^{z_k}} \\
P(y=3|x) &= \frac{e^{z_3}}{\sum_{k=1}^{K} e^{z_k}} \\
P(y=4|x) &= \frac{e^{z_4}}{\sum_{k=1}^{K} e^{z_k}}
\end{align*}
\]
Goal: Minimize loss ⇒ Maximize the probability of true class

• (Per-sample) Negative log-likelihood loss, e.g., for the i-th sample, \((x^i, y^i)\)

\[
L(x^i, y^i; \theta) = -\log(P(y = y_i | x^i)) = -\log\left(\frac{e^{z_{yi}}}{\sum_{k=1}^{K} e^{z_k}}\right)
\]

• Training: Minimizing the loss w.r.t parameters over the whole training set using backpropagation

\[
\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} L(x^i, y^i; \theta)
\]
Regularization reduces overfitting (as we have seen before):

$$L = L_{data} + L_{reg}$$

- $\lambda = 0.001$
- $\lambda = 0.01$
- $\lambda = 0.1$
examples of regularization terms

- **L2 regularization**: encourages small weights
  \[ L_{reg} = \lambda \frac{1}{2} \| W \|_2^2 \]

- **L1 regularization**: encourages sparse weights
  \[ L_{reg} = \lambda \| W \|_1 = \lambda \sum_{i,j} |W_{ij}| \]

- **Elastic net**: combines L1 and L2 regularization terms
  \[ L_{reg} = \lambda_1 \| W \|_1 + \lambda_2 \| W \|_2^2 \]

- **Max norm**: clamps (clips) weights to some maximum norm
  \[ \| W \|_2^2 \leq c \]