Module 3: Convolutional Neural Networks

Convolutional Neural Networks -Architecture, Convolution operation, Motivation, pooling.

Variants of convolution functions, Structured outputs, Data types, Efficient convolution algorithms, Applications of Convolutional Networks,

Pre-trained convolutional Architectures :AlexNet, ZFNet, ResNet

1. Convolutional Neural Networks (CNNs)

CNNs are specialized neural networks used primarily for image-related tasks, such as classification, object detection, and recognition.(They are designed to process grid-like data, such as images.) They are inspired by the visual processing mechanisms in the human brain, CNNs excel at capturing hierarchical patterns and spatial dependencies within images.

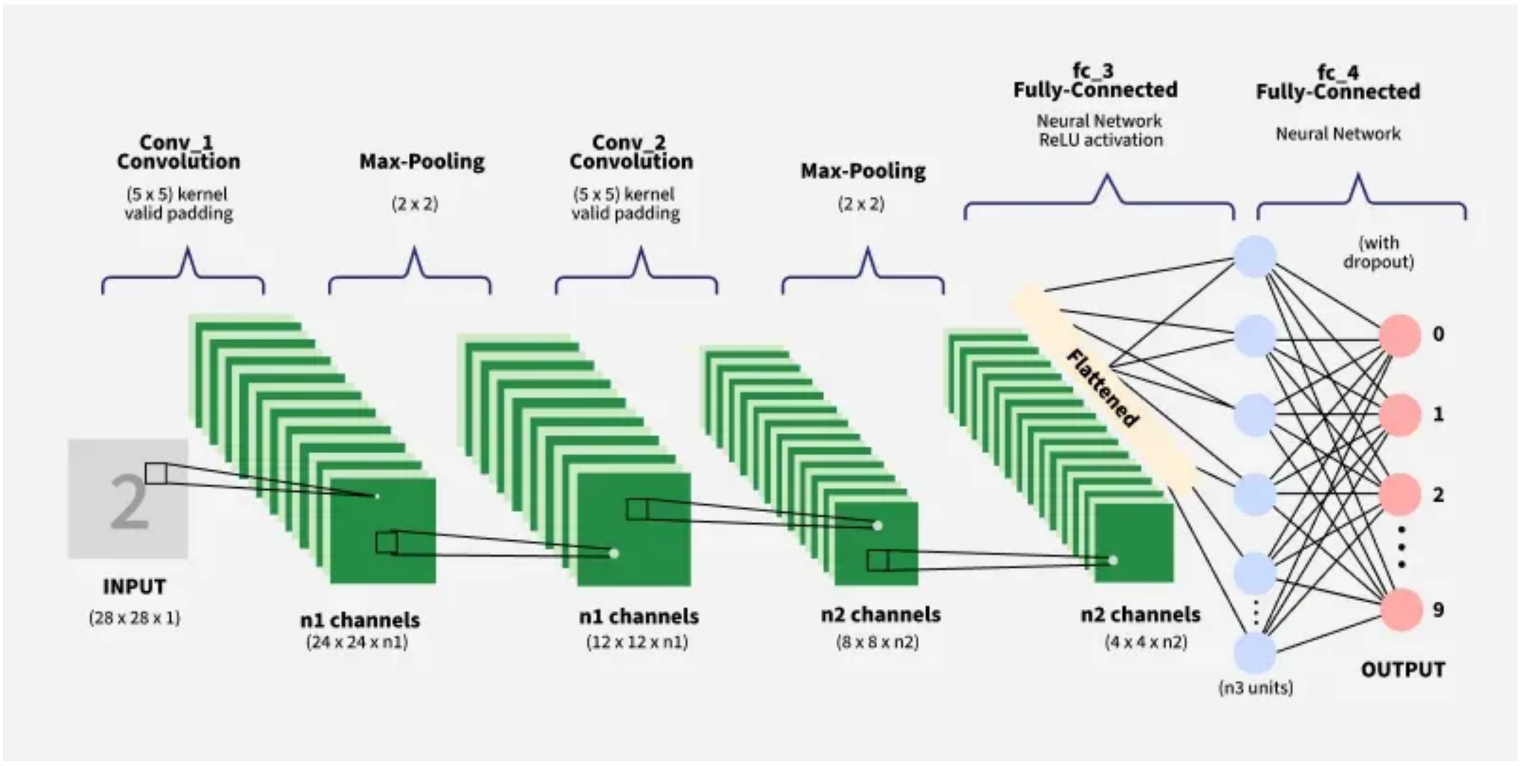
2. CNN Architecture

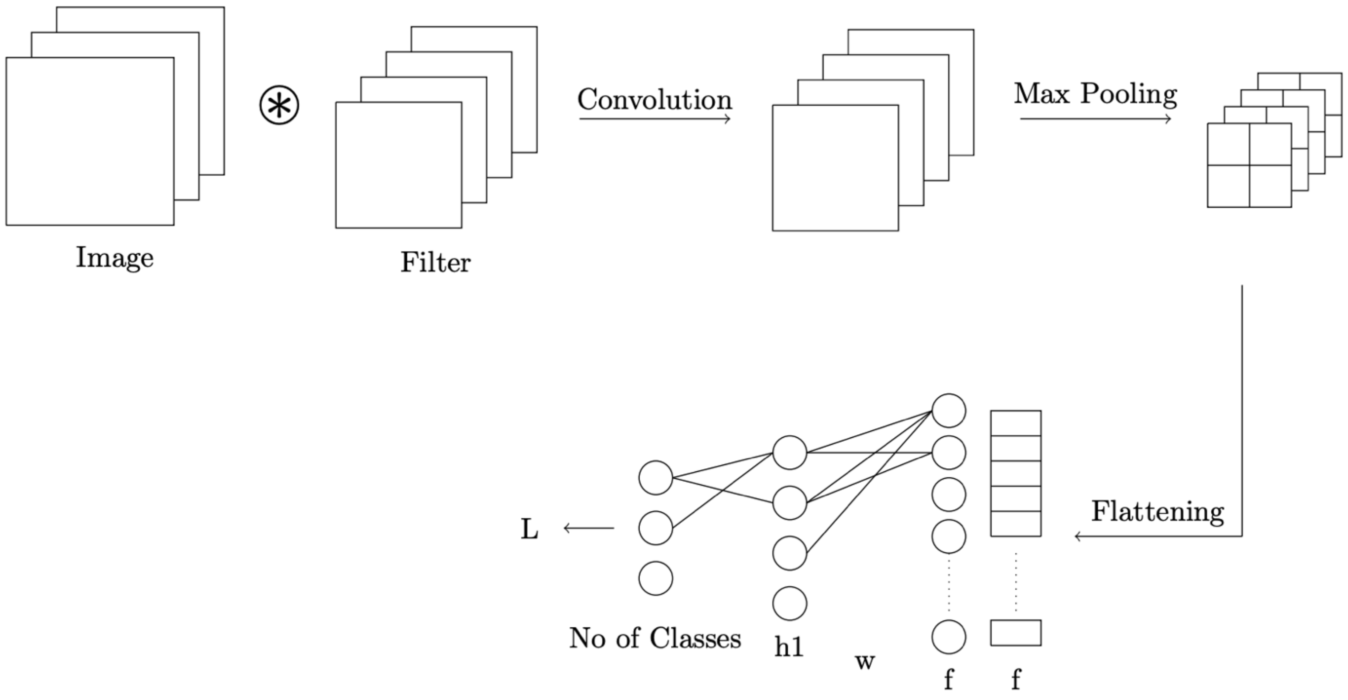
A typical CNN consists of multiple layers, including:

- **Input Layer:** Accepts image data (e.g., RGB images with size 1024x1024x3).
- **Convolutional Layers:** Perform feature extraction using convolution operations.
- **Activation Function (ReLU):** Introduces non-linearity.
- **Pooling Layers:** Reduces spatial dimensions.
- **Fully Connected Layers:** Perform classification based on extracted features.
- **Output Layer:** Provides class probabilities.

Key Components of a Convolutional Neural Network

- **Convolutional Layers**: These layers apply convolutional operations to input images, using filters (also known as kernels) to detect features such as edges, textures, and more complex patterns. Convolutional operations help preserve the spatial relationships between pixels.
- **Pooling Layers:** They downsample the spatial dimensions of the input, reducing the computational complexity and the number of parameters in the network. (Max pooling is a common pooling operation, selecting the maximum value from a group of neighboring pixels.)
- **Activation Functions:** They introduce non-linearity to the model, allowing it to learn more complex relationships in the data.
- **Fully Connected Layers**: These layers are responsible for making predictions based on the high-level features learned by the previous layers. They connect every neuron in one layer to every neuron in the next layer.





How CNNs Work?

- Input Image: The CNN receives an input image, which is typically preprocessed to ensure uniformity in size and format.
- Convolutional Layers: Filters are applied to the input image to extract features like edges, textures, and shapes.
- Pooling Layers: The feature maps generated by the convolutional layers are downsampled to reduce dimensionality.
- Fully Connected Layers: The downsampled feature maps are passed through fully connected layers to produce the final output, such as a classification label.
- Output: The CNN outputs a prediction, such as the class of the image.

Convolutional Neural Network Training

CNNs are trained using a supervised learning approach. This means that the CNN is given a set of labeled training images. The CNN then learns to map the input images to their correct labels.

The training process for a CNN involves the following steps:

- Data Preparation: The training images are preprocessed to ensure that they are all in the same format and size.
- Loss Function: A loss function is used to measure how well the CNN is performing on the training data. The loss function is typically calculated by taking the difference between the predicted labels and the actual labels of the training images.
- Optimizer: An optimizer is used to update the weights of the CNN in order to minimize the loss function.
- Backpropagation: Backpropagation is a technique used to calculate the gradients of the loss function with respect to the weights of the CNN. The gradients are then used to update the weights of the CNN using the optimizer.

CNN Evaluation

After training, CNN can be evaluated on a held-out test set. A collection of pictures that the CNN has not seen during training makes up the test set. How well the CNN performs on the test set is a good predictor of how well it will function on actual data.The efficiency of a CNN on picture categorization tasks can be evaluated evaluation metrices like:

- Accuracy: Accuracy is the percentage of test images that the CNN correctly classifies.
- Precision: Precision is the percentage of test images that the CNN predicts as a particular class and that are actually of that class.
- Recall: Recall is the percentage of test images that are of a particular class and that the CNN predicts as that class.
- F1 Score: The F1 Score is a harmonic mean of precision and recall. It is a good metric for evaluating the performance of a CNN on classes that are imbalanced.

Advantages of CNN

- High Accuracy: CNNs achieve state-of-the-art accuracy in various image recognition tasks.
- Efficiency: CNNs are efficient, especially when implemented on GPUs.
- Robustness: CNNs are robust to noise and variations in input data.
- Adaptability: CNNs can be adapted to different tasks by modifying their architecture.

Disadvantages of CNN

- Complexity: CNNs can be complex and difficult to train, especially for large datasets.
- Resource-Intensive: CNNs require significant computational resources for training and deployment.
- Data Requirements: CNNs need large amounts of labeled data for training.
- Interpretability: CNNs can be difficult to interpret, making it challenging to understand their predictions.

3. Convolution Operation

Convolution Operation

The convolution operation is fundamental to CNNs and involves the following steps:

- **Filter Application**: A filter (or kernel), which is a small matrix of weights, slides over the input data.
- **Element-wise Multiplication**: At each position, the filter performs an element-wise multiplication with the corresponding section of the input.
- **Summation**: The results of these multiplications are summed to produce a single value in the output feature map.

Convolution is the process of applying a filter (kernel) to an image to extract features.

Mathematical Definition

The convolution operation between two functions f and g (denoted by f ∗ g) is defined as:

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

Convolution essentially combines the information of the two input functions by measuring the overlap between them at every point and integrating the product over the entire range of possible shifts. This operation is often visualized as sliding one function (g) over another (f), multiplying them at each point of overlap, and summing up the results.

Convolution is widely used in signal processing for bandpass filtering and noise reduction. This operation is particularly effective for filtering because it allows for the extraction of specific frequency components from a signal while suppressing others.

For instance, in bandpass filtering, convolution enables the isolation of frequency bands of interest by convolving the input signal with a suitable filter kernel. In case of image classification, we will use the matrix form of convolution wherein we apply filters to the input image by convolving them with appropriate kernel matrices. This involves sliding the filter matrix over the input image, computing element-wise multipications, summing these products, and aggregating the results to generate the output image.

(In image processing, convolution is performed using matrices where:

1. The **input image** is represented as a matrix.
2. The **filter (kernel)** is a small matrix applied over the image.
3. The **output** is obtained by multiplying corresponding values and summing them up.)

Why Use Convolution Instead of Fully Connected Layers?

1. **Reduces Computation:** Instead of connecting every pixel to every neuron (which leads to a huge number of parameters), convolution significantly reduces the number of trainable parameters.
2. **Preserves Spatial Features:** Local spatial patterns (e.g., edges, corners) are better captured by convolutional filters.
3. **Translation Invariance:** Features detected by filters remain consistent regardless of their position in the image.

In Convolutional Neural Networks (CNNs), the use of multiple filters is crucial for:

1. Feature Hierarchy: Different filters capture features at varying abstraction levels, forming a hierarchical representation.

2. Variability Handling: Filters with diverse orientations and scales handle variations in object orientation and size.

3. Spatial Hierarchies: Filters learn spatial hierarchies, capturing details and relationships across different scales.

4. Class-Specific Features: Filters specialize in extracting features relevant to specific object classes.

5. Increased Model Capacity: Multiple filters enhance the network's capacity, allowing it to model a broader range of features.

6. Regularization and Generalization: Diverse filters act as regularization, preventing overfitting and improving generalization.

4. Pooling

Pooling layers reduce the size of feature maps, improving computational efficiency and controlling overfitting.

Types of Pooling

1. **Max Pooling:** Retains only the maximum value in each region.
    - Example: A **2×2 filter** applied to a **4×4 matrix** results in a **2×2 output**, keeping the max value from each region.
2. **Average Pooling:** Computes the average of pixel values in the filter region.

Formula for Output Size in Pooling

For an image of size n×n, filter size f×f, and stride s:

Output Size=⌊((n−f)/s)+1⌋

Why Use Pooling?

- Reduces **dimensionality** while preserving essential information.
- Helps in making the model **robust to small transformations** in the image.
- Acts as a form of **regularization**, reducing overfitting.

Padding

Purpose:

- Ensures that important **edge features** are not lost during convolution.
- Helps maintain the spatial size of the image after applying filters.

Concept:

- When a convolution operation is performed, pixels in the **center** of the image influence multiple output pixels, while **corner and edge pixels** contribute to fewer.
- **Padding** adds extra layers of zeros around the image to **give all pixels equal importance**.

Formula:

1. **If ppp layers of zeros are added**, the new image size becomes: (n+2p)×(n+2p)
2. **If a filter of size f×f is applied after padding**, the output size is: ((n+2p)−f+1)×((n+2p)−f+1)
3. **For same padding** (to keep output size equal to input size): 2p=f−1,p=(f−1)2,where f is odd

Types of Padding:

- **Valid Padding:** No padding applied. The output is **smaller** than the input.
- **Same Padding:** Padding is added so the output has the **same size** as the input.

Striding

Purpose:

- Controls the step size when sliding the filter across the image.
- Reduces computation by skipping over some pixels.

Concept:

- Instead of moving the filter **one pixel at a time (stride = 1)**, it can move by **2, 3, or more pixels**.
- This **reduces the output size**, making the network faster.

Formula for Output Size with Stride s:
For an input grid of size **n1×n2,** a filter of size **f×f**, and stride **s**:

$\lfloor((n1-f)/s)+1\rfloor \times \lfloor((n2-f)/s)+1\rfloor$
Example: If **n1=5,n2=5,f=3,s=2**

  - Output size: $\lfloor(5-3)/2+1\rfloor=2$
  - The final output is **2×2**

Effects of Stride:

- **Higher stride (e.g., 2, 3, 4)** → Smaller output size, fewer computations.
- **Lower stride (e.g., 1)** → More detailed feature maps but higher computational cost.

5.Variants of the Basic Convolution Function

In neural networks, convolution is an operation that consists of applying multiple convolution operations in parallel. This allows CNNs to efficiently extract features from input data.

Convolution Operation in CNNs

- A convolution operation involves a **kernel (K)** and an **input (V)**.
- The kernel **Ki,j,k,l** represents the connection strength between an output unit in channel i and an input unit in channel j, with an offset of k rows and l columns.
- The input Vi,j,k consists of **channels, rows, and columns**.

The output Z has the same format as V, representing the result of applying the convolution operation.

Full Convolution

- Uses **zero padding** with a **stride of 1** to ensure the entire input is processed while maintaining spatial dimensions.
- When using a **stride greater than 1**, convolution with a stride of 1 is performed first, followed by **downsampling** to reduce the output size.

Effect of Zero Padding

- **Without zero padding**, the spatial dimensions of the feature map shrink as it moves through layers.
- **Zero padding** helps control the kernel width and output size independently.

Types of Zero Padding:

1. **Valid Padding**
   - No zero padding is used.
   - Limits the number of layers that can be applied.

2. **Same Padding**
   - Keeps the output size the same as the input.
   - Allows an **unlimited number of layers** to be stacked.
   - Border pixels contribute to fewer output values than center pixels.
3. **Full Padding**
   - Adds enough zeros so that every pixel in the input is processed multiple times.
   - The output size becomes larger than the input.
   - Difficult to learn a single kernel that works well across all positions.

Finding the Right Amount of Padding

- The best padding choice is usually between **Valid** and **Same** padding.

Unshared Convolution

- Instead of applying the same kernel across the entire image, different filters are used for different regions.
- **Useful when:**
  - Features should be detected in specific parts of an image (e.g., identifying a mouth in the lower half of a face).
  - Connectivity should be restricted to certain areas.

Advantages of Unshared Convolution:

- **Reduces memory consumption** by limiting redundant parameters.
- **Increases statistical efficiency**, making the model less prone to overfitting.
- **Improves computational efficiency**, reducing the number of calculations needed during training.

Tiled Convolution

- A set of kernels is used and rotated through different regions of the image.
- **Unlike standard convolution**, where the same kernel is applied everywhere, tiled convolution allows neighboring locations to have different filters.
- This approach reduces memory usage while still allowing more variation than fully unshared convolution.

Use Case:

- When **tiled convolution layers are combined with max pooling**, the network learns different transformations of the same features, making the model **invariant to transformations**.

Backpropagation in Convolutional Layers

- During training, the network updates kernel weights by computing gradients.
- **Main Components in Backpropagation:**
  - **Kernel Stack** (set of filters used for convolution).
  - **Input Image** (original data passed into the network).
  - **Output Feature Map** (result of convolution).
  - **Gradient Map** (used to update the kernel weights).

Bias in Convolutional Layers

- **Local Connected Layers**: Each unit has its own bias.
- **Tiled Convolution Layers**: Biases follow the same tiling pattern as the kernels.
- **Standard Convolution Layers**:
  - A single bias per output channel is shared across all locations.
  - If the input has a fixed size, a separate bias can be learned for each output location.

Summary

1. **Full Convolution** maintains spatial dimensions using zero padding.
2. **Zero Padding** can be:
   - **Valid** (no padding, limited depth).
   - **Same** (keeps input and output size equal, allowing deeper networks).
   - **Full** (ensures every pixel is processed multiple times, but increases complexity).
3. **Unshared Convolution** applies different filters to different regions, useful for **region-specific features**.
4. **Tiled Convolution** alternates between different kernels, making the model more robust to transformations.
5. **Backpropagation in CNNs** updates kernel weights using gradients, with biases adjusted depending on the layer type.

6. Structured Output

Convolutional Neural Networks (CNNs) can be used for more than just classification or regression; they can **generate high-dimensional structured outputs** instead of just predicting a single class label or numerical value.

Example Use Case:

- The CNN might output a tensor **S**, where each element represents the **probability that a specific pixel (j, k) in the input belongs to class i**.
- This is useful for **segmentation tasks**, where every pixel is assigned a class label instead of classifying an entire image.

Issue: Output Plane is Smaller than Input Plane

- When CNNs process images, operations like convolution and pooling often **reduce spatial resolution**, causing the output feature map to be smaller than the original input.
- This size reduction can be problematic for tasks that require per-pixel predictions.

Strategies to Address Size Reduction:

1. **Avoid Pooling Entirely**
   - Eliminates downsampling, preserving spatial resolution.
   - Increases computational cost but retains fine details.
2. **Emit a Lower-Resolution Grid of Labels**
   - Instead of predicting labels for every pixel, generate a **coarse prediction** and use interpolation to refine it.
3. **Use Pooling Operators with Unit Stride**
   - Ensures that pooling does not reduce the size of the feature map.

Refining Pixel-Wise Predictions

One approach to **image segmentation** is to:

1. **Generate an Initial Guess**
   - A rough prediction is made for each pixel's class.

2. **Refine the Prediction Using Neighboring Pixels**
    - The model updates its predictions by considering relationships between adjacent pixels.
3. **Repeat the Refinement Process Multiple Times**
    - This process corresponds to applying the **same convolutional layer iteratively** and sharing weights across the last layers of the deep network.

Recurrent Convolutional Network (RCN)

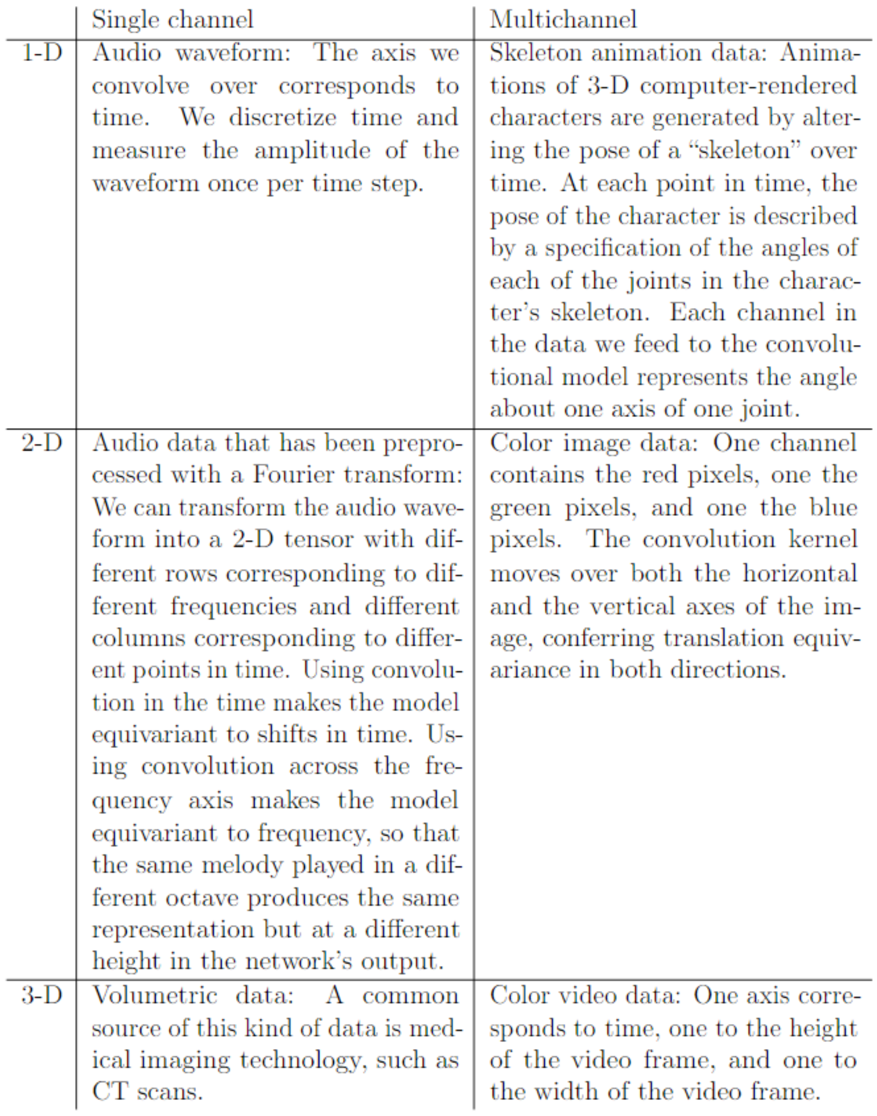
- A **Recurrent Convolutional Network** (RCN) applies the **same convolution multiple times** to refine pixel-wise predictions.
- By iterating over the image, the network improves its understanding of local structures and relationships.
- **Benefit**: Helps in tasks like **semantic segmentation**, where fine-grained details matter.

Summary

- **Structured output** allows CNNs to predict a high-dimensional tensor instead of a single label.
- **Example**: Assigning a class label to each pixel in an image.
- **Challenge**: The output resolution is often smaller than the input.
- **Solutions**: Avoid pooling, use a lower-resolution grid, or apply unit-stride pooling.
- **Refinement Process**: Initial predictions are improved by **iteratively updating** pixel labels, often using **Recurrent Convolutional Networks (RCNs)**.

---

7. Data Type



	Single channel	Multichannel
1-D	Audio waveform: The axis we convolve over corresponds to time. We discretize time and measure the amplitude of the waveform once per time step.	Skeleton animation data: Animations of 3-D computer-rendered characters are generated by altering the pose of a "skeleton" over time. At each point in time, the pose of the character is described by a specification of the angles of each of the joints in the character's skeleton. Each channel in the data we feed to the convolutional model represents the angle about one axis of one joint.
2-D	Audio data that has been preprocessed with a Fourier transform: We can transform the audio waveform into a 2-D tensor with different rows corresponding to different frequencies and different columns corresponding to different points in time. Using convolution in the time makes the model equivariant to shifts in time. Using convolution across the frequency axis makes the model equivariant to frequency, so that the same melody played in a different octave produces the same representation but at a different height in the network's output.	Color image data: One channel contains the red pixels, one the green pixels, and one the blue pixels. The convolution kernel moves over both the horizontal and the vertical axes of the image, conferring translation equivariance in both directions.
3-D	Volumetric data: A common source of this kind of data is medical imaging technology, such as CT scans.	Color video data: One axis corresponds to time, one to the height of the video frame, and one to the width of the video frame.

Table 9.1: Examples of different formats of data that can be used with convolutional networks.

8. Efficient Convolution Algorithms

Speeding up convolution is crucial for improving **real-time performance** in deep learning applications. Several techniques can be used to optimize convolution operations.

1. Parallel Computation Resources

- Modern hardware, such as **GPUs and TPUs**, are designed for parallel computation, making it possible to process multiple convolutions simultaneously.
- **Batch processing** helps improve efficiency by computing multiple inputs at once instead of one at a time.

2. Selecting Appropriate Algorithms

Choosing the right algorithm based on the **size of the input, kernel, and available computational resources** can significantly reduce computation time.

3. Fourier Transform-Based Convolution

- Converts the **input** and **kernel** into the **frequency domain** using the **Fourier Transform**.

- Performs **point-wise multiplication** instead of a computationally expensive spatial convolution.
- Uses the **Inverse Fourier Transform** to convert the result back to the original domain.
  Advantage: Reduces computational complexity for large kernels.

4. Separable Convolutions

- A **d-dimensional kernel** can sometimes be expressed as the **outer product of multiple 1D vectors**.
- **Separable Kernels** allow performing **multiple 1D convolutions** instead of a single large convolution, significantly improving efficiency.
- **Example:** Instead of a **3x3 convolution**, perform two **1D convolutions** (first in one direction, then in the other).
  Advantages:
    - Reduces computational cost.
    - Requires fewer parameters, leading to lower memory usage.

5. Efficiency Considerations for Deployment

- Even if some optimization techniques only speed up **forward propagation**, they are still valuable for **real-world applications**.
- In commercial settings, **more resources are allocated to deploying neural networks** than to training them, making efficient convolution techniques essential for real-time inference.

9. Applications of Convolutional Networks

CNNs are widely used across industries:

a. Image Classification

- **Face Recognition:** Used in **security systems (Face ID)**.
- **Medical Diagnosis: Detects tumors in MRIs and CT scans**.

b. Object Detection

- **Autonomous Vehicles: YOLO, SSD, Faster R-CNN** help identify pedestrians, traffic signs.

c. Image Segmentation

- **Medical Imaging:** Separates **organs, tumors, and tissues** from scans.
- **Agriculture:** Identifies **crop diseases** from satellite imagery.

d. Natural Language Processing (NLP)

- **Text Classification:** CNNs extract patterns from text for **sentiment analysis**.
- **Speech Recognition: 1D convolutions** in **waveform analysis**.

e. Generative Models

- **GANs (Generative Adversarial Networks)** use CNNs for image synthesis (e.g., **DeepFake, DALL-E**).

f. Robotics

- **Visual Navigation:** CNNs guide **robot arms** in industrial applications.

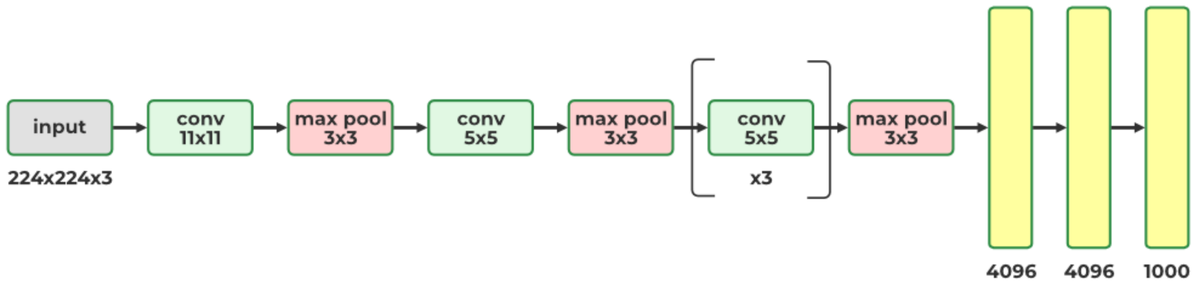
10. Pre-trained convolutional Architectures :AlexNet, ZFNet, ResNet

AlexNet, ZFNet, and ResNet are pre-trained convolutional neural network (CNN) architectures that have been instrumental in the advancement of computer vision, with AlexNet pioneering deep learning in image recognition, ZFNet building upon it with improved visualization, and ResNet introducing "skip connections" to enable training deeper networks.

Here's a more detailed look at each:

1. AlexNet:

- **Pioneering Deep Learning:** AlexNet was a groundbreaking architecture that demonstrated the power of deep learning for image classification, winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012.
- **Key Features:**
  - **8 Layers:** Consists of five convolutional layers and three fully connected layers.
  - **ReLU Activation:** Used ReLU (Rectified Linear Unit) activation functions, which helped with faster training and better performance compared to traditional sigmoid functions.
  - **GPU Training:** Trained on two GPUs to handle the large number of parameters and computational demands.
  - **Dropout:** Employed dropout regularization to prevent overfitting.
  - **Batch Normalization:** Used batch normalization to stabilize training and improve generalization.
- **Impact:** AlexNet's success sparked widespread interest and research in deep learning for computer vision.



The AlexNet CNN architecture won the 2012 ImageNet ILSVRC challenges of deep learning algorithm by a large variance by achieving 17% with top-5 error rate as the second best achieved 26%!

It was introduced by Alex Krizhevsky (name of founder), The Ilya Sutskever and Geoffrey Hinton are quite similar to LeNet-5, only much bigger and deeper and it was introduced first to stack convolutional layers directly on top of each other models, instead of stacking a pooling layer top of each on CN network convolutional layer.

AlexNNet has 60 million parameters as AlexNet has total 8 layers, 5 convolutional and 3 fully connected layers.

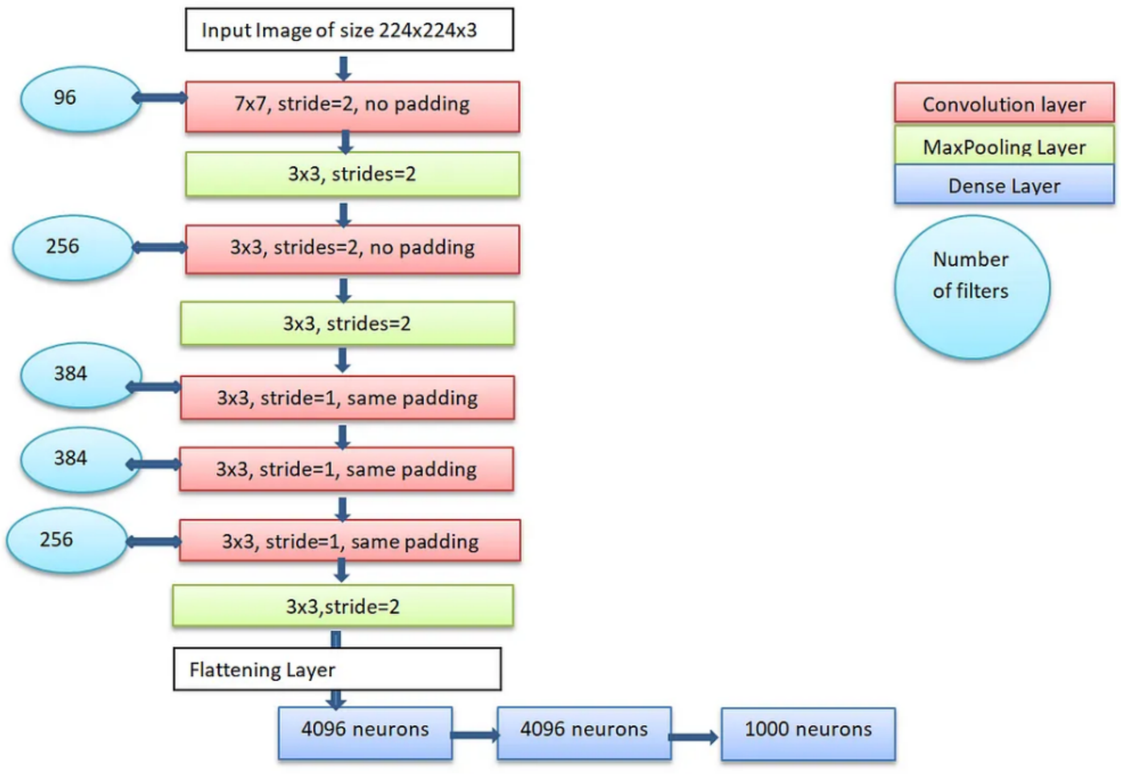
AlexNNet is first to execute (ReLUs) Rectified Linear Units as activation functions

it was the first CNN architecture that uses GPU to improve the performance.

2. ZFNet (Zeiler and Fergus Network):

- **Building on AlexNet:** ZFNet, developed by Matthew Zeiler and Rob Fergus, built upon AlexNet's architecture with modifications to improve visualization and performance.
- **Key Features:**
  - **Visualization:** Introduced a novel visualization technique using deconvolutional networks to understand the internal workings of CNNs.
  - **Hyperparameter Tuning:** Made adjustments to hyperparameters, such as filter sizes and strides, to optimize performance.
  - **7x7 Filters:** Used 7x7 filters in the initial layers instead of 11x11 filters to prevent loss of features.

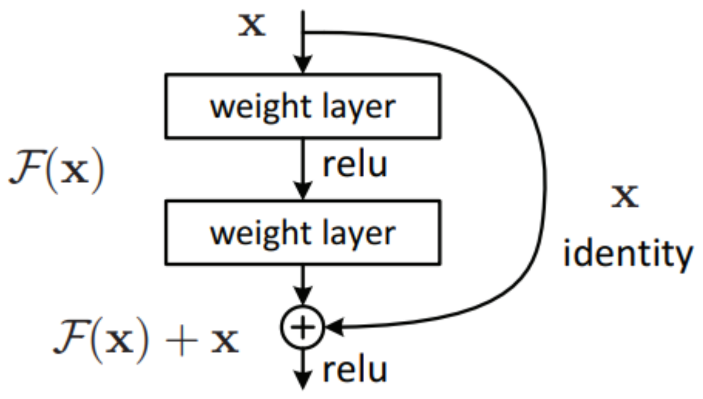
- **Impact:** ZFNet's visualization techniques helped researchers gain a deeper understanding of how CNNs learn and process visual information.



ZFNet Architecture (image by me)

3. ResNet (Residual Network):

- **Overcoming Vanishing Gradients:** ResNet, proposed by researchers at Microsoft Research, introduced the concept of "residual blocks" to address the problem of vanishing gradients in deep networks.
- **Key Features:**
  - **Residual Connections (Skip Connections):** Used skip connections, or residual connections, to allow information to bypass layers and directly connect to later layers.
  - **Deep Networks:** Enabled the training of extremely deep networks, which previously struggled due to vanishing gradients.
  - **Residual Blocks:** Consisted of residual blocks, which are groups of layers that include skip connections.
- **Impact:** ResNet's architecture revolutionized deep learning, enabling the development of much deeper and more accurate models.


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Residual Network (ResNet), the winner of the ILSVRC 2015 challenge, was developed by Kaiming He and delivered an impressive top-5 error rate of 3.6% with an extremely deep CNN composed of 152 layers. An essential factor enabling the training of such a deep network is the use of skip connections (also known as shortcut connections). The signal that enters a layer is added to the output of a layer located higher up in the stack. Let's explore why this is beneficial.

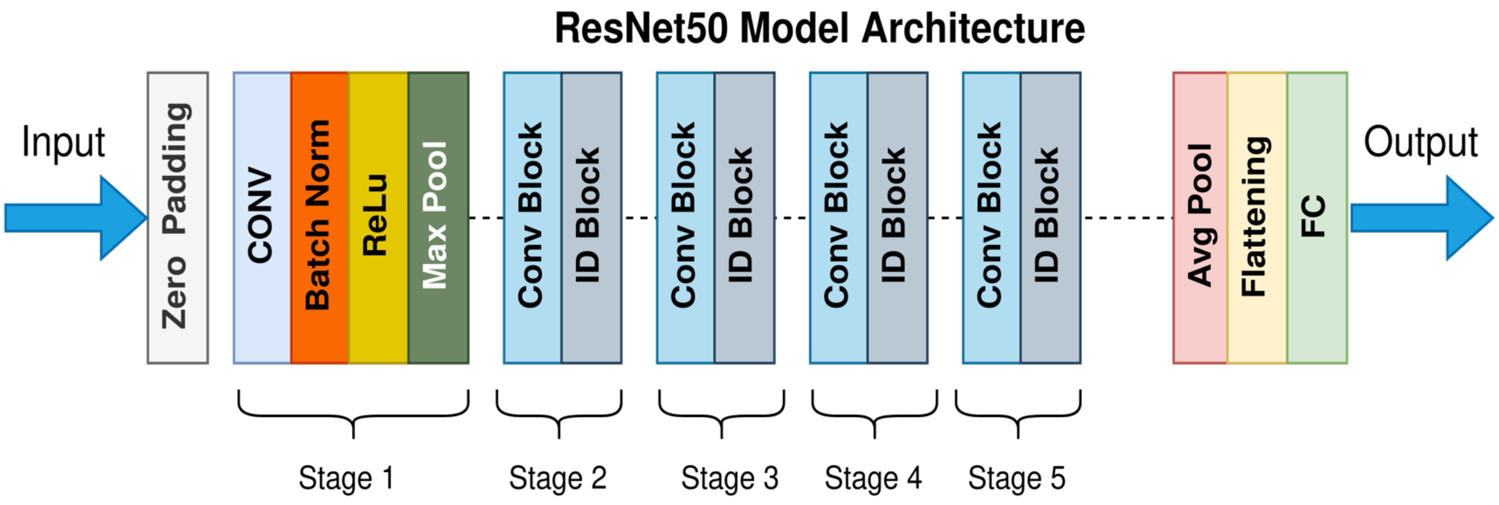
When training a neural network, the goal is to make it replicate a target function h(x). By adding the input x to the output of the network (a skip connection), the network is made to model f(x) = h(x) – x, a technique known as residual learning.

F(x) = H(x) - x which gives H(x) := F(x) + x.

When initializing a regular neural network, its weights are near zero, resulting in the network outputting values close to zero. With the addition of skip connections, the resulting network outputs a copy of its inputs, effectively modeling the identity function. This can be beneficial if the target function is similar to the identity function, as it will accelerate training. Furthermore, if multiple skip connections are added, the network can begin to make progress even if several layers have not yet begun learning.

the target function is fairly close to the identity function (which is often the case), this will speed up training considerably. Moreover, if you add many skin connections, the network can start making progress even if several

The deep residual network can be viewed as a series of residual units, each of which is a small neural network with a skip connection



1. AlexNet (2012)

AlexNet

Objective

- A leading **pre-trained model** in Computer Vision.
- Introduced in **2012** by Alex Krizhevsky and colleagues.
- Won the **ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)** with breakthrough performance.
- Inspired by **LeNet-5**, but with deeper architecture.

AlexNet Architecture

- **8 layers** with learnable parameters.
- **Input:** RGB images of **227×227×3**.
- **Uses ReLU activation** in all layers except the output layer.

- **Trained on the ImageNet dataset** (14 million images, 1000 classes).
- **Total parameters: 62.3 million**.

Convolution & Max-Pooling Layers

1. **Conv1:** 96 filters, **11×11**, stride 4 → Output: **55×55×96**
    - **ReLU activation**
2. **Max-Pool1: 3×3**, stride 2 → Output: **27×27×96**
3. **Conv2:** 256 filters, **5×5**, stride 1, padding 2 → Output: **27×27×256**
    - **ReLU activation**
4. **Max-Pool2: 3×3**, stride 2 → Output: **13×13×256**
5. **Conv3:** 384 filters, **3×3**, stride 1, padding 1 → Output: **13×13×384**
    - **ReLU activation**
6. **Conv4:** 384 filters, **3×3**, stride 1, padding 1 → Output: **13×13×384**
    - **ReLU activation**
7. **Conv5:** 256 filters, **3×3**, stride 1, padding 1 → Output: **13×13×256**
    - **ReLU activation**
8. **Max-Pool3: 3×3**, stride 2 → Output: **6×6×256**

Fully Connected & Dropout Layers

1. **Dropout Layer** (0.5 dropout rate).
2. **FC1:** 4096 neurons, **ReLU activation**.
3. **Dropout Layer** (0.5 dropout rate).
4. **FC2:** 4096 neurons, **ReLU activation**.
5. **FC3 (Output Layer):** 1000 neurons, **Softmax activation**.

Key Features

- **Increased network depth** compared to earlier CNNs.
- **Uses ReLU activation**, which speeds up training (6× faster).
- **Dropout layers** to prevent overfitting.
- **Uses padding** to maintain feature map size.

Applications

- Image Recognition (**ImageNet, CIFAR-10, CIFAR-100**).
- Medical Image Analysis.
- Autonomous Driving.
- Object Detection (**YOLO, Faster R-CNN**).

FAQs

Q1: What is the use of AlexNet?

- A pioneering **CNN model** used for **image recognition and classification**.
- Marked a **breakthrough in deep learning** and influenced modern CNN architectures.

Q2: Why is AlexNet better than traditional CNNs?

- **Deeper network with more layers**.
- **Uses ReLU activation** instead of sigmoid/tanh for faster training.
- **Dropout layers** prevent overfitting.
- **Max-pooling & padding** help retain important features.

Summary

- **8 layers** (5 convolutional + max pooling, 3 fully connected).
- **ReLU activation** speeds up training.
- **Dropout layers** prevent overfitting.
- **Softmax activation** in the output layer.
- **Total parameters: 62.3 million**.
- A **milestone in deep learning** for computer vision.

AlexNet was introduced by **Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton** in 2012. It won the **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)** by significantly outperforming previous models.

Key Features of AlexNet

- **Deeper than previous models** like LeNet-5, with 8 layers.
- **Uses ReLU (Rectified Linear Unit) activation** to speed up training.
- **Dropout layers** are included to prevent overfitting.
- **Trained on the ImageNet dataset** with 14 million images across 1000 classes.

AlexNet Architecture

AlexNet consists of **five convolutional layers**, followed by **three fully connected layers**:

1. **Input Layer**:
   - Image size: **227 × 227 × 3 (RGB images)**.
2. **First Convolution Layer**:
   - **96 filters of size 11×11**, stride **4**, with **ReLU activation**.
   - Output feature map: **55 × 55 × 96**.
   - Followed by **Max Pooling (3×3, stride 2)** → Output **27 × 27 × 96**.
3. **Second Convolution Layer**:
   - **256 filters of size 5×5**, stride **1**, with padding **2** and **ReLU activation**.
   - Output feature map: **27 × 27 × 256**.
   - Followed by **Max Pooling (3×3, stride 2)** → Output **13 × 13 × 256**.
4. **Third Convolution Layer**:
   - **384 filters of size 3×3**, stride **1**, padding **1**, and **ReLU activation**.
   - Output feature map: **13 × 13 × 384**.
5. **Fourth Convolution Layer**:
   - **384 filters of size 3×3**, stride **1**, padding **1**, and **ReLU activation**.
   - Output feature map: **13 × 13 × 384**.
6. **Fifth Convolution Layer**:
   - **256 filters of size 3×3**, stride **1**, padding **1**, and **ReLU activation**.
   - Output feature map: **13 × 13 × 256**.
   - Followed by **Max Pooling (3×3, stride 2)** → Output **6 × 6 × 256**.
7. **Fully Connected Layers**:
   - **4096 neurons**, followed by **ReLU activation** and **Dropout (0.5 probability)**.
   - Another **4096 neurons**, with **ReLU activation** and **Dropout**.
   - **Final layer with 1000 neurons (for ImageNet classes) and Softmax activation**.

Why Was AlexNet a Breakthrough?

- It was significantly **deeper** than previous models, leading to better feature extraction.
- **ReLU activation** accelerated training by six times compared to traditional sigmoid/tanh functions.
- **Dropout regularization** reduced overfitting.
- **Trained on two GPUs**, which allowed parallel computation.

2. ZFNet (2013)

ZFNet

Introduction

- **ZFNet (Zeiler & Fergus Network)** was introduced by **Matthew D. Zeiler & Rob Fergus**.
- Won **ILSVRC 2013**, improving on **AlexNet** by **visualizing** and tuning parameters for better performance.
- **Main contribution: Deconvolution layers** to understand how CNNs extract features.

Key Features

1. **Convolutional Layers:**
   - Extract important features from images.
2. **MaxPooling Layers:**
   - Downsample feature maps using a **3×3 kernel, stride 2**.
3. **ReLU Activation:**
   - Used after each convolution layer for **non-linearity** and better learning.
4. **Fully Connected Layers:**
   - Extract final patterns from feature maps.
5. **SoftMax Activation:**
   - Used in the last layer for **classification (1000 classes in ImageNet)**.
6. **Deconvolution Layers:**
   - **Key innovation** in ZFNet. Helps visualize how CNNs **learn features** by projecting activations back to the input space.

ZFNet Architecture

- **Input:** Image of size **224×224×3**.

Layer	Filters	Kernel Size	Stride	Other Details
Conv1	96	7×7	2	ReLU, MaxPooling (3×3, stride 2), Contrast Normalization
Conv2	256	3×3	2	ReLU, MaxPooling (3×3, stride 2), Contrast Normalization
Conv3	384	3×3	1	ReLU
Conv4	384	3×3	1	ReLU
Conv5	256	3×3	1	ReLU, MaxPooling (3×3, stride 2), Contrast Normalization
FC1	4096	-	-	Fully Connected, ReLU
FC2	4096	-	-	Fully Connected, ReLU
Output	1000	-	-	Fully Connected, SoftMax

Differences Between AlexNet & ZFNet

Aspect	AlexNet	ZFNet
Architecture	8 layers (5 Conv + 3 FC)	Retained AlexNet structure with improvements
Filter Sizes	**11×11, 5×5, 3×3**	**7×7 (Conv1), 3×3 (others)**
Stride	**Stride 4 in Conv1**	**Stride 2 in Conv1**
Normalization	**Local Response Normalization (LRN)**	**Local Contrast Normalization (LCN)**
Visualization	No visualization	**Deconvolution layers for feature visualization**

Summary

- **ZFNet improved AlexNet** by refining filter sizes, strides, and normalization techniques.

- Introduced **Deconvolution layers** for feature visualization.
- Used **7×7 filters** instead of **11×11** to capture finer details.
- **Outperformed AlexNet** in the **ILSVRC 2013** competition.

ZFNet (Zeiler & Fergus Net) was introduced in **2013 by Matthew D. Zeiler and Rob Fergus**. It **improved upon AlexNet** by visualizing how convolutional layers work and tuning hyperparameters for better performance.

Key Features of ZFNet

- **Built on AlexNet's structure but with modifications.**
- **Smaller filters (first convolution layer uses 7×7 instead of 11×11)** to capture finer details.
- **Lower stride (2 instead of 4 in AlexNet)** for better feature preservation.
- **Introduced Deconvolutional Layers** for **visualizing feature maps** and understanding how CNNs extract features.

ZFNet Architecture

1. **Input Layer**:
   - Image size: **224 × 224 × 3 (RGB images)**.
2. **First Convolution Layer**:
   - **96 filters of size 7×7, stride 2**, followed by **ReLU activation**.
   - Max Pooling **(3×3, stride 2)** and contrast normalization.
3. **Second Convolution Layer**:
   - **256 filters of size 3×3, stride 2**, followed by **ReLU activation**.
   - Max Pooling **(3×3, stride 2)** and contrast normalization.
4. **Third and Fourth Convolution Layers**:
   - **384 filters of size 3×3**, stride **1**, padding **same**.
5. **Fifth Convolution Layer**:
   - **256 filters of size 3×3**, stride **1**.
   - Max Pooling **(3×3, stride 2)** and contrast normalization.
6. **Fully Connected Layers**:
   - **Two layers with 4096 neurons**, followed by **ReLU activation** and **dropout**.
   - **Final softmax layer for 1000-class classification**.

Differences Between AlexNet and ZFNet

Feature	AlexNet	ZFNet
First Conv Layer	**11×11, stride 4**	**7×7, stride 2**
Normalization	Local Response Normalization	Local Contrast Normalization
Visualization	No feature visualization	Introduced **deconvolution** to visualize features

Why ZFNet was Important?

- Showed **how CNNs process images** by **visualizing learned features**.
- Used **smaller filters** to extract better details.
- Improved accuracy over AlexNet by fine-tuning hyperparameters.

---

4. ResNet (2015)

ResNet (Residual Network)

Problem in Deep Networks

- Vanishing/exploding gradients in deep networks.
- Training becomes difficult as layers increase.
- Accuracy degrades due to ineffective weight updates.

Introduction

- ResNet was introduced by Microsoft in 2015.
- Won the ImageNet competition with a **3.6% error rate**.
- Addresses training challenges in deep networks using **residual learning**.

Solution - Residual Learning

- Introduces **shortcut (skip) connections**.
- Allows gradients to flow directly to earlier layers.
- Learns residuals instead of direct mappings: F(x)=H(x)−x⇒H(x)=F(x)+xF(x) = H(x) - x \Rightarrow H(x) = F(x) + xF(x)=H(x)−x⇒H(x)=F(x)+x

ResNet Architecture

- Composed of **residual blocks** with skip connections.
- Residual blocks:
  - Enhance backpropagation efficiency.
  - Prevent vanishing gradients.
- **Variants:** ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152.
  - The number represents the total layers in the network.

Advantages of ResNet

- Enables training of deep networks **(100+ layers)**.
- Improves accuracy without increasing complexity.
- Widely used in **image classification, object detection, segmentation**.

Applications

- **Image recognition** (ImageNet, CIFAR-10, CIFAR-100).
- **Medical image analysis**.
- **Autonomous driving**.
- **Object detection** (YOLO, Faster R-CNN).

Conclusion

- ResNet **solved the vanishing gradient problem** in deep networks.
- It remains a **widely used** architecture in computer vision.
- Future advancements continue to improve residual learning.

ResNet (**Residual Network**) was introduced by **Microsoft in 2015** and won the **ImageNet competition** with a **record-breaking error rate of 3.6%**. It **solved the vanishing gradient problem**, allowing CNNs to go **deeper than ever before** (over 100 layers).

Problem in Deep Networks

- Traditional deep networks suffer from **vanishing gradients**, making it hard to train very deep models.
- Accuracy **decreases** as depth increases due to ineffective weight updates.

Solution: Residual Learning (Skip Connections)

- **ResNet introduced shortcut (skip) connections**, allowing gradients to flow directly to earlier layers.
- Instead of learning the full transformation, **ResNet learns the residual (difference) between input and output**.

ResNet Architecture

1. **Residual Blocks**:
   - Each block contains identity (skip) connections that allow efficient backpropagation.
   - Prevents vanishing gradients.
2. **Variants of ResNet**:
   - **ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152**.
   - The number represents the **total layers** in the model.

Why ResNet Was a Breakthrough?

- Enabled training of **very deep networks (over 100 layers)**.
- Improved accuracy **without making the model more complex**.
- Became the foundation for modern architectures in **image classification, object detection, and segmentation**.

Applications of ResNet

- **Image recognition** (e.g., ImageNet, CIFAR-10, CIFAR-100).
- **Medical image analysis**.
- **Autonomous driving**.
- **Object detection** (YOLO, Faster R-CNN).

Final Comparison

Model	Year	Key Innovation	Depth
AlexNet	2012	Deep CNN with ReLU and Dropout	8 layers
ZFNet	2013	Smaller filters, visualization with deconvolution	8 layers
ResNet	2015	Residual connections to train very deep networks	18–152 layers