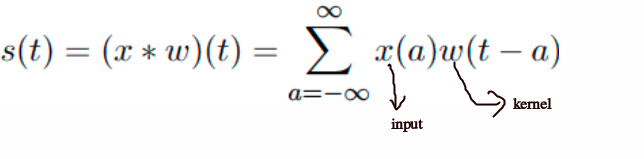
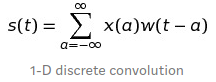
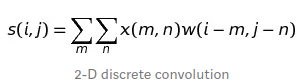
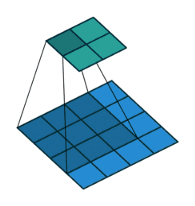
**Convolutional Neural Networks**

Convolutional networks also known as convolutional neural networks, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology.

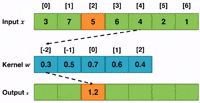
Convolution Operation

The convolution operates on the input with a kernel (weights) to produce an output map given by:  
[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/01.updated_convolution_operation.png)

* 1-D discrete convolution operation can be given by:  
  [](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/03.1-D-cnn.png)
* 2-D discrete convolution operation can be given by:  
  [](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/04.2-D-cnn.png)

2-D convolution operation can be visualized as below:  
[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/16.convolution_operation_2.gif)

Example Demonstrating Convolution Operation

[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/01.convolution_operation.gif)

**Properties of Convolution Operation and Cross-Correlation**

* **Commutative Property**

Convolution operation is commutative.

Commutative property arises because we have ﬂipped the kernel relative to the input  
[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/05.commutative_lhs.png) [](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/06.commutative_rhs.png)

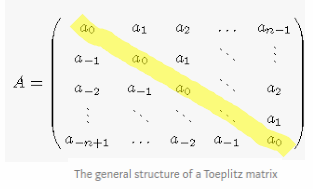
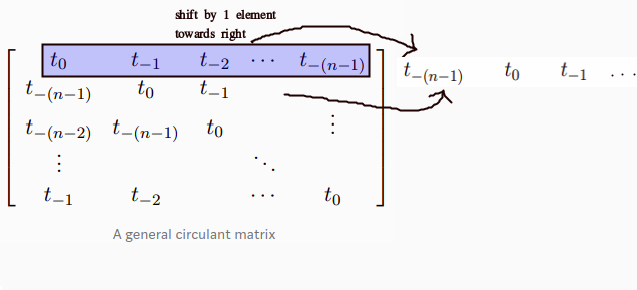
* **Cross-Correlation**

Function which is analogous to convolution operation without flipping the kernel is called cross-correlation operation.

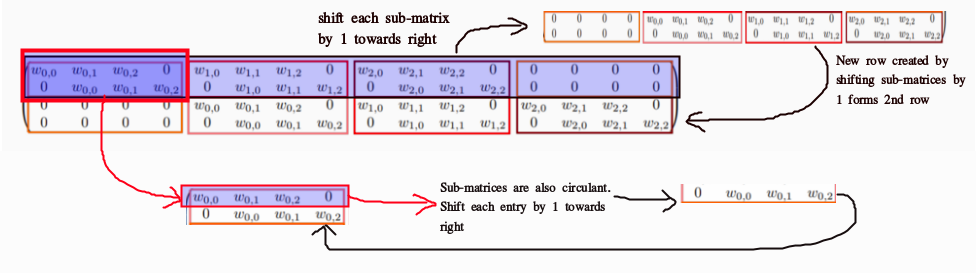
Cross-correlation is not commutative.

* **Convolution operation:**[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/06.commutative_rhs.png)

Correlation operation:  
[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/07.cross_correlation.png)

* **Toeplitz Matrix**
* 1D convolution operation can be represented as a matrix vector product.
* The kernel marix is obtained by composing weights into a Toeplitz matrix.
* Toeplitz matrix has the property that values along all diagonals are constant.  
  [](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/08.toeplitz_matrix_1d.png)
* **Block-Circulant and Doubly-Block-Circulant Matrix**
* To extend the concept of Toeplitz matrix towards 2-D input, we need to convert 2-D input to 1-D vector.
* Kernel needs to be modified as before but this time resulting in a block-circulant matrix.
* A circulant matrix is a special case of a Toeplitz matrix where each row is equal to the row above shifted by one element.  
  [](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/09.circulant_matrix.png)
* A matrix which is circulant with respect to its sub-matrices is called a block circulant matrix.

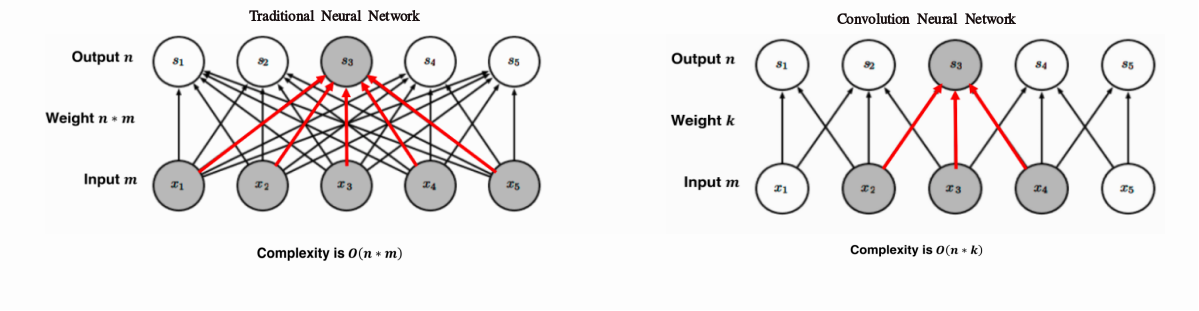
[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/10.curculant_matrix.png)

* If each of the submatrices is itself circulant, the matrix is called doubly block-circulant matrix.  
  [](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/11.doubly_circulant_matrix.png)

**Motivation**

**Sparse Interactions**

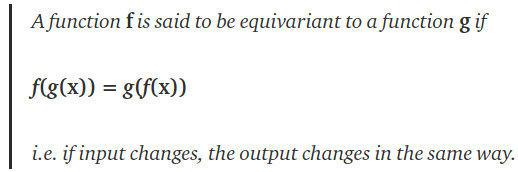
* In traditional Neural Networks, every output unit interacts with every input unit.
* Convolutional networks, however, typically have sparse interactions, by making kernel smaller than input.
* Reduces memory requirements
* Improves statistical eﬃciency
* In a deep convolutional network, units in the deeper layers may indirectly interact with a larger portion of the input.

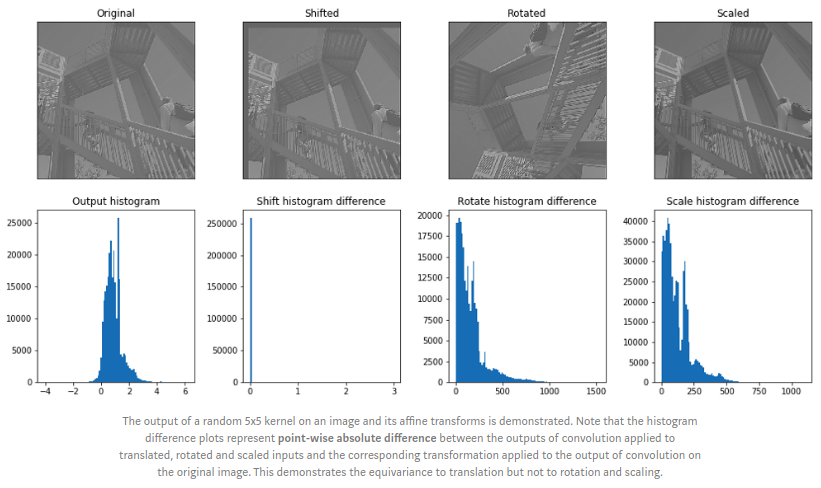
[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/13.comparison_sparse_interactions.png)

**Parameter Sharing**

* Parameter sharing refers to using same parameter for more than one function in a model.
* In convolutional neural net, each member of kernel is used at every position of input i.e. parameters used to compute different output units are tied together (all times their values are same).
* Sparse interactions and parameter sharing combined can improve eﬃciency of a linear function for detecting edges in an image

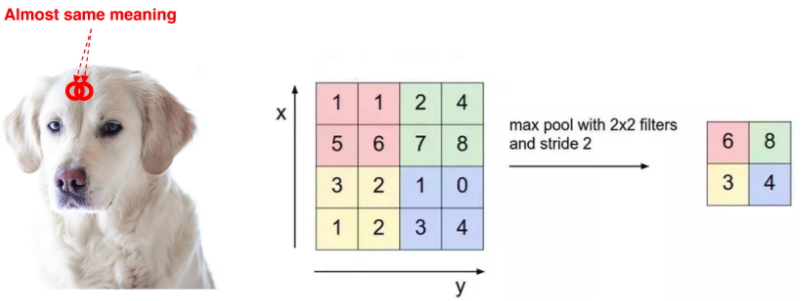
**Equivariance**

* Parameter sharing in a convolutional network provides equivariance to translation.  
  [](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/14.equivariance.png)
* Translation of image results in corresponding translation in the output map.
* Convolution operation by itself is not equivariant to changes in scale or rotation.

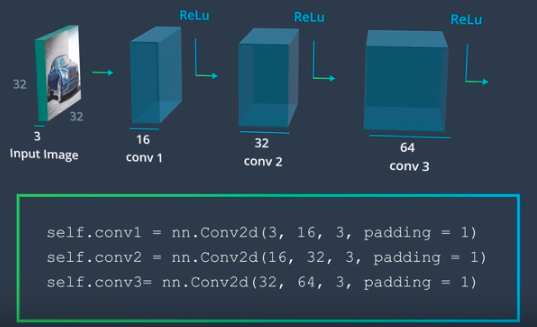
[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/15.equivariance_rotate_scale.png)

**Pooling**

* A convolution layer consists of 3 layers -
* Convolution
* Activation (Detector Stage)
* Pooling
* A pooling function replaces the output of net at a certain location with summary statistic of nearby outputs.
* Common summary statistics are : mean, median, weighted average.

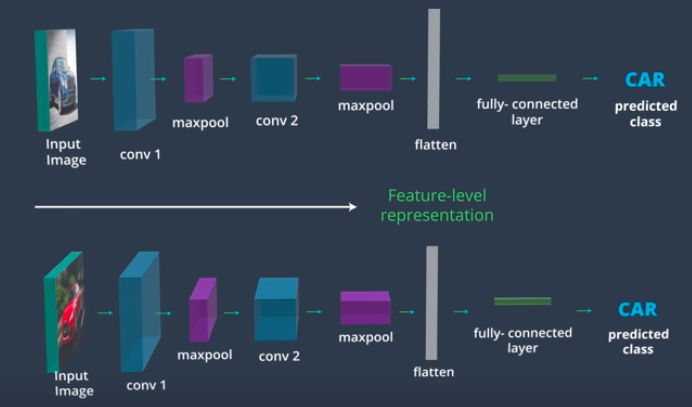
[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/17.max_pooling.png)

* Pooling makes the representation slightly translation invariant, in that small translations in the input do not cause large changes in output map.
* It allows detection of a particular feature if we only care about its existence, not its position in an image.
* Pooling reduces input size to the next layer in turn reducing the number of computations required upstream.

[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/35.image_classification_01.png)

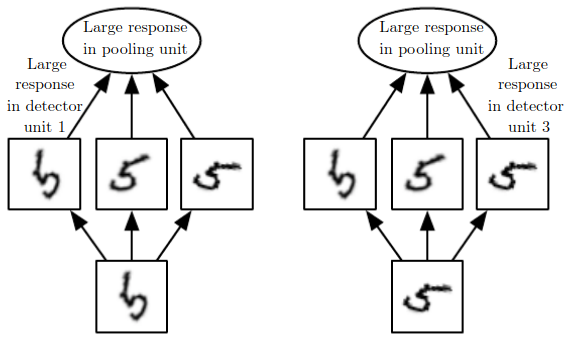
**Inputs having Variable Size**

* Classification layers requires fixed size of their inputs.
* Pooling makes their output fixed size by changing their pooling size, stride etc.

[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/36.image_classification_02.png)

**Learned Invariances**

* Pooling over feature channels can be used to develop invariance to certain transformations of the input.
* Units in a layer may be developed to learn rotated features and then pooled over. This property has been used in Maxout networks.

[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/16.maxout_function.png)

**Convolution and Pooling as an Infinitely Strong Prior**

**Weight Prior**

Assumptions about weights (before learning) in terms of acceptable values and range are encoded into the prior distribution of weights.

| **S.No.** | **Prior Type** | **Variance/Confidence Type** |
| --- | --- | --- |
| 1. | Weak | High Variance, Low Confidence |
| 2. | Strong | Narrow range of values about which we are confident before learning begins. |
| 3. | Infinitely strong | Demarkates certain values as forbidden completely assigning them zero probability. |

* Convolution imposes an infinitely strong prior by making the following restrictions on weights:
* Adjacent units must have the same weight but shifted in space.
* Except for a small spatially connected region, all other weights must be zero.
* Features should be translation invariant.
* If tasks relies on preserving specific spatial information, then pooling can cause on all features can increase training error.

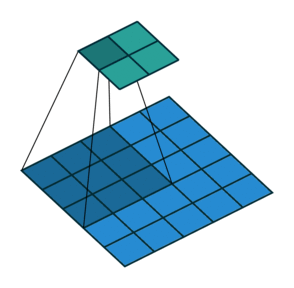
**Variants of the Basic Convolution Function**

In practical implementations of the convolution operation, certain modifications are made which deviate from standard discrete convolution operation -

* In general a convolution layer consists of application of several different kernels to the input. Since, convolution with a single kernel can extract only one kind of feature.
* The input is generally not real-valued but instead vector valued.
* Multi-channel convolutions are commutative iff number of output and input channels is the same.

**Effect of Strides**

* Stride is the number of pixels shifts over the input matrix.
* 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.
* Strides can be used to reduce the representation size.
* Below is an example representing 2-D Convolution, with (3 \* 3) Kernel and Stride of 2 units.

[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/19.stride_2.gif)

**Effect of Zero Padding**

* Convolution networks can implicitly zero pad the input V, to make it wider.
* Without zero padding,the width of representation shrinks by one pixel less than the kernel width at each layer.
* Zero padding the input allows to control kernel width and size of output independently.

**Zero Padding Strategies**

3 common zero padding strategies are:

| **Zero Padding Type** | **Properties** | **Example** |
| --- | --- | --- |
| Valid Zero-Padding | 1. No zero padding is used. 2. Output is computed only at places where entire kernel lies inside the input. 3. Shrinkage > 0 4. Limits #convolution layers to be used in network 5. Input's width = m, Kernel's width = k, Width of Output = m-k+1 |  |
| Same Zero-Padding | 1. Just enough zero padding is added to keep: 1.a. Size(Ouput) = Size(Input) 2. Input is padded by (k-1) zeros 3. Since the #output units connected to border pixels is less than that for centre pixels, it may under-represent border pixels. 4. Can add as many convolution layers as hardware can support 5. Input's width = m, Kernel's width = k, Width of Output = m |  |
| Strong Zero-Padding | 1. The input is padded by enough zeros such that each input pixel is connected to same #output units. 2. Allows us to make an arbitrarily deep NN. 3. Can add as many convolution layers as hardware can support 4. Input's width = m, Kernel's width = k, Width of Output = m+k-1 |  |

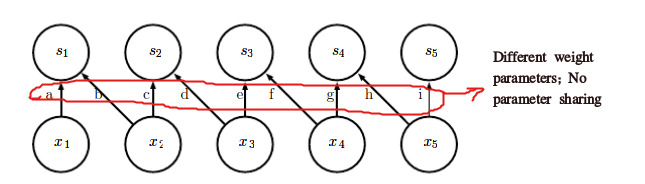
**Types of Convolution**

Comparing Unshared, Tiled and Traditional Convolutions

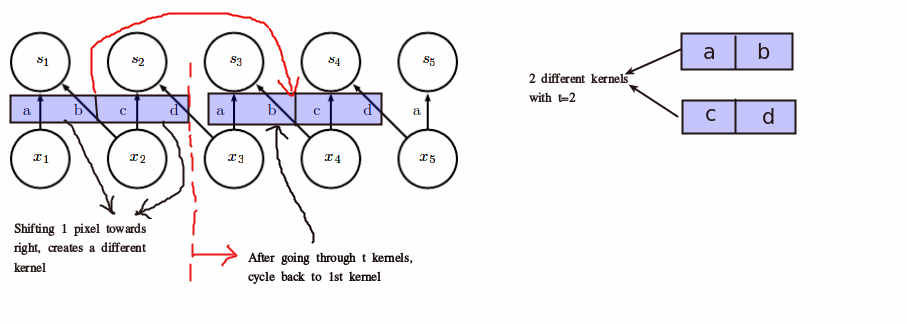
| **Convolution Type** | **Properties** | **Advantages and Disadvantages** |
| --- | --- | --- |
| Unshared Convolution | 1. No Parameter sharing. 2. Each output unit performs a linear operation on its neighbourhood but parameters are not shared across output units. 3. Captures local connectivity while allowing different features to be computed at different spatial locations. | Advantages 1. Reducing memory consumption 2. Increasing statistical eﬃciency 3. Reducing the amount of computation needed to perform forward and back-propagation. Disadvantages 1. requires much more parameters than the convolution operation. |
| Tiled Convolution | 1. Offers a compromise b/w unshared and traditional convoltion. 2. Learn a set of kernels and cycle/rotate them through space. 3. Makes use of parameter sharing. | Advantages 1. Reduces the #parameters in model. |
| Traditional Convolution | 1. Equivalent to tiled convolution with t=1. 2. Has the same connectivity as unshared convolution |  |

**Examples of Unshared, Tiled and Traditional Convolutions**

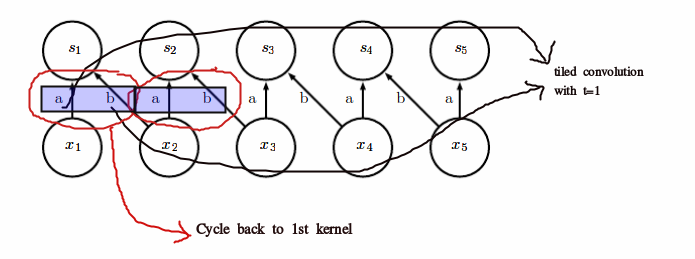
**Unshared Convolution**

[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/22.unshared_convolution.png)

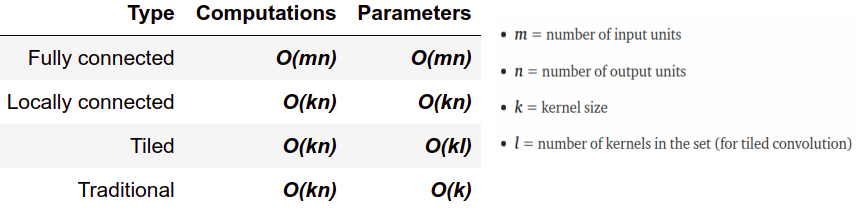
**Tiled Convolution**

[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/23.tiled_convolution.png)

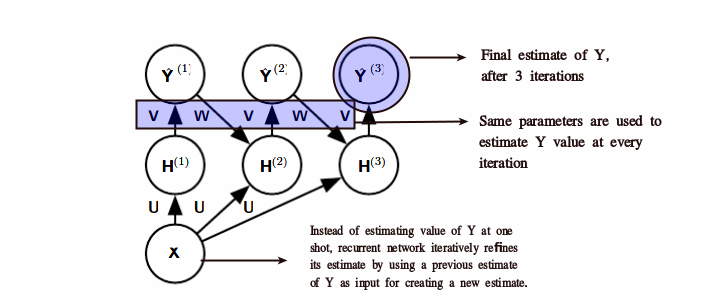
**Traditional Convolution**

[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/24.traditional_convolution.png)

**Comparing Computation Times**

[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/25.computation.png)

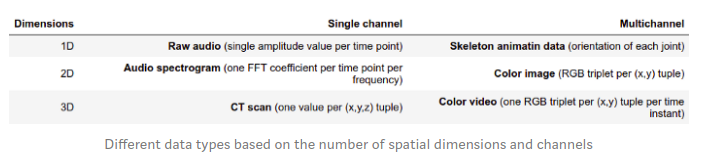
**Structured Outputs**

* Convolutional networks can be trained to output high-dimensional structured output rather than just a classification score.
* To produce an output map as same size as input map, only same-padded convolutions can be stacked.
* 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  
  [](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/26.structured_outputs.png)

| **Variable** | **Description** |
| --- | --- |
| X | Input image tensor |
| Y | Probability distribution over tensor for each pixel |
| H | Hidden representation |
| U | Tensor of convolution kernels |
| V | Tensor of kernels to produce estimation of lables |
| W | Kernel tensor to convolve over Y to provide input to H |

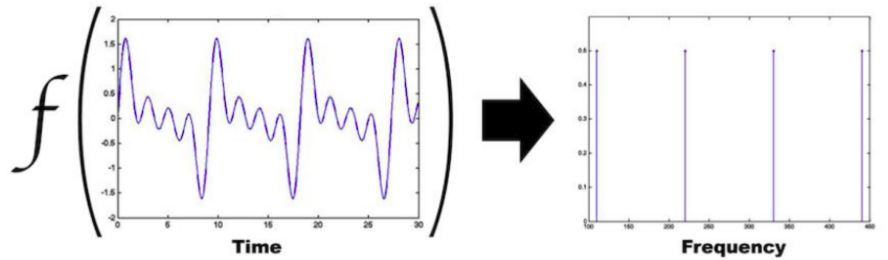
**Data Types**

* 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.
* When output is variable sized, no extra design change needs to be made.
* When output requires fixed size (classification), a pooling stage with kernel size proportional to input size needs to be used.

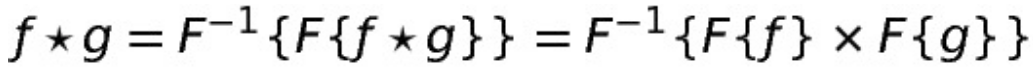
[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/27.single_vs_multichanel.png)

**Efficient Convolution Algorithms**

* **Fourier Transform**

The Fourier Transform is a tool that breaks a waveform (a function or signal) into an alternate representation, characterized by sine and cosines.  
[](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/33.fourier_transform.png)

**Separable Kernels**

* Convolution is equivalent to converting both input and kernel to frequency domain using a Fourier transform, performing point-wise multiplication of two signals:  
  [](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/30.eca_1.png)
* Converting back to time domain using an inverse Fourier transform.  
  [](https://github.com/purvasingh96/Deep-learning-with-neural-networks/blob/master/Chapter-wise%20notes/Ch_8_Convolutional_Neural_Networks/images/31.eca_2.png)
* When a d-dimensional kernel can be expressed as outer product of d vectors, one vector per dimension, the kernel is called separable.
* The kernel also takes fewer parameters to represent as vectors.

| **Kernel Type** | **Runtime complexity for d-dimensional kernel with w elements wide** |
| --- | --- |
| Traditional Kernel |  |
| Separable Kernel |  |

**Random and Unsupervised Features**

To reduce the cost of convolutional network training, we have to use features that are not trained in a supervised way:

* **Random Initialization:**
* Layers consisting of convolution followed by pooling naturally become frequency selective and translation invariant when assigned random weights.
* Randomly initialize several CNN architectures and just train the last classification layer.
* Once a winner is determined, train that model using a more expensive approach (supervised approach).
* Hand Designed Kernels: \* Used to detect edges at a certain orientation or scale.
* Unsupervised Training: \* Unsupervised pre-training may offer regularization effect. \* It may also allow for training of larger CNNs because of reduced computation cost.

**Greedy Layer-wise Pre-training**

Instead of training an entire convolutional layer at a time, we can train a model of a small patch:

* Train the ﬁrst layer in isolation.
* Extract all features from the ﬁrst layer only once.
* Once the first layer is trained, its output is stored and used as input for training the next layer.
* We can train very large models and incur a high computational cost only at inference time.