TensorFlow:

Tensors and Operations:

tf.constant: Create constant tensors.

tf.Variable: Create mutable variables used for model parameters.

tf.add, tf.subtract, tf.multiply, etc.: Element-wise arithmetic operations.

tf.matmul: Matrix multiplication.

tf.nn: Module containing various neural network operations like activation functions, pooling, normalization, etc.

tf.reduce_sum, tf.reduce_mean, etc.: Functions for reducing tensors along specific dimensions.

Automatic Differentiation:

tf.GradientTape: A context manager for automatic differentiation to compute gradients with respect to variables.

Optimizers:

tf.optimizers: Module containing various optimization algorithms like SGD, Adam, RMSprop, etc.

Layers and Models:

tf.keras.layers: High-level API to create various types of neural network layers (e.g., Dense, Conv2D, LSTM).

tf.keras.Model: High-level API to define custom models by subclassing and assembling layers.

Datasets and Data Pipeline:

tf.data.Dataset: Efficient API for managing large datasets, enabling operations like batching, shuffling, prefetching, etc.

tf.data.experimental: Module containing additional experimental functionalities for datasets.

Model Training and Evaluation:

tf.keras.Model.compile: Method to configure the model for training, specifying loss function, optimizer, and metrics.

tf.keras.Model.fit: Method to train the model on the provided data.

tf.keras.Model.evaluate: Method to evaluate the model's performance on a dataset.

tf.keras.Model.predict: Method to obtain predictions from the model.

Save and Restore Models:

tf.saved_model: Save and load models for deployment using the SavedModel format.

GPU and TPU Support:

TensorFlow can utilize GPUs and TPUs for accelerated computations.

Distributed Training:

TensorFlow supports distributed training across multiple devices and machines.

TensorBoard:

A visualization tool for monitoring and visualizing TensorFlow graphs and training progress.

Keras:

Sequential API:

keras.models.Sequential: The Sequential class allows you to create a linear stack of layers, where each layer has one input tensor and one output tensor. It's suitable for building feedforward neural networks.

Functional API:

The Functional API allows for more complex models with multiple inputs and outputs and includes functionalities like shared layers and model branching. It provides greater flexibility in model architecture compared to the Sequential API.

Layers:

keras.layers: Keras provides various built-in layers such as Dense (fully connected), Conv2D (2D convolution), LSTM (Long Short-Term Memory), Dropout, BatchNormalization, etc. These layers can be added to the model using the Sequential or Functional API.

Model Compilation:

model.compile: Method to specify the loss function, optimizer, and metrics used for training the model.

Model Training:

model.fit: Method to train the model on the training data using specified loss, optimizer, and metrics. It allows for specifying batch size, number of epochs, and validation data.

Model Evaluation:

model.evaluate: Method to evaluate the performance of the trained model on a given dataset.

Model Prediction:

model.predict: Method to generate predictions from the trained model.

Callbacks:

keras.callbacks: Provides various callback functions that can be used during model training. Examples include ModelCheckpoint (saving the best model), EarlyStopping (stopping training based on validation performance), etc.

Saving and Loading Models:

model.save: Method to save the model and its weights to disk.

keras.models.load_model: Function to load a saved model from disk.

Pre-Trained Models and Transfer Learning:

keras.applications: Module containing pre-trained models (e.g., VGG16, ResNet, etc.) that can be used for transfer learning or fine-tuning on specific tasks.

Custom Layers and Models:

Keras allows you to define custom layers and models by subclassing existing layer classes or the Model class.

Text Preprocessing:

keras.preprocessing.text: Module containing utilities for text data preprocessing, including tokenization and sequence padding.

Image Preprocessing:

keras.preprocessing.image: Module with utilities for image data preprocessing, augmentation, and loading.

Callbacks:

keras.callbacks: Module containing various built-in callbacks for monitoring training progress and taking actions during training.