#_ the Tensorflow Foundational List

1. Basics:

- `import tensorflow as tf`: Import TensorFlow module.
- `tf.__version__`: Check TensorFlow version.
- `tf.constant(value)`: Creates a constant tensor.
- `tf.Variable(value)`: Creates a variable tensor.
- `tf.zeros(shape)`: Create a tensor with all zeros.
- `tf.ones(shape)`: Create a tensor with all ones.

2. Operations:

- `tf.add(x, y)`: Add two tensors.
- `tf.subtract(x, y)`: Subtract two tensors.
- `tf.multiply(x, y)`: Multiply two tensors.
- `tf.divide(x, y)`: Divide two tensors.
- `tf.pow(x, y)`: X raised to power Y.
- `tf.exp(x)`: Exponential of X.
- `tf.sqrt(x)`: Square root of X.

3. Shapes & Reshaping:

- `tf.shape(tensor)`: Get shape of a tensor.
- `tf.reshape(tensor, shape)`: Reshape a tensor.
- `tf.transpose(tensor)`: Transpose matrix.
- `tf.squeeze(tensor)`: Removes dimensions of size 1.
- `tf.expand_dims(tensor, axis)`: Inserts a new axis.

4. Data & I/O:

- `tf.data.Dataset.from_tensor_slices(data)`: Create a dataset from tensors.
- `dataset.batch(size)`: Batch the dataset.
- `dataset.shuffle(buffer_size)`: Shuffle the dataset.
- `tf.data.experimental.make_csv_dataset()`: Load CSV data.

By: Waleed Mousa

5. Neural Network Components:

- `tf.keras.layers.Dense(units, activation)`: Fully connected layer.
- `tf.keras.layers.Conv2D(filters, kernel_size)`: 2D convolutional layer.
- `tf.keras.layers.MaxPooling2D(pool_size)`: Max pooling layer.
- `tf.keras.layers.Dropout(rate)`: Dropout layer.
- `tf.keras.layers.Flatten()`: Flatten layer.

6. Model Management:

- `model.compile(optimizer, loss, metrics)`: Compile the model.
- `model.fit(data, labels, epochs)`: Train the model.
- `model.evaluate(data, labels)`: Evaluate the model's performance.
- `model.save(filepath)`: Save a model to a file.
- `tf.keras.models.load_model(filepath)`: Load a model from a file.

7. Optimizers, Losses, and Metrics:

- `tf.keras.optimizers.Adam(learning_rate)`: Adam optimizer.
- `tf.keras.losses.SparseCategoricalCrossentropy()`: Cross-entropy loss.
- `tf.keras.metrics.Accuracy()`: Accuracy metric.

8. Advanced Layers and Operations:

- `tf.keras.layers.BatchNormalization()`: Batch normalization layer.
- `tf.keras.layers.Embedding(input_dim, output_dim)`: Embedding layer.
- `tf.image.resize(images, size)`: Resize images.
- `tf.signal.stft(signals, frame_length, frame_step)`: Short-time Fourier transform.

9. Gradient & Training:

- `with tf.GradientTape() as tape: ...`: Record operations for automatic differentiation.
- `tape.gradient(target, sources)`: Compute the gradient of target w.r.t sources.

By: Waleed Mousa

- `optimizer.apply_gradients(zip(gradients, variables))`: Apply gradients to variables.

10. Custom & Advanced Models:

- Define a custom layer: subclass `tf.keras.layers.Layer`.
- Define a custom model: subclass `tf.keras.Model`.
- `@tf.function`: Decorator to compile function into a TensorFlow graph for faster execution.

11. Debugging & Inspection:

- `tf.debugging.assert_shapes()`: Assertion for tensor shapes.
- `tf.print(tensor)`: Print the value of a tensor.

12. Distributed Training:

- `tf.distribute.MirroredStrategy()`: Synchronous training on multiple GPUs.

13. Transfer Learning:

- `tf.keras.applications.MobileNetV2()`: Pre-trained MobileNetV2 model.
- `base_model.trainable = False`: Freeze the base model for transfer learning.

14. Callbacks:

- `tf.keras.callbacks.TensorBoard(log_dir)`: TensorBoard visualization callback.
- `tf.keras.callbacks.EarlyStopping(patience)`: Stop training when a monitored metric has stopped improving.

15. Regularization:

- `tf.keras.regularizers.l1(l=0.01)`: L1 regularization.
- `tf.keras.regularizers.l2(l=0.01)`: L2 regularization.

By: Waleed Mousa

16. Functional API:

- `input = tf.keras.Input(shape)`: Define model input.
- `x = layer(input)`: Connect a layer to another.

17. Eager Execution:

- `tf.executing_eagerly()`: Check if running in eager mode.

18. Advanced Neural Network Components:

- `tf.keras.layers.GRU(units)`: Gated Recurrent Unit.
- `tf.keras.layers.LSTM(units)`: Long Short-Term Memory layer.
- `tf.keras.layers.Bidirectional(layer)`: Bidirectional wrapper for RNNs.

19. Mobile & Edge Deployment:

- TensorFlow Lite: Convert models for mobile and edge devices.
- `tf.lite.TFLiteConverter.from_keras_model(model)`: Converter to create TensorFlow Lite model.

20. Extended Backend Operations:

- `tf.math.sin(x)`: Sin function.
- `tf.linalg.inv(matrix)`: Inverse of a matrix.
- `tf.reduce_mean(tensor)`: Compute the mean of tensor elements.

21. Performance & Optimization:

- `tf.function`: Convert functions to high-performance TensorFlow graphs.
- `tf.TensorArray()`: TensorFlow equivalent of a list for use in tf.function.

22. Generative Adversarial Networks (GAN):

- `tf.keras.layers.LeakyReLU()`: Leaky Rectified Linear Unit activation function often used in GANs.

By: Waleed Mousa

23. Object Detection & Image Segmentation:

- TensorFlow Object Detection API: Set of utilities to perform object detection.
- `tf.image.draw_bounding_boxes()`: Draw bounding boxes on batch of images.

24. Sequence to Sequence & Attention Mechanisms:

- `tf.keras.layers.Attention()`: Dot-product attention mechanism layer.
- `tf.keras.layers.AdditiveAttention()`: Additive attention mechanism layer.

25. Transformers & NLP:

- `tf.keras.layers.MultiHeadAttention()`: Multi-head attention layer, used in transformers.
- Tokenization & Sequence Padding: Convert text to numerical data.

26. Time Series & Forecasting:

- `tf.keras.layers.Conv1D(filters, kernel_size)`: 1D convolutional layer for sequences.
- `tf.data.Dataset.window(size, shift, stride)`: Create a sliding window over a dataset.

27. Reinforcement Learning:

- TensorFlow Agents: Library for reinforcement learning in TensorFlow.

28. Model Interpretability:

- `tf.keras.layers.Activation(activation)`: Applies an activation function.
- Feature importance: Identifying important input features to the model predictions.

29. TensorFlow Hub:

- Pre-trained models and reusable parts for TensorFlow.
- `hub.KerasLayer("URL")`: Use a model from TensorFlow Hub in Keras.

30. TensorFlow Extended (TFX):

- End-to-end platform for deploying production machine learning pipelines.
- Data validation, transformation, and model serving.

31. TensorFlow.js:

- `tfjs.converters.save_keras_model(model, path)`: Save Keras model for TensorFlow.js.
- Browser-based machine learning.

32. Miscellaneous:

- TensorFlow Datasets: Library to load and preprocess datasets.
- `tf.autograph`: Convert Python constructs into their TensorFlow equivalents.
- `tf.data.experimental.cardinality(dataset)`: Determine the cardinality of a dataset.
- `tf.saved_model.load(path)`: Load a SavedModel.
- `tf.keras.utils.get_file()`: Downloads a file from a URL.
- `tf.keras.utils.plot_model(model)`: Plots a model as a graph.
- TensorFlow Serving: Library developed by Google to serve TensorFlow models.
- TensorFlow Graphics: Computer graphics and 3D vision library.
- TensorFlow Quantum: Library for quantum machine learning.