TensorFlow 2.0 Beta is available **Learn more** (/beta/)

TensorFlow Guide

The documents in this unit dive into the details of how TensorFlow works. The units are as follows:

High Level APIs

- Keras (https://www.tensorflow.org/guide/keras), TensorFlow's high-level API for building and training deep learning models.
- Eager Execution (https://www.tensorflow.org/guide/eager), an API for writing TensorFlow code imperatively, like you would use Numpy.
- Importing Data (https://www.tensorflow.org/guide/datasets), easy input pipelines to bring your data into your TensorFlow program.
- Estimators (https://www.tensorflow.org/guide/estimators), a high-level API that provides fully-packaged models ready for large-scale training and production.

Estimators

- Premade Estimators (https://www.tensorflow.org/guide/premade_estimators), the basics of premade Estimators.

- Checkpoints (https://www.tensorflow.org/guide/checkpoints), save training progress and resume where you left off.
- Feature Columns (https://www.tensorflow.org/guide/feature_columns), handle a variety of input data types without changes to the model.
- Datasets for Estimators (https://www.tensorflow.org/guide/datasets_for_estimators) use tf.data (https://www.tensorflow.org/api_docs/python/tf/data) to input data

, use **tf.data** (https://www.tensorflow.org/api_docs/python/tf/data) to input data.

- Creating Custom Estimators (https://www.tensorflow.org/guide/custom_estimators) , write your own Estimator.

Accelerators

- Using GPUs (https://www.tensorflow.org/guide/using_gpu) explains how TensorFlow assigns operations to devices and how you can change the arrangement manually.
- Using TPUs (https://www.tensorflow.org/guide/using_tpu) explains how to modify `Estimator` programs to run on a TPU.

Low Level APIs

- Introduction (https://www.tensorflow.org/guide/low_level_intro), which introduces the basics of how you can use TensorFlow outside of the high Level APIs.
- Tensors (https://www.tensorflow.org/guide/tensors), which explains how to create, manipulate, and access Tensors--the fundamental object in TensorFlow.
- Variables (https://www.tensorflow.org/guide/variables), which details how to represent shared, persistent state in your program.
- Graphs and Sessions (https://www.tensorflow.org/guide/graphs), which explains:
  - dataflow graphs, which are TensorFlow's representation of computations as dependencies between operations.
  - sessions, which are TensorFlow's mechanism for running dataflow graphs across one or more local or remote devices. If you are programming with the low-level TensorFlow API, this unit is essential. If you are programming with a high-level TensorFlow API such as Estimators or Keras, the high-level API creates and manages graphs and sessions for you, but understanding graphs and sessions can still be helpful.
- Save and Restore (https://www.tensorflow.org/guide/saved_model), which explains

how to save and restore variables and models.

- Ragged Tensors (https://www.tensorflow.org/guide/ragged_tensors), which explains how to use Ragged Tensors to encode nested variable-length lists.

ML Concepts

- Embeddings (https://www.tensorflow.org/guide/embedding), which introduces the concept of embeddings, provides a simple example of training an embedding in TensorFlow, and explains how to view embeddings with the TensorBoard Embedding Projector.

Debugging

- TensorFlow Debugger (https://www.tensorflow.org/guide/debugger), which explains how to use the TensorFlow debugger (tfdbg).

Performance

Performance is an important consideration when training machine learning models. Performance speeds up and scales research while also providing end users with near instant predictions.

- Performance overview (https://www.tensorflow.org/guide/performance/overview) contains a collection of best practices for optimizing your TensorFlow code.
- Data input pipeline (https://www.tensorflow.org/guide/performance/datasets) describes the `tf.data` (https://www.tensorflow.org/api_docs/python/tf/data) API for building efficient data input pipelines for TensorFlow.
- Benchmarks (https://www.tensorflow.org/guide/performance/benchmarks) contain a collection of benchmark results for a variety of hardware configurations.

Additionally, TensorFlow Lite (https://www.tensorflow.org/lite) has optimization techniques (https://www.tensorflow.org/lite/performance/best_practices) for mobile and embedded devices.

Extend

This section explains how developers can add functionality to TensorFlow's capabilities.

- TensorFlow architecture (https://www.tensorflow.org/guide/extend/architecture) presents an architectural overview.
- Create an op (https://www.tensorflow.org/guide/extend/op), which explains how to create your own operations.
- Custom filesystem plugin (https://www.tensorflow.org/guide/extend/filesystem), which explains how to add support for your own shared or distributed filesystem.
- Custom file and record formats (https://www.tensorflow.org/guide/extend/formats), which details how to add support for your own file and record formats.
- Language bindings (https://www.tensorflow.org/guide/extend/bindings), Python is currently the only language supported by TensorFlow's API stability promises. However, TensorFlow also provides functionality to create or develop features in other languages.
- Model files (https://www.tensorflow.org/guide/extend/model_files), for creating tools compatible with TensorFlow's model format.

XLA (Accelerated Linear Algebra) (https://www.tensorflow.org/xla) is an experimental compiler for linear algebra that optimizes TensorFlow computations.

Misc

- TensorFlow Version Compatibility (https://www.tensorflow.org/guide/version_compat), which explains backward compatibility guarantees and non-guarantees.
- Frequently Asked Questions (https://www.tensorflow.org/guide/faq), which contains frequently asked questions about TensorFlow.