INTRODUCTION :

Neural networks are a subset of machine learning, and they are at the heart of deep learning algorithms. They are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

While we primarily focused on feedforward networks in that article, there are various types of neural nets, which are used for different use cases and data types. For example, recurrent neural networks are commonly used for natural language processing and speech recognition whereas convolutional neural networks (ConvNets or CNNs) are more often utilized for classification and computer vision tasks. Prior to CNN, manual, time-consuming feature extraction methods were used to identify objects in images.

However, convolutional neural networks now provide a more scalable approach to image classification and object recognition tasks, leveraging principles from linear algebra, specifically matrix multiplication, to identify patterns within an image. That said, they can be computationally demanding, requiring graphical processing units (GPUs) to train models.

Convolutional network:

Within Deep Learning, a Convolutional Neural Network or CNN is **a type of artificial neural network, which is widely used for image/object recognition and classification**. Deep Learning thus recognizes objects in an image by using a CNN.

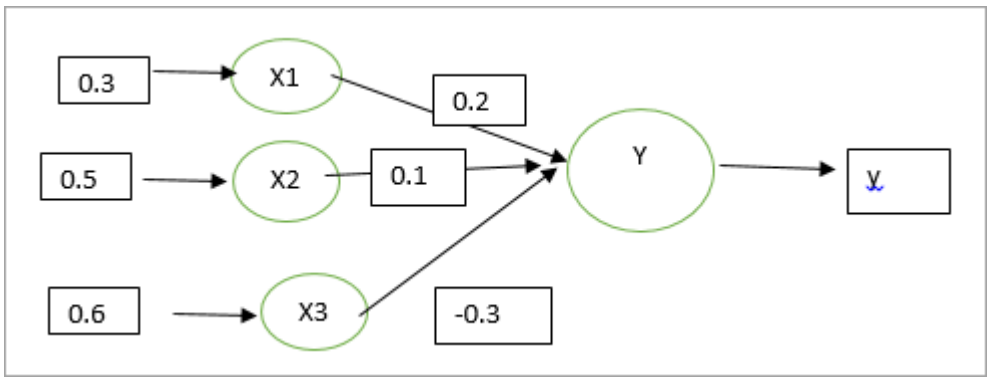
TYPES OF NEURAL NETWORKS:

Artificial Neural Networks (ANN) :

Artificial Neural Network is analogous to a biological neural network. A biological neural network is a structure of billions of interconnected neurons in a human brain. The human brain comprises of neurons that send information to various parts of the body in response to an action performed.

Example Of Artificial Neuron Network

Let's take the below network with the given input and calculate the net input neuron and obtain the output of the neuron Y with activation function as binary sigmoidal.



The input has 3 neurons X1, X2 and X3, and single output Y.

The weights associated with the inputs are: {0.2, 0.1, -0.3}

Inputs= {0.3, 0.5, 0.6}

Net input ={x1*w1+x2*w2+ x3*w3}

Net input = (0.3*0.2) + (0.5*0.1) + (0.6*-0.3)

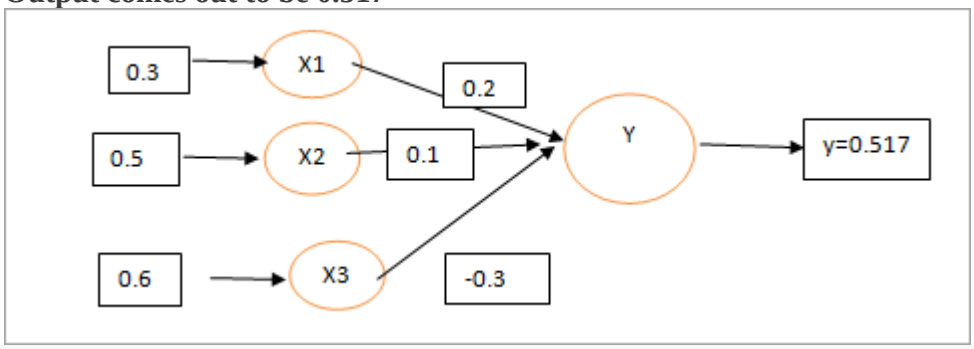
Net input= -0.07

Output for Binary Sigmoidal:

$$f(x) = \frac{1}{1 + e^{-x}}$$

X is -0.07

Output comes out to be 0.517



Convolution Neural Networks (CNN):

Within Deep Learning, a Convolutional Neural Network or CNN is **a type of artificial neural network, which is widely used for image/object recognition and classification**. Deep Learning thus recognizes objects in an image by using a CNN.

Examples:

Face recognition

Image recognition in computer vision

Recurrent Neural Networks (RNN):A RNN is designed to mimic the human way of processing sequences: we consider the entire sentence when forming a response instead of words by themselves. For example, consider the following sentence: “The concert was boring for the first 15 minutes while the band warmed up but then was terribly exciting.”

Examples:

Sentences

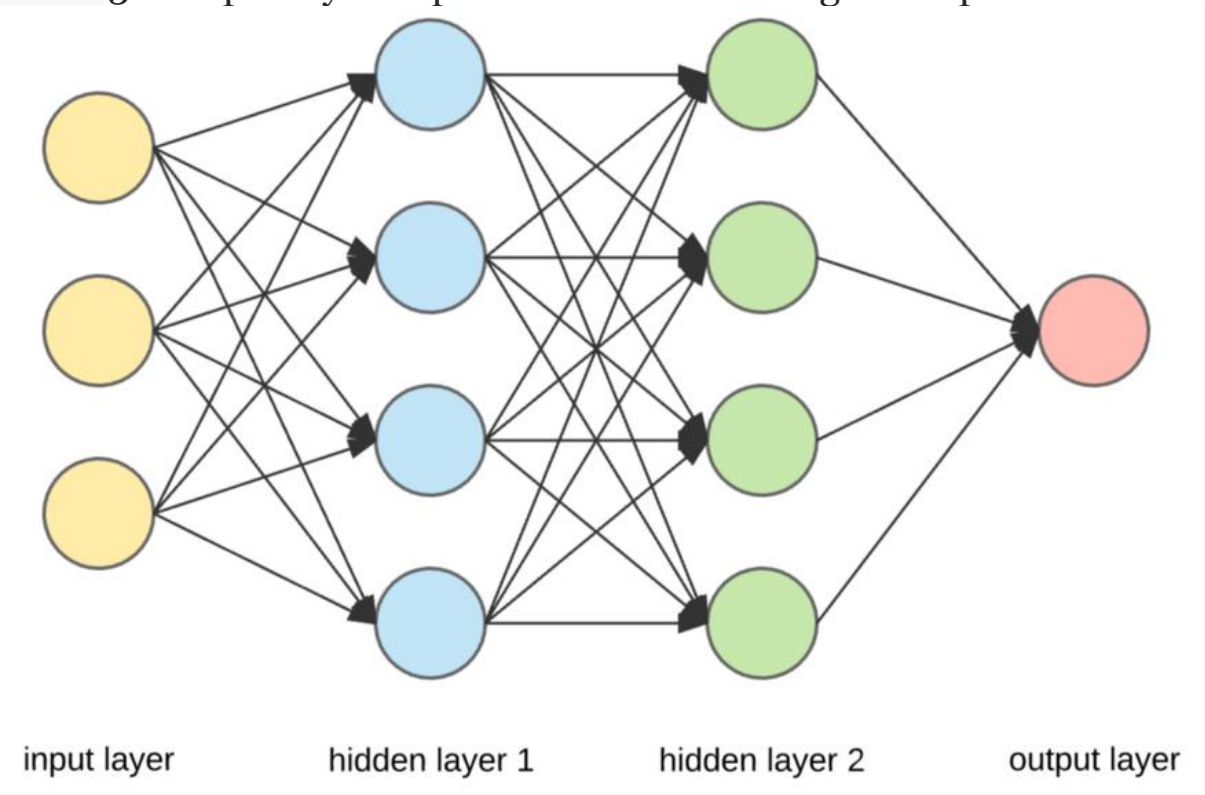
Daily stock prices

Solving time series problems

Self driving cars

The Neural Network is constructed from 3 types of layers:

1. Input layer — initial data for the neural network.
2. Hidden layers — an intermediate layer between the input and output layer and the place where all the computation is done.
3. Output layer — produce the result for given inputs.



APPLICATIONS:

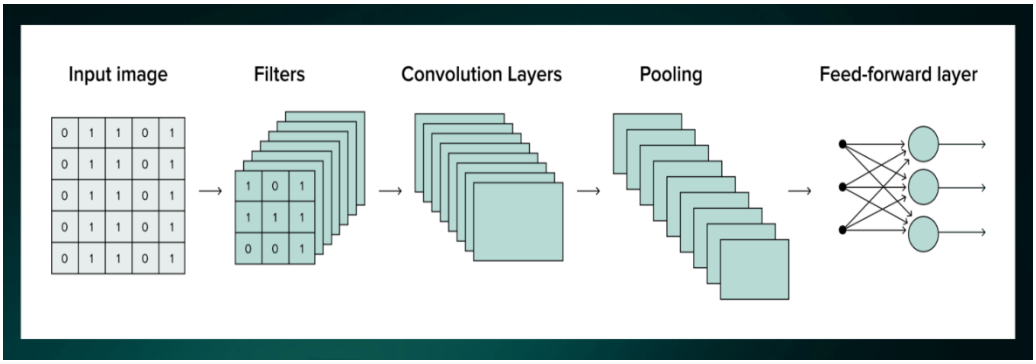
Autonomous driving. **Images** can be modeled using convolutional neural networks (CNN), which are used to model spatial information. CNN's are regarded as

universal non-linear function approximators because of their superior ability to extract features from images such as obstacles and interpret street signs.

The Convolution Operation (working of convolutional neural networks)

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

- Convolutional layer
- Pooling layer
- Fully connected (FC) layer



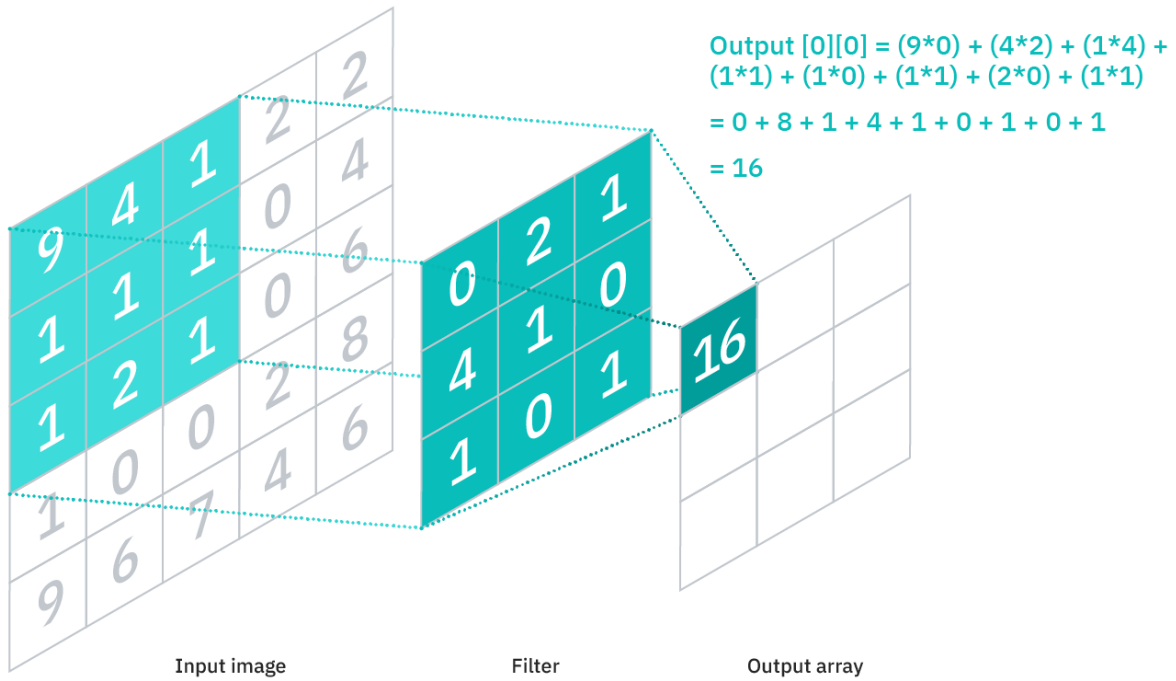
The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

Convolutional Layer

The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let’s

assume that the input will be a color image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions a height, width, and depth—which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution.

The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.



As you can see in the image above, each output value in the feature map does not have to connect to each pixel value in the input image. It only needs to connect to the receptive field, where the filter is being applied. Since the output array does not need to map directly to each

input value, convolutional (and pooling) layers are commonly referred to as “partially connected” layers. However, this characteristic can also be described as local connectivity.

Note that the weights in the feature detector remain fixed as it moves across the image, which is also known as parameter sharing. Some parameters, like the weight values, adjust during training through the process of backpropagation and gradient descent. However, there are three hyperparameters that affect the volume size of the output that needs to be set before the training of the neural network begins. These include:

1. The **number of filters** affects the depth of the output. For example, three distinct filters would yield three different feature maps, creating a depth of three.

2. **Stride** is the distance, or number of pixels, that the kernel moves over the input matrix. While stride values of two or greater is rare, a larger stride yields a smaller output.

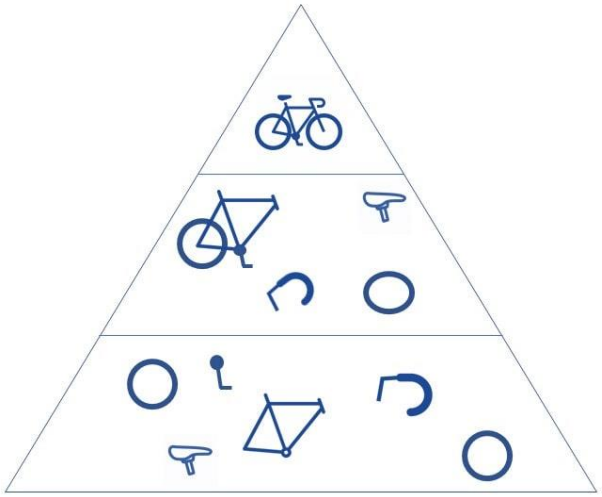
3. **Zero-padding** is usually used when the filters do not fit the input image. This sets all elements that fall outside of the input matrix to zero, producing a larger or equally sized output. There are three types of padding:

- **Valid padding:** This is also known as no padding. In this case, the last convolution is dropped if dimensions do not align.
- **Same padding:** This padding ensures that the output layer has the same size as the input layer
- **Full padding:** This type of padding increases the size of the output by adding zeros to the border of the input.

After each convolution operation, a CNN applies a Rectified Linear Unit (ReLU) transformation to the feature map, introducing nonlinearity to the model.

As we mentioned earlier, another convolution layer can follow the initial convolution layer. When this happens, the structure of the CNN can become hierarchical as the later layers can see the pixels within

the receptive fields of prior layers. As an example, let's assume that we're trying to determine if an image contains a bicycle. You can think of the bicycle as a sum of parts. It is comprised of a frame, handlebars, wheels, pedals, et cetera. Each individual part of the bicycle makes up a lower-level pattern in the neural net, and the combination of its parts represents a higher-level pattern, creating a feature hierarchy within the CNN.



Ultimately, the convolutional layer converts the image into numerical values, allowing the neural network to interpret and extract relevant patterns.

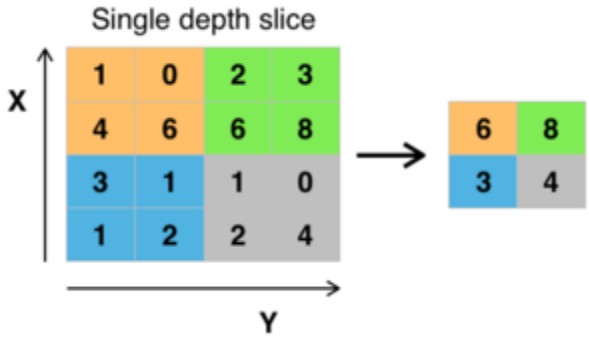
Pooling Layer

Pooling is nothing other than the downsampling of an image. The most common pooling layer filter is of size 2x2, which discards three fourth of the activations. The role of the pooling layer is to reduce the resolution of the feature map but retain features of the map required for classification through translational and rotational invariants. In addition to spatial invariance robustness, pooling will reduce the computation cost by a great deal.

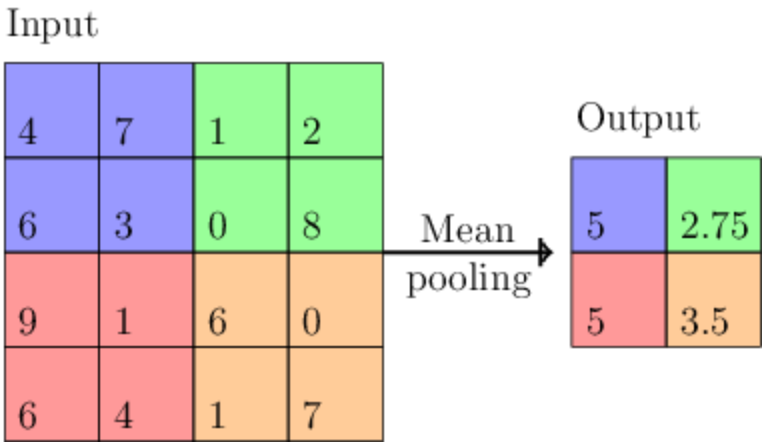
- Backpropagation is used for training of pooling operation
- It again helps the processor to process things faster.

There are many pooling techniques. They are as follows

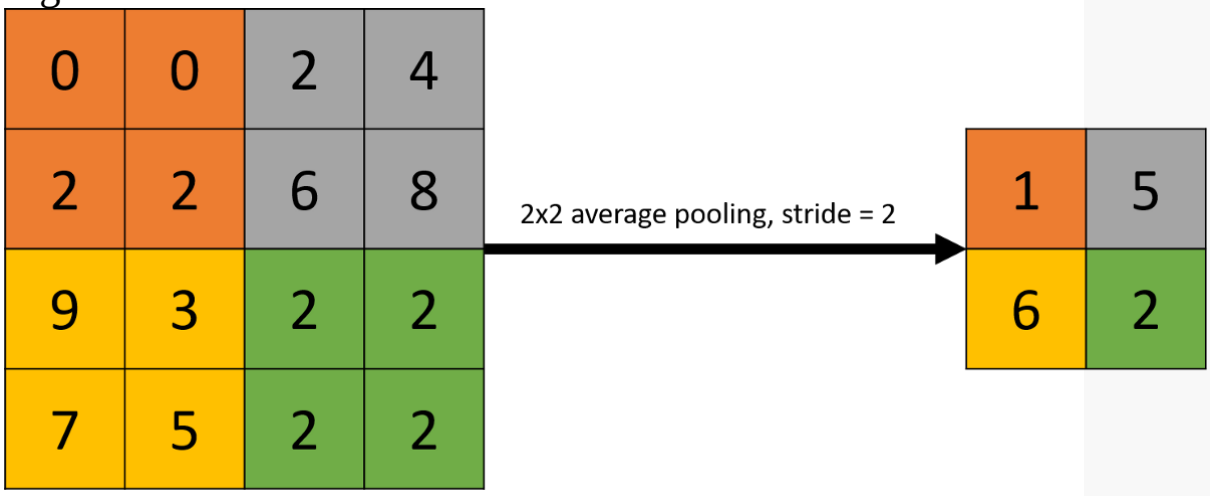
- **Max pooling** where we take largest of the pixel values of a segment.



- **Mean pooling** where we take largest of the pixel values of a segment.



- **Avg pooling** where we take largest of the pixel values of a segment.



As cross validation is expensive for big network, remedy of over-fitting in a modern neural network is considered through two roots:

- Reducing the number of the parameter by representing the model more effectively.
- Regularization

So dominant architecture in recent times for image classification is a convolution neural network, where the number of the parameter is reduced effectively through the convolution technique in initial layers and fully connected layers at the very end of the network.

Usually, regularization is performed through data augmentation, dropout or batch normalization. Most of these regularization techniques have difficulties implementing in convolutional layers. So, alternatively, such responsibility can be carried over by pooling layers in a convolutional neural network.

There are three variants of pooling operation depending on the roots of the regularization technique:

- **Stochastic pooling**: Randomly picked activation within each pooling region is considered more than deterministic pooling operations for regularization of the network. Stochastic pooling performs a reduction of feature size but denies the role of selecting features judiciously for the sake of regularization. Although clipping of negative output from ReLU activation helps to carry some of the selection responsibility.
- **Overlapping pooling**: Overlapping pooling operation shares the responsibility of local connection beyond the size of the previous convolutional filter, which breaks orthogonal responsibility between the pooling layer and convolutional layer. So, no information is gained if pooling windows overlap
- **Fractional pooling**: The reduction ratio of filter size due to pooling can be controlled by a fractional pooling concept, which helps to increase the depth of the network. Unlike stochastic pooling, the randomness is related to the choice of pooling regions, not the way pooling is performed inside each of the pooling regions.

There are other variants of pooling as follows:

- Min pooling
- wavelet pooling
- tree pooling
- max-avg pooling
- spatial pyramid pooling

Pooling makes the network invariant to translations in shape, size and scale. Max pooling is generally predominantly used in objection recognition.

Fully-Connected Layer

The name of the full-connected layer aptly describes itself. As mentioned earlier, the pixel values of the input image are not directly connected to the output layer in partially connected layers. However, in the fully-connected layer, each node in the output layer connects directly to a node in the previous layer.

This layer performs the task of classification based on the features extracted through the previous layers and their different filters. While convolutional and pooling layers tend to use ReLu functions, FC layers usually leverage a softmax activation function to classify inputs appropriately, producing a probability from 0 to 1.

The Convolution Operation:

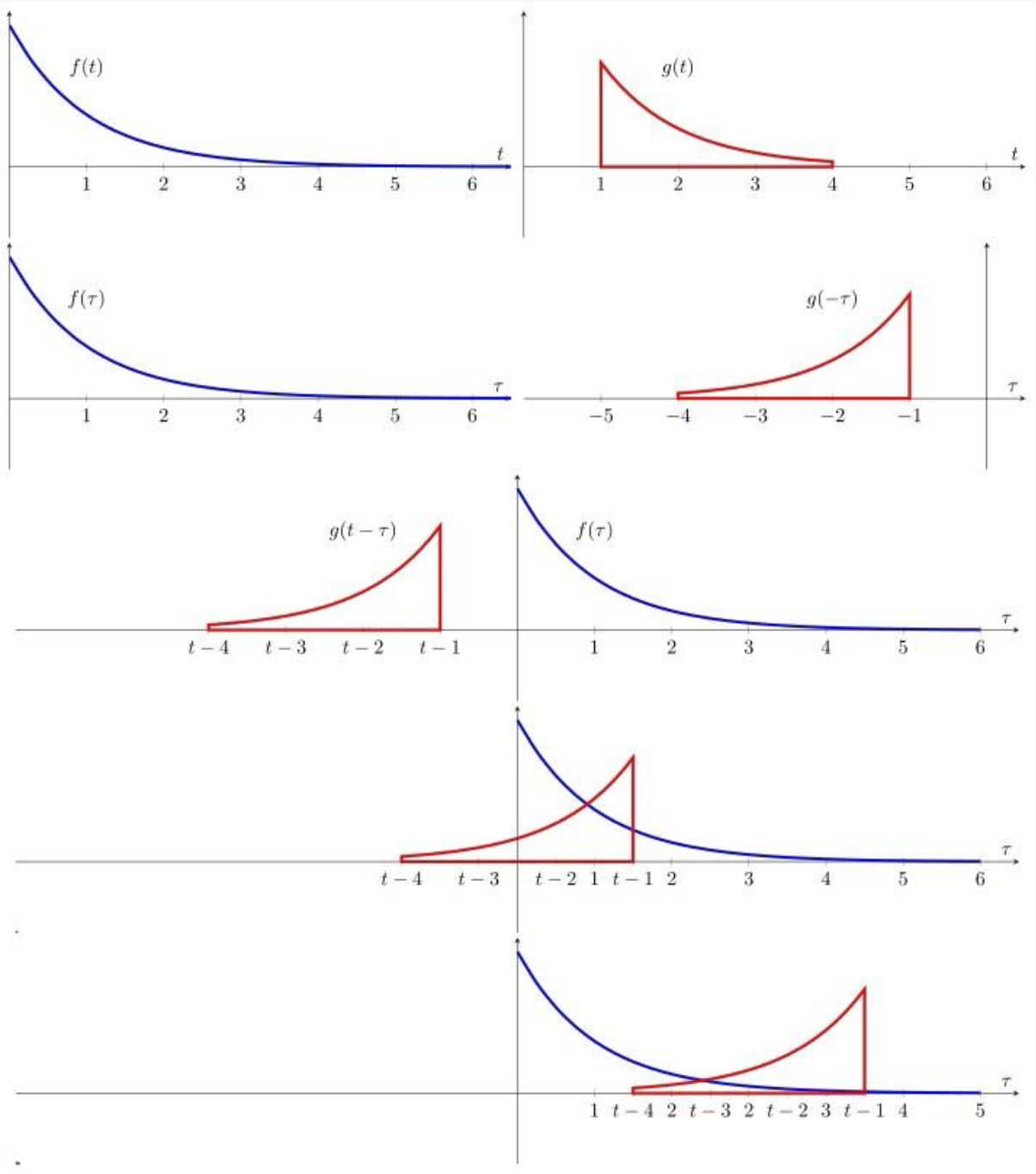
The convolution operates on the **input** with a **kernel** (weights) to produce an **output map** given by:

$$s(t) = \int f(\tau)g(t-\tau)d\tau$$

Continuous domain convolution

Let us break down the formula. The steps involved are:

1. Express each function in terms of a dummy variable τ
2. Reflect the function g i.e. $g(\tau) \rightarrow g(-\tau)$
3. Add a time offset i.e. $g(\tau) \rightarrow g(t-\tau)$. Adding the offset shifts the input to the right by t units (by convention, a negative offset shits it to the left)
4. Multiply f and g point-wise and accumulate the results to get output at instant t. Basically, we are calculating the area of overlap between f and shifted g



source: https://en.wikipedia.org/wiki/Convolution

For our application, we are interested in the discrete domain formulation:

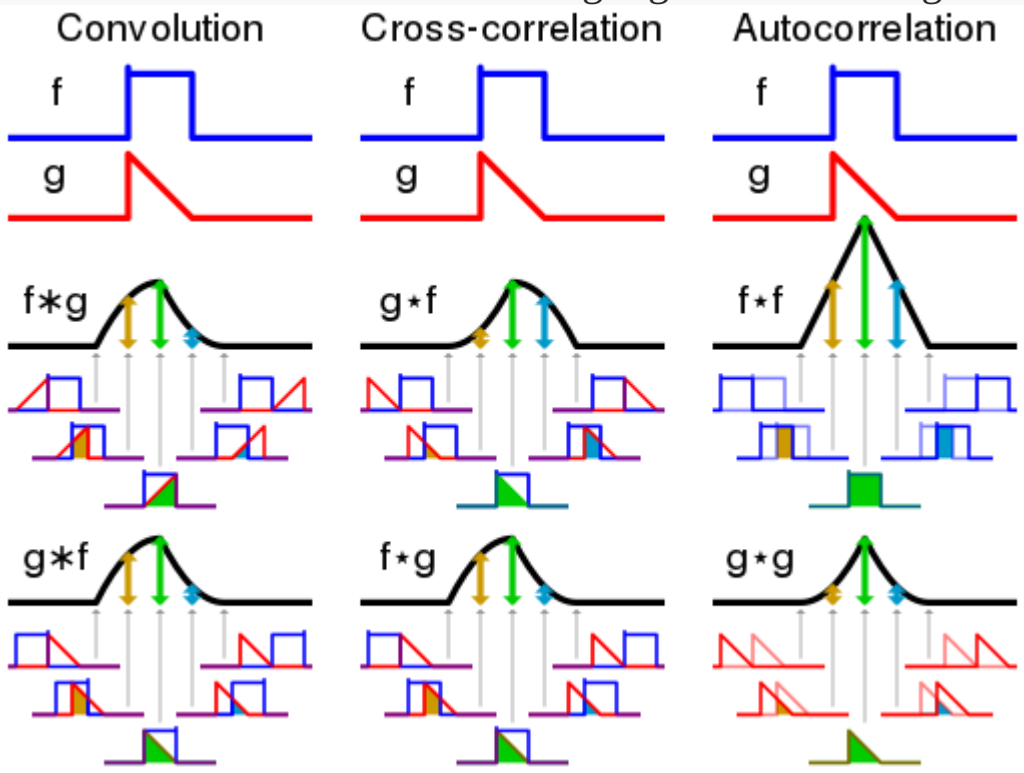
$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

1-D discrete convolution

$$s(i,j) = \sum_{m}\sum_{n} x(m,n)w(i-m, j-n)$$

2-D discrete convolution

When the kernel is not flipped in its domain, we obtain the **cross-correlation** operation. The basic difference between the two operations is that convolution is commutative in nature, i.e. *f* and *g* can be interchanged without changing the output. Cross-correlation is not commutative. This difference is highlighted in the image below:

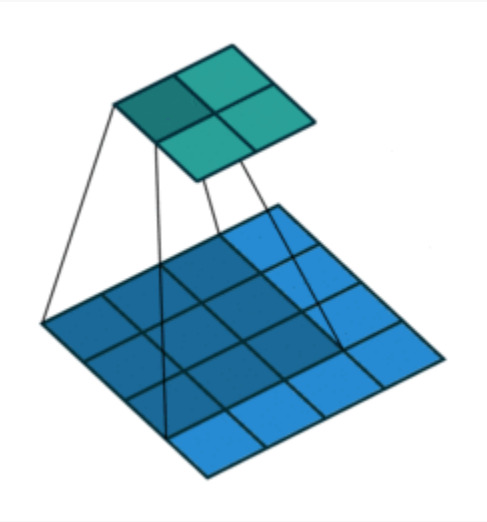


source: https://en.wikipedia.org/wiki/Convolution

Although these equations imply that the *domains* for both *f* and *g* are infinite, in practice, these two variables are non-zero only in a finite region. As a result, the output is non-zero only in a finite region (where the non-zero regions of *f* and *g* overlap).

The intuition for convolution in 1-D can be extended to *n*-dimensions by nesting the convolution operations. *Vision* provide an in-

depthanalysis of how input and output shapes and computations are tied. Below is their visualization of a 2-D convolution operation:



2D convolution (source)

The 1D convolution operation can be represented as a matrix-vector product. The kernel matrix is obtained by composing weights into a **Toeplitz matrix**. A Toeplitz matrix has the property that values along all diagonals are constant.

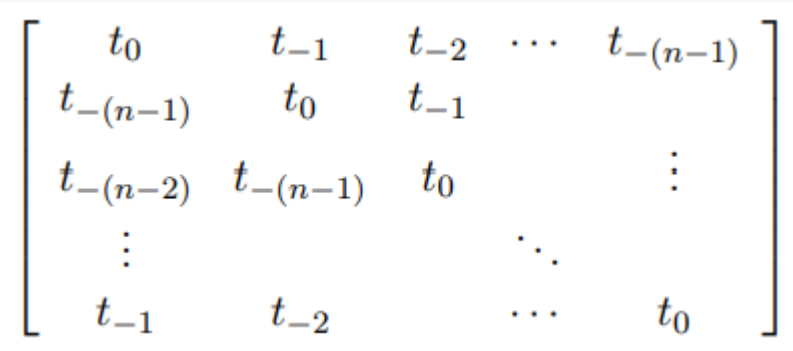
$$\begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{-(n-1)} \\ t_1 & t_0 & t_{-1} & & \\ t_2 & t_1 & t_0 & & \vdots \\ \vdots & & & \ddots & \\ t_{n-1} & & & \cdots & t_0 \end{bmatrix}$$

The general structure of a Toeplitz matrix

$$\begin{bmatrix} s0 \\ s1 \\ s2 \\ s3 \\ s4 \\ \dots \end{bmatrix} = \begin{bmatrix} g0 & 0 & 0 & 0 \\ g1 & g0 & 0 & 0 \\ g2 & g1 & g0 & 0 \\ g3 & g2 & g1 & g0 \\ g4 & g3 & g2 & g1 \\ & \dots & & \end{bmatrix} \times \begin{bmatrix} f0 \\ f1 \\ f2 \\ f3 \end{bmatrix}$$

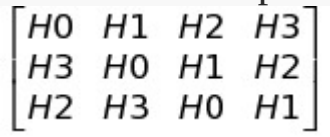
Using the Toeplitz matrix of the kernel for matrix-vector implementation of convolution

To extend this principle to 2D input, we first need to unroll the 2D input into a 1D vector. Once this is done, the kernel needs to be modified as before but this time resulting in a **block-circulant matrix**. What's that?



A general circulant matrix

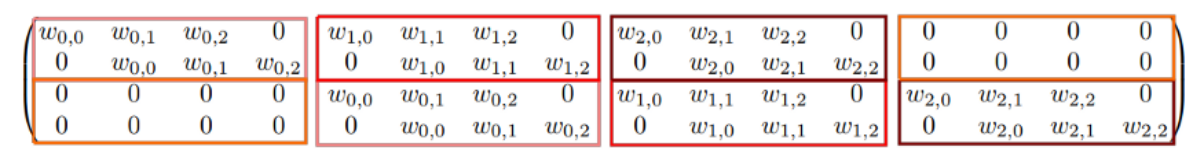
A **circulant matrix** is a special case of a Toeplitz matrix where each row is a circular shift of the previous row. To see that it is a special case of the Toeplitz matrix is trivial.



A block circulant matrix (each Hi is a matrix)

A matrix which is circulant with respect to its sub-matrices is called a **block circulant matrix**. If each of the submatrices is itself circulant, the matrix is called **doubly block-circulant matrix**.

Now, given a 2D kernel, we can create the block-circulant matrix that will act allow matrix-vector implementation of convolution as below:



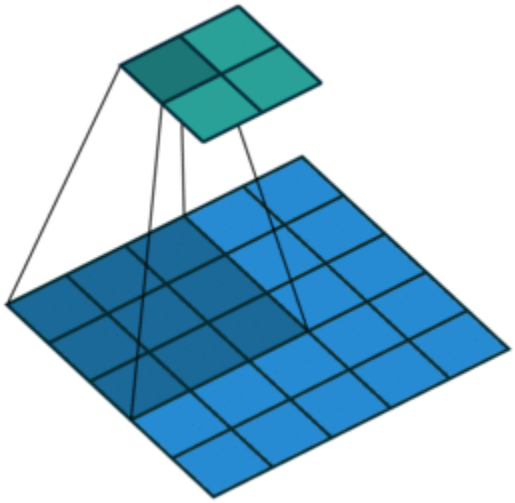
Convince yourself that the resultant of convolving a 3x3 kernel on a 4x4 input (16x1 unrolled vector) results in a 2x2 output (4x1 vector)

[refer to gif above] and hence the required kernel matrix must be of shape 4x1

Variants of the Basic Convolution Function

In practical implementations of the convolution operation, certain modifications are made which deviate from the discrete convolution formula mentioned above:

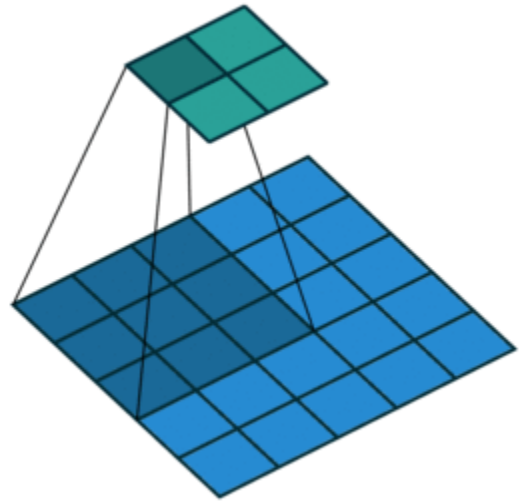
- In general a convolution layer consists of application of several different kernels to the input. This allows the extraction of several different features at all locations in the input. This means that in each layer, a single kernel (filter) isn't applied. Multiple kernels (filters), usually a power of 2, are used as different feature detectors.
- The input is generally not real-valued but instead vector valued (e.g. RGB values at each pixel or the feature values computed by the previous layer at each pixel position). Multi-channel convolutions are commutative only if number of output and input channels is the same.
- In order to allow for calculation of features at a **coarser level** *strided convolutions* can be used. The effect of strided convolution is the same as that of a convolution followed by a downsampling stage. This can be used to reduce the representation size.

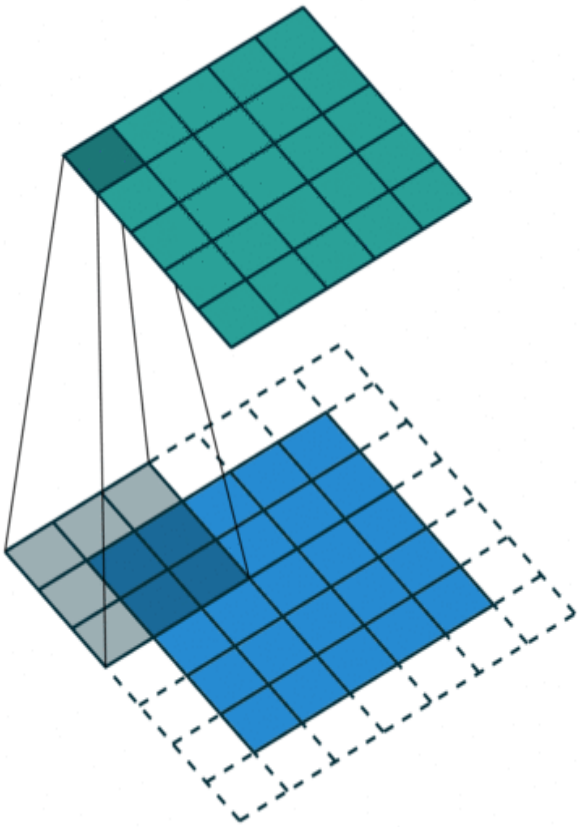


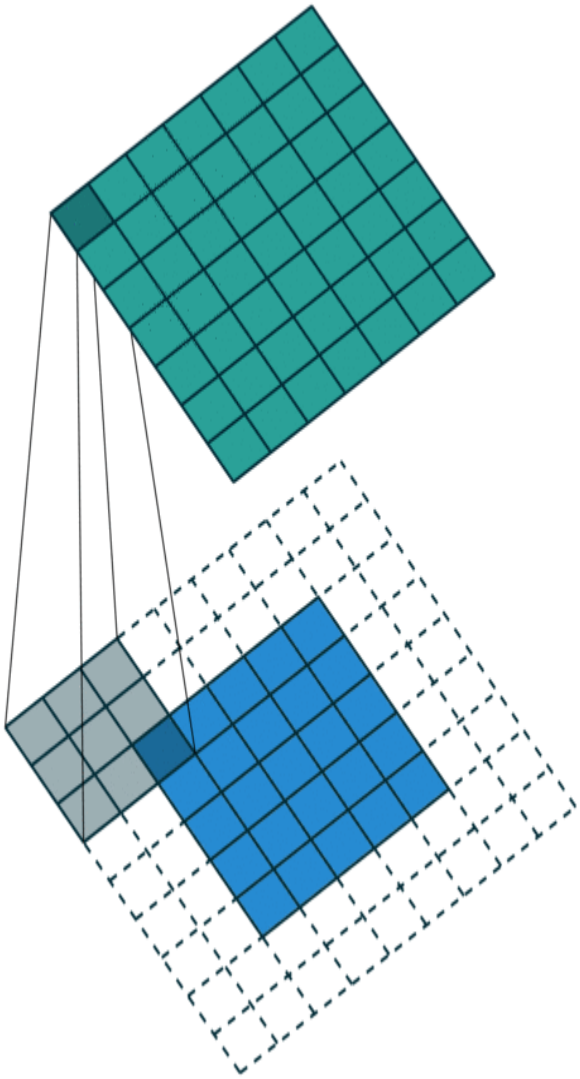
2D convolution 3x3 kernel and stride of 2 units (source)

- Zero padding helps to make output dimensions and kernel size independent. 3 common zero padding strategies are:

1. **valid**: The output is computed only at places where the entire kernel lies inside the input. Essentially, no zero padding is performed. For a kernel of size **k** in any dimension, the input shape of **m** in the direction will become **m-k+1** in the output. This shrinkage restricts architecture depth.
2. **same**: The input is zero padded such that the spatial size of the input and output is same. Essentially, for a dimension where kernle size is **k**, the input is padded by **k-1** zeros in that dimension. Since the number of output units connected to border pixels is less than that for centre pixels, it may under-represent border pixels.
3. **full**: The input is padded by enough zeros such that each input pixel is connected to the same number of output units. In terms of test set accuracy, the optimal padding is somewhere between *same* and *valid*.







valid(left), **same**(middle) and **full**(right) padding (source). The extreme left one is for stride=2.

- Besides locally-connected layers and tiled convolution, another extension can be to restrict the kernels to operate on certain input channels. One way to implement this is to connect the first **m** input channels to the first **n** output channels, the next **m** input channels to the next **n** output channels and so on. This method decreases the number of parameters in the model without dereasing the number of output units.

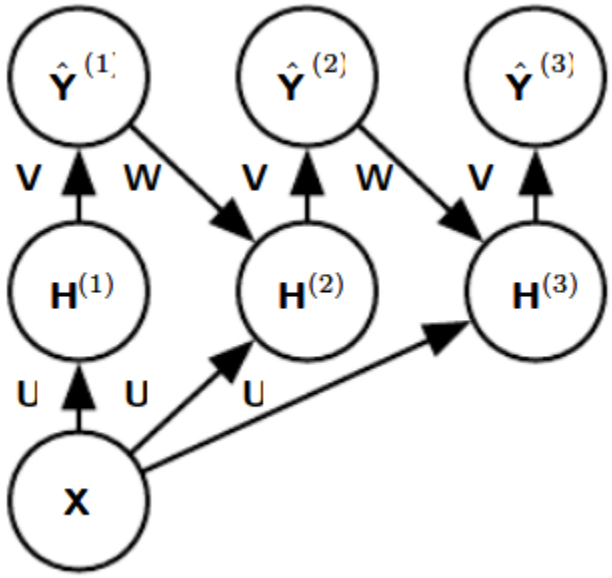
- When max pooling operation is applied to locally connected layer or tiled convolution, the model has the ability to become **transformation invariant** because adjacent filters have the freedom to learn a transformed version of the same feature. *This essentially similar to the property leveraged by pooling over channels rather than spatially*.
- **Bias** terms can be used in different ways in the convolution stage. For locally connected layer and tiled convolution, we can use a bias per output unit and kernel respectively. In case of traditional convolution, a single bias term per output channel is used. *If the input size is fixed, a bias per output unit may be used to counter the effect of regional image statistics and smaller activations at the boundary due to zero padding*.

Structured Outputs

Convolutional networks can be trained to output high-dimensional structured output rather than just a classification score. A good example is the task of image segmentation where each pixel needs to be associated with an object class. Here the output is the same size (spatially) as the input. The model outputs a tensor ***S*** where ***S[i,j,k]*** is the probability that pixel **(j,k)** belongs to class **i**.

To produce an output map as the same size as the input map, only **same-padded** convolutions can be stacked. Alternatively, a coarser segmentation map can be obtained by allowing the output map to shrink spatially.

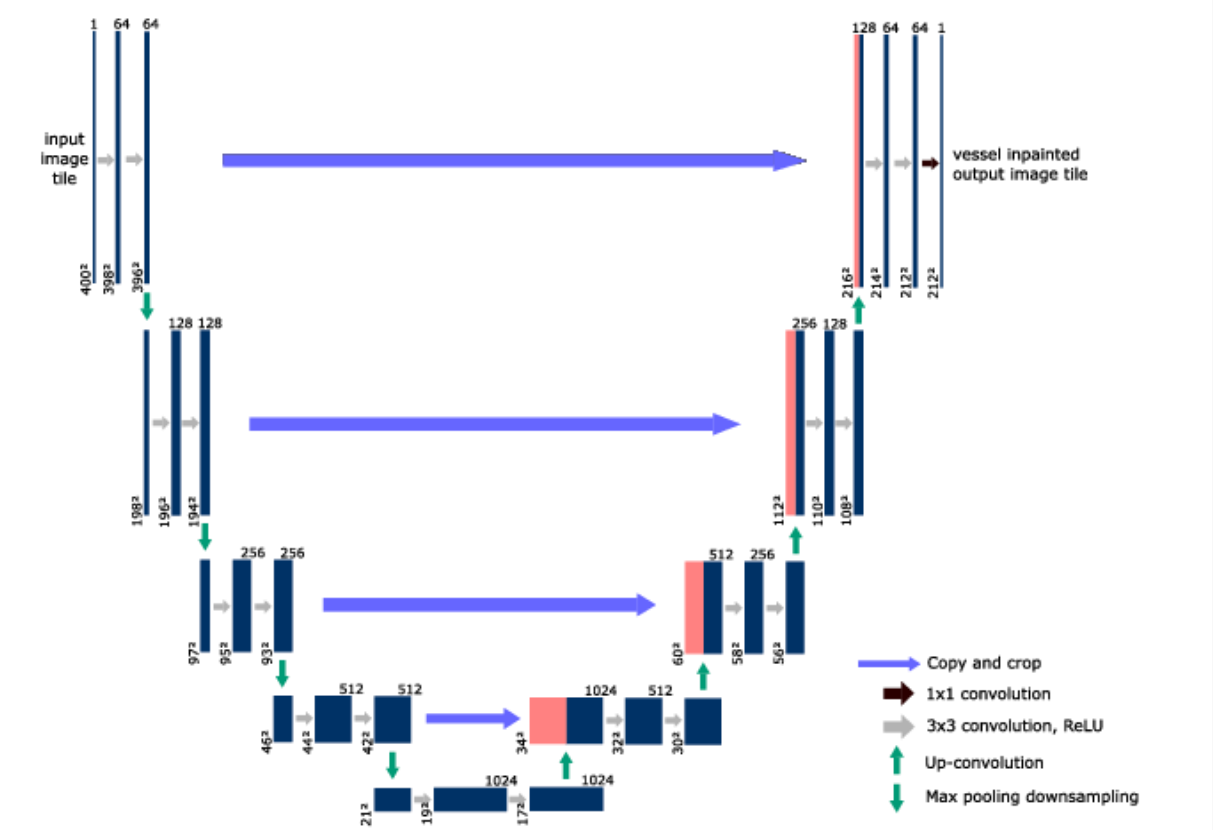
The output of the first labelling stage can be refined successively by another convolutional model. If the models use tied parameters, this gives rise to a type of **recursive model** as shownbelow. (H^1, H^2, H^3 share parameters)



Recursive refinement of the segmentation map

The output can be further processed under the assumption that contiguous regions of pixels will tend to belong to the same label. Graphical models can describe this relationship. Alternately, *CNNs can learn to optimize the graphical models training objective*.

Another model that has gained popularity for segmentation tasks (especially in the medical imaging community) is the *U-Net*. The *up-convolution* mentioned is just a direct upsampling by repetition followed by a convolution with *same padding*.



U-Net architecture for medical image segmentation (*source*)

Data Types

The data used with a convolutional network usually consist of several channels, each channel being the observation of a different quantity at some point in space or time.

One advantage to convolutional networks is that they can also process inputs with varying spatial extents.

When the output is accordingly variable sized, no extra design change needs to be made. If however the output is fixed sized, as in the classification task, a pooling stage with kernel size proportional to the input size needs to be used.

Dimensions	Single channel	Multichannel
1D	**Raw audio** (single amplitude value per time point)	**Skeleton animatin data** (orientation of each joint)
2D	**Audio spectrogram** (one FFT coefficient per time point per frequency)	**Color image** (RGB triplet per (x,y) tuple)
3D	**CT scan** (one value per (x,y,z) tuple)	**Color video** (one RGB triplet per (x,y) tuple per time instant)

Different data types based on the number of spatial dimensions and channels

Efficient Convolution Algorithms

In some problem settings, performing convolution as pointwise multiplication in the frequency domain can provide a speed up as compared to direct computation. This is a result from the property of convolution:

$$F\{f \star g\} = F\{f\} \times F\{g\}$$

Convolution in the source domain is multiplication in the frequency domain. F is the transformation operation

$$f \star g = F^{-1}\{F\{f \star g\}\} = F^{-1}\{F\{f\} \times F\{g\}\}$$

When a **d**-dimensional kernel can be broken into the outer product of **d** vectors, the kernel is said to be separable. The corresponding convolution operations are more efficient when implemented as **d** 1-dimensional convolutions rather than a direct **d**-dimensional convolution. Note however, it may not always be possible to express a kernel as an outer product of lower dimensional kernels.

This is not to be confused with **depthwise separable convolution** (explained brilliantly *here*). This method restricts convolution kernels to operate on only one input channel at a time

followed by 1x1 convolutions on all channels of the intermediate output.

Devising faster ways of performing convolution or approximate convolution without harming the accuracy of the model is an active area of research.

Random and Unsupervised Features

To reduce the computational cost of training the CNN, we can use features not learned by supervised training.

1. **Random initialization** has been shown to create filters that are *frequency selective and translation invariant*. This can be used to inexpensively select the model architecture. Randomly initialize several CNN architectures and just train the last classification layer. Once a winner is determined, that model can be fully trained in a supervised manner.
2. **Hand designed kernels** may be used; e.g. to detect edges at different orientations and intensities.
3. **Unsupervised training** of kernels may be performed; e.g. *applying k-means clustering to image patches* and using the centroids as convolutional kernels. *Unsupervised pre-training may offer regularization effect (not well established)*. It may also allow for training of larger CNNs because of reduced computation cost.

Another approach for CNN training is **greedy layer-wise pretraining** most notably used in *convolutional deep belief network*. For example, in the case of multi-layer perceptrons, starting with the

first layer, each layer is trained in isolation. Once the first layer is trained, its output is stored and used as input for training the next layer, and so on.

The Neuroscientific Basis for Convolutional Networks

Hubel and Wiesel studied the activity of neurons in a cat's brain in response to visual stimuli. Their work characterized many aspects of brain function.

In a simplified view, we have:

1. The light entering the eye stimulates the **retina**. The image then passes through the the optic nerve and a region of the brain called the **LGN (lateral geniculate nucleus)**
2. **V1 (primary visual cortex)**: The image produced on the retina is transported to the V1 with minimal processing. The properties of V1 that have been replicated in CNNs are:

- The V1 response is localized spatially, i.e. the upper image stimulates the cells in the upper region of V1 **[localized kernel]**.
- V1 has simple cells whose activity is a linear function of the input in a small neighbourhood **[convolution]**.
- V1 has complex cells whose activity is invariant to shifts in the position of the feature **[pooling]** as well as some changes in lighting which cannot be captured by spatial pooling **[cross-channel pooling]**.

3. There are several stages of V1 like operations **[stacking convolutional layers]**.

4. In the **medial temporal lobe**, we find **grandmother cells**. These cells respond to specific concepts and are invariant to several transforms of the input. In the **medial temporal lobe**, researchers also found neurons spiking on a particular concept, e.g. the **Halle Berry neuron** fires when looking at a photo/drawing of Halle Berry or even reading the text Halle Berry. Of course, there are neurons which spike at other concepts like Bill Clinton, Jennifer Aniston, etc.

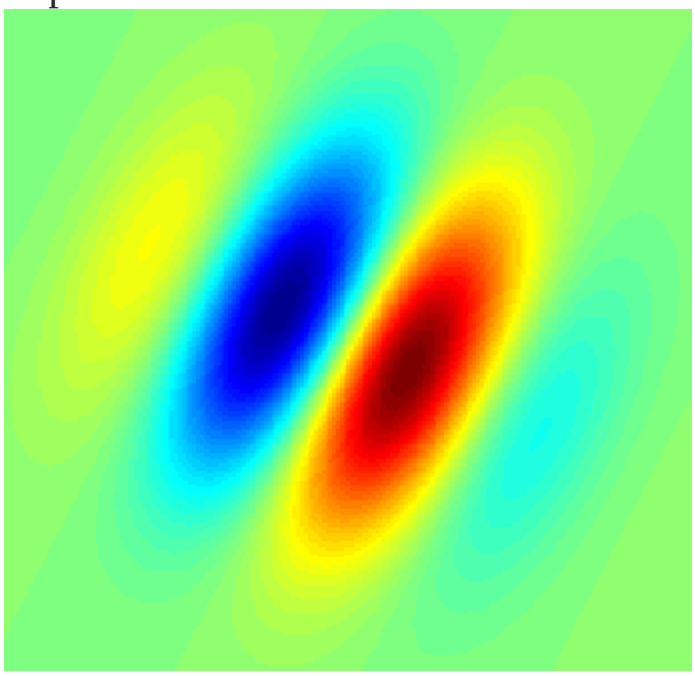
The medial temporal neurons are more generic than CNN in that they respond even to specific ideas. A closer match to the function of the last layers of a CNN is the **IT (inferotemporal cortex)**. When viewing an object, information flows from the retina, through LGN, V1, V2, V4 and reaches IT. This happens within 100ms. When a person continues to look at an object, the brain sends top-down feedback signals to affect lower level activation.

Some of the major differences between the human visual system (HVS) and the CNN model are:

- The human eye is low resolution except in a region called **fovea**. Essentially, the eye does not receive the whole image at high resolution but stiches several patches through eye movements called **saccades**. This attention based gazing of the input image is an active research problem. *Note:* attention mechanisms have been shown to work on natural language tasks.

- Integration of several senses in the HVS while CNNs are only visual.
- The HVS processes rich 3D information, and can also determine relations between objects. CNNs for such tasks are in their early stages.
- The feedback from higher levels to V1 has not been incorporated into CNNs with substantial improvement.
- While the CNN can capture firing rates in the IT, the similarity between intermediate computations is not established. The brain probably uses different activation and pooling functions. Even the linearity of filter response is doubtful as recent models for V1 involve quadratic filters.

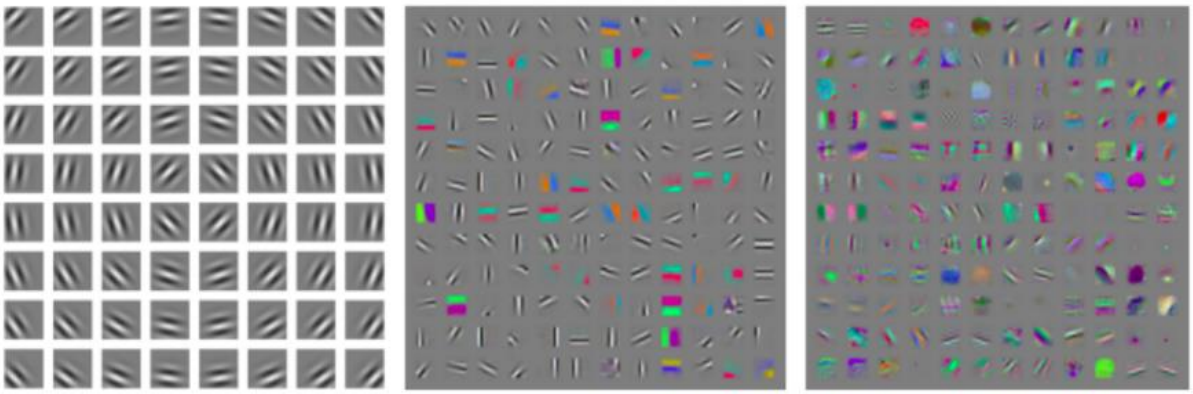
Neuroscience tells us very little about the training procedure. Backpropogation which is a standard training mechanism today is not inspired by neuroscience and sometimes considered biologically implausible.



The heatmap of a 2D Gabor filter (source)

In order to determine the filter parameters used by neurons, a process called **reverse correlation** is used. The neuron activations are measured by an electrode when viewing several white noise images and a linear model is used to approximate this behaviour. It has been shown experimentally that the weights of the fitted model of V1 neurons are described by **Gabor functions**. If we go by the simplified version of the HVS, if the simple cells detect Gabor-like features, then complex cells learn a function of simple cell outputs which is invariant to certain translations and magnitude changes.

A wide variety of statistical learning algorithms (from unsupervised (sparse code) to deep learning (first layer features)) learn features with Gabor-like functions when applied to natural images. This goes to show that while no algorithm can be touted as the **right method** based on Gabor-like feature detectors, a **lack** of such features may be taken as a **bad sign**.



(Left) Gabor functions with different values of the parameters that control the coordinate system. (Middle) Weights learned by an unsupervised learning algorithm (Right) Convolution kernels learned by the first layer of a fully supervised convolutional maxout network.

Convolutional neural networks and computer vision

Convolutional neural networks power image recognition and computer vision tasks. Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs, and based on those inputs, it can take action. This ability to provide recommendations distinguishes it from image recognition tasks. Some common applications of this computer vision today can be seen in:

- **Marketing:** Social media platforms provide suggestions on who might be in photograph that has been posted on a profile, making it easier to tag friends in photo albums.
- **Healthcare:** Computer vision has been incorporated into radiology technology, enabling doctors to better identify cancerous tumors in healthy anatomy.
- **Retail:** Visual search has been incorporated into some e-commerce platforms, allowing brands to recommend items that would compliment an existing wardrobe.
- **Automotive**: While the age of driverless cars hasn't quite emerged, the underlying technology has started to make its way into automobiles, improving driver and passenger safety through features like lane line detection