TENSORFLOW:

TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow APIs are arranged hierarchically, with the high-level APIs built on the low-level APIs.In short it collects Data, it prepares Data, it trains a model and it evaluates the model.

Methods:

1.Tensorflow log() method

The module tensorflow.math provides support for many basic mathematical operations. Function tf.log() [alias tf.math.log] provides support for the natural logarithmic function in Tensorflow. It expects the input in form of complex numbers as $a+bi$ or floating point numbers.

Syntax: tf.log(x, name=None) or tf.math.log(x, name=None)

2.Tensorflow nn.tanh() method

The module tensorflow.nn provides support for many basic neural network operations.One of the many activation functions is the hyperbolic tangent function (also known as tanh) which is defined as tanh(x) = (e^z - e^{-z}) / (e^z + e^{-z}) .

Syntax: tf.nn.tanh(x, name=None) or tf.tanh(x, name=None)

3.Tensorflow exp() method

Function tf.exp() [alias tf.math.exp] provides support for the exponential function in Tensorflow. It expects the input in form of complex numbers as $a+bi$ or floating point numbers. The input type is tensor and if the input contains more than one element, an element-wise exponential value is computed, y=e^x.

Syntax: tf.exp(x, name=None) or tf.math.exp(x, name=None)

4.Tensorflow cosh() method

Function tf.cosh() [alias tf.math.cosh] provides support for the hyperbolic cosine function in Tensorflow. It expects the input in radian form. The input type is tensor and if the input contains more than one element, element-wise hyperbolic cosine is computed.

Syntax: tf.cosh(x, name=None) or tf.math.cosh(x, name=None)

5.Tensorflow tan() method

Function tf.tan() [alias tf.math.tan] provides support for the tangent function in Tensorflow. It expects the input in radian form. The input type is tensor and if the input contains more than one element, element-wise tangent is computed.

Syntax: tf.tan(x, name=None) or tf.math.tan(x, name=None)

6.Tensorflow acos() method

Function tf.acos() [alias tf.math.acos] provides support for the inverse cosine function in Tensorflow. It expects the input to be in the range [-1, 1] and gives the output in radian form. It returns nan if the input does not lie in the range [-1, 1]. The input type is tensor and if the input contains more than one element, element-wise inverse cosine is computed.

Syntax: tf.acos(x, name=None) or tf.math.acos(x, name=None)

7.Tensorflow atan() method

Function tf.atan() [alias tf.math.atan] provides support for the inverse tangent function in Tensorflow. It gives the output in radian form. The input type is tensor and if the input contains more than one element, element-wise inverse tangent is computed.

Syntax: tf.atan(x, name=None) or tf.math.atan(x, name=None)

8. Tensorflow sin() method

Function tf.sin() [alias tf.math.sin] provides support for the sine function in Tensorflow. It expects the input in radian form and the output is in the range [-1, 1]. The input type is tensor and if the input contains more than one element, element-wise sine is computed.

Syntax: tf.sin(x, name=None) or tf.math.sin(x, name=None)

9.Tensorflow reciprocal() method

Function tf.reciprocal() [alias tf.math.reciprocal] provides support to calculate the reciprocal of input in Tensorflow. It expects the input in form of complex numbers as $a+bi$, floating point numbers and integers. The input type is tensor and if the input contains more than one element, an element-wise reciprocal is computed, $y=1/x$.

Syntax: tf.reciprocal(x, name=None) or tf.math.reciprocal(x, name=None)

10.Tensorflow log1p() method

Function tf.log1p() [alias tf.math.log1p] provides support for the natural logarithmic function in Tensorflow. It expects the input in form of complex numbers as $a+bi$ or floating point numbers. The input type is tensor and if the input contains more than one element, an element-wise logarithm of 1+x is computed, y=log_e (1+x).

Syntax: tf.log1p(x, name=None) or tf.math.log1p(x, name=None)

11.Tensorflow logical_and() method

Function tf.logical_and() [alias tf.math.logical_and] provides support for the logical AND function in Tensorflow. It expects the input of bool type. The input types are tensor and if the tensors contains more than one element, an element-wise logical AND is computed, x AND y.

Syntax: tf.logical_and(x, y, name=None) or tf.math.logical_and(x, y, name=None)

12.Tensorflow logical_or() method

Function tf.logical_or() [alias tf.math.logical_or] provides support for the logical OR function in Tensorflow. It expects the input of bool type. The input types are tensor and if the tensors contains more than one element, an element-wise logical OR is computed, x OR y .

Syntax: tf.logical_or(x, y, name=None) or tf.math.logical_or(x, y, name=None)

13.Tensorflow logical_not() method

Function tf.logical_not() [alias tf.math.logical_not or tf.Tensor.__invert__] provides support for the logical NOT function in Tensorflow. It expects the input of bool type. The input type is tensor and if the input contains more than one element, an element-wise logical NOT is computed, NOT x .

Syntax: tf.logical_not(x, name=None) or tf.math.logical_not(x, name=None) or tf.Tensor.__invert__(x, name=None)

14.Tensorflow logical_xor() method

Function tf.logical_xor() [alias tf.math.logical_xor] provides support for the logical XOR function in Tensorflow. It expects the inputs of bool type. The input types are tensor and if the tensors contains more than one element, an element-wise logical XOR is computed, x XOR y = (x || y) &&!(x && y) .

Syntax: tf.logical_xor(x, y, name=None) or tf.math.logical_xor(x, y, name=None)

15.Tensorflow acosh() method

Function tf.acosh() [alias tf.math.acosh] provides support for the inverse hyperbolic cosine function in Tensorflow. It expects the input in the range [1, ∞) and returns nan for any input outside this range. The input type is tensor and if the input contains more than one element, element-wise inverse hyperbolic cosine is computed.

Syntax: tf.acosh(x, name=None) or tf.math.acosh(x, name=None)

16.Tensorflow atanh() method

Function tf.atanh() [alias tf.math.atanh] provides support for the inverse hyperbolic tangent function in Tensorflow. Its domain is in the range [-1, 1] and it returns nan for any input outside this range. The input type is tensor and if the input contains more than one element, element-wise inverse hyperbolic tangent is computed.

Syntax: tf.atanh(x, name=None) or tf.math.atanh(x, name=None)

17.Tensorflow acosh() method

Function tf.acosh() [alias tf.math.acosh] provides support for the inverse hyperbolic cosine function in Tensorflow. It expects the input in the range [1, ∞) and returns nan for any input outside this range. The input type is tensor and if the input contains more than one element, element-wise inverse hyperbolic cosine is computed.

Syntax: tf.acosh(x, name=None) or tf.math.acosh(x, name=None)

18.Tensorflow abs() method

Function tf.abs() [alias tf.math.abs] provides support for the absolute function in Tensorflow. It expects the input in form of complex numbers as a+bi or floating point numbers. The input type is tensor and if the input contains more than one element, an element-wise absolute value is computed.

Syntax: tf.abs(x, name=None) or tf.math.abs(x, name=None)

KERAS:

Keras is an open-source deep learning framework that provides a high-level API for building and training neural networks. It is designed to be user-friendly, modular, and extensible. Keras was initially developed as a standalone library but was later integrated into TensorFlow as the official high-level API.

Methods in Keras:

Model Creation:

keras.models.Sequential: Creates a sequential model where layers are stacked sequentially.

keras.models.Model: Allows the creation of complex models with shared layers or multiple inputs/outputs.

Layers:

keras.layers.Dense: Fully connected (dense) layer.

keras.layers.Conv2D: 2D convolutional layer.

keras.layers.MaxPooling2D: 2D max pooling layer.

keras.layers.Dropout: Dropout layer for regularization.

keras.layers.Embedding: Embedding layer for handling text or categorical data.

Activation Functions:

keras.activations.relu: Rectified Linear Unit (ReLU) activation function.

keras.activations.sigmoid: Sigmoid activation function.

keras.activations.softmax: Softmax activation function.

keras.activations.tanh: Hyperbolic tangent activation function.

Optimizers:

keras.optimizers.SGD: Stochastic Gradient Descent optimizer.

keras.optimizers.Adam: Adam optimizer.

keras.optimizers.RMSprop: RMSprop optimizer.

Loss Functions:

keras.losses.mean_squared_error: Mean Squared Error (MSE) loss.

keras.losses.categorical_crossentropy: Categorical Crossentropy loss.

keras.losses.binary_crossentropy: Binary Crossentropy loss.

Metrics:

keras.metrics.accuracy: Accuracy metric.

keras.metrics.precision: Precision metric.

keras.metrics.recall: Recall metric.

keras.metrics.mean_squared_error: Mean Squared Error metric.

Training:

model.compile: Configures the model for training, specifying the optimizer, loss function, and metrics.

model.fit: Trains the model on training data.

model.evaluate: Evaluates the model on test data.

model.predict: Generates predictions for new data.

Callbacks:

keras.callbacks.ModelCheckpoint: Saves the model during training based on specific conditions.

keras.callbacks.EarlyStopping: Stops training early based on a monitored metric.

keras.callbacks.TensorBoard: Enables visualization and monitoring of training progress using TensorBoard.