# CNN

Convolutional Neural Network (CNN), also known as ConvNet, is a type of deep learning model specifically designed for processing structured grid-like data, such as images or time series. CNNs are widely used in various computer vision tasks, including image classification, object detection, segmentation, and more.

## History of CNN

1. **Development at AT&T Bell Labs (Late 1980s - Early 1990s)**:
   * Convolutional Neural Networks were initially developed by Yann LeCun, along with collaborators at AT&T Bell Labs, in the late 1980s and early 1990s.
   * LeCun's seminal work on CNNs, particularly in the context of handwritten digit recognition, laid the foundation for the practical application of CNNs in computer vision tasks.
2. **Handwriting Recognition (1990s)**:
   * CNNs gained prominence in the 1990s for their successful application in handwriting recognition systems.
   * One of the notable projects was the development of the LeNet-5 architecture by Yann LeCun and his team, which achieved state-of-the-art performance on the MNIST dataset, a benchmark dataset for handwritten digit recognition.
3. **Commercial Applications (Late 1990s - Early 2000s)**:
   * Microsoft's development of handwriting recognition systems, particularly for tasks such as digit recognition in postal services and handwritten text input in tablet PCs, further popularized the use of CNNs.
   * The ability of CNNs to automatically learn hierarchical features from raw input data made them well-suited for these real-world applications.
4. **Advancements and Current Applications (2000s - Present)**:
   * CNNs have continued to evolve and advance over the years, with improvements in architecture, training techniques, and computational resources.
   * In recent years, CNNs have become the backbone of many computer vision applications, including facial recognition, object detection, image segmentation, and autonomous driving.
   * Companies like Google, Facebook, and NVIDIA are actively developing CNN-based systems for a wide range of applications, leveraging the power of deep learning and CNNs' ability to extract meaningful features from complex data.
5. CNN Intution

## CNN intuition

1. **Inspired by Human Visual System**:
   * CNNs are inspired by the structure and function of the human visual system. Just like how our brains process visual information by recognizing the shapes, edges, and textures, CNNs are designed to mimic this process by learning hierarchical representations of feature from the raw input data.
2. **Layered Structure**:

CNN consist of multiple layers, each serving a specific purpose:

* Convolutional Layers: These layers are like detectives looking for patterns in a picture. They slide small windows (filters) across the image, checking for specific features like edges or shapes.
* Activation Layers: Once the patterns are found, activation layers decide whether they're important or not. They're like traffic lights: green means keep the feature; red means ignore it.
* Pooling Layers: These layers shrink down the picture by merging nearby information. It's like squishing several Lego blocks into one, which makes it easier to handle.
  + Fully Connected Layers: Finally, the fully connected layers make sense of all the features found. They connect all the dots and decide what the picture is about.

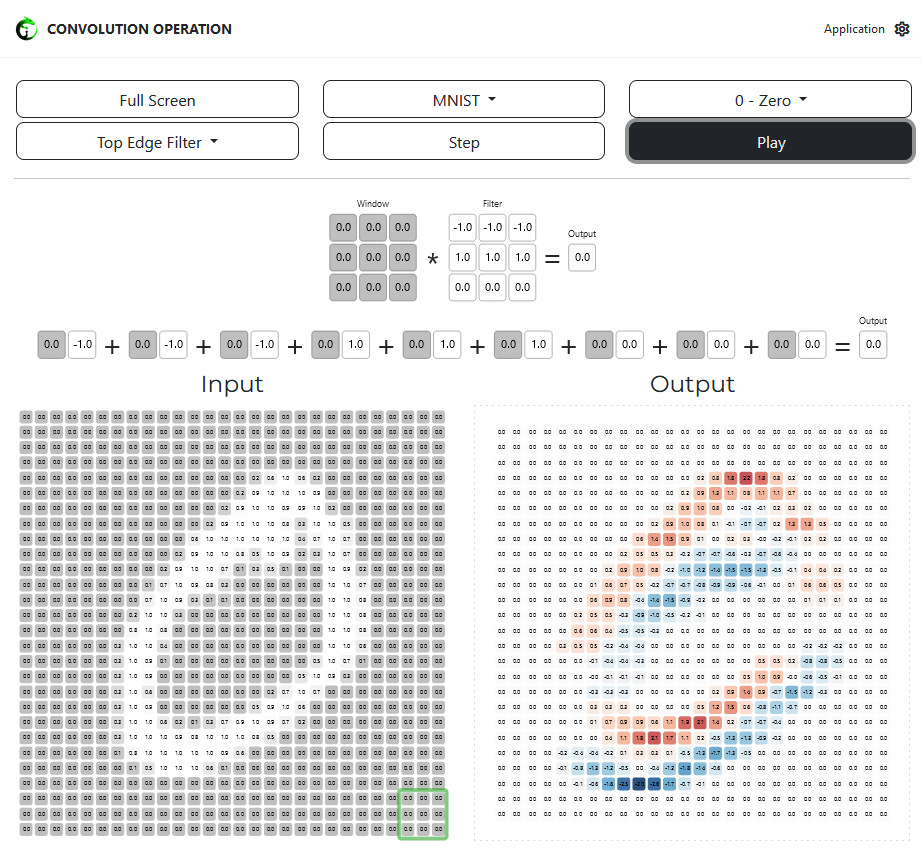
1. Feature Extraction:
   * CNNs automatically learn relevant features from the input data during the training process. Convolutional layers act as feature extractors, capturing low-level features like edges and textures in early layers and high-level features like object shapes and structures in deeper layers.
2. **Parameter Sharing and Translation Invariance**:
   * CNNs leverage parameter sharing and local connectivity to achieve translation invariance, meaning they can recognize patterns regardless of their position or orientation within the input data. This is achieved by using shared weights across different regions of the input data, enabling the network to generalize better to new, unseen examples.
3. **Training and Optimization**:
   * CNNs are trained using backpropagation and gradient descent, where the network learns to minimize a loss function by adjusting its weights and biases. Training involves presenting the network with labelled examples, and the network learns to make predictions by adjusting its parameters based on the error between its predictions and the true labels.
4. **Applications**:
   * CNNs are widely used in computer vision tasks such as image classification, object detection, and image segmentation, as well as other domains like natural language processing and speech recognition. Their ability to automatically learn hierarchical representations of features from raw data makes them well-suited for a wide range of tasks.

## CNN V/s Visual Cortex | The Famous Cat experiment

1. **Biological Connection**:
   1. CNNs are indeed inspired by the structure and function of the visual system in animals, particularly mammals like cats and primates.
   2. The famous CAT experiment involved studying the visual cortex of cats to understand how they perceive and process visual information.
2. **Processing in the Visual System**:
   1. When we see something, the information is captured by our eyes and transmitted as 2D images to the brain via the optic nerve.
   2. The information undergoes preprocessing in the lateral geniculate nucleus (LGN) before being sent to the visual cortex, the part of the brain responsible for processing visual stimuli.
3. **Feature Detection in the Visual Cortex**:
   1. In the visual cortex, there are specialized cells that respond to different features of visual stimuli, such as edges, shapes, and textures.
   2. Simple cells, also known as orientation cells or feature detectors, have small receptive fields and respond to basic features like edges of specific orientations (horizontal, vertical).
   3. Complex cells, on the other hand, have larger receptive fields and integrate the outputs of simple cells to detect more complex patterns and structures.
4. **Hierarchical Feature Representation**:
   1. The arrangement of simple and complex cells in the visual cortex forms a hierarchical structure, where simple cells detect basic features and complex cells combine these features to detect more complex patterns.
   2. This hierarchical feature representation is similar to the layers in CNNs, where lower layers detect simple features like edges, and higher layers combine these features to recognize more complex objects.
5. **CNNs as Artificial Neural Networks**:
   1. CNNs mimic this hierarchical feature extraction process in the visual cortex using layers of artificial neurons.
   2. Convolutional layers in CNNs act as feature detectors, analogous to simple cells, while subsequent layers combine these features to detect more complex patterns, similar to complex cells.

## Convolutional Operations:

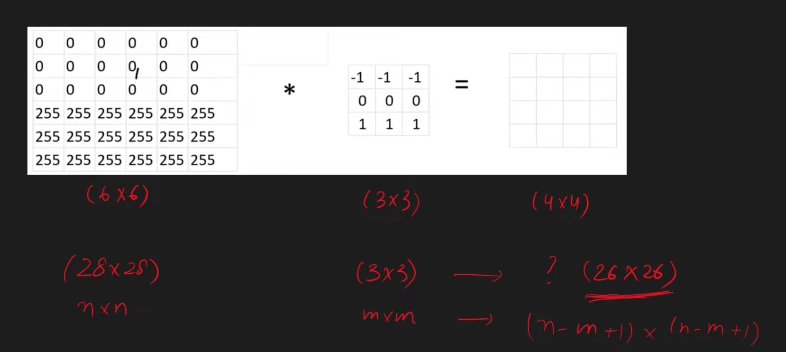
* **Understanding Images**:
  + Images can be grayscale (black and white) or RGB (color).
  + They are represented as grid of pixels, where each pixel holds a value (e.g., intensity or color) represented in 2D array.
  + For coloured images (RGB), there are three channels representing the red, green and blue components of the color.
* **Convolutional operations:**
  + Convolutional operations involve using small filters (called kernels) to detect patterns in the input image.
  + These filters slide over the image, performing mathematical operations at each position to detect features like edges or textures.
  + The result is a feature map that highlights areas where specific features are detected.
* **Filter/Kernel:**
  + A filter or kernel is a small matrix used for convolutional operations.
  + It contains weights that are multiplied with pixel values of the input image to produce the feature map.
  + Initially, these weights are randomly initialized, but during training (backpropagation), they are adjusted to minimize the loss function and improve performance.

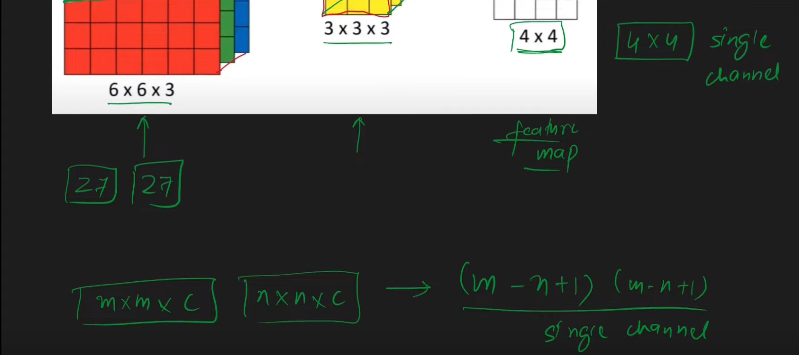


Size of a feature map after sliding the filter/kernel over an input image:

For Grayscale:

For RGB:





## Padding & Strides

### Padding

**Definition:** Padding involves adding extra border pixels around the input image before applying the convolution operation.

**Purpose:** Padding is typically used to preserve the spatial dimensions of the input image and the feature maps produced by the convolutional layers.

**Types of Padding:**

**Valid (No Padding):** No padding is added, and the filter is only applied to positions where it fully overlaps with input image. This results in feature maps smaller than the input image.

**Same (Zero Padding):** Padding is added symmetrically around the input image so that the size of the feature map remains the same as the input image.

### Strides

**Definition:** Stride refers to the number of pixels by which the convolution filter moves across the input image.

**Purpose:** Strides control the amount of overlap between adjacent regions covered by the filter and determine the spatial dimensions of the resulting feature map.

**Types of Strides**:

**Stride 1**: The filter moves one pixel at a time, resulting in adjacent regions overlapping by one pixel

**Stride >1:** The filter moves more than one pixel at a time, resulting in larger gaps between adjacent regions.

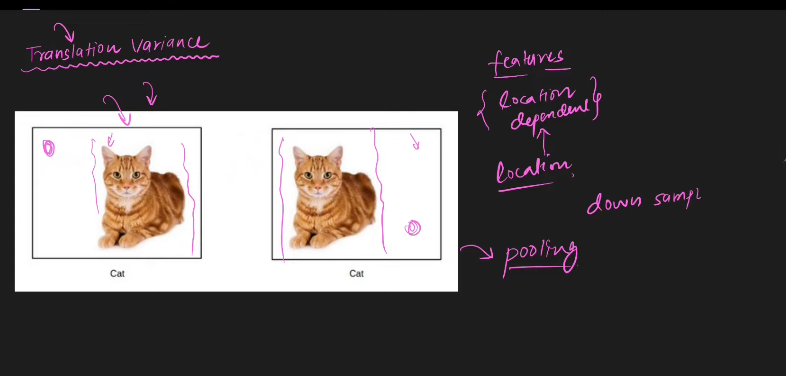
**When and Why Padding and Strides are applied:**

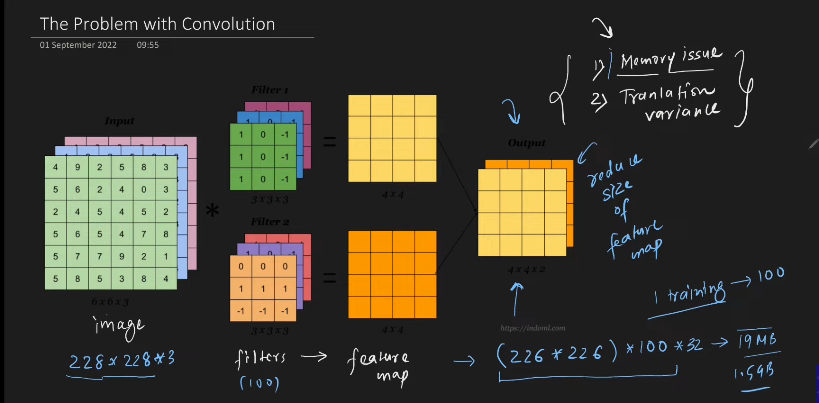
* **Preserving Spatial Dimensions:** Padding is applied when we want the spatial dimension of the input image and feature maps to remain same. It helps prevent information loss at the borders of the image
* **Control Over Output Size:** Strides are used to control the size of the feature maps. A larger stride results in smaller feature maps, while a smaller stride produces larger feature maps.

## Pooling

Pooling is a technique used in CNN to reduce the spatial dimensions (width and height) of feature maps while retaining important information. Pooling layers are typically inserted between convolution layers in CNN architectures.

1. **Pooling Operation:**
   * **Definition:** Pooling involves dividing the input feature map into non-overlapping regions and computing a summary statistic (e.g. Maximum, Average) for each region
   * **Types of Pooling:**
     1. **Max Pooling:** Computes the maximum value within each region.
     2. **Average Pooling:** Computes the average value withing each region**.**
   * **Stride:** Pooling operations also have a stride parameter, which determines the step size for moving the pooling window across the input feature map.
2. **Purpose of Pooling:**
   * **Dimensionality Reduction:** Pooling reduces the spatial dimension of the feature maps, leading to a smaller volume of data and lower computational complexity
   * **Translation invariance:** Pooling helps introduce a degree of translation invariance by reducing the sensitivity of the network to small variation in the position of features within the input range. (Translation invariance refers to the ability of a system to recognize an object or pattern regardless of its position or location within the input data.)
3. **Advantages of Pooling:**
   * **Reduced Computational Complexity:** Pooling reduces the number of parameters and computations in the network, making it computationally efficient.
   * **Feature invariance:** Pooling helps the network become more robust to variations in the position and orientation of features in input image, enhancing generalization.
   * **Dimensionality reduction:** By reducing the spatial dimensions of feature maps, pooling helps control overfitting and prevents the network from memorizing noise in the data.
4. **Disadvantages of Pooling:**
   * **Information Loss:** Pooling operations discard some spatial information from input feature maps, potentially leading to loss of fine-grained details.
   * **Reduced resolution:** Aggressivepooling can result in significant reduction in spatial resolution, which may be undesirable for tasks requiring precise location of features.





## Famous CNN architecture

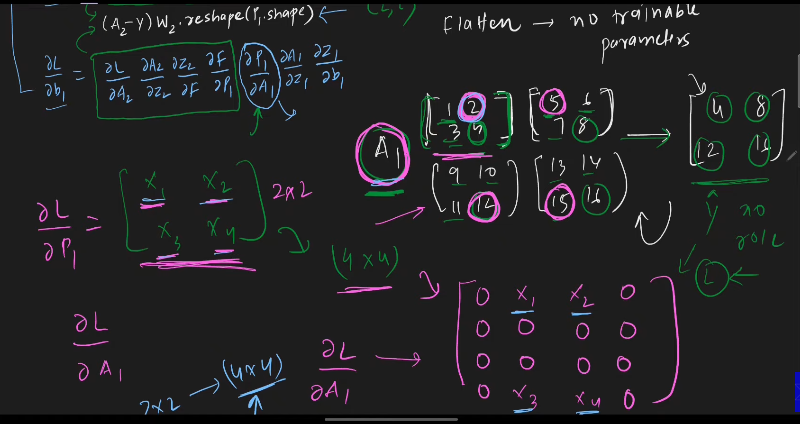
* LeNet-5:
  + **Introduction**: Developed by Yann LeCun et al. in 1998.
  + **Key Features**: Comprises convolutional layers followed by average pooling layers. Utilized for handwritten digit recognition.
  + **Impact**: Pioneering architecture that laid the foundation for modern CNNs.
* AlexNet:
  + **Introduction**: Developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012.
  + **Key Features**: Introduced deeper architectures with eight layers, including convolutional layers with ReLU activation and max-pooling layers.
  + **Impact**: Significantly improved image classification performance on the ImageNet dataset, winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012.
* ResNet (Residual Network):
  + **Introduction**: Proposed by Kaiming He et al. in 2015.
  + **Key Features**: Utilizes residual blocks with skip connections to address the problem of vanishing gradients in deep networks. Deeper architectures (up to hundreds of layers) can be trained effectively.
  + **Impact**: Achieved state-of-the-art performance on various computer vision tasks, including image classification and object detection.
* VGGNet (Visual Geometry Group Network):
  + **Introduction**: Developed by the Visual Geometry Group at the University of Oxford in 2014.
  + **Key Features**: Comprises deep architectures with small 3x3 convolutional filters stacked on top of each other. Simplicity and uniformity in architecture.
  + **Impact**: Demonstrated strong performance on the ImageNet dataset, providing a baseline for deep convolutional architectures.
* GoogLeNet (Inception):
  + **Introduction**: Developed by Szegedy et al. at Google in 2014.
  + **Key Features**: Introduced the concept of inception modules, which perform convolutions at multiple scales and concatenate the results. Utilizes global average pooling and auxiliary classifiers.
  + **Impact**: Achieved high accuracy on the ImageNet dataset while maintaining computational efficiency, inspiring subsequent architectures.
* Inception (Inception-v3, Inception-v4):
  + **Introduction**: Continuation of the GoogLeNet project, with improved versions released in subsequent years.
  + **Key Features**: Inception-v3 and Inception-v4 architectures incorporate advancements such as factorization, batch normalization, and aggressive use of regularization techniques.
  + **Impact**: Improved performance over earlier versions and served as a basis for further research in efficient and effective CNN architectures.

## Backpropagation

Back propagation is crucial algorithm used to train CNNs. It allows the network to learn by adjusting its weights and biases based on error between its prediction and the acutal labels. Below is the breakdown of backpropagation in CNNs:

Lets consider a simple CNN with two layers:

* Convolution Layer:
  + Input: 3x3 image (grayscale).
  + Filter: 2x2 matrix with learnable weights
  + Output: 2x2 feature map
* Fully Connected Layer:
  + Input: Flatten 2x2 feature map into a 4-dimensional vector.
  + Weights: Matrix connecting each element of the input vector to output neuron
  + Output: Single neuron with sigmoid activation, representing the predicted class probability.
* **Forward Pass**:
  + **Input**: We start with a 3x3 image representing the input data.
  + **Convolution**: The filter slides across the image, performing element-wise multiplication with the corresponding elements in the image and summing the products. This is done for each position of the filter to create a feature map. In our example, we’ll get a 2x2 feature map capturing local features like edges or corners.
  + **Activation**: The elements in the feature map are passed through an activation (e.g. ReLU) to introduce non-linearity and prevent vanishing gradients.
  + **Flattening**: The 2x2 feature map is flattened into 4-dimensional vector.
  + **Fully Connected Layer:** The flattened vector is multipled by the weights matrix of fully connected layer, and the sum is passed through the activation function (e.g. sigmoid) to obtain the final output, which represents the predicted probability of the image belonging to a specific class.
* **Backpropagation:**
  + **Error Calculation:** We compare the predicted output with the actual label using a loss function (e.g., cross-entropy). This gives us the error associated with the prediction.
  + **Fully connected Layer:**
    - **Gradient of Loss w.r.t Output:** We calculate the derivative of the loss function with respect to the output neuron’s activation using chain rule.
    - **Gradient of Lss w.r.t Weights:** We use the calculated output gradient and the input vector to compute the gradient of the loss function with respect to each weight in the fully connected layer using the chain rule again.
  + **Reshaping and Convolution:**
    - **Gradiet w.r.t Feature Map:** The gradient from the fully connected layer needs to be propagated back to convolutional layer. We reshape the gradient into original feature map size.
    - **Flipped Filter Convolutional:** We flip the filter horizontally and vertically (180° rotation). This is because the convolution operation in the forward pass essentially correlates the filter with the input. Flipping the filter helps calculate how each element in the filter contributed to the overall error.
    - **Element-wise Multiplication:** We perform element-wise multiplication between the flipped filter and the gradient from the previous layer. This effectively distributes the error signal back to the input of the convolutional layer.
    - **Gradient of Loss w.r.t Filter:** We average the element-wise product from the previous step to obtain the gradient of the loss function with respect to each element in the filter.



## Data Augmentation

Data augmentation is a technique used to artificially increase the size and diversity of a dataset by applying various transformations to the existing data samples. It is commonly used in deep learning tasks, especially in computer vision, to improve the generalization and robustness of Deep learning models.

Operation of data augmentation:

1. Existing dataset: Start with a dataset containing a limited number of samples.
2. Transformations: Apply a variety of transformations to the original data samples, such as rotation, scaling, cropping, flipping, translation, brightness adjustment and noise addition.
3. Augmented Dataset: Generate new data samples by applying these transformations to the original samples.
4. Training: Use the augmented dataset, along with the original data samples, for training deep learning models.

Benefits of Data Augmentation:

1. Increase Diversity: Augmented data introduces variations in the dataset, aking the model more robust to different scenarios.
2. Regularization: data augmentation acts as a form of regularization, helping prevent overfitting by exposing the model to a wider range of input variations.
3. Improved Generalization: Models trained on augmented data tend to generalize better to unseen data, leading to improved performance on test datasets
4. Reduced Data Dependency: With data augmentation, models can be trained effectively even with limited amounts of original data.

Common Data Augmentation Technique:

1. Rotation: Rotate the image by a certain angle
2. Scaling: Zoom in or out of the image
3. Translation: Shift the image horizontally and vertically
4. Flipping: Flip the image horizontally or vertically
5. Cropping: Crop a portion of the image
6. Brightness Adjustment: Adjust the brightness of the image
7. Noise Addition: Add random noise to the image.

Code to do Data Augmentation

From tensorflow.keras.preprocessing.image import ImageDataGenerator

Datagen = ImageDataGenerator(

rotation\_range=20

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=true,

fill\_mode=’neares’

)

augmented\_data = datagen.flow(x\_train,y\_train,batch\_size=32)

## Pre-trained model

* **Imagenet**: A visual database created by Fei Fei Li containing 1.4 million images with 20,000 categories, labeled and annotated with bounding boxes for object localization.
* **ILSVRC**: The ImageNet Large Scale Visual Recognition Challenge, where researchers compete to achieve the lowest error rates on various image classification tasks.
* **AlexNet**: Introduced by Geoffrey Hinton in 2012, it utilized CNNs and ReLU activation functions, achieving a significant reduction in error rates on ILSVRC tasks.
* **ResNet**: Residual Networks introduced in subsequent years achieved even lower error rates, with ResNet models currently reaching around 3.5% error rates.

## Transfer Learning

* **Definition**: The practice of storing knowledge gained while solving one problem and applying it to a different but related problem.
* **Benefits**: Saves time and computational resources by leveraging pre-trained models instead of training from scratch. Especially useful when training data is limited.

There are two ways of transfer learning

* **Feature Extraction**: Involves freezing the convolutional parameters of a pre-trained model and only training the fully connected layers on new data.
* **Fine-tuning**: Allows training of the last few convolutional layers along with the fully connected layers, adapting the pre-trained model to the new data.

### Advantages of Transfer Learning:

* **Efficiency**: Reduces the need for large amounts of labeled training data and extensive computational resources.
* **Generalization**: Leverages knowledge learned from one task to improve performance on related tasks, even if the domains are different.
* **Flexibility**: Provides a starting point for custom model development, allowing researchers to focus on specific problem domains.