UNIT-4

· Convolutional Networks

The convolution operation, Pooling, Convolution, Basic Convolution Functions, Structured outputs, Data Types, Efficient Convolution Algorithms, Random or Unsupervised Features, Basis for Convolutional Networks.

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

Ex Time-series data, which can be thought of as a 1D grid taking values at regular time intervals.

Image data which can be thought of as a 2D grid of pixels.

The name "convolutional NN" indicates that the network employs a mathematical operation called convolution. It is a specialized kind of linear operation.

Convolutional networks are simply NN that use convolution in place of general matrix multiplication in at least one of their layers.

Convolutional networks stand out as an example of neuroscientific principles influencing deep learning.

CNNs (or Conv Nets are a class of Deep neural networks primarily designed for tasks involving visual data, such as image and video recognition, object detection, and image classification.

CNNs have been successful in various computer vision tasks due to their ability to automatically learn hierarchical representations from data.

Convolutional Layers:

- Convolutional layers are the core building blocks of CNNs. They apply convolution operations to input data using filters also called kernels to extract local patterns or features.

Convolution helps capture spatial hierarchies of features and reduces the need for manual feature engineering.

Pooling Layers:

Pooling layers are used to down-sample the spatial dimensions of the input volume, reducing computational complexity and controlling overfitting.

Common pooling operations include max pooling and average pooling, which retain the most important information from a group of values.

Activation Functions:

Non-linear activation functions eg. ReLU, are applied after convolution and pooling layers to introduce non-linearity into the model, allowing it to learn complex patterns.

Fully Connected Layers:

Fully connected layers are typically used towards the end of the network to perform high-level reasoning on the extracted features.

These layers connect every neuron in one layer to every neuron in the next layer, forming a dense, fully connected structure.

Loss Function: The loss function measures the difference b/w the predicted o/p and the actual target.
common loss functions for classification tasks include cross-entropy loss.

Training:
CNNs are trained using backpropagation and optimization algorithms like SGD to adjust the weights and biases of the network.

Architecture:
CNNs often follow a specific architecture pattern, such as alternating convolutional and pooling layers followed by fully connected layers.
Popular CNN architectures include LeNet-5, AlexNet, VGGNet, GoogLeNet (Inception), ResNet, and more.

Data Augmentation:
Data augmentation techniques such as rotation, flipping, and scaling, are often employed during training to increase the diversity of the training dataset and improve model generalization.

Convolutional networks have proven highly effective in image-related tasks, and their principles have been extended to other domains like NLP eg. using 1D convolutions for text processing and signal processing.

The Convolution Operation :-

In its most general form, convolution is an operation on two functions of a real valued argument.

$$s(t) = \int x(a)\, w(t-a)\, da.$$

This operation is called convolution. The convolution operation is typically denoted with an asterisk:

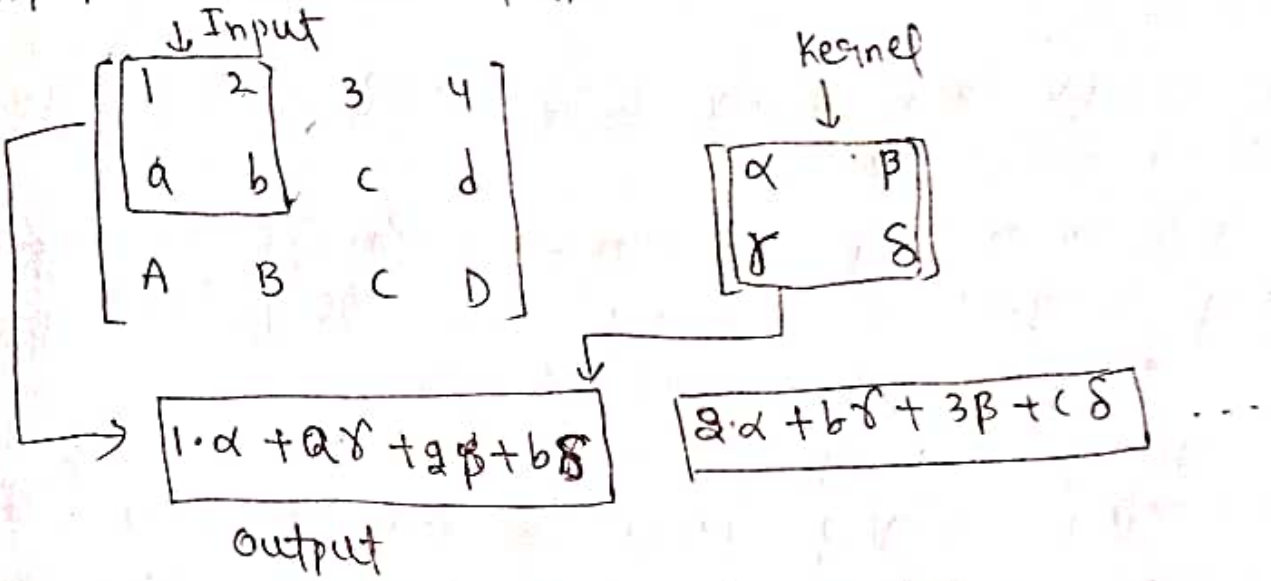
$$s(t) = (x * w)(t)$$

In convolutional network terminology, the first argument to the convolution is often referred to as the i/p and the second argument as the kernel. The output is referred to as the feature map.

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

In ML applications, the i/p is usually a multi-dimensional array of data and the kernel is usually a multidimensional array of parameters that are adapted by the learning algorithm.
We will refer to these multidimensional arrays to tensors.
many NN libraries implement a related function called the cross-correlation, which is the same as convolution but without flipping the kernel.
Discrete convolution can be viewed as multiplication by a matrix.
For univariate discrete convolution, each row of the matrix is constrained to be equal to the row above shifted by one element. This is known as a Toeplitz matrix.

In 2-dimensions, a doubly block circulant matrix corresponds to convolution.



2D convolution without kernel-flipping.

Motivation:

Convolution leverages three important ideas that can help improve ML system.

→ Sparse interactions (or) Sparse connectivity (or) sparse weights.

→ Parameter sharing

→ Equivariant representations.

Traditional NN layers use matrix multiplication by a matrix of parameters with a separate parameter describing the interaction b/w each i/p unit and each o/p unit.

Ex: When processing an image the i/p image might have thousands (or) millions of pixels, but we can detect small, meaningful features such as edges with kernels that occupy only tens (or) hundreds of pixels.

Parameter sharing refers to using the same parameter for more than one function in a model.

In a traditional NN, each element of the weight matrix is used exactly once when computing the o/p of a layer.

In a CNN, each member of the kernel is used at every position of the input.

Sparse connectivity and parameter sharing can dramatically improve the efficiency of a linear function for detecting edges in an image.

In case of convolution, the particular form of parameter sharing causes the layer to have a property called eigenvariance to translation. This means that if the i/p changes, the o/p changes in the same way.

When processing images, it is useful to detect edges in the first layer of a convolutional network.

Convolution operations are fundamental to CNN and are used to exact features from input data, particularly in image processing tasks.

The convolution operation involves sliding a filter also called a kernel over the input data and computing the dot product at each step.

Filter / Kernel:

A filter or kernel is a small matrix that contains learnable parameters.

The size of the filter is usually smaller than the input data.

Stride:

The stride is the step size at which the filter moves across the input data.

If stride = 1, means the filter moves one pixel at a time. Strides affect the spatial dimensions of the o/p.

Padding:

Padding involves adding extra pixels (usually zeros) around the input data.

Padding is often used to ensure that the filter can be applied to the border pixels without reducing the spatial dimensions too much.

Ex: valid - no padding
same - zero padding.

Mathematically, the convolution operation at position (i,j) of the o/p feature map O can be represented as:

$$O(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n).$$

where, I is the input data, K is the filter and m,n iterate over the filter dimensions.

Feature Map:

The result of the convolution operation is a feature map, which represents the presence of specific features in the i/p data.

The o/p of one convolutional layer becomes the input to the next layer.

Pooling:-

A typical layer of convolutional network consists of three stages. In first stage, the layer performs several convolutions in parallel to produce a set of linear activations.

In the 2nd stage, each linear activation is run through

a non-linear AF, such as the rectified linear AF. This stage is sometimes called the detector stage.
In 3rd stage, we use a pooling function to modify the o/p of the layer further.
The max pooling operation reports the max output within a rectangular neighborhood.
Other popular pooling functions include the avg of a rectangular neighborhood, the L2 norm of a rectangular neighborhood, or a weighted avg based on the distance from the central pixel.
Invariance to local translation can be a very useful property if we care more about whether some feature is present than exactly where it is.
When the number of parameters in the next layer is a function of its i/p size, this reduction in the i/p size can also result in improved statistical efficiency and reduced memory requirements for storing the parameters.
Pooling can complicate some kinds of NN architectures, that use top-down information, such as Boltzmann machines and autoencoders.

Pooling layers are a crucial component in CNNs and are used to down-sample the spatial dimensions of the input features maps.
Pooling helps reduce the computational load, control overfitting, and focus on the most important information in the data.
The two most common types of pooling are max pooling and avg pooling.

Max Pooling:

Max pooling is a type of pooling operation where, for each local region in the i/p feature map, the max value is taken.

Typically, a square filter (eg. 2x2, 3x3) is used.

Max pooling helps retain the most prominent features and reduces spatial dimensions.

Mathematically, represented as at position (i, j).

$$O(i,j) = \max_{m,n} I(i \times stride + m, j \times stride + n).$$

Average Pooling:

Avg pooling computes the avg value within each local region of the i/p feature map.

A square filter is typically used.

Avg pooling is computationally less intensive compared to max pooling.

Mathematically, represented as at position (i, j)

$$O(i,j) = \frac{1}{\text{filter size}} \sum_{m,n} I(i \times stride + m, j \times stride + n)$$

Global Average Pooling:

Global avg pooling is applied to the entire feature map.

Global avg pooling reduces the spatial dimensions to 1x1 for each feature map.

Pooling layers are usually inserted between consecutive convolutional layers in a CNN.

They help in reducing the spatial dimensions, this reduction is particularly important in large-scale networks to manage computational complexity and improve the models ability to generalize to new data.

Convolution and Pooling as an Infinitely Strong Prior

Priors can be considered weak or strong depending on how concentrated the probability density in the prior is.

A weak prior is a prior distribution with high entropy, such as a Gaussian distribution with high variance. Such a prior allows the data to move the parameters more or less freely.

A strong prior has very low entropy, such as a Gaussian distribution with low variance. Such a prior plays a more active role in determining where the parameters end up.

An infinitely strong prior places a zero probability on some parameters and says that these parameter values are completely forbidden, regardless of how much support the data gives to those values.

Implementing a convolutional net as a fully connected net with an infinitely strong prior would be extremely computationally wasteful.

Some convolutional network architectures are designed to use pooling on some channels but not on other channels, in order to get both highly invariant features and features that will not underfit when the translation invariance prior is incorrect.

Basic Convolution Functions :-

A color image has a red, green, and blue intensity at each pixel. In a multilayer convolutional network, the input to the second layer is the o/p of the first layer.

When working with images, we usually think of the input and output of the convolution as being 3-D tensors, with one index into the different channels and two indices into the spatial coordinates of each channel.

Convolutional networks usually use multi-channel convolution.

Zero padding the input allows us control the kernel width and the size of the o/p independently.

No-zero padding is called valid convolution.

Zero padding is called same convolution.

Tiled convolution offers a compromise b/w a convolutional layer and a locally connected layer.

Both locally connected layers and tiled convolutional layers have an interesting interaction with max-pooling: the detector units of these layers are driven by different filters.

In DL, convolutional operations are fundamental for feature extraction in tasks like image recognition.

Convolution Operation:

Inputs: Input data (image) and a convolutional filter (kernel).

Operation: The filter is applied to the input data through a convolution operation, where the filter "slides" on "convolves" over the input, computing the dot product at each position.

Output: The result is a feature map that represents the learned features in the input data.

Stride:

Stride is the step size with which the convolutional filter moves across the input data.

Effect: A larger stride reduces the spatial dimensions of the o/p feature map.

Padding:

Padding involves adding extra pixels around the input data.

Effect: Padding helps in applying the filters to border pixels and can be crucial for preserving spatial dimensions.

Activation Function:

After convolution, an AF (eg ReLU) is often applied element-wise to introduce non-linearity.

Effect: Non-linearity allows the model to learn complex patterns and relationships in the data.

Pooling:

Pooling layers are applied to down-sample the spatial dimensions of the feature maps.

Effect: Pooling reduces computational complexity and focuses on important features.

Fully Connected Layer:

Fully connected layers are often used for high level reasoning.

Effect: These layers connect every neuron in one layer to every neuron in the next, allowing the network to make predictions.

Loss Function:

The loss function measures the difference b/w predicted and actual output.

Optimization: During training, the model adjusts its parameters using optimization algorithms like stochastic gradient descent to minimize the loss.

Backpropagation:
The gradient of the loss is propagated backward through the network using the chain rule.

Purpose: It allows the model to update its parameters efficiently during training.

Each layer in the network extracts different levels of abstraction, allowing the model to recognize patterns and make accurate predictions.

Structured Outputs:-

Convolutional networks can be used to output a high-dimensional, structured object, rather than just predicting a class label for a classification task or real value for a regression task.

One issue that often comes up is that the o/p plane can be smaller than the i/p plane.

The greatest reduction in the spatial dimensions of the network comes from using pooling layers with large stride.

In order to produce an o/p map of similar size as the i/p, one can avoid pooling altogether.

Another strategy is to simply emit a lower-resolution grid of labels.

One strategy for pixel-wise labeling of images is to produce an initial guess of the image labels, then refine this initial guess using the interactions b/w neighboring pixels.

Repeating this refinement step several times corresponds to using the same convolutions at each stage, sharing weights b/w the last layers of the deep net.
This makes the sequence of computations performed by the successive convolutional layers with weights shared across layers of particular kind of recurrent network.

Once a prediction for each pixel is made, various methods can be used to further process these predictions in order to obtain a segmentation of the image into regions.

Graphical models can describe the probabilistic relationships b/w neighboring pixels.

Structured o/ps in CNNs refer to the scenario where the network is designed to predict complex, structured o/p data rather than a simple scalar or class label.

Image Segmentation:

In image segmentation, the goal is to assign a label or category to each pixel in an image.
The o/p is a structured map where each pixel corresponds to a specific class (eg. person, car).
CNN architectures designed for image segmentation typically o/p dense predictions across the entire i/p image.

Object Detection:

Object detection involves identifying and localizing multiple objects within an image.
The o/p in this case is a structured set of bounding

boxes, each associated with a class label.
CNN architectures for object detection often include both classification and regression components to predict class labels and bounding box coordinates.

Key-point Detection:
Keypoint detection tasks involve predicting the locations of specific points or landmarks in an image.
Ex: Facial keypoint detection aims to locate key points
- key points on a face, such as the eyes, nose, and mouth.

Sequence-to-Sequence Tasks:
In tasks like image captioning or language translation, CNNs may be part of a larger architecture that generates sequences of outputs.
The CNN might be responsible for extracting features from input data (eg. an image), and entire model generates a structured o/p sequence (eg. a sentence).
The CNN is often combined with other types of NN layers, such as recurrent layers (eg. LSTM) or attention mechanisms to handle the structured o/p.
The design of the o/p layer and the choice of the AF depend on the specific task and the desired o/p structure.
Ex: Softmax AF might be used for multi-class classification.
Sigmoid AF may be employed for binary classification.
In summary, structured o/ps in CNNs are essential for tasks where the o/p data has a more complex structure than a simple label.

Data Types:-

The data used with a convolutional network usually consists 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. These kinds of input simply cannot be represented by traditional, matrix multiplication-based NN.

Convolution is straightforward to apply; the kernel is simply applied a different number of times depending on the size of the input, and the o/p of the convolution operation scales accordingly.

	Single channel	Multi-channel.
1-D	Audio waveform	Skeleton animation data.
2-D	Audio data that has been preprocessed with a Fourier transform.	Color image data.
3-D	Volumetric data.	Color video data.

Convolution does not make sense if the i/p has variable size because it can optionally include different kinds of observations.

Input Data:

Image Data: The primary input to a CNN is often image data, represented as pixel values.

Images can be in various formats, such as grayscale or RGB and have specific dimensions (h, w and channels).

Convolutional filters / kernels:

Weights: Convolutional layers in CNN use filters or kernels,

which are small learnable matrices of weights. These weights are adjusted during training to capture patterns and features in the o/p data.

Activations:

Feature Maps: The o/p of convolutional layers is represented by feature maps, which are generated by applying the convolution operation to the i/p data using learned filters.

Activation Maps: An activation map is obtained by applying an AF (eg ReLU) to the feature maps, introducing non-linearity.

Pooling:

Pooled Features: Pooling layers down-sample the spatial dimensions of feature maps.

The o/p of pooling layers consists of pooled features, which represent the most important information in the local regions.

Fully Connected Layers:

Weights and Biases: In fully connected layers, each neuron is connected to every neuron in the previous layer.

The weights and biases associated with these connections are adjusted during training.

Output Layer:

The final layer of the CNN produces o/p predictions.

Loss Function: During training, the CNN compares its predictions with ground truth labels or annotations. The loss function quantifies the diff b/w predictions and ground truth, and its result is used to update the model parameters.

Understanding the data types involved in a CNN is essential for designing and implementing effective architectures.

It involves managing the I/p data format, adjusting filter weights, handling AFs, and interpreting the o/p predictions.

Efficient Convolution Algorithms:-

Modern convolutional network applications often involve networks containing more than one million units.

In many cases it is also possible to speed up convolution by selecting an appropriate convolution algorithm.

Convolution is equivalent to converting both the input and the kernel to the frequency domain using a Fourier transform, performing point-wise multiplication of the two signals, and converting back to the time domain using an inverse Fourier transform.

When a d-dimensional kernel can be expressed as the outer product of d vectors, one vector per dimension, the kernel is called "seperable".

When the kernel is separable, naive convolution is inefficient.

Efficient convolution algorithms are crucial for the performance of CNNs in DL. Convolution is computationally intensive operation, and optimizing its computation can significantly speed up training and inference.

Fast Fourier Transform (FFT) based Convolution:

This approach leverages the convolution theorem from signal processing, which states that convolution in the spatial domain is equivalent to pointwise multiplication in the frequency domain.
FFT based convolution can be faster for large kernel sizes, as the complexity is reduced from $O(N^2)$ to $O(N \log N)$, where N is the size of the input.

Winograd Minimal Filtering Algorithm:

Winograd minimal filtering algorithm is an algorithmic optimization that reduces the number of multiplications required for convolution.
It is particularly effective for small convolutional filters.

Depthwise Separable Convolution:

Depthwise separable convolution decomposes the standard convolution into two separate operations: depthwise convolution and pointwise convolution.

This reduces the number of parameters and computations, ~~making~~ & especially in mobile and edge computing scenarios.

Im2Col and Col2Im Techniques:

Image to column and column to image techniques reshape the input data and filters to simplify the convolution operation, allowing for more efficient matrix multiplication-based implementations.

These techniques are often used in libraries like CUDA Deep NN library for GPU-accelerated convolutional operations.

Dilated / Atrous Convolution:

Dilated convolution introduces gaps between filter elements, effectively increasing the receptive field without significantly increasing computation.

It is commonly used in architectures like DRN- Dilated Residual Network and is particularly useful for capturing larger contextual information.

Grouped Convolution:

Grouped convolution divides the input channel into groups, and each group is convolved separately. The results are then concatenated.

It is especially useful when the I/P has a large number of channels.

Pruning and Quantization:

Model pruning involves removing unnecessary weights, and quantization involves reducing the precision of weights and activations.

These techniques reduce the overall computation and memory requirements, making models more efficient for deployment.

Efficient convolution algorithms are often implemented in DL libraries and frameworks, and their adoption depends on factors such as hardware architecture,

available memory, and the specific characteristics of the model and data.

Random or Unsupervised Features:

The most expensive part of convolutional network training is learning the features.

The o/p layer is usually relatively inexpensive due to the small number of features provided as input to this layer after passing through several layers of pooling.

When performing supervised training with gradient descent, every gradient step requires a complete run of forward propagation and backward propagation through the entire network.

One way to reduce the cost of convolutional network training is to use features that are not trained in a supervised fashion.

There are three basic strategies for obtaining convolution kernels without supervised training.

One is to simply initialize them randomly. Another is to design them by hand, for example by setting each kernel to detect edges at a certain orientation or scale.

Finally, one can learn the kernels with an unsupervised criterion. For example, apply k-means clustering to small image patches, then use each learned centroid as a convolution kernel.

Learning the features with an unsupervised criterion allows them to be determined separately from the

classifier layer at the top of the architecture.

An expensive way to choose the architecture of a convolutional network is first evaluate the performance of several convolutional network architectures by training only the last layer, then take the best of these architectures and train the entire architecture using a more expensive approach.

An intermediate approach is to learn the features, but using methods that do not require full forward and back-propagation at every gradient step.

As with multilayer perceptrons, we use greedy layer-wise pretraining, to train the first layer in isolation, then extract all features from the first layer only once, then train the second layer in isolation given those features, and so on.

The canonical example of greedy layer-wise pretraining of a convolutional model is the convolutional deep belief network.

Today, most convolutional networks are trained in a purely supervised fashion, using full forward and back-propagation through the entire network on each training iteration.

Unsupervised pretraining may offer some regularization relative to supervised training, & it may simply allow us to train much larger architectures due to the reduced computational cost of the learning rule.

Random & unsupervised features in CNN refer to the idea of learning features from the data without explicit supervision, meaning that the model is not provided with labeled training examples during the learning process.

Randomly Initialization of filters:-

In the initial layers of a CNN, filters are often initialized randomly before being trained using SL. The idea is that these filters can capture low-level features like edges, corners, and textures, which are common across various types of images.

These randomly initialized filters are refined during the training process through backpropagation and GD, adjusting their weights to learn more specific features relevant to the given task.

Unsupervised Pre-training:

Unsupervised pre-training involves training a CNN on a large dataset using an USL approach before fine-tuning the network for a specific supervised task.

Autoencoders and RBMs - Restricted Boltzmann Machines are examples of USL models that can be used for pre-training.

The motivation behind incorporating unsupervised & randomly initialized features in CNNs is ~~to~~ to automatically learn meaningful representations from the data.

Random initialization provides the network with a starting point, and unsupervised pre-training allows the model to discover patterns and relationships in the absence of labeled data.

The choice blw random initialization, unsupervised pre-training, & direct end-to-end supervised training depends on factors such as the availability of labeled data, computational resources etc.

The Neuroscientific Basis for Convolutional Networks:

Convolutional networks are perhaps the greatest success story of biologically inspired AI.

The history of convolutional networks begins with neuroscientific experiments long before the relevant computational models were developed.

Neurophysiologists David Hubel and Torsten Wiesel collaborated for several years to determine many of the most basic facts about how the mammalian vision system works. They won nobel prize.

A part of the brain called V1, also known as the primary visual cortex. V1 is the first area of the brain that begins to perform significantly advanced processing of visual input.

In this cartoon view, images are formed by light arriving in the eye and stimulating the retina, the light sensitive tissue in the back of the eye.

The image then passes through the optic nerve and a brain region called the lateral geniculate nucleus.

A convolutional network layer is designed to capture three properties of V1:

→ V1 is arranged in a spatial map.
→ V1 contains many simple cells.
→ V1 also contains many complex cells.

The closest analog to a convolutional networks last layer of features is a brain area called the inferotemporal cortex (IT).

The human eye is mostly very low resolution, except for a tiny patch called the fovea. The fovea only observes an area about the size of a thumbnail held at arms length.

The human brain makes several eye movements called saccades to glimpse the most visually salient or task-relevant parts of a scene.

In the content of DL, attention mechanisms have been most successful for natural language processing.

The human visual system is integrated with many other senses, such as hearing, and factors like our moods and thoughts.

The human visual system does much more than just recognize objects. It is able to understand entire scenes including many objects and relationships b/w objects, and processes rich 3D geometric information need for our bodies to interface with the world.

The brain probably uses very different activation and pooling functions. An individual neuron's activation probably is not well characterized by a single linear filter response.

Lang and Hinton (1988) introduced the use of back-propagation to train time delay NN (TDNNs). To use contemporary terminology, TDNNs are one-dimensional convolutional networks applied to time seri

The Gabor function describes the weight at a 2D-point in the image. The Gabor function for $w(x,y)$ is

$$w(x,y;\alpha,\beta_x,\beta_y,f,\phi,x_0,y_0,T) = \alpha \exp(-\beta_x x'^2 - \beta_y y'^2)\cos(fx+\phi$$

Local Receptive Fields:

CNNs are inspired by the idea of mimicking the visual perception of animals, where neurons in the visual cortex respond to specific regions of the visual field.

In CNNs this concept is implemented through local receptive fields.

Neurons in the first layer of a CNN are connected only to a small region of the i/p data, known as the receptive field.

Hierarchical Structure:

CNNs typically have a hierarchical structure with multiple convolutional and pooling layers. Each layer learns complex features.

Translation Invariance:

CNNs inherently exhibit translation invariance, meaning that they can recognize patterns regardless of their spatial location in the i/p.

These principles have been crucial in the success of CNNs in computer vision applications.