

TensorFlow Cheatsheet

TensorFlow Installation

Install TensorFlow

```
pip install tensorflow
```

Install TensorFlow with GPU support

```
pip install tensorflow-gpu
```

Check TensorFlow version

```
python -c 'import tensorflow as tf; print(tf.__version__)'
```

Check available GPUs

```
python -c 'import tensorflow as tf; print(tf.config.list_physical_devices("GPU"))'
```

TensorFlow Basics

Create a constant tensor

```
tf.constant(value)
```

Create a variable tensor

```
tf.Variable(value)
```

Create a placeholder for input

```
tf.placeholder(dtype, shape=None, name=None)
```

Create a model

```
tf.keras.Model()
```

Add a dense layer to your model

```
tf.keras.layers.Dense(units)
```

Model Training

Configure the model for training

```
model.compile(optimizer, loss, metrics)
```

Train the model

```
model.fit(x, y, epochs, batch_size)
```

Evaluate the model

```
model.evaluate(x_test, y_test)
```

Generate predictions

```
model.predict(x)
```

TensorFlow Profiling

Start profiling

```
tf.profiler.experimental.start('logdir')
```

Stop profiling

```
tf.profiler.experimental.stop()
```

Launch TensorBoard to visualize profiling data

```
tensorboard --logdir=path/to/logs
```

TensorFlow Debugging

Assert tensors are equal

```
tf.debugging.assert_equal(x, y)
```

Check tensor for NaN or Inf

```
tf.debugging.check_numerics(tensor, message)
```

Print tensor values during execution

```
tf.print(tensor)
```

Log device placement

```
tf.debugging.set_log_device_placement(True)
```

TensorBoard

Create a file writer for logs

```
writer = tf.summary.create_file_writer('logs/')
```

Log scalar values

```
with writer.as_default():
    tf.summary.scalar('loss', loss, step=step)
```

Launch TensorBoard server

```
tensorboard --logdir=logs/
```

Model Saving/Loading

Save entire model

```
model.save('path/to/model')
```

Load entire model

```
model = tf.keras.models.load_model('path/to/model')
```

Save model weights

```
model.save_weights('path/to/weights')
```

Load model weights

```
model.load_weights('path/to/weights')
```

Distributed Training

Create distribution strategy for multiple GPUs

```
strategy = tf.distribute.MirroredStrategy()
```

Define model within strategy scope

```
with strategy.scope(): model = tf.keras.Model()
```

TensorFlow Serving

Pull TensorFlow Serving Docker image

```
docker pull tensorflow/serving
```

Run TensorFlow Serving container

```
docker run -p 8501:8501 --mount
type=bind,source=/path/to/model,target=/models/
my_model -e MODEL_NAME=my_model tensorflow/
serving
```

Make prediction request to TF Serving

```
curl -d '{"instances": [[1.0, 2.0, 3.0]]}' -X
POST http://localhost:8501/v1/models/
my_model:predict
```