Vidyavardhini's College of Engineering & Technology

Department of Computer Science and Engineering (Data Science)

Academic Year 2023-24

Subject: CSC701 – Deep Learning

Name:	DIPTI R. SHARNAGAT
Roll No:	55
Class:	BE/CSE(DS)
EXPERIMENT NO:	01
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Keras:

- **Intro:-**
  Keras is an open-source high-level Neural Network library, which is written in Python and is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

- **Methods:-**
  keras.models.Sequential(): This function creates a linear stack of layers, which is the most common type of model in Keras. It allows you to add layers one by one in a sequential manner.
  model.add(): This method is used to add layers to the model created with Sequential(). You can add various types of layers such as Dense (fully connected), Conv2D (2D convolutional), LSTM (Long Short-Term Memory), etc.
  model.compile(): This method is used to configure the learning process of the model. You need to specify the optimizer, loss function, and optionally, metrics to evaluate during training.
  model.fit(): This method is used to train the model on a given dataset. It takes input data and target labels, and performs the training using the specified loss function and optimizer. You can also specify the number of epochs (iterations over the entire dataset) and batch size.
  model.predict(): This method is used to make predictions on new data using the trained model. It takes input data as input and returns the predicted outputs.
  model.evaluate(): This method is used to evaluate the performance of the model on a test dataset. It computes the specified metrics (e.g., accuracy, loss) on the given test data.
  model.summary(): This method provides a summary of the model's architecture, showing the number of parameters and the shape of each layer's output.
  model.save(): This method is used to save the model's architecture, weights, and

optimizer state to a file. It allows you to save and load the model for later use or deployment.

Tensorflow:-

- **Intro:-**

  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! Let us first try to understand what the word TensorFlow actually mean! TensorFlow is basically a software library for numerical computation using data flow graphs where:

  - **nodes** in the graph represent mathematical operations.
  - **edges** in the graph represent the multidimensional data arrays (called tensors) communicated between them. (Please note that tensor is the central unit of data in TensorFlow).

- **Methods and Operations**

  1. **tf.constant():** This method creates a constant tensor with a specified value.
  2. **tf.Variable()**: This method creates a variable tensor that can be updated and trained during model training.
  3. **tf.placeholder():** This method creates a placeholder tensor that acts as a placeholder for input data during the execution of a computational graph.
  4. **tf.add():** This operation adds two tensors element-wise.
  5. **tf.matmul():** This operation performs matrix multiplication between two tensors.
  6. **tf.reduce_mean():** This operation computes the mean value of a tensor along a specified axis.

7. **tf.argmax():** This operation returns the indices of the maximum values along a specified axis of a tensor.
8. **tf.nn.softmax():** This operation computes the softmax activation function, which normalizes the values of a tensor into a probability distribution.
9. **tf.layers.dense():** This method creates a fully connected (dense) layer with a specified number of units and activation function.
10. **tf.losses.mean_squared_error():** This method computes the mean squared error loss between predicted and target values.
11. **tf.train.GradientDescentOptimizer():** This method creates a gradient descent optimizer for updating the variables of a model based on computed gradients.
12. **tf.train.AdamOptimizer():** This method creates an Adam optimizer, which is a popular optimization algorithm for deep learning.
13. **tf.train.Saver():** This method creates a saver object to save and restore the variables of a model during training and inference.
14. **tf.Session():** This class represents a TensorFlow session that encapsulates the execution environment for running computational graphs.
15. **tf.global_variables_initializer():** This operation initializes all the variables in the computational graph.
16. **tf.trainable_variables():** This function returns a list of all trainable variables in the computational graph.
17. **tf.keras.Model():** This class provides an API for building and training models using the Keras high-level API within TensorFlow.
18. **tf.data.Dataset()**: This class provides a way to represent and manipulate input data for training models, allowing for efficient data loading and preprocessing.