# **Derin Öğrenme Konusu Hakkında Araştırma Raporu**

**Hazırlayan**: Marya Hijazi

**Tarih**: 27/10/2025\_\_\_\_\_\_

## **INTRODUCTION**

The advent of Artificial Intelligence has marked a pivotal moment in technological advancement, fundamentally transforming how machines interact with and interpret the world around them. No longer confined to theoretical research, AI technologies have become integral to modern society, driving innovation across healthcare, commerce, transportation, and countless other fields that shape our daily experiences.

Understanding Artificial Intelligence requires navigating through a carefully structured hierarchy of concepts and methodologies. At its foundation, AI encompasses the broad ambition of creating intelligent systems capable of human-like reasoning and decision-making. This overarching goal is primarily achieved through Machine Learning, which enables computers to learn from experience and data rather than following rigid, pre-programmed instructions. The most sophisticated manifestation of this approach is Deep Learning, which leverages complex, multi-layered neural networks inspired by the biological structure of the human brain.

* 1. **Artificial intelligence**

Artificial Intelligence is a technology that enables machines and computers to simulate human capabilities in learning[1], understanding, problem-solving, and decision-making. Despite its various definitions, the common understanding links it to systems capable of interpreting data, learning from it, and achieving specific goals through flexible adaptation, without the need for continuous human intervention.

With rapid technological development, Artificial Intelligence has become an essential part of our daily lives, to the extent that some of its applications are no longer considered AI due to their prevalence, such as Optical Character Recognition and Siri. Its applications extend to include e-commerce where it predicts suitable products for shoppers, voice assistants like Alexa, self-driving cars, social media platforms that determine content displayed to users, and music streaming services that suggest appropriate songs.[2]

In the medical field, Artificial Intelligence contributes to cancer detection, reviewing radiology results, accelerating diagnoses, and identifying new treatments, opening wide horizons for improving healthcare and saving lives.

* 1. **Machine learning**

Machine Learning is a subfield of Artificial Intelligence that focuses on developing algorithms capable of learning from data and extracting patterns, then applying these patterns to make decisions or predictions about new data without the need for explicit programming or continuous human intervention. The process begins with providing high-quality data used to train models, where model performance is improved through a process called "model training," enabling it to deliver accurate predictions in real-world scenarios.[4]

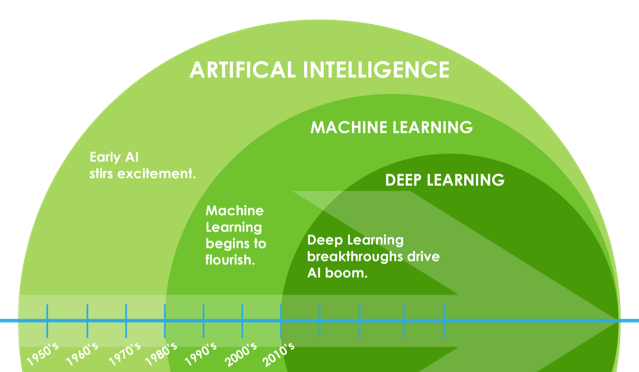
Machine Learning has come to dominate the field of Artificial Intelligence, providing the backbone for most modern systems, from predictive models to self-driving vehicles to large language models and generative AI tools. This discipline is closely related to data science, where it can be understood as a set of techniques for automating data analysis and applying lessons learned from it.

Machine Learning algorithms are used in a wide range of practical applications, including recommending products to consumers, predicting stock market fluctuations, translating texts, medical applications, email filtering, and computer vision, where developing traditional algorithms is difficult or impossible. Machine Learning is also known as predictive analytics when applied to solving business challenges.[3]

* 1. **The relationship between AI and ML**

Artificial Intelligence is the broader and more comprehensive field in computer science, aiming to create systems and machines capable of mimicking human intelligence, solving problems in a human-like manner, and performing tasks that typically require human cognitive abilities such as logical thinking, decision-making, natural language processing, and pattern recognition. Machine Learning, on the other hand, is a narrow branch of Artificial Intelligence, serving as a method to achieve AI by enabling systems to learn and improve automatically from experience and data without being explicitly programmed for each specific task. Both fields go beyond basic automation and programming to generate outputs based on complex data analysis.

The relationship between them is complementary, as Machine Learning is a part of Artificial Intelligence and not vice versa. Artificial Intelligence represents the ultimate goal of creating intelligent machines, while Machine Learning is one of the means or tools to achieve this goal. AI and ML solutions are suitable for complex tasks that generally involve accurate results based on acquired knowledge. For example, in a facial recognition program, the traditional AI approach requires manually writing specific rules about the shape of eyes, nose, and other features, while the Machine Learning approach relies on feeding the system thousands of images of different faces so it can learn on its own how to recognize faces by discovering patterns in the data. In short, all Machine Learning is Artificial Intelligence, but not all Artificial Intelligence is Machine Learning, as there are other AI techniques that do not rely on learning from data.[5]



**Figure:1.1. ML&AI**

* 1. **Deep learning**

Deep Learning is an advanced branch of Machine Learning that uses multi-layered neural networks which more accurately mimic the human brain's ability to make complex decisions. Deep Learning is based on the principle of extracting features from raw data using multiple layers to identify different aspects related to input data. Deep neural networks consist of an input layer, at least three hidden layers that sometimes reach hundreds, plus an output layer, unlike neural networks used in traditional Machine Learning models which typically contain only one or two hidden layers.[7]

Deep Learning techniques include Convolutional Neural Networks, Recurrent Neural Networks, and Deep Neural Networks, all of which use artificial neural networks to process large amounts of data. These multiple layers enable unsupervised learning, as they are capable of automatically extracting features from large, unlabeled, and unstructured datasets, and making their own predictions about what this data represents without the need for continuous human intervention. In the past, the use of Machine Learning was limited due to its inability to process raw input data directly, and deep expertise in feature extraction was necessary to transform raw data into a suitable form. However, Deep Learning has overcome this limitation and become an effective and useful technique, especially with the accelerated advancement in computer hardware.[6]

Because Deep Learning does not require significant human intervention, it enables Machine Learning on an extremely large scale. It is perfectly suited for natural language processing, computer vision, and other tasks that require accurately and quickly identifying complex patterns and relationships in large amounts of data.

## Artificial Neural Networks

Artificial Neural Networks are adaptive artificial systems inspired by the workings of the human brain, representing a computational model based on the structure and functions of biological neural networks. These networks are nonlinear statistical data modeling tools capable of modifying their internal structure according to a functional objective, and are used to describe complex interactions between inputs and outputs or to discover patterns. Since neural networks change or learn based on inputs and outputs, the information passing through the network modifies the structure of the artificial neural network, making them particularly suitable for solving nonlinear problems as they are capable of reconstructing the ambiguous rules that govern the optimal solution to these problems.[8]

* 1. **Artificial Neural Networks layers**

**2.2.1. Input Layer**

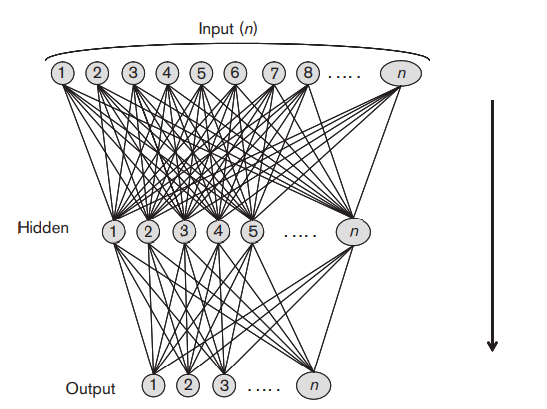
Comprises a set of artificial neurons that transmit information from initial neuron layers to the system for processing. This layer initiates the workflow in the neural network.

**2.2.2. Hidden Layer**

Located between the input and output layers where the inputs and outputs of artificial neurons are weighted. The network may contain one or several hidden layers depending on the complexity of the task.

**2.2.3.Output Layer**

Represents the final layer of neurons that provides specific outputs to the programmer. Since it is the final "executing" node of the network, neurons in this layer can be constructed and processed differently.



**Figure:2.1. Layers of ANN**

Data enters the network through inputs and travels through the network's layers one by one until it reaches the outputs. All processing units within the network are connected to other processing units in their vicinity through connections characterized by excitatory or inhibitory strength, where positive values indicate excitatory connections and negative values indicate inhibitory connections.[8]

The basic elements of an artificial neural network are nodes or processing elements and the connections between them, where each node has its own input through which it receives connections from other nodes and its own output through which it communicates with them, in addition to a function through which it transforms its total input into an output. The connections between nodes can change over time, initiating a learning process throughout the entire artificial neural network, and the way nodes change is called the "learning rule." Each of these connections has what is called a weight that modifies the value of inputs or outputs, and the value of these connection weights is determined during the training process. This function is the basis of the artificial neural network's ability to learn.[8]

* 1. **Activation Functions in Artificial Neural Networks**

Activation functions are a vital component in artificial neural networks, determining whether a particular neuron should be activated based on the weighted inputs it receives. These functions transform linear inputs into nonlinear outputs, enabling the network to learn and represent complex relationships in data. Without activation functions, a neural network would simply be a linear model incapable of solving complex problems. An activation function acts as a decision gate that determines how much information should pass to the next layer in the network, helping to introduce the necessary nonlinearity for learning from complex data.

* + 1. **The types of activation functions**
  1. **Sigmoid Function**

This function uses real numbers as inputs and limits the output between 0 and 1, making it useful for binary classification tasks. The sigmoid function curve is S-shaped, and it is commonly used in the final layers of networks when we need probabilities.

* 1. **Tanh Function**

The tanh function is similar to sigmoid since both use real numbers as their inputs. However, the tanh function limits its output between -1 and 1. This wider range makes it more effective than sigmoid in some applications, especially when data is centered around zero.

* 1. **ReLU Function**

ReLU stands for Rectified Linear Unit and is the most common activation function used in feature extraction using neural networks. It converts all inputs into positive numbers, returning the value itself if positive and zero if negative. The computational load of ReLU is relatively lower than other functions, making it efficient for rapid training of deep networks.

* 1. **Softmax Function**

his function is primarily used in the output layer of neural networks designed for multi-class classification tasks. Softmax converts a set of values into probabilities ranging between 0 and 1, such that the sum of all resulting probabilities equals 1. This makes it ideal for determining the most likely category among several options, giving each category a probability that reflects the model's confidence that the input belongs to that category.

## **3. Deep learning models**

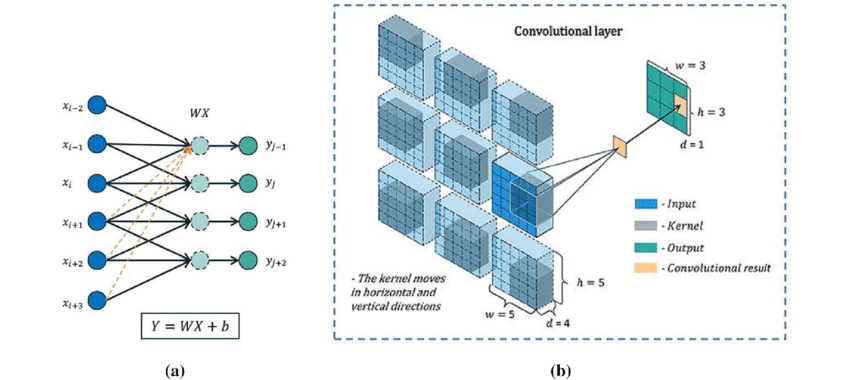
**3.1. Convolutional Neural Networks**

## Convolutional Neural Networks are deep learning models designed to process data with grid-like structure, such as images, and form the foundation for most modern computer vision applications for detecting features within visual data. These networks perceive images as collections of numbers, commonly known as matrices, where each number represents light intensity at a specific point called a pixel. **3.1.2. Layers of Convolutional Neural Networks**

## **1. Convolutional Layers** These layers apply convolution operations to input images using filters or kernels to detect features such as edges, textures, and more complex patterns, helping to preserve spatial relationships between pixels. The convolutional layer is defined by three fundamental parameters: kernel size (the size of the sliding kernel filter), stride length (the number of kernels that slide before forming product points and generating output pixels), and padding (the size of the zero frame prepared around the input feature map).

The convolution mechanism is based on applying a filter of a specific size (such as 3×3 or 5×5) to the input image, where this filter analyzes small portions of the image at a time, and the filter is moved across the image using a specified stride. At each step, the convolutional output is calculated through dot product between pixel values in the image portion and filter values, and these values are then aggregated into a feature map.

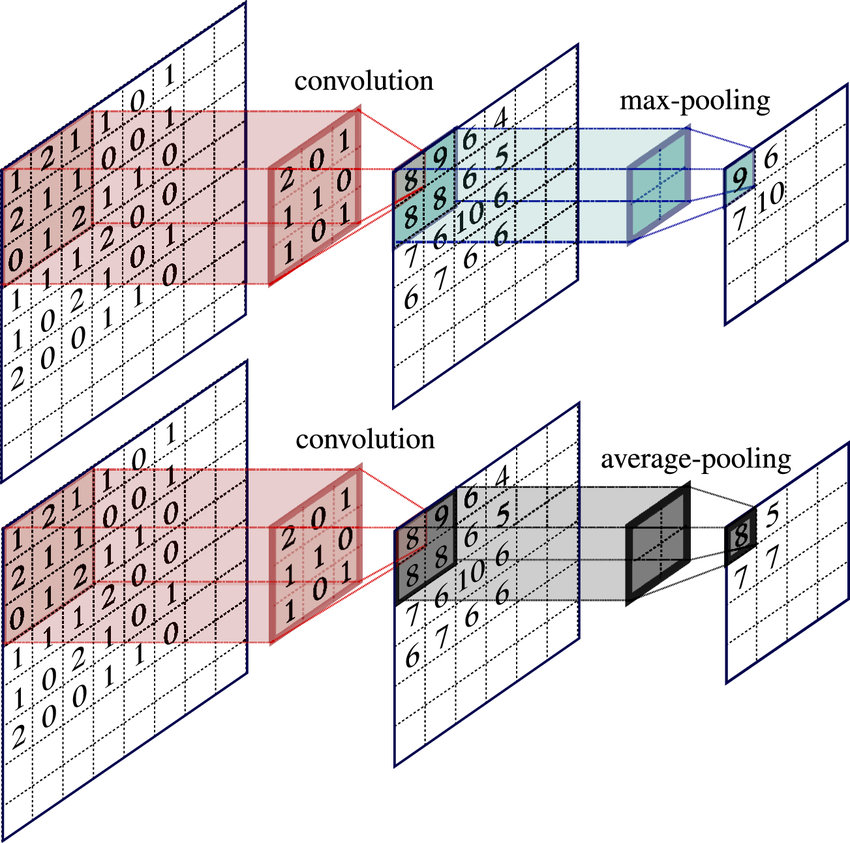
An important characteristic of convolutional layers is that they automatically learn the most important patterns in the data, where the first layers become responsible for capturing basic features such as edges and boundaries, while deeper layers specialize in analyzing more complex patterns such as shapes and complete objects. This allows the network to generalize and handle different forms of the same category without the need for manual feature design, making CNNs powerful in computer vision tasks.



**Figure:3.1.Convolutional Layers**

**2. Pooling Layers**

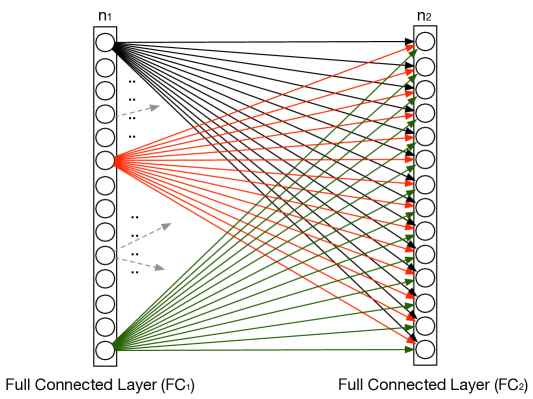
These layers reduce the spatial dimensions of input data, thereby reducing computational complexity and the number of parameters in the network. The pooling layer combines two consecutive convolutional layers and reduces the number of parameters and computational loads by creating downsampled representations. Max pooling is a common pooling operation, where the maximum value is selected from a group of neighboring pixels. The function in the pooling layer can produce a maximum or average value, and maximization composition is often used to obtain an optimal function. The pooling layer also helps reduce overfitting or computational weights.



**Figure:3.2.Pooling Layers**

**3. Fully Connected Layer**

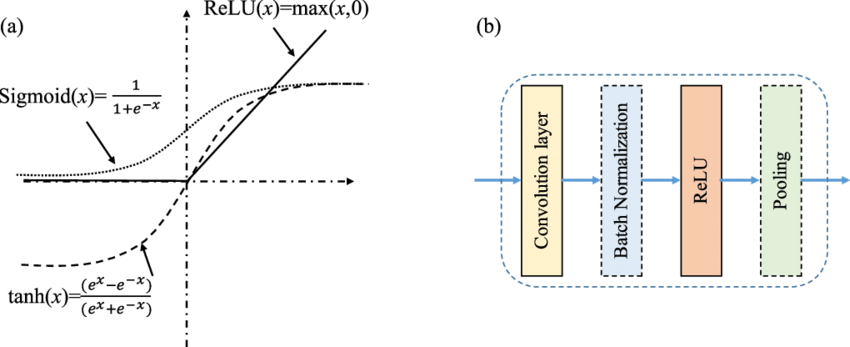
The third layer is the fully connected layer, commonly called the convolutional output layer. This layer resembles a feedforward neural network and is typically located in the lower layer of the network. The layer receives inputs from the final pooling or convolutional output layer and is flattened before being sent to the subsequent layer. The uniform distribution of output means unwrapping all result values obtained after the last pooling layer or convolutional layer into a vector (three-dimensional matrix). This method is a simple technique for studying high-level nonlinear combinations of features represented by the convolutional output layer. Each neuron in this layer connects to every neuron in the next layer and is responsible for making predictions based on the high-level features learned by previous layers.



**Figure:3.3.Fully Connected Layer**

## **4. Nonlinearity Layer (Activation Functions)**

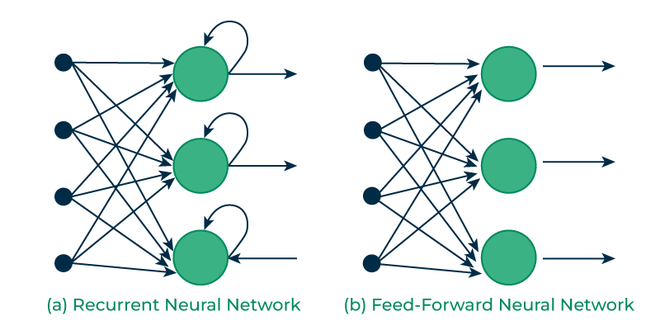
The activation function plays a fundamental role in CNN layers, as it adds nonlinearity to the model by enabling it to learn more complex relationships in the data. The filtered output provides another mathematical function called activation, and the main goal of the activation function is to determine the final output of the neural network, such as "yes" or "no". The activation function maps output values between -1 and 1, or between 0 and 1, and so on.



**Figure:3.4.Nonlinearity Layer**

**3.2. Recurrent Neural Networks**

Recurrent Neural Networks are an advanced class of artificial neural networks specifically designed to process sequential data and time series, representing a comprehensive set of feedforward neural networks enhanced with the ability to pass information across time steps[10]. Unlike traditional neural networks such as feedforward and convolutional networks that assume inputs and outputs are independent of each other, recurrent networks can process examples one by one while maintaining a state or memory that reflects an arbitrarily long context window, making their outputs dependent on previous elements within the sequence. These networks capture temporal dynamics through cycles in the graph, where the current state of the network's outputs is connected to a combination of the network's inputs and its previous states by feeding back the outputs of the hidden layer or network outputs to the input layer.



**Figure:3.4. RNN vs FNN**

Recurrent Neural Networks are characterized by their "memory" feature, where they leverage information from previous inputs to influence current inputs and outputs, and they are a rich family of models capable of performing nearly arbitrary computations. A well-known result from 1991 demonstrated that a finite-sized recurrent neural network with sigmoid activation functions can simulate a universal Turing machine, reflecting its enormous computational capacity. Another distinctive characteristic of recurrent networks is that they share parameters across each layer of the network. While feedforward networks have different weights for each node, recurrent neural networks share the same weight parameter within each layer of the network, though these weights are adjusted through backpropagation and gradient descent processes to facilitate reinforcement learning.

Recurrent Neural Networks can be used to predict daily flood levels based on previous flood, tide, and daily meteorological data, and they can also be used to solve sequential or temporal problems such as language translation, natural language processing, sentiment analysis, speech recognition, and image captioning. Although future events may also be useful in determining the outputs of a given sequence, unidirectional recurrent neural networks cannot take these events into account in their predictions, which represents one of the limitations facing these models.

**3.2.1. Types of Recurrent Neural Networks**

**1. Standard Recurrent Neural Networks** which represent the basic form of this architecture,[9]

**2. Bidirectional Recurrent Neural Networks (BRNNs)** which process data in both forward and backward directions to obtain more complete context, [9]

**3. Long Short-Term Memory (LSTM) network**s which overcome the vanishing gradient problem and retain information for longer time periods, [9]

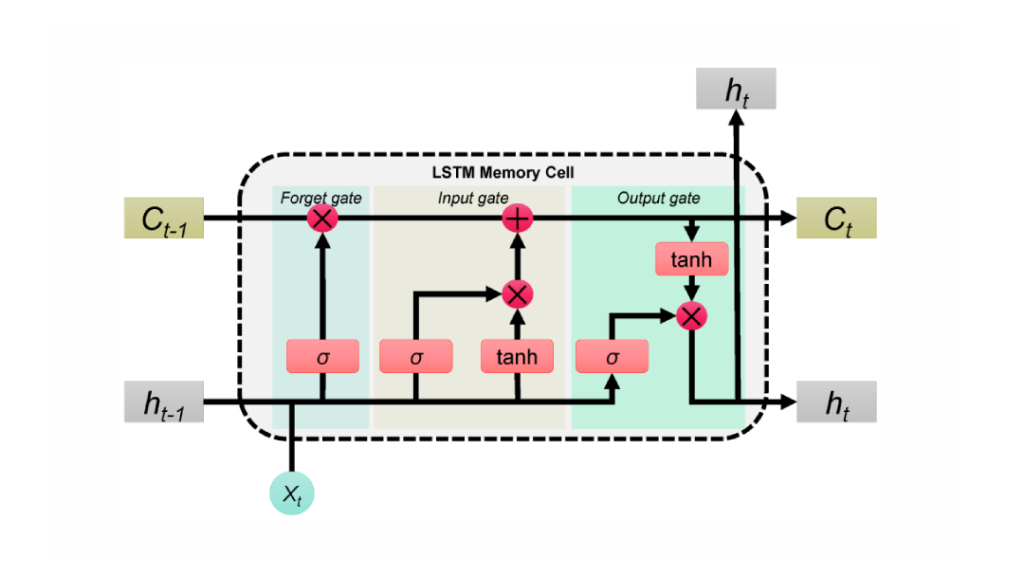
**4. Gated Recurrent Units (GRUs)** which offer a simplified structure with performance similar to LSTM, [9]

**5. Encoder-Decoder RNN networks** used in tasks such as machine translation where the encoder processes the input sequence while the decoder generates the output sequence.[9]

**3.3. Long Short-Term Memory Network (LSTM)**

Long Short-Term Memory is a powerful recurrent neural network model specifically designed to overcome the vanishing and exploding gradient problems that typically arise when learning long-term dependencies, even when minimum time lag intervals are very long. This problem can be prevented using a Constant Error Carousel mechanism that maintains the error signal within each unit's cell, where these cells are recurrent networks themselves with an interesting structure in which this mechanism is extended with additional features: an input gate, an output gate, and a forget gate to form the memory cell. The basic LSTM unit consists of a cell that remembers values over arbitrary time intervals, and three gates that regulate the flow of information associated with the cell, where the idea behind the memory block is to maintain its state over time and regulate the flow of information through nonlinear gating units.

LSTM works through several sequential steps beginning with the block input which combines the current input and the LSTM unit output from the last iteration, then comes the input gate which determines what information should be retained in the network's cell states by selecting candidate values that can be added to cell states and activation values for input gates. Next comes the forget gate which determines what information should be removed from previous cell states based on current inputs, outputs, and the state of memory cells in the previous time step. Then the cell value is calculated, which combines block inputs, the input gate, and forget gate values with the previous cell value. Finally, the output gate is calculated, which combines current inputs, LSTM unit outputs, and cell value from the last iteration, then the block output is calculated, which combines the current cell value with the current output gate value. The logistic sigmoid function is used as the activation function for the gate, while the hyperbolic tangent function is often used as the activation function for block input and output.

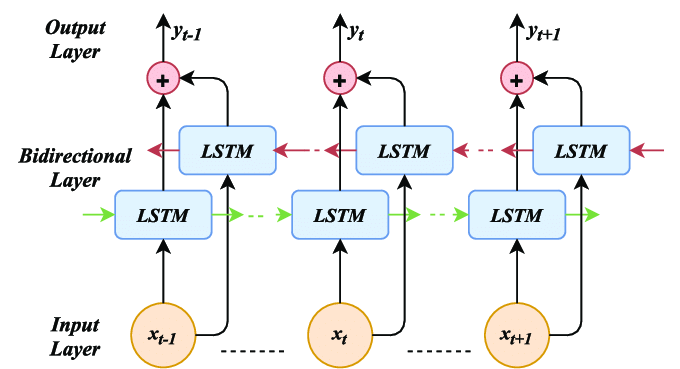


**Figure:3.5.LSTM**

**3.3.1. Types of LSTM**

**1. Bidirectional LSTM**

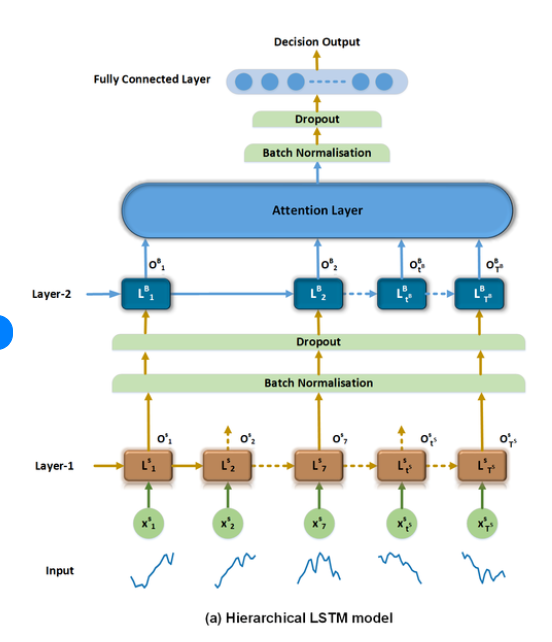
This network processes data in both forward and backward directions simultaneously, allowing the model to access both future and past context information for each time point in the sequence. This makes it particularly useful for tasks such as speech recognition and natural language processing where understanding complete context is essential.[11]



**Figure:3.6.Bidirectional LSTM**

1. **Hierarchical and Attention-based LSTM**

This architecture combines hierarchical structures with attention mechanisms to process data at different levels of abstraction, where the attention mechanism focuses on the most relevant parts of the sequence rather than processing all elements equally, improving performance in tasks such as text summarization and machine translation.[11]



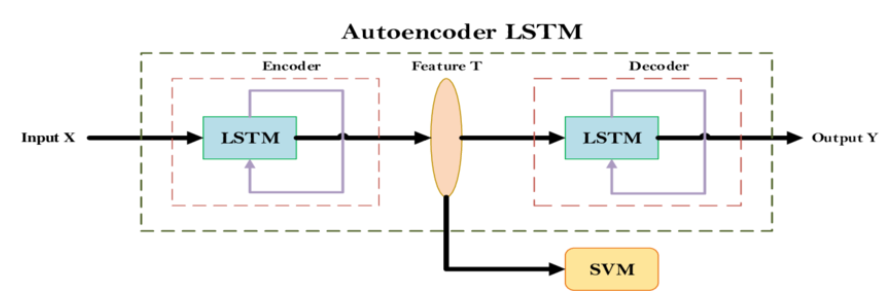
**Figure:3.7.Hierarchical and Attention-based LSTM**

1. **Convolutional LSTM**

This network integrates the capabilities of convolutional networks to extract spatial features with LSTM's capabilities to model temporal dependencies, making it ideal for processing spatiotemporal data such as weather forecasting and video analysis where understanding both spatial patterns and temporal changes is needed.[11]

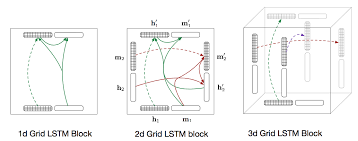
1. **LSTM Autoencoder**

This model uses an encoder-decoder architecture to learn compressed representations of sequential data, where the encoder transforms the sequence into a fixed-length representation while the decoder reconstructs the original sequence. It is useful in anomaly detection, data compression, and sequence generation.[11]

  **Figure:3.8.LSTM Autoencoder**

1. **Grid LSTM**

This architecture extends traditional one-dimensional LSTM networks to multiple dimensions, allowing more effective modeling of multi-dimensional data. It is useful in applications requiring processing of complex information with multi-dimensional structures such as advanced machine translation and language modeling.[11]



**Figure:3.9.Grid LSTM**

1. **Cross-modal and Associative LSTM**

These networks are designed to work with different types of data simultaneously such as text, images, and audio, where they learn correlations and relationships between different modalities, making them powerful in applications such as image captioning, video-text matching, and multimodal emotion recognition.[11]

**4. Model Training and Optimization Process**

**4.1. Dataset Preparation and Splitting**

The dataset preparation phase is a fundamental step in building effective deep learning models, beginning with collecting appropriate data for the required task and then splitting it into three main sets. The training set is used to teach the model and update weights, typically comprising the largest portion of data at around 70-80% of total data, where the model learns patterns and relationships in the data. The validation set is used to tune hyperparameters and evaluate model performance during training to detect overfitting problems, typically comprising 10-15% of the data, and helps make decisions such as when to stop training or how to adjust the learning rate. The test set is used for final evaluation of model performance on data it has never seen before, comprising about 10-15% of the data, and provides an objective measure of the model's ability to generalize to new data. It is important that these splits be random and balanced to ensure fair representation of all data classes in each set.

**4.2. Data Preprocessing**

Data preprocessing includes a set of necessary techniques to improve data quality and model performance. Normalization is the process of transforming data to a specific range, such as converting pixel values in images from the range 0-255 to the range 0-1 or -1 to 1, which helps neural networks converge faster and more stably during training. For example, if you have an image with pixel values 0-255, it can be normalized by dividing each value by 255 to get values between 0 and 1. Data augmentation is used to artificially increase the size of the training set by applying various transformations to the original data, such as rotation, flipping, cropping, and brightness changes for images, or adding noise and changing speed for audio data, which helps the model learn better and reduces overfitting by exposing the model to greater data diversity.

**4.3.Loss Functions**

Loss functions are metrics that determine how far the model's predictions are from the true values and guide the learning process toward performance improvement. Mean Squared Error is primarily used in regression problems where we predict continuous values, calculating the average of squared differences between predicted and true values. For example, if you're predicting house prices and the true price is 200,000 and the prediction is 180,000, the error would be (200,000 - 180,000)² = 400,000,000. Cross-Entropy Loss is used in multi-class classification problems, measuring the difference between the predicted probability distribution and the true distribution of classes. If you have a model to classify images into cat, dog, or bird, this function calculates how confident the model is in the correct classification. Binary Cross-Entropy Loss is used in binary classification problems where there are only two classes such as positive or negative, working similarly to regular cross-entropy but simplified for the binary case, such as determining whether an email is spam or not.

**4.4.Optimization Algorithms**

Optimization algorithms are methods used to update neural network weights based on gradients calculated from the loss function. Stochastic Gradient Descent is a basic optimization algorithm that updates model weights in the opposite direction of the gradient, taking a single random sample or small batch of data at each update step, making it fast but convergence paths may be oscillatory. Adam combines advantages of other optimization algorithms by maintaining moving averages of both the gradient and squared gradient, providing adaptive learning rates for each parameter, and is one of the most popular algorithms because it works well with most problems without requiring significant hyperparameter tuning. AdamW is an improved version of Adam that adds weight decay separately from gradient calculation, improving generalization and reducing overfitting, and is preferred in many modern deep learning applications especially with large models.

**4.5.Fundamental Training Concepts**

**4.5.1.An epoch** is one complete cycle where the model passes through all training data. If you have 1000 images for training and complete processing them all once, this means you've completed one epoch, and typically the model needs many epochs to learn well, such as 50 or 100 epochs.

**4.5.2.Batch size** determines the number of examples processed together before updating model weights. For example, if you have 1000 images and batch size 32, the data will be divided into about 31 batches. Larger batches affect training stability and memory usage; small batches like 16 or 32 provide more frequent but noisier updates, while large batches like 128 or 256 provide more stable updates but require more memory.

**4.5.3. Learning rate** controls the step size the model takes when updating weights and is one of the most important hyperparameters. If the learning rate is too large like 0.1, the model may overshoot the optimal solution and oscillate, while if it's too small like 0.00001, training may take very long or get stuck in suboptimal local solutions. Common values range between 0.001 and 0.0001.

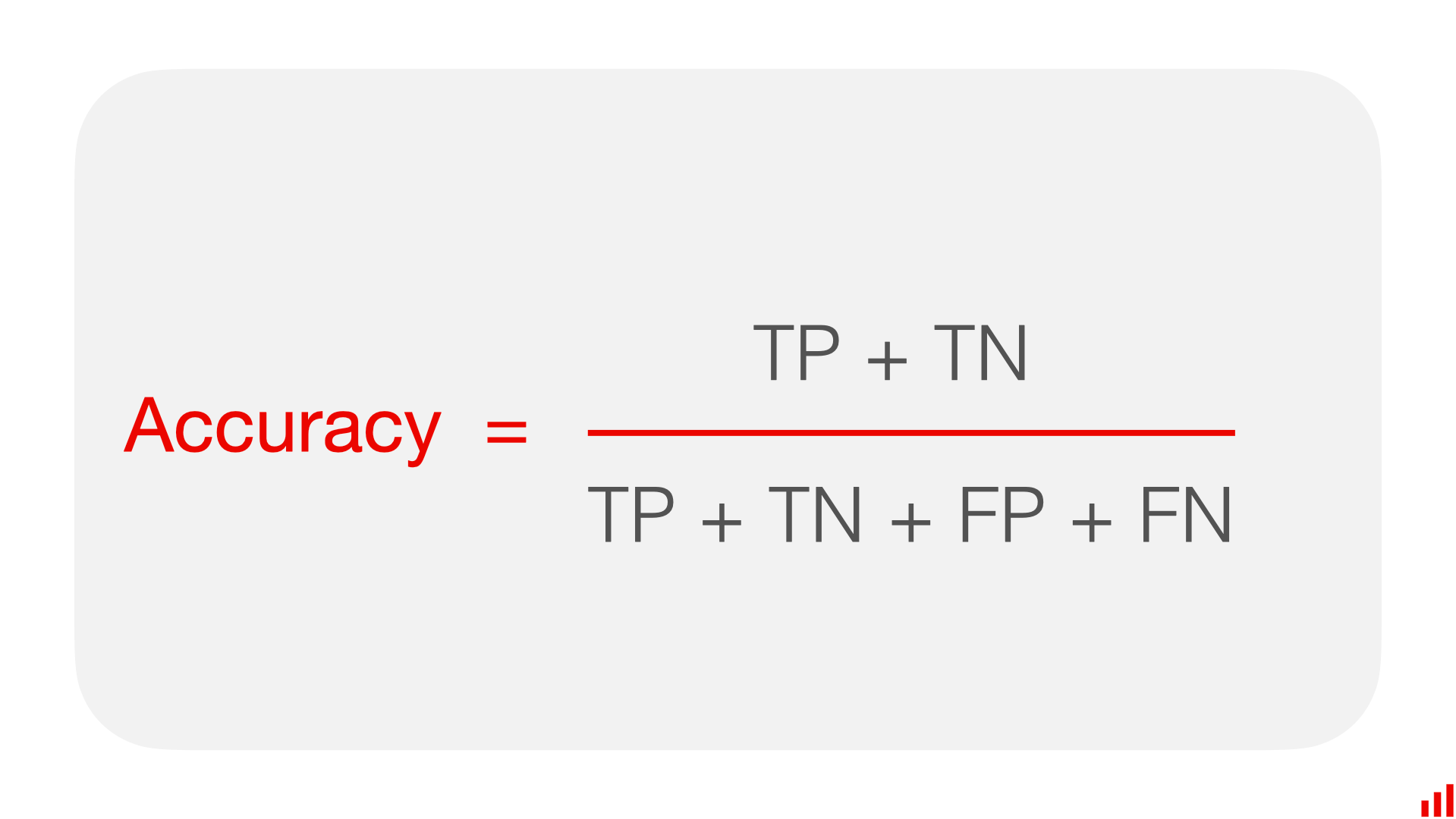
**4.5.4.Overfitting** occurs when the model learns the details of training data so well that it loses the ability to generalize to new data, like a student who memorizes answers to previous exam questions without understanding concepts and fails when facing new questions in different formats. Signs of overfitting include training accuracy being very high like 98% while validation accuracy is low like 70%, or training loss continuing to decrease while validation loss starts increasing. To address overfitting, techniques can be used such as increasing training data size, applying data augmentation, using dropout which randomly disables some neurons during training, applying weight decay which adds a penalty on large weights, using early stopping where training is halted when validation performance stops improving, or using simpler models with fewer parameters.

## **5. Model Performance Evaluation**

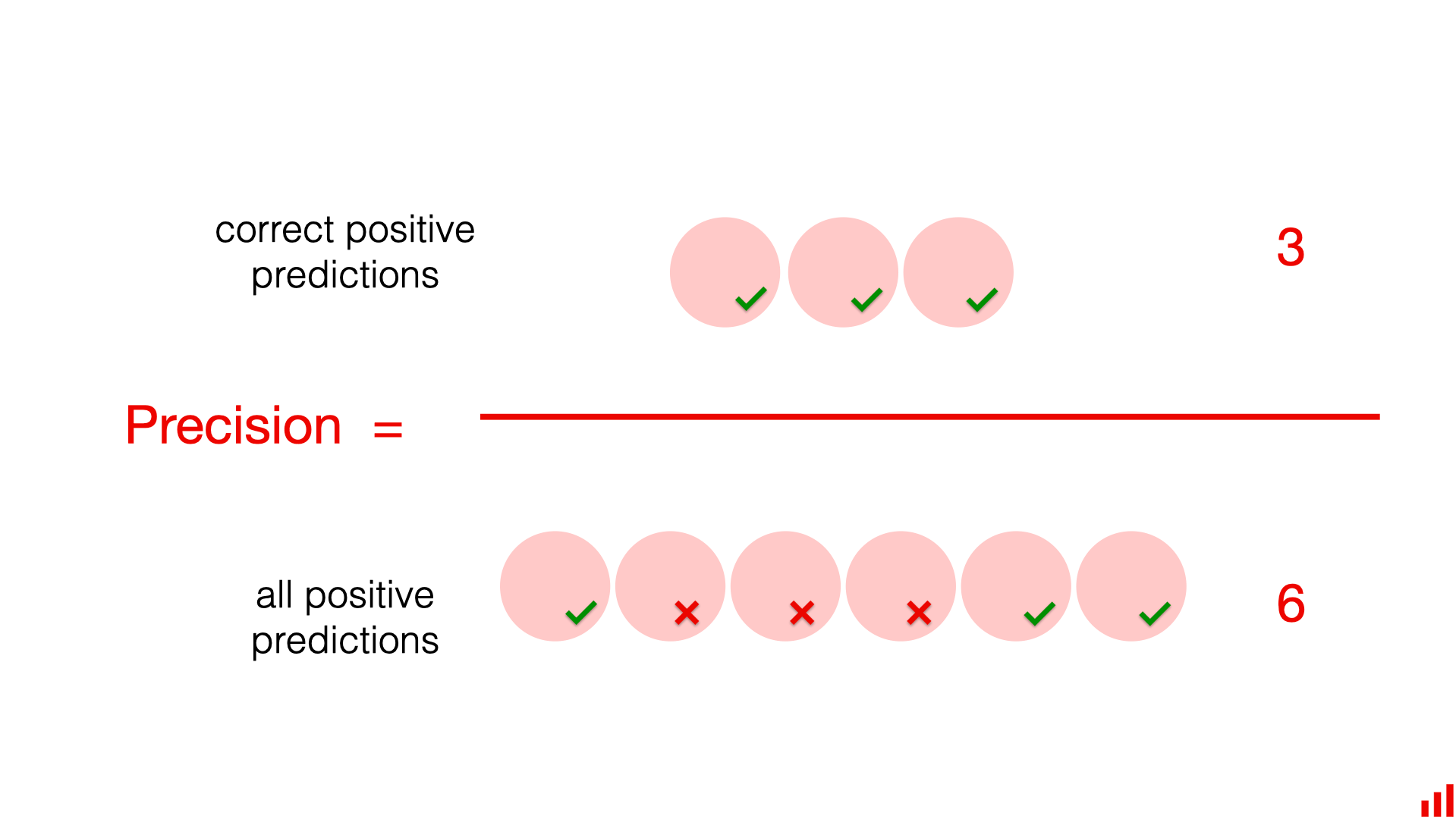
**5.1. Classification Metrics**

In classification tasks, we use several metrics to evaluate the model's ability to classify data correctly.

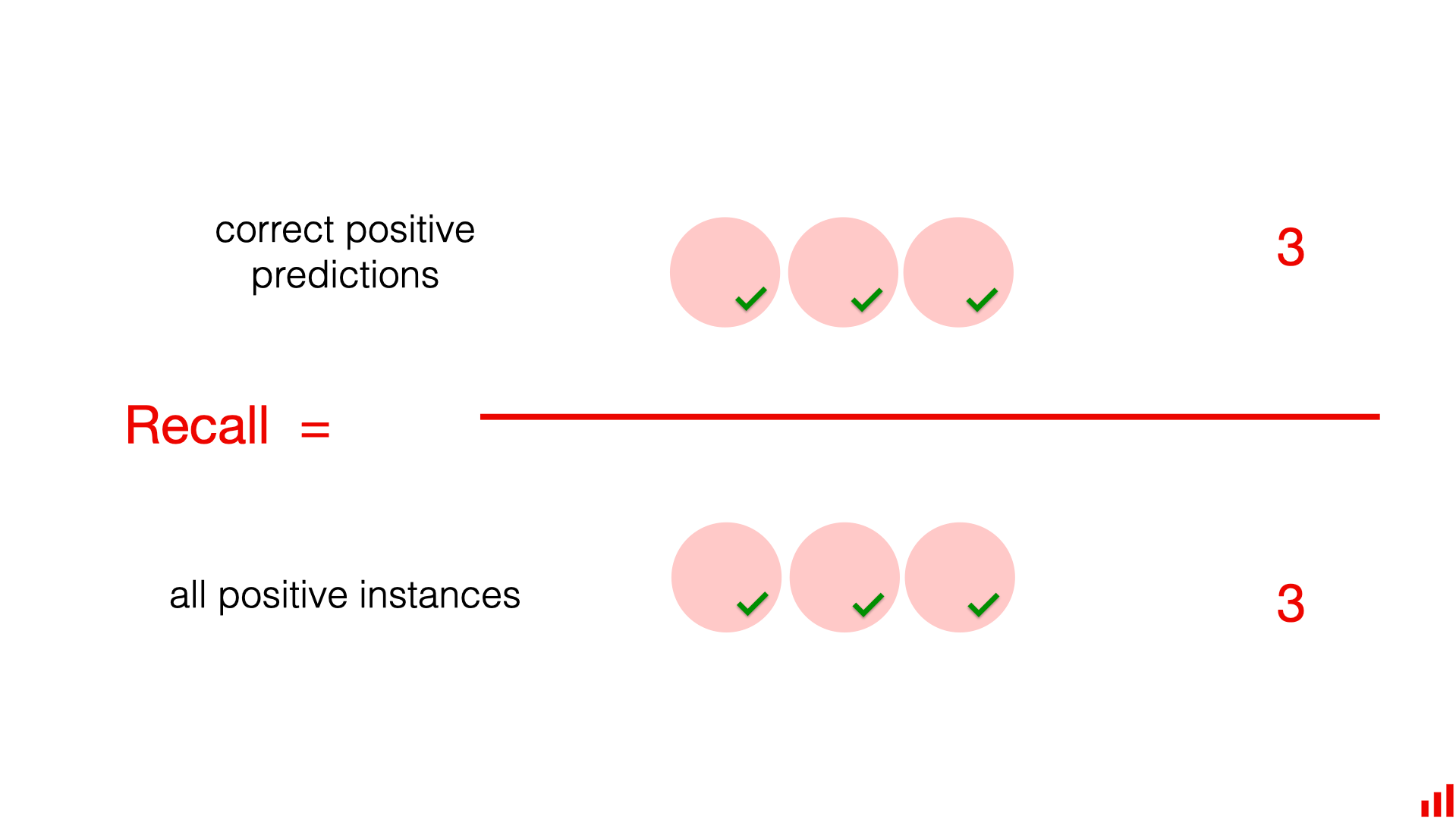
**5.1.1.Accuracy** measures the percentage of correct predictions out of total predictions, calculated by dividing the number of correct predictions by the total number of predictions. For example, if you have 100 images and the model correctly classified 85 of them, the accuracy is 85%. Although this metric is simple and easy to understand, it can be misleading in cases of imbalanced data. If you have 95 cat images and 5 dog images, and a model that classifies everything as a cat, you'll get 95% accuracy but the model is actually poor.



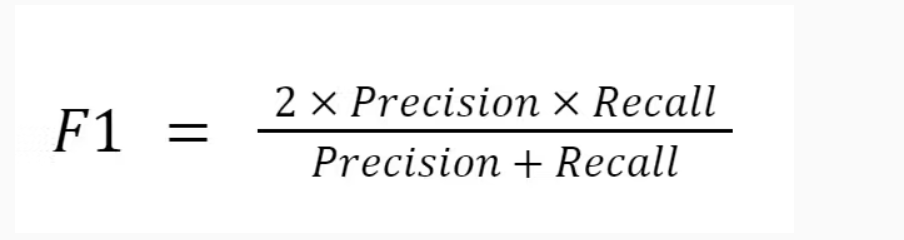
**5.1.2.Precision** measures among all cases the model predicted as positive, how many were actually positive, calculated by dividing true positives by the sum of true positives and false positives. For example, in a disease diagnosis system, if the model diagnosed 100 cases as sick, and 80 were actually sick while 20 were healthy, precision would be 80%. This metric is important when the cost of false positives is high, such as sending annoying alerts to users.



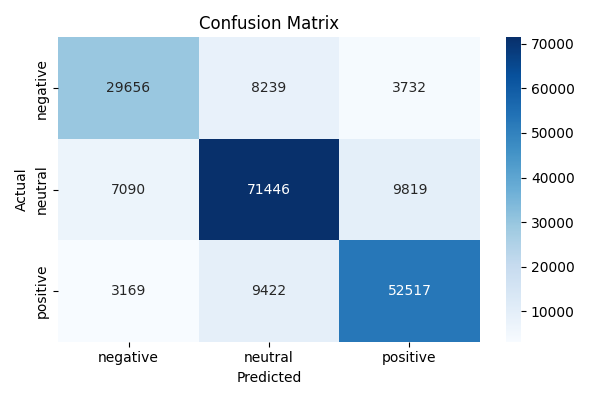
**5.1.3.Recall** or sensitivity measures among all actual positive cases, how many the model was able to detect, calculated by dividing true positives by the sum of true positives and false negatives. For example, if there are 100 actual patients and the model detected 75 of them, recall would be 75%. This metric is crucial when the cost of missing positive cases is high, such as failing to detect a serious disease or security threat.



**5.1.4.F1-score** is the harmonic mean between precision and recall, providing a balance between these two metrics, calculated using the formula: F1 = 2 × (Precision × Recall) / (Precision + Recall). For example, if precision is 80% and recall is 75%, the F1-score would be approximately 77.4%. This metric is particularly useful when you need a single metric that balances avoiding false positives and false negatives, and is widely used with imbalanced data.



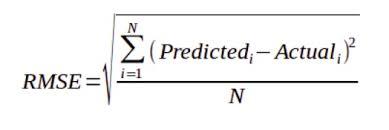
**5.1.5.Confusion matrix** is a table that shows classification model performance in detail, displaying the number of correct and incorrect predictions for each class. In binary classification, the matrix contains four elements: true positives which are cases the model predicted as positive and were actually positive, false positives which are cases the model predicted as positive but were actually negative, true negatives which are cases the model predicted as negative and were actually negative, and false negatives which are cases the model predicted as negative but were actually positive. For example, in a spam detection system, if you have 50 actual spam messages and 50 normal messages, and the model predicted 45 spam messages correctly and 5 incorrectly, and 40 normal messages correctly and 10 incorrectly, the confusion matrix clearly shows these numbers and helps understand where the model makes mistakes.



**5.2.**Regression Metrics****

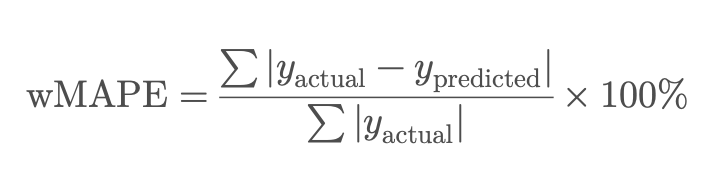
In regression tasks where we predict continuous values, we use different metrics to evaluate prediction accuracy.

**5.2.1.Root Mean Squared Error** measures the square root of the average squared differences between predicted and actual values, and is in the same unit as the original values making it easy to interpret. For example, if you're predicting house prices in dollars and the true value is 500,000 and the prediction is 480,000, this error contributes to calculating RMSE, and if the final RMSE is 25,000, this means the model's predictions deviate on average by about 25,000 from the true values. RMSE gives greater weight to large errors due to squaring, making it sensitive to outliers.



**5.2.2.Mean Absolute Error** calculates the average of absolute differences between predictions and actual values, and is less sensitive to outliers compared to RMSE because it doesn't square errors. For example, in the same house price example, if you have three predictions with errors of 20,000, 15,000, and 25,000, MAE would be the average of these absolute values which is 20,000, meaning the model errs on average by about 20,000 in either direction.

**5.2.3.Mean Absolute Percentage Error** measures the average percentage of errors, expressing error as a percentage of the actual value, making it useful for comparing performance across different datasets or value ranges. For example, if the actual value is 100 and the prediction is 90, the percentage error is 10%, and if the actual value is 1000 and the prediction is 900, the percentage error is also 10%, allowing fair comparison of error magnitude regardless of value scale. However, MAPE can be problematic when actual values are close to zero or equal zero as this leads to division by zero or very large values.



**5.2.4.Mean Squared Error** calculates the average of squared differences between predictions and actual values, similar to RMSE but without taking the square root, making its unit the square of the original values' unit. For example, if errors are 10, 20, and 15, MSE would be (100 + 400 + 225) / 3 = 241.67. MSE is commonly used as a loss function during training because its mathematical properties make the optimization process easier.

**5.2.5.R-squared coefficient** measures the proportion of variance in the data explained by the model, ranging between 0 and 1 where a value of 1 means the model perfectly explains all variance in the data, while a value of 0 means the model explains no variance and is no better than simply using the mean. For example, if R² equals 0.85 or 85%, this means the model explains 85% of variance in the data, which is considered very good performance, while R² of 0.40 indicates the model explains only 40% of variance and may need improvement. R² values can be negative in rare cases when model performance is worse than simply predicting the mean for all values, indicating a very poor model.

## SOURCE

[1]https://www.ibm.com/think/topics/artificial-intelligence

1. <https://www.britannica.com/technology/artificial-intelligence>

[3] https://www.ibm.com/think/topics/machine-learning

[4]<https://www.researchgate.net/publication/354306269_Study_On_Machine_Learning_Algorithms>

[5]<https://aws.amazon.com/compare/the-difference-between-artificial-intelligence-and-machine-learning/>

[6]https://www.researchgate.net/publication/350664352\_The\_Understanding\_of\_Deep\_Learning\_A\_Comprehensive\_Review

[7]https://www.ibm.com/think/topics/artificial-intelligence

[8]https://www.researchgate.net/publication/5847739\_Introduction\_to\_artificial\_neural\_networks

[9]https://www.ibm.com/think/topics/recurrent-neural-networks

[10]https://www.researchgate.net/publication/277603865\_A\_Critical\_Review\_of\_Recurrent\_Neural\_Networks\_for\_Sequence\_Learning

[11]https://www.sciencedirect.com/science/article/pii/S2212827121003796