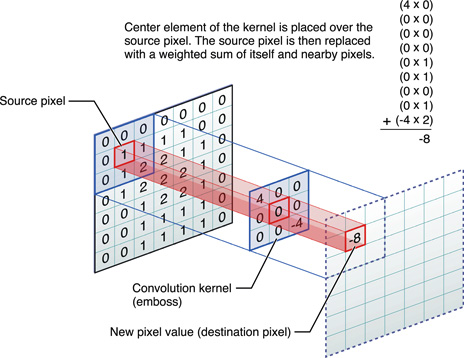
# Convolutional Neural Network

[if you know how to detect a feature in one place, you will know how to detect a feature in another place]

## Convolution



source: https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

Formal equation:

**Kernel convolution**: Slide the kernel through the entire matrix for a given stride length (# of steps). Notice that you lose some information after convolution. For a 3x3 kernel, you lose 1 row and 1 column. On the contrary, you are preserving useful information about every 3x3 area in the image while reducing the number of input parameters.

Here, we reduce a 7x7 matrix (= 49 pixels) to 5x5 matrix (= 25 pixel).

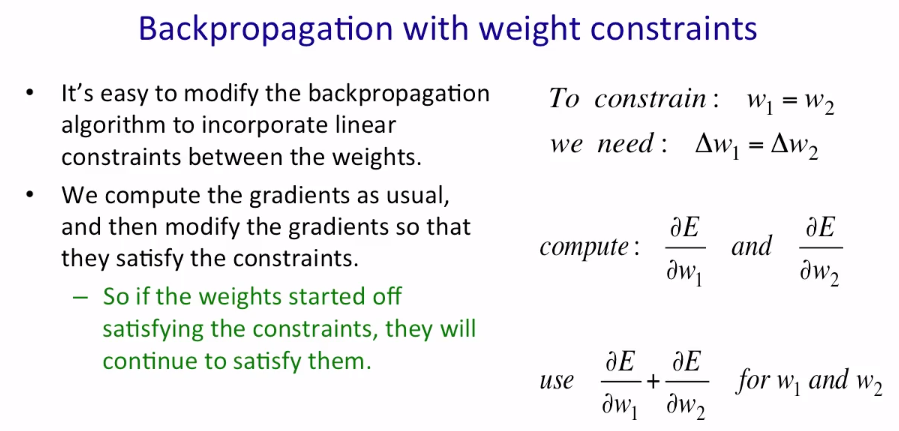
More hands-on math: <https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cs6505/fft.html>

* CNN is very good at reusing the same feature in different dimensions or different positions in the same dimension.
* This reduces the number of parameters that the network must learn.
* We can do this by incorporating linear constraints between two weights for the same feature that may appear in different positions or dimensions.

## Backpropagation with weight constraints

**Objective: translational equivariance**

* (update weights using) Rate of change in error (target vs prediction) wrt change in weight

1. Incorporate linear constraints between the weights by making the change in one weight equal to the change in the other.
2. To do this, we compute the gradients as usual and modify them to satisfy the constraints.

## Pooling

**Objective: translational invariance**

1. Average/take max of four neighbouring replicated feature detectors to give a single output in the next level.
   * Reduces the number of inputs to the next layer **->** allows more feature maps
   * Problem: we lose information on the precise location of things
     + Suppose we are doing facial recognition
     + Max pooling four replicated features (four dimensions) for the eyes will bring it to a single dimension
     + We do the same for the nose and lips
     + Now we have three max-pooled features in three different dimensions, one for the eyes, one for the nose and one for the lips
     + Knowing that there are eyes, nose and lips is enough to determine that it is a person’s face
     + **However**, we cannot recognize whose face it is because we lost information on where these are located in the face image and how they are aligned wrt each other **->** a person’s unique facial feature

