Intel® Optimization for TensorFlow\* and

Intel® Extension for TensorFlow\* (ITEX) Cheat Sheet

Get started with Intel® Optimization for TensorFlow\* and Intel® Extension for TensorFlow\* (ITEX) using these commands!

# Intel® Optimization for TensorFlow\*(Google’s public release)

Features and optimizations for TensorFlow on Intel hardware are frequently up streamed and included in stock TensorFlow releases. As of TensorFlow v2.9, oneDNN optimization is automatically enabled

See [TensorFlow](https://www.tensorflow.org/) website for additional information.

|  |  |
| --- | --- |
| Basic Installation using PyPi | pip install tensorflow |
| Basic Installation using Anaconda | conda install -c conda-forge tensorflow |
| Import TensorFlow | import tensorflow as tf |
| Capture Verbose log (Command Prompt) | export ONEDNN\_VERBOSE=1 |
| Parallelize execution (in-code) | tf.config.threading.set\_intra\_op\_parallelism\_threads()  tf.config.threading.set\_intra\_op\_parallelism\_threads()  tf.config.set\_soft\_device\_placement(enabled=1) |
| Parallelize execution (Command Prompt) | export TF\_NUM\_INTEROP\_THREADS=<number of physical cores per socket>  export TF\_NUM\_INTRAOP\_THREADS=<number of sockets> |
| Non-uniform memory access (NUMA) | numactl --cpunodebind N --membind N python <script> |
| Enable BF16 training | from tensorflow.keras import mixed\_precision  policy = mixed\_precision.Policy('mixed\_bfloat16')  mixed\_precision.set\_global\_policy(policy) |

# Intel® Optimization for TensorFlow\*(Intel’s public release)

In addition to the performance tuning options listed under Google’s public release, Intel’s public release offers OpenMP optimizations for further performance enhancement

See [Intel® Optimization for TensorFlow Installation Guide](https://www.intel.com/content/www/us/en/developer/articles/guide/optimization-for-tensorflow-installation-guide.html) for additional installation methods.

See [Maximize TensorFlow\* Performance on CPU](https://www.intel.com/content/www/us/en/developer/articles/technical/maximize-tensorflow-performance-on-cpu-considerations-and-recommendations-for-inference.html) for additional installation methods.

|  |  |
| --- | --- |
| Basic Installation using PyPi | pip install intel-tensorflow |
| Basic Installation using Anaconda | conda install tensorflow (Linux/MacOS)  conda install tensorflow-mkl (Windows) |
| Import TensorFlow | import tensorflow as tf |
| Capture Verbose log (Command Prompt) | export ONEDNN\_VERBOSE=1 |
| Parallelize execution (in-code) | tf.config.threading.set\_intra\_op\_parallelism\_threads()  tf.config.threading.set\_intra\_op\_parallelism\_threads()  tf.config.set\_soft\_device\_placement(enabled=1) |
| Parallelize execution (Command Prompt) | export TF\_NUM\_INTEROP\_THREADS=<number of physical cores per socket>  export TF\_NUM\_INTRAOP\_THREADS=<number of sockets> |
| Non-uniform memory access (NUMA) | numactl --cpunodebind N --membind N python <script> |
| Enable BF16 training | from tensorflow.keras import mixed\_precision  policy = mixed\_precision.Policy('mixed\_bfloat16')  mixed\_precision.set\_global\_policy(policy) |
| Set max number of threads (Command Prompt) | export OMP\_NUM\_THREADS=num physical cores |
| Bind OpenMP threads to physical processing units | export KMP\_AFFINITY=granularity=fine,compact,1,0 |
| Set wait time (ms) after completing execution of a parallel region before sleeping | export KMP\_BLOCKTIME=<time>  Recommended to be to 0 for CNN or 1 for non-CNN (user should verify empirically) |
| Print OpenMP run-time library env variables during execution | export KMP\_SETTINGS=TRUE |

# Intel® Extension for TensorFlow\* (ITEX)

Intel® Extension for TensorFlow\* provides the most up-to-date features and optimizations on Intel hardware, most of which will eventually be up streamed to stock TensorFlow releases. Additionally, while users can get many optimization benefits by default without the need for additional setup, Intel® Extension for TensorFlow\* provides further tuning and custom operations to boost performance even more.

See [Intel® Extension for TensorFlow\* Installation Guide](https://intel.github.io/intel-extension-for-tensorflow/latest/docs/install/installation_guide.html) for additional installation methods

See [Intel® Extension for TensorFlow\*](https://intel.github.io/intel-extension-for-tensorflow/latest/get_started.html) Website for more information

|  |  |
| --- | --- |
| Basic GPU Installation using PyPi | pip install --upgrade intel-extension-for-tensorflow[gpu] |
| Import Intel® Extension for TensorFlow\* | import intel\_extension\_for\_tensorflow as itex |
| Get XPU backend type | itex.get\_backend() |
| Toggle GPU backend (in-code) – set by default | itex.set\_backend(‘GPU’) |
| Toggle GPU backend (Command Prompt) – set by default | export ITEX\_XPU\_BACKEND="GPU" |
| Advanced Auto Mixed Precision (In-Code) – improved inference speed with reduced memory consumption (Basic Configuration) | auto\_mixed\_precision\_options = itex.AutoMixedPrecisionOptions()  auto\_mixed\_precision\_options.data\_type = itex.BFLOAT16 # or itex.FLOAT16 |
| Advanced Auto Mixed Precision (Command Prompt) – improved inference speed with reduced memory consumption (Basic Configuration) | export ITEX\_AUTO\_MIXED\_PRECISION=1  export ITEX\_AUTO\_MIXED\_PRECISION\_DATA\_TYPE="BFLOAT16" # or "FLOAT16" |
| Customized AdamW optimizer (In-Code) | itex.ops.AdamWithWeightDecayOptimizer(  weight\_decay\_rate=0.001,  learning\_rate=0.001, beta\_1=0.9,  beta\_2=0.999,  epsilon=1e-07, name='Adam',  exclude\_from\_weight\_decay=["LayerNorm",  "layer\_norm", "bias"], \*\*kwargs  ) |
| Customized Layer normalization (In-Code) | itex.ops.LayerNormalization(  axis=-1, epsilon=0.001, center=True,  scale=True,  beta\_initializer='zeros',  gamma\_initializer='ones',  beta\_regularizer=None,  gamma\_regularizer=None,  beta\_constraint=None,  gamma\_constraint=None, \*\*kwargs  ) |
| Customized GELU (In-Code) | itex.ops.gelu(  features, approximate=False, name=None  ) |
| Customized LSTM (In-Code) | itex.ops.ItexLSTM(  200, activation='tanh',  recurrent\_activation='sigmoid',  use\_bias=True,  kernel\_initializer='glorot\_uniform',  recurrent\_initializer='orthogonal',  bias\_initializer='zeros', \*\*kwargs  ) |

For more information and support, or to report any issues, please visit:

[Intel® Extension for TensorFlow\*](https://github.com/intel/intel-extension-for-tensorflow/issues)

[TensorFlow](https://github.com/tensorflow/tensorflow/issues)

[Intel® AI Analytics Toolkit Forum](https://community.intel.com/t5/Intel-oneAPI-AI-Analytics/bd-p/ai-analytics-toolkit)

Sign up and try for free using: [Intel® DevCloud for oneAPI](https://devcloud.intel.com/oneapi/get_started/)