Core TensorFlow (tf module)

1. **Tensor Creation and Operations**
    a. tf.constant(value, dtype=None, shape=None, name='Const'): Creates a constant tensor.
    b. tf.Variable(initial_value, trainable=True, name='Variable'): Creates a variable tensor.
    c. tf.add(x, y, name=None): Adds two tensors element-wise.
    d. tf.matmul(a, b, transpose_a=False, transpose_b=False, adjoint_a=False, adjoint_b=False, ...): Performs matrix multiplication of tensors.
    e. tf.reduce_mean(input_tensor, axis=None, keepdims=False, name=None): Computes the mean of elements across dimensions of a tensor.
2. **Control Flow**
    a. tf.cond(pred, true_fn=None, false_fn=None, name=None): Conditionally executes operations.
    b. tf.while_loop(cond, body, loop_vars, shape_invariants=None, parallel_iterations=10, ...): Constructs a loop that executes a body function while a condition is true.
    c. tf.control_dependencies(control_inputs): Returns a context manager that specifies control dependencies.
3. **Session Management**
    a. tf.Session(config=None): Creates a new TensorFlow session.
    b. tf.Session.run(fetches, feed_dict=None): Executes operations and evaluates tensors in a session.
4. **Gradient Computation**
    a. tf.GradientTape(persistent=False, watch_accessed_variables=True): Records operations for automatic differentiation.
    b. tf.gradients(ys, xs, grad_ys=None, name='gradients', colocate_gradients_with_ops=False, ...): Computes the gradients of ys with respect to xs.
5. **Random Tensors**
    a. tf.random.uniform(shape, minval=0, maxval=None, dtype=tf.float32, seed=None, name=None): Generates a tensor with values uniformly distributed in a specified range.
    b. tf.random.normal(shape, mean=0.0, stddev=1.0, dtype=tf.float32, seed=None, name=None): Generates a tensor with values from a normal distribution.
6. **Serialization and IO**
    a. tf.io.write_graph(graph_or_graph_def, logdir, name, as_text=True): Writes a graph proto to a file.
    b. tf.saved_model.save(obj, export_dir, signatures=None): Saves a model to TensorFlow SavedModel format.

Keras API (tf.keras module)

1. **Model Creation and Compilation**
    a. tf.keras.Sequential(): Initializes a linear stack of layers for a sequential model.
    b. tf.keras.Model(): Allows defining a more flexible architecture where layers can be connected in arbitrary ways.
    c. tf.keras.layers.Layer(): Base layer class for Keras models.
    d. tf.keras.Input(): Defines input tensor(s) for model.
2. **Layer Creation and Configuration**
    a. tf.keras.layers.Dense(): Fully connected layer.
    b. tf.keras.layers.Conv2D(): 2D convolutional layer.
    c. tf.keras.layers.Dropout(): Applies Dropout to the input.
    d. tf.keras.layers.BatchNormalization(): Batch normalization layer.
3. **Losses and Optimizers**
    a. tf.keras.losses.BinaryCrossentropy(): Computes the cross-entropy loss between true labels and predicted labels.
    b. tf.keras.optimizers.Adam(): Optimizer that implements the Adam algorithm.
4. **Callbacks**
    a. tf.keras.callbacks.ModelCheckpoint(): Callback to save the Keras model or model weights at some frequency.
5. **Model Training and Evaluation**
    a. tf.keras.Model.compile(): Configures the model for training.
    b. tf.keras.Model.fit(): Trains the model for a fixed number of epochs.
    c. tf.keras.Model.evaluate(): Evaluates the model on a dataset.
6. **Saving and Loading Models**
    a. tf.keras.models.save_model(): Saves a model to TensorFlow SavedModel format.
    b. tf.keras.models.load_model(): Loads a model saved with save_model().

Utility Functions

1. **Data Preprocessing**
    a. tf.keras.utils.to_categorical(y, num_classes=None, dtype='float32'): Converts a class vector (integers) to binary class matrix (one-hot encoding).
    b. tf.keras.utils.normalize(x, axis=-1, order=2): Normalizes a tensor along a specified axis.
2. **Learning Rate Scheduling**
    a. tf.keras.callbacks.LearningRateScheduler(schedule, verbose=0): Callback that schedules learning rate changes during training.