Unit 3 : Convolutional Neural Network (I)

Introduction

1. The connection pattern of neurons of CNNs is similar to a unique feed forward artificial neural network, modelled after the visual cortex.

CNN Architecture

1. Consists of 2 components:
2. Feature extraction → uses a convolution tool to separate and identify distinct characteristics of a picture for study
3. A fully connected layer makes use of convolution process output and determines the class of the picture.
4. CNN feature extraction minimises the quantity of features and it generates a new features that compile an initial set of features into a single new feature.

Layers

Made up of 3 kinds of layers:

1. Fully connected layers
2. Pooling layers
3. Convolutional layers

Convolutional layer →

1. First layer → extract different characteristics from the input photos.
2. A filter of a specific size MxM is applied on to the the input image. The dot product of the filter and the input image pixels is computed to find the feature map.
3. Feature map → provides information about different features such as edges, corners, and the feature map reflects the activation of these features across the image.
4. The result is transferred to the further layers of the CNN to extract high level features by combining information from initial feature maps.
5. Stride → Refers to the number of pixels by which the filter moves across the input image at each step during the convolution operations.
6. Stride = 1, moves 1 pixel at a time, scanning the input image.
7. If more than 2, filter skips over some pixels, reducing the size of the feature map. (strided convolution)
8. Strided convolution → reduces the cost of convolution operation, beneficial for large images.
9. Low level characteristics like edge, color are captured by the first layer.
10. During convolution the dimensionality of the feature map can either reduce or stay the same, or increase.

11. Padding → Control the dimension of the output feature maps. Involves adding extra pixels before the convolution operation.
    a. Valid padding → no padding added. results in decrease in the size
    b. Same padding → padding added to the input image, zeroes are added around the input image. preserves information about the image
12. Padding ensures → information at the edges of the input is considered during convolution, preventing loss of information at the borders.

> 💡 IN SHORT :
>
> In convolutional layer :
>
> Image pixels * filter = feature map
>
> strides, padding (valid and same)

Pooling layer →

1. Used to reduce the size of convolved features.
2. This is used to decrease the burden for processing the data.
3. Also extracts dominant characteristics that are invariant to rotation and translation.
4. 2 types of pooling
    a. Max pooling → returns the max value from the area of input image covered by the pooling kernel.
    b. Average pooling → average of all the values from the area of the input image covered by the kernel is returned.
5. Noise suppression and dimensionality reduction → max pooling acts as a noise suppressant because it can return the max value from the region,

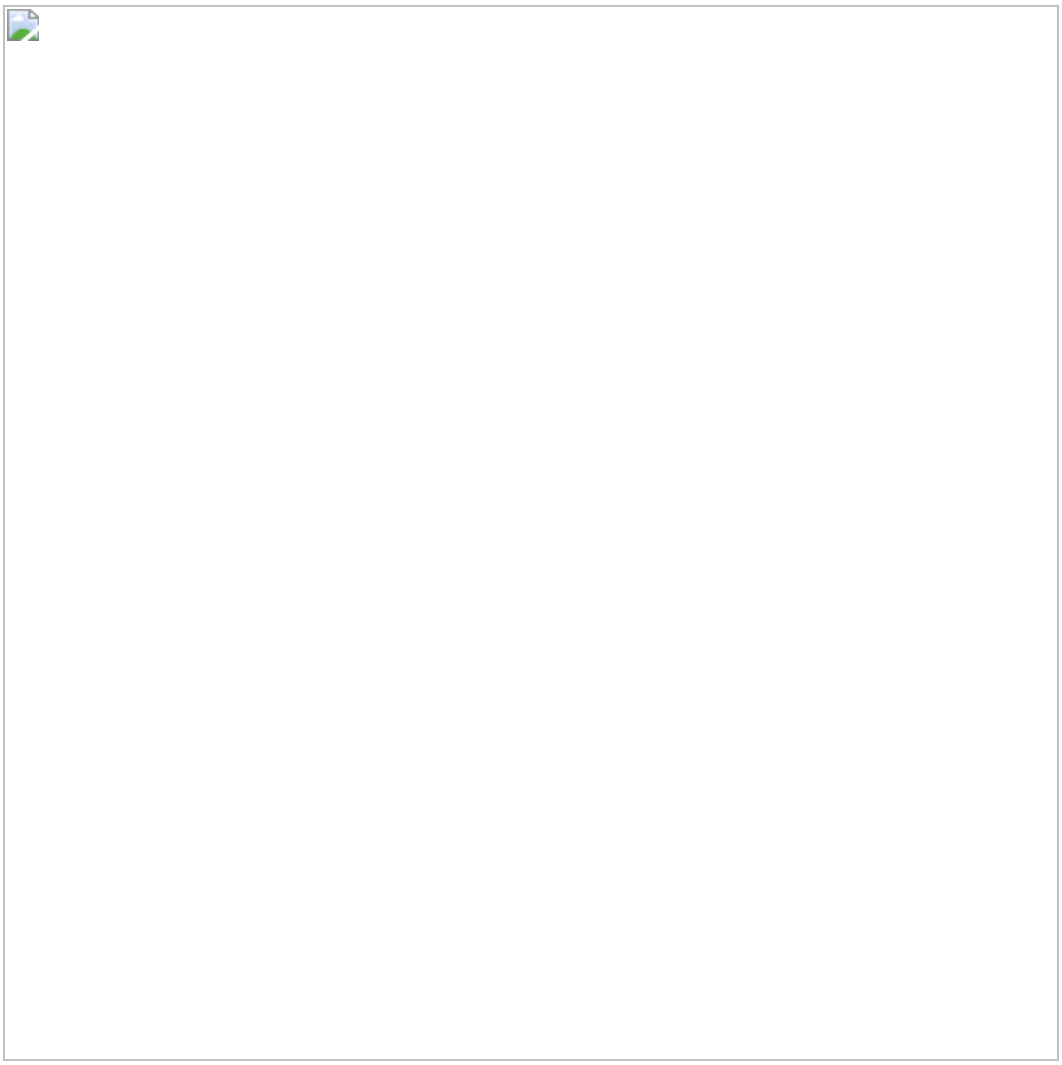
effectively reducing the impact of noisy activations. and by selecting a subset of features, it reduces the dimensions.

6. Pooling + convolutional layer = initial layer of CNN
7. After pooling and convolutional layers, the output is flattened to prepare it for classification purposes.

> 💡 Pooling layer → pooling - (max and avg), noise suppression, dimensionality reduction and flattening

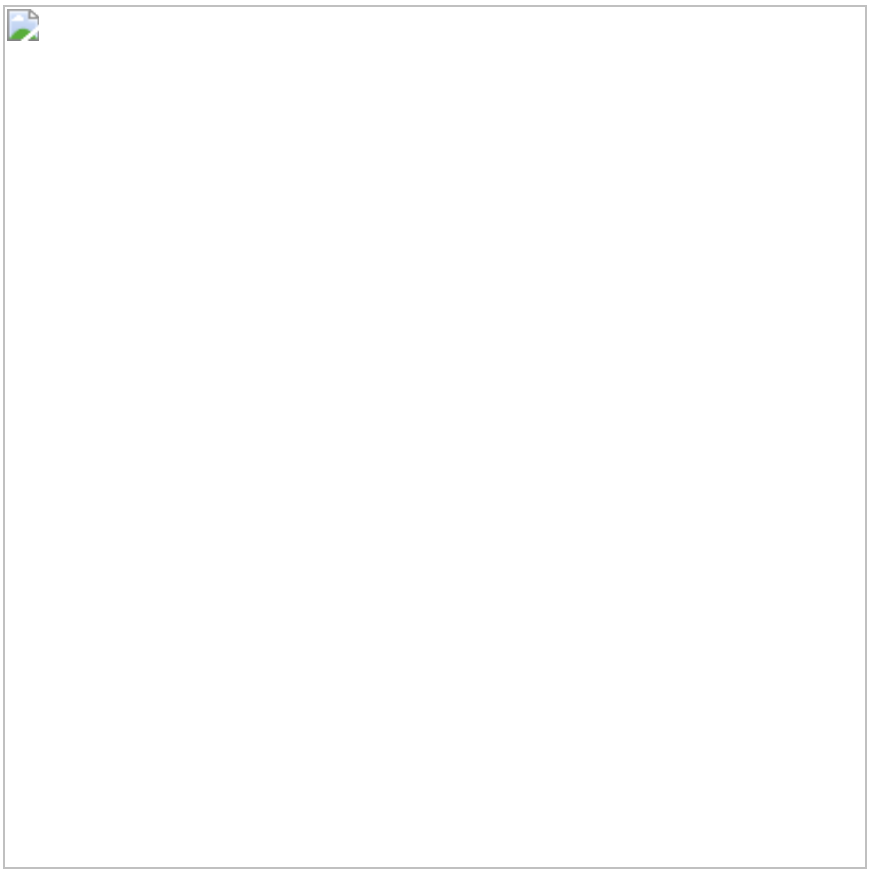
Fully connected layer →

1. After the convolutional layers and pooling, fully connected layers are designed to learn from non linear combinations of the high level features.
2. Outputs of convolutional layers are converted into a column vector and a multi layer perceptron is added to it.
3. A flattened output is recieved from the fully connected layer, it consists of multiple layers of neurons.
4. Backpropogation → adjusts the weights of the connections between neurons to minimize the difference between predicted and actual output.
5. Softmax classification → final layer of the network is applied to a softmax function, for multi class classification
6. Training → model is trained across different images, it learns to identify dominant features.

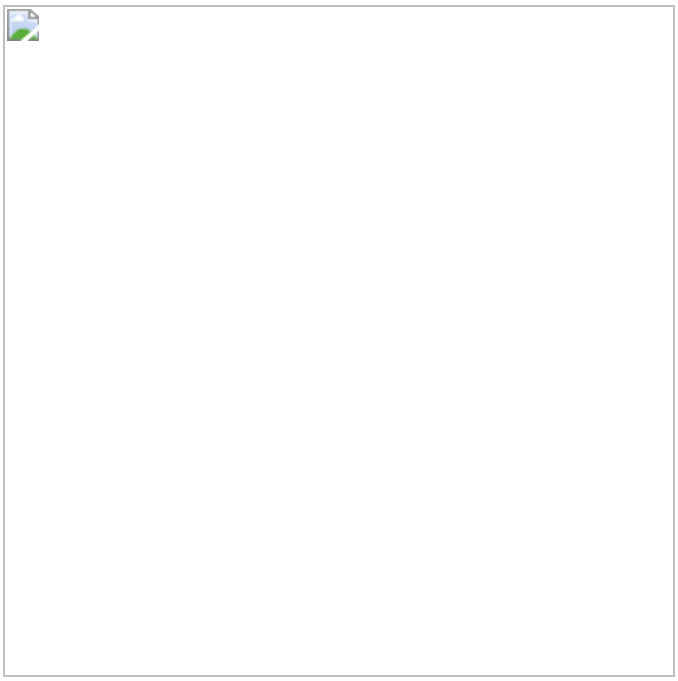


ReLU

1. Hidden layers of trained CNN have neurons have abstract characteristics.
2. ReLU is the most prevalent non negative activation function as its abstract representations are independent of one another, and they are preferred for CNNs.



3. ReLU can be calculated easily because all that is required is to compare the input to the value 0.
4. Computing the gradient of the neuron is inexpensive.



5. ReLu also leads to the absense of the “vanishing gradient” prb that frequently occurs when employing sigmoidal functions. This issue occurs during the training of deep neural networks, particularly those with many layers. It occurs when the gradients of the loss function with respect to the parameters (weights) of the network become extremely small as they are propagated back through the network during the training process. This problem is especially pronounced in networks that utilize activation functions with derivatives that tend to be very close to zero in certain regions.

Explain the RELU Layer in detail and what are the advantages of sigmoid over RELU (PYQ)

The Rectified Linear Unit (ReLU) is a type of activation function commonly used in neural networks, including convolutional neural networks (CNNs). Let's delve into its details and advantages over the sigmoid activation function:

ReLU Layer:

1. **Definition**: The ReLU activation function is defined as $f(x) = \max(0, x)$, which means it outputs the input directly if it's positive, and outputs zero otherwise.

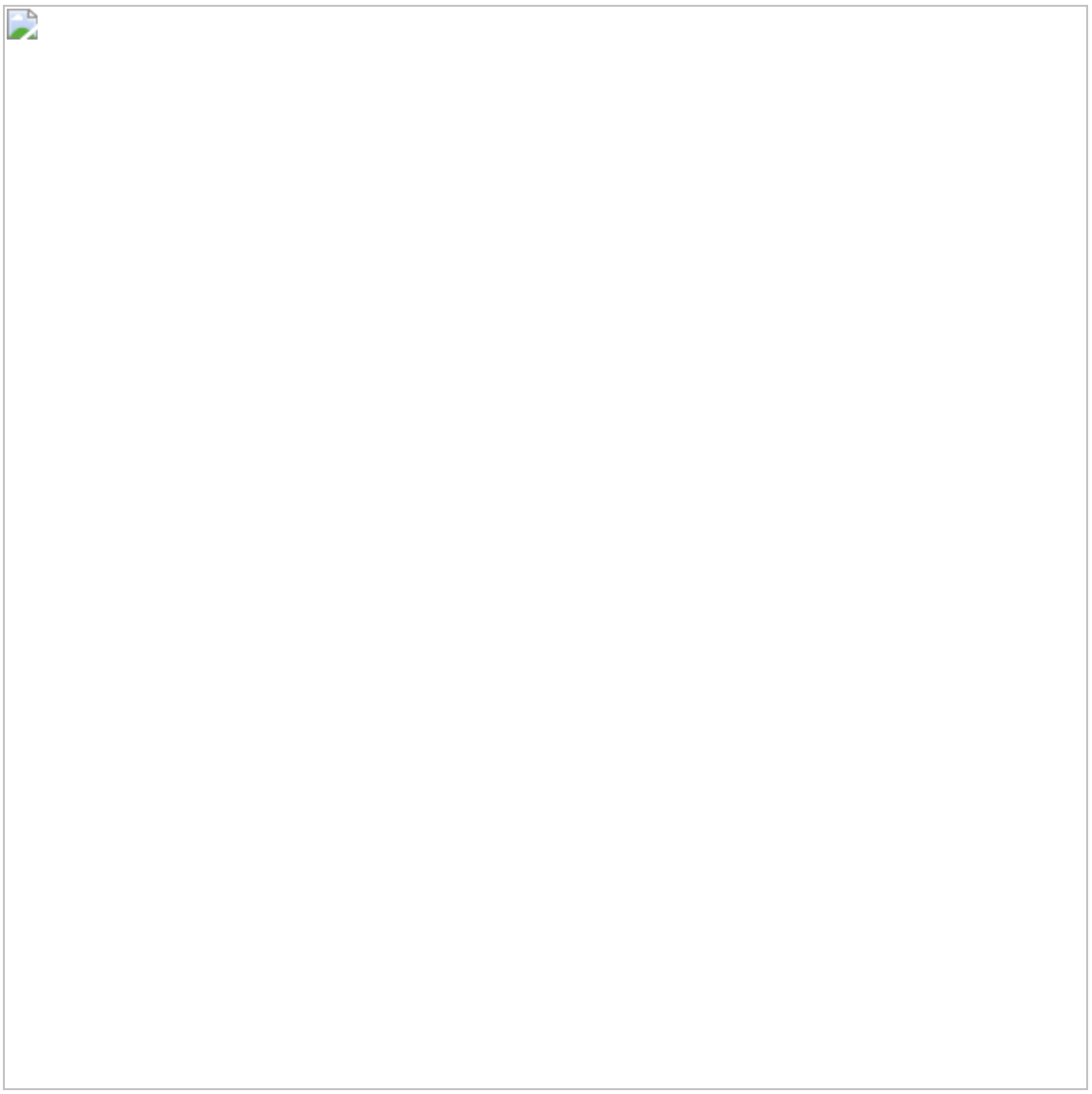
2. **Simplicity**: ReLU is computationally efficient to compute, as it involves only a simple thresholding operation. This simplicity contributes to faster training and inference times in neural networks.
3. **Sparsity**: ReLU introduces sparsity in the network because it outputs zero for all negative inputs. This sparsity can help reduce the likelihood of overfitting by limiting the number of active neurons and encouraging the network to learn more robust representations.
4. **Non-linearity**: Despite its simplicity, ReLU is a non-linear activation function. It allows neural networks to learn complex, non-linear relationships between inputs and outputs, which is essential for solving many real-world problems.
5. **Addressing Vanishing Gradient Problem**: ReLU helps mitigate the vanishing gradient problem, which can occur with activation functions like sigmoid and tanh. By avoiding saturation in the positive range and allowing gradients to flow more freely during backpropagation, ReLU facilitates the training of deeper networks.
6. **Advantages over Sigmoid**:
    - **Avoiding Saturation**: Sigmoid activation functions can suffer from saturation, particularly for extreme input values where the gradient approaches zero. ReLU does not suffer from this issue, as it only saturates in one direction (at zero).
    - **Faster Training**: ReLU typically leads to faster convergence during training compared to sigmoid and tanh. This is because ReLU does not have the vanishing gradient problem to the same extent, enabling more efficient weight updates and faster learning.
    - **Sparse Activation**: ReLU encourages sparsity in the network, which can be beneficial for memory and computational efficiency, especially in large-scale models.

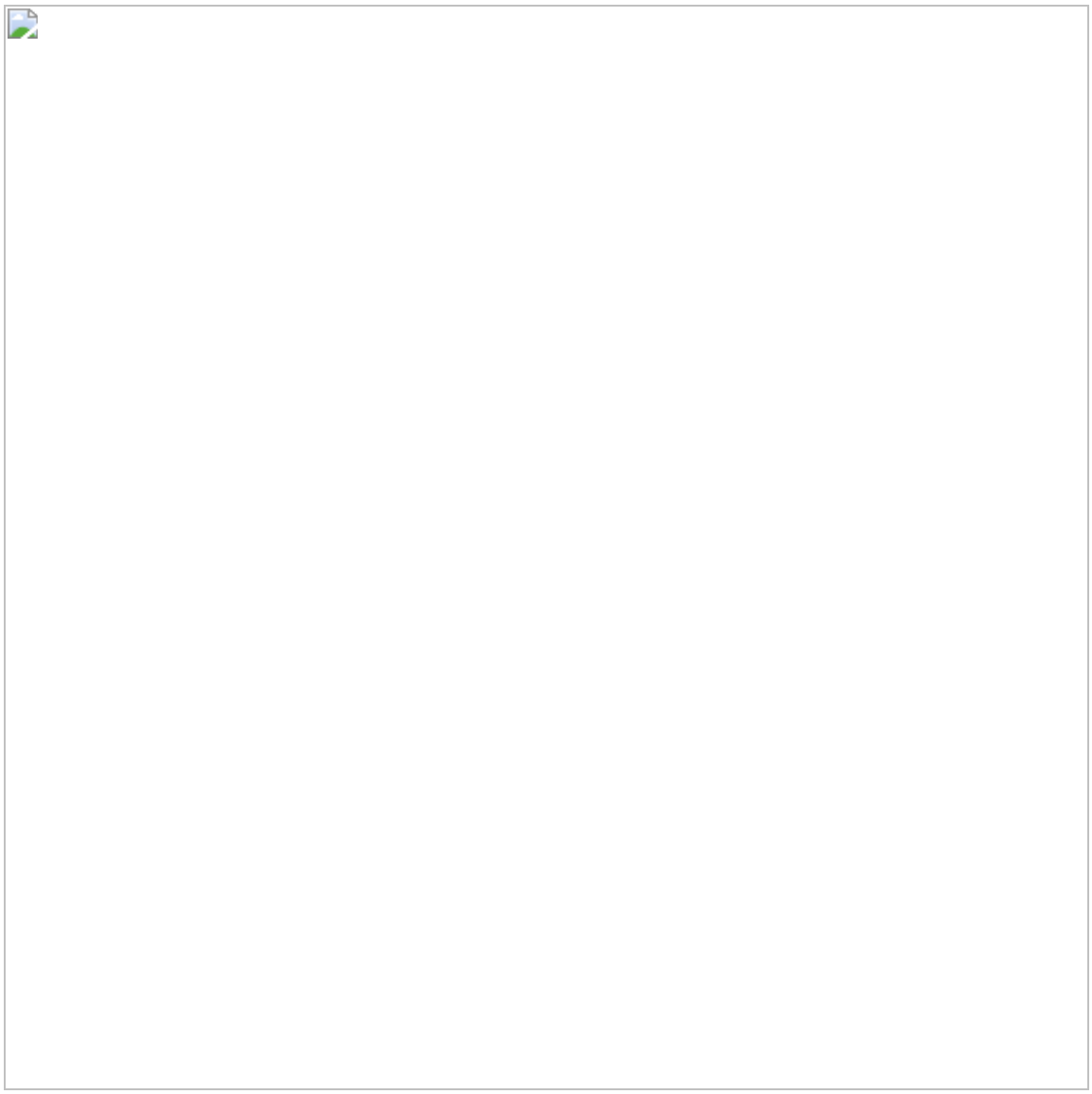
In summary, ReLU is a widely used activation function in neural networks due to its simplicity, non-linearity, and effectiveness in mitigating the vanishing gradient problem. Its advantages over sigmoid include faster training, avoidance of saturation issues, and promotion of sparsity in the network.

💡 1. Give the function
2. simple
3. reduces overfitting
4. non linear
5. solves vanishing gradient

Dropout layer

1. This layer acts as a mask, eliminating some neurons contribution to the subsequent layers while maintaining the functionality of other neurons.
2. This avoids overfitting, otherwise the first set of examples would have a large impact on training, and hence the exampls in the subsequent layers wont be learned.





The dropout layer is a regularization technique used in neural networks, including convolutional neural networks (CNNs), to prevent overfitting and improve generalization performance. Here's an explanation of the dropout layer in CNNs:

Definition:
The dropout layer randomly sets a fraction of the input units (neurons) to zero during training. This means that the output of the dropout layer for each training example is a "masked" version of the input, where a random subset of neurons is deactivated. During inference (testing), dropout is typically turned off, and all neurons are used.

How it works:

1. **Random Dropout:** During training, each neuron in the dropout layer has a probability \(p \) of being "dropped out" (set to zero). This probability is a hyperparameter typically set between 0.2 and 0.5.
2. **Stochastic Training:** By randomly dropping out neurons, the network is forced to learn redundant representations of the data. This prevents neurons from relying too heavily on specific features and encourages them to learn more robust and generalizable representations.
3. **Ensemble Effect:** Dropout can be seen as training multiple "subnetworks" in parallel, where each subnetwork consists of a different combination of active neurons. During inference, the predictions of these subnetworks are averaged, leading to an ensemble effect that improves generalization performance.

Advantages:

1. **Regularization:** Dropout helps prevent overfitting by reducing the co-adaptation of neurons and encouraging the network to learn more robust features.
2. **Improved Generalization:** By training with dropout, the network learns to make predictions that are more resilient to noise and variations in the input data, leading to better generalization performance on unseen examples.
3. **Reduced Sensitivity to Initialization:** Dropout reduces the network's sensitivity to the choice of initial weights, as it encourages the network to learn redundant representations that are less dependent on specific weight configurations.

Implementation:
Dropout layers can be easily incorporated into CNN architectures using popular deep learning libraries like TensorFlow or PyTorch. They are typically added after the fully connected layers or convolutional layers in the network architecture.

In summary, the dropout layer is a powerful regularization technique for training CNNs, helping to prevent overfitting and improve generalization performance by randomly dropping out neurons during training.

Interleaving between layers

Interleaving of layers → Different layers of CNN mixed or repeated any number of times

1. Interleaving between layers in a CNN refers to the idea of introducing connections between non-adjacent layers in the network architecture.
2. This concept is typically associated with skip connections or shortcut connections, which allow information to bypass certain layers and be propagated deeper into the network.
3. The most common form of interleaving in CNNs is through the use of skip connections in architectures like residual networks (ResNets) and densely connected networks (DenseNets).
4. In these architectures, skip connections are added between layers that are not necessarily adjacent to each other. These connections allow gradients and information to flow more directly from earlier layers to later layers during training, mitigating the vanishing gradient problem and enabling the training of very deep networks.

By interleaving connections between layers, CNNs can facilitate the flow of information through the network and potentially improve the representational capacity and learning ability of the model. This can lead to better performance on tasks such as image classification, object detection, and semantic segmentation.

Local Response Normalisation

Local Response Normalization (LRN) is a technique used in convolutional neural networks (CNNs) to normalize the responses within a local neighborhood across feature maps. Here's an explanation of how LRN works in CNNs:

Definition:
LRN operates on each spatial location independently and normalizes the responses of a neuron by the responses of its neighboring neurons within the same feature map. The normalization is done using a formula that considers the squared sum of activations in a local neighborhood.

How it works:

1. **Local Neighborhood:** For each neuron in a feature map, LRN considers a local neighborhood centered around that neuron. The size of the neighborhood is determined by a hyperparameter called the depth radius or the kernel size.
2. **Normalization:** LRN normalizes the activation of each neuron by dividing it by a factor that includes the sum of the squared activations of neurons within the local neighborhood. This normalization is done independently for each neuron and for each feature map.
3. **Formula:** The formula for LRN normalization can vary, but a common formulation is:

   $$\text{LRN}(x_{i,j}) = \frac{x_{i,j}}{\left(k + \alpha \sum_{l=max(0,i-n/2)}^{min(N-1,i+n/2)} x_{l,j}^2 \right)^\beta}$$

   - $x_{i,j}$ is the activation of the neuron at spatial position (i, j) within a feature map.
   - N is the total number of neurons in the local neighborhood.
   - n is the depth radius or kernel size, determining the size of the local neighborhood.
   - $k, \alpha,$ and β are hyperparameters controlling the normalization process.

Advantages:

1. **Normalization:** LRN normalizes the responses within local neighborhoods, which can help in controlling the scale of the responses and improving the stability of the training process.
2. **Contrast Enhancement:** LRN can enhance the contrast between different features by normalizing responses relative to their local surroundings, potentially improving the discriminative power of the network.

Implementation:
LRN layers were commonly used in earlier CNN architectures like AlexNet, but they have become less common in modern architectures. However, LRN functionality can be achieved through other normalization techniques like batch

normalization or group normalization, which are more widely used in current CNN architectures.

In summary, Local Response Normalization (LRN) is a technique used in CNNs to normalize responses within local neighborhoods across feature maps. It can help control the scale of activations and enhance contrast between features, though it's less commonly used in modern architectures compared to other normalization techniques.

Training a convolutional network

1. Input layer
    - This step resets the data
    - The size of the layer is equal to the square root of the number of pixels.
    - We also need to specify if the image is color or not. I
    - If its color, size 3×3 for RGB otherwise its 1.
2. Convolutional Layer
    - Consistent layers are created, and we apply various filters to learn important featuers of the network, and the size of the kernel and the volume of the filter is decided.
3. Pooling layer
    - This layer reduces the size of input, it does that by taking the maximum value of the sub matrix. (max pooling and avg pooling)
4. Additional convolutional and pooling layers
    - We can add as many convolutional and pooling layers we want
5. Dense layer / Fully connected layer
    - Flattens the previous layers to form fully joined layers, and we can use different activation functions for dropout.
6. Train the model

- Tensorflow, keras, pytorch can be used for building the cnn architecture and train and test it.

7. Test and evaluate the model
    - Test and evaluate the performance of the model