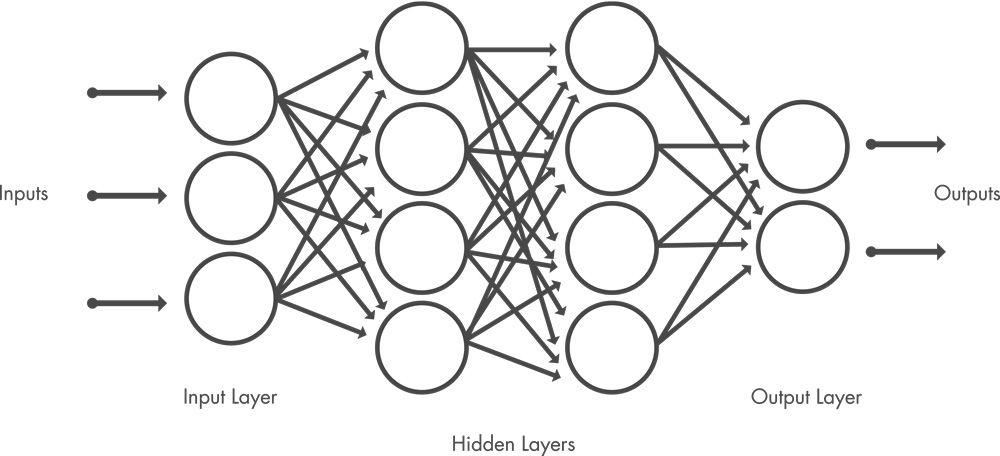
**DEEP LEARNING**

**- Convolutional neural network:**

**-> A convolutional neural network can have tens or hundreds of layers that each learn to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start with very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object.**

**1/Feature Learning, Layers, and Classification**

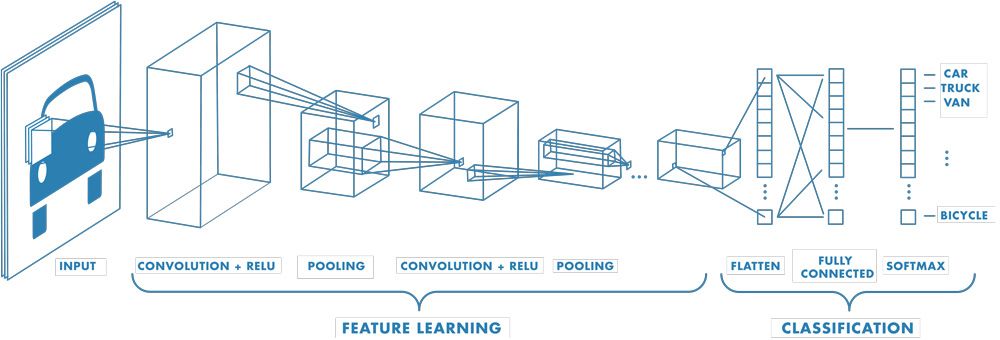
**A CNN is composed of an input layer, an output layer, and many hidden layers in between.**

**[[](https://www.mathworks.com/discovery/convolutional-neural-network.html)](https://www.mathworks.com/discovery/convolutional-neural-network.html)**

**These layers perform operations that alter the data with the intent of learning features specific to the data. Three of the most common layers are convolution, activation or ReLU, and pooling.**

* **Convolution puts the input images through a set of convolutional filters, each of which activates certain features from the images.**
* **Rectified linear unit (ReLU) allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as *activation*, because only the activated features are carried forward into the next layer.**
* **Pooling simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn.**

**These operations are repeated over tens or hundreds of layers, with each layer learning to identify different features.**

**[[](https://www.mathworks.com/discovery/convolutional-neural-network.html)](https://www.mathworks.com/discovery/convolutional-neural-network.html)**

**Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer.**

**Convolution Operation Step-by-Step**

**+Let's break down the convolution operation step-by-step using a specific example.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động**

**Convolution Operation**

1. **Place the filter at the top-left corner of the input image.**
2. **First Position Calculation (top-left corner):**
   * **Multiply each element of the filter with the corresponding element of the image patch.**
   * **Sum the results to get the output value for that position.**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động**

1. **Move the filter one step to the right (stride = 1):**
   * **Recalculate for the next position.**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động**

1. **Continue moving the filter across the image:**
   * **Repeat the calculation for each new position until the entire image is covered.**

**Output Feature Map:**

**The output matrix will be smaller than the input image if no padding is used:**

**Ảnh có chứa ảnh chụp màn hình, số, Phông chữ, đồng hồ

Mô tả được tạo tự động**

**NOTE:**

**1. Training the Neural Network**

**(a) Initial Weights and Biases:**

* **At the start, the neural network initializes the weights and biases randomly or with small values close to zero.**
* **For example:**
  + **Weight wi and bias b are randomly assigned.**

**(b) Feedforward Step:**

* **The training data (inputs and corresponding outputs) is passed through the network.**
* **The output of the network is compared with the actual target value (label) using a loss function.**

**(c) Loss Function:**

* **The loss function measures how far the network's predictions are from the actual target values.**
* **For regression tasks, we use Mean Squared Error (MSE):**

**Ảnh có chứa Phông chữ, đồng hồ, ảnh chụp màn hình, thuật in máy

Mô tả được tạo tự động**

**Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, thiết kế

Mô tả được tạo tự động**

**(d) Backpropagation:**

* **The network adjusts the weights and biases based on how much they contributed to the error.**
* **This is done using gradient descent and the chain rule of calculus.**

**2. Learning Process**

**(a) Gradient Descent:**

* **During backpropagation, gradients of the loss function with respect to weights and biases are calculated.**
* **The weights and biases are updated iteratively using:**

**Ảnh có chứa ảnh chụp màn hình, Phông chữ, văn bản, số

Mô tả được tạo tự động**

* **Where:**
  + **η: Learning rate (a small constant controlling step size).**

**(b) Epochs:**

* **Training is repeated over the dataset multiple times (epochs) to minimize the loss.**

**3. Activation Functions**

* **The choice of activation functions (like ReLU or sigmoid) depends on the problem:**
  + **ReLU is widely used for hidden layers because it works well in deep networks.**
  + **Softmax is used for classification tasks in the output layer.**

**4. Hyperparameters**

**These are manually chosen parameters that control the training process:**

* **Learning rate (η): Controls the step size in weight updates.**
* **Number of hidden layers and nodes: Decides the model's capacity to learn.**
* **Epochs and batch size: Defines how often and in what chunks data is fed to the model.**

**Example of the Process:**

**Imagine training the network to predict if a student will pass based on their study hours and attendance:**

1. **Provide labeled data:**
   * **Input: [5,80]**
   * **Target: 1 (pass)**
2. **Start with random weights and biases.**
3. **Compute the output of the network using the forward pass.**
4. **Compare the prediction with the actual label and calculate the error.**
5. **Update weights and biases to reduce the error using backpropagation.**

**2/Shared Weights and Biases:**

**Unlike a traditional neural network, a CNN has shared weights and bias values, which are the same for all hidden neurons in a given layer.**

**This means that all hidden neurons are detecting the same feature, such as an edge or a blob, in different regions of the image. This makes the network tolerant to translation of objects in an image. For example, a network trained to recognize cars will be able to do so wherever the car is in the image.**

**a. What Does Shared Weights and Biases Mean?**

**In a CNN, instead of each neuron having its own set of weights and biases (as in fully connected layers), one set of weights and biases is shared across all neurons in a convolutional layer. This means:**

* **A filter (or kernel) with a fixed set of weights slides over the entire input (e.g., an image) to detect patterns like edges or textures.**
* **This filter is applied to all regions of the input, producing a feature map.**
* **All the neurons in a convolutional layer use the same weights and bias but operate on different parts of the input.**

**b. Why Share Weights and Biases?**

**The primary reasons are:**

**(a) Feature Detection Across Regions:**

* **Each filter detects the same feature (e.g., edges, corners, textures) in different locations of the input.**
* **For example:**
  + **If a filter detects a vertical edge in one region of the image, the same filter will detect vertical edges elsewhere in the image.**

**(b) Translation Invariance:**

* **Shared weights make the network tolerant to the position of features in the input.**
* **Example:**
  + **If a car appears in the top-left corner of an image or in the bottom-right, the same filter will still detect it.**

**(c) Fewer Parameters:**

* **Traditional fully connected layers require separate weights for every connection, which becomes computationally expensive for large inputs like images.**
* **Sharing weights drastically reduces the number of parameters, making the network more efficient and less prone to overfitting.**

**c. How It Works in Practice?**

**Consider a grayscale image as input:**

**Example: Detecting an Edge**

1. **Input: A 5×5image (a small matrix of pixel values).**
2. **Filter: A 3×3 filter (a small matrix of weights).**
   * **Filter weights are shared across all regions of the input.**
   * **For example, a filter might look like:**

**Ảnh có chứa ảnh chụp màn hình, Phông chữ, biểu đồ, số

Mô tả được tạo tự động**

**This filter detects vertical edges.**

1. **Convolution Operation:**
   * **The filter slides (or convolves) over the input image.**
   * **At each position, the dot product between the filter and the image patch is computed, producing a single value.**
   * **These values form a feature map.**

**d. Advantages of Shared Weights in CNNs**

* **Efficiency:**
  + **Fewer parameters mean faster training and less memory usage.**
* **Better Generalization:**
  + **Sharing weights ensures the network focuses on general features rather than overfitting to specific regions.**
* **Robustness to Translation:**
  + **Objects can appear in different parts of the image, and the CNN can still detect them.**

**3/Classification Layers**

**After learning features in many layers, the architecture of a CNN shifts to classification.**

**The next-to-last layer is a fully connected layer that outputs a vector of K dimensions (where K is the number of classes able to be predicted) and contains the probabilities for each class of an image being classified.**

**The final layer of the CNN architecture uses a classification layer to provide the final classification output.**

**a. Transition from Feature Learning to Classification**

* **In the earlier layers of a CNN:**
  + **The convolutional and pooling layers extract features from the input (like edges, textures, and patterns).**
  + **These features capture meaningful spatial hierarchies of the input (e.g., in an image of a cat, the CNN learns low-level features like edges, mid-level features like whiskers, and high-level features like the face).**
* **In the final layers:**
  + **The network shifts focus from learning features to classifying the input based on the features.**

**b. Fully Connected Layer**

* **After the convolutional layers, the feature maps are flattened into a 1D vector. This vector represents the extracted features.**
* **The fully connected layer (FC) acts like a traditional neural network layer. It connects every input (flattened feature) to every output (class probability).**

**Output of Fully Connected Layer:**

* **The fully connected layer outputs a vector of size K, where K is the number of classes the network is trained to predict.**
* **Each value in this vector represents the score or logit for a class (e.g., the likelihood of an input belonging to a specific class).**

**c. Softmax Activation Function**

**The final classification layer typically uses the softmax activation function to convert the scores (logits) into probabilities.**

**Formula for Softmax:**

**Ảnh có chứa Phông chữ, ảnh chụp màn hình, đồng hồ, thiết kế

Mô tả được tạo tự động**

**Where:**

* **P(y=k): Probability of the input belonging to class kkk.**
* **zk : Logit (score) for class kkk.**
* **k: Total number of classes.**

**Properties of Softmax:**

1. **The output values are in the range [0,1].**
2. **The sum of all probabilities equals 1, making it easy to interpret as a distribution over classes.**

**d. Final Classification**

* **The class with the highest probability in the softmax output is chosen as the final prediction.**
* **For example:**
  + **If the softmax output is [0.1, 0.7, 0.2] for classes [A, B, C] the network predicts class B because 0.7 is the highest probability.**

**e. Overall Workflow**

1. **Input: An image is fed into the CNN.**
2. **Feature Extraction: Convolutional and pooling layers learn hierarchical features from the image.**
3. **Flattening: The feature maps are flattened into a single vector.**
4. **Fully Connected Layer:**
   * **Applies learned weights and biases to the flattened vector to generate scores for each class.**
5. **Softmax Layer:**
   * **Converts scores into probabilities.**
6. **Output:**
   * **The final prediction is the class with the highest probability.**

**Example: Image Classification with 3 Classes**

**Suppose a CNN is trained to classify images into 3 categories: Cat, Dog, and Bird.**

* **Input Image: A picture of a dog.**
* **Convolutional Layers:**
  + **Detect features like fur texture, ear shape, etc.**
* **Flattened Features:**
  + **A vector of features, say [1.2, 0.8, −0.5, 2.3], is passed to the FC layer.**
* **Fully Connected Layer:**
  + **Outputs raw scores (logits): [2.5, 3.1, 1.2] (one score for each class: Cat, Dog, Bird).**
* **Softmax Layer:**
  + **Converts logits into probabilities: [PCat, PDog, PBird] = [0.2, 0.7, 0.1]**
* **Prediction:**
  + **The network predicts "Dog" as the final output because PDog​=0.7 is the highest.**

**f. Advantages of This Structure**

* **The fully connected and classification layers ensure that the CNN can generalize well and predict the correct output class.**
* **Using softmax makes the output interpretable as probabilities, enabling applications like multi-class classification.**