Deep Learning

TENSORFLOW:-

TensorFlow: Tensorflow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google’s Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as welL.

Methods in Tensorflow:-

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. The input type is tensor and if the input contains more than one element, an element-wise logarithm is computed, y=loge x.

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

Parameters: x: A Tensor of type bfloat16, half, float32, float64, complex64 or complex128. name (optional): The name for the operation.

Return type: A Tensor with the same size and type as that of x.

2.Tensorflow nn.tanh():- 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}).The function tf.nn.tanh() [alias tf.tanh] provides support for the hyperbolic tangent function in Tensorflow.

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

Parameters: x: A tensor of any of the following types: float16, float32, double, complex64, or complex128.

name (optional): The name for the operation.

Return : A tensor with the same type as that of x.

3.Tensorflow exp() method:- The module tensorflow.math provides support for many basic mathematical operations. 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)

Parameters: x: A Tensor of type bfloat16, half, float32, float64, complex64 or complex128. name (optional): The name for the operation.

Return type: A Tensor with the same size and type as that of x.

4.Tensorflow cosh() method:- The module tensorflow.math provides support for many basic mathematical operations. 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)

Parameters: x: A tensor of any of the following types: float16, float32, float64, complex64, or complex128. name (optional): The name for the operation.

Return type: A tensor with the same type as that of x.

5. Tensorflow exp() method:- The module tensorflow.math provides support for many basic mathematical operations. 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)

Parameters:

x: A Tensor of type bfloat16, half, float32, float64, complex64 or complex128. name (optional): The name for the operation.

Return type: A Tensor with the same size and type as that of x.

6. Tensorflow acos() method:- The module tensorflow.math provides support for many basic mathematical operations. Function tf.acos() [alias tf.math.acos] provides support for the inverse cosine function in Tensorflow.

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

Parameters x: A tensor of any of the following types: bfloat16, half, float32, float64, int32, int64, complex64, or complex128. name (optional): The name for the operation.

7. Tensorflow atan() method:- The module tensorflow.math provides support for many basic mathematical operations. Function tf.atan() [alias tf.math.atan] provides support for the inverse tangent function in Tensorflow. It gives the output in radian form.

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

Parameters:

x: A tensor of any of the following types: bfloat16, half, float32, float64, int32, int64, complex64, or complex128. name (optional): The name for the operation.

Return type: A tensor with the same type as that of x.

8. Tensorflow sin() method:- The module tensorflow.math provides support for many basic mathematical operations. 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].

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

Parameters: x: A tensor of any of the following types: float16, float32, float64, complex64, or complex128. name (optional): The name for the operation.

Return type: A tensor with the same type as that of x.

9. Tensorflow reciprocal() method:- The module tensorflow.math provides support for many basic mathematical operations. 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)

Parameters: x: A Tensor of type bfloat16, half, float32, float64, int32, int64, complex64 or complex128.

name (optional): The name for the operation.

10. Tensorflow log1p() method:- The module tensorflow.math provides support for many basic mathematical operations. 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=loge (1+x).

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

Parameters: x: A Tensor of type bfloat16, half, float32, float64, complex64 or complex128. name (optional): The name for the operation.

Return type: A Tensor with the same size and type as that of x.

11. Tensorflow logical_and() method:- The module tensorflow.math provides support for many basic logical operations. 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)

Parameters: x: A Tensor of type bool. y: A Tensor of type bool. name (optional): The name for the operation.

12. Tensorflow logical_xor() method:- The module tensorflow.math provides support for many basic logical operations. 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.

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

Parameters:

x: A Tensor of type bool. y: A Tensor of type bool. name (optional): The name for the operation.

13. Tensorflow logical_or() method:- The module tensorflow.math provides support for many basic mathematical operations. 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)

Parameters:

x: A Tensor of type bool. y: A Tensor of type bool. name (optional): The name for the operation.

14. Tensorflow logical_not() method:- The module tensorflow.math provides support for many basic logical operations. 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)

Parameters:

x: A Tensor of type bool. name (optional): The name for the operation.

16. Tensorflow abs() method:- The module tensorflow.math provides support for many basic mathematical operations. 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.