torch.cuda

This package adds support for CUDA tensor types.

It implements the same function as CPU tensors, but they utilize GPUs for computation.

It is lazily initialized, so you can always import it, and use `is_available()` to determine if your system supports CUDA.

CUDA semantics has more details about working with CUDA.

`StreamContext`	Context-manager that selects a given stream.
`can_device_access_peer`	Check if peer access between two devices is possible.
`current_blas_handle`	Return cublasHandle_t pointer to current cuBLAS handle
`current_device`	Return the index of a currently selected device.
`current_stream`	Return the currently selected `Stream` for a given device.
`cudart`	Retrieves the CUDA runtime API module.
`default_stream`	Return the default `Stream` for a given device.
`device`	Context-manager that changes the selected device.
`device_count`	Return the number of GPUs available.
`device_memory_used`	Return used global (device) memory in bytes as given by *nvidia-smi* or *amd-smi*.
`device_of`	Context-manager that changes the current device to that of given object.
`get_arch_list`	Return list CUDA architectures this library was compiled for.
`get_device_capability`	Get the cuda capability of a device.
`get_device_name`	Get the name of a device.
`get_device_properties`	Get the properties of a device.
`get_gencode_flags`	Return NVCC gencode flags this library was compiled with.
`get_sync_debug_mode`	Return current value of debug mode for cuda synchronizing operations.
`init`	Initialize PyTorch's CUDA state.
`ipc_collect`	Force collects GPU memory after it has been released by CUDA IPC.
`is_available`	Return a bool indicating if CUDA is currently available.
`is_initialized`	Return whether PyTorch's CUDA state has been initialized.
`memory_usage`	Return the percent of time over the past sample period during which global (device) memory was being read or written as given by *nvidia-smi*.
`set_device`	Set the current device.
`set_stream`	Set the current stream.This is a wrapper API to set the stream.
`set_sync_debug_mode`	Set the debug mode for cuda synchronizing operations.

`synchronize`	Wait for all kernels in all streams on a C DA evice to complete.
`utilization`	Return the percent of time over the past sample perio uring which one or more kernels was executing on the GP as given y *nvidia-smi*.
`temperature`	Return the average temperature of the GP sensor in Degrees C (Centigra es).
`power_draw`	Return the average power raw of the GP sensor in mW (MilliWatts)
`clock_rate`	Return the clock spee of the GP SM in Hz Hertz over the past sample perio as given y *nvidia-smi*.
`OutOfMemoryError`	Exception raise when evice is out of memory

Ran om Num er Generator

`get_rng_state`	Return the ran om num er generator state of the specifie GP as a ByteTensor.
`get_rng_state_all`	Return a list of ByteTensor representing the ran om num er states of all evices.
`set_rng_state`	Set the ran om num er generator state of the specifie GP .
`set_rng_state_all`	Set the ran om num er generator state of all evices.
`manual_seed`	Set the see for generating ran om num ers for the current GP .
`manual_seed_all`	Set the see for generating ran om num ers on all GP s.
`seed`	Set the see for generating ran om num ers to a ran om num er for the current GP .
`seed_all`	Set the see for generating ran om num ers to a ran om num er on all GP s.
`initial_seed`	Return the current ran om see of the current GP .

Communication collectives

`comm.broadcast`	Broa casts a tensor to specifie GP evices.
`comm.broadcast_coalesced`	Broa cast a sequence of tensors to the specifie GP s.
`comm.reduce_add`	Sum tensors from multiple GP s.
`comm.scatter`	Scatters tensor across multiple GP s.
`comm.gather`	Gathers tensors from multiple GP evices.

Streams an events

`Stream`	Wrapper aroun a C DA stream.
`ExternalStream`	Wrapper aroun an externally allocate C DA stream.
`Event`	Wrapper aroun a C DA event.

Gra hs (eta)

`is_current_stream_capturing`	Return True if C DA graph capture is un erway on the current C DA stream, False otherwise.
`graph_pool_handle`	Return an opaque token representing the i of a graph memory pool.
`CUDAGraph`	Wrapper aroun a C DA graph.
	Context-manager that captures C DA work into a

`make_graphed_callables`	versions.

Memory management

`empty_cache`	Release all unoccupie cache memory currently hel y the caching allocator so that those can e use in other GP application an visi le in *nvidia-smi*.
`get_per_process_memory_fraction`	Get memory fraction for a process.
`list_gpu_processes`	Return a human-rea a le printout of the running processes an their GP memory use for a given evice.
`mem_get_info`	Return the glo al free an total GP memory for a given evice using cu aMemGetInfo.
`memory_stats`	Return a ictionary of C DA memory allocator statistics for a given evice.
`memory_summary`	Return a human-rea a le printout of the current memory allocator statistics for a given evice.
`memory_snapshot`	Return a snapshot of the C DA memory allocator state across all evices.
`memory_allocated`	Return the current GP memory occupie y tensors in ytes for a given evice.
`max_memory_allocated`	Return the maximum GP memory occupie y tensors in ytes for a given evice.
`reset_max_memory_allocated`	Reset the starting point in tracking maximum GP memory occupie y tensors for a given evice.
`memory_reserved`	Return the current GP memory manage y the caching allocator in ytes for a given evice.
`max_memory_reserved`	Return the maximum GP memory manage y the caching allocator in ytes for a given evice.
`set_per_process_memory_fraction`	Set memory fraction for a process.
`memory_cached`	Deprecate ; see `memory_reserved()`.
`max_memory_cached`	Deprecate ; see `max_memory_reserved()`.
`reset_max_memory_cached`	Reset the starting point in tracking maximum GP memory manage y the caching allocator for a given evice.
`reset_peak_memory_stats`	Reset the "peak" stats tracke y the C DA memory allocator.
`caching_allocator_alloc`	Perform a memory allocation using the C DA memory allocator.
`caching_allocator_delete`	Delete memory allocate using the C DA memory allocator.
`get_allocator_backend`	Return a string escri ing the active allocator acken as set y `PYTORCH_CUDA_ALLOC_CONF`.
`CUDAPluggableAllocator`	C DA memory allocator loa e from a so file.
`change_current_allocator`	Change the currently use memory allocator to e the one provi e .
`MemPool`	MemPool represents a pool of memory in a caching allocator.
`MemPoolContext`	MemPoolContext hol s the currently active pool an stashes the previous pool.
`caching_allocator_enable`	Ena le or isa le the C DA memory allocator.

CLASS torch.cu a.use_mem_pool(*pool*, *device=None*) [SO RCE]

A context manager that routes allocations to a given pool.

Parameters

- **ool** (*torch.cuda.MemPool*) – a MemPool o ect to e ma e active so that allocations route to this pool.

NVIDIA Tools Extension (NVTX)

`nvtx.mark`	Descri e an instantaneous event that occurre at some point.
`nvtx.range_push`	Push a range onto a stack of neste range span.
`nvtx.range_pop`	Pop a range off of a stack of neste range spans.
`nvtx.range`	Context manager / ecorator that pushes an NVTX range at the eginning of its scope, an pops it at the en .

Jiterator (eta)

`jiterator._create_jit_fn`	Create a iterator-generate cu a kernel for an elementwise op.
`jiterator._create_multi_output_jit_fn`	Create a iterator-generate cu a kernel for an elementwise op that supports returning one or more outputs.

Tuna leO

Some operations coul e implemente using more than one li rary or more than one technique. For example, a GEMM coul e implemente for C DA or ROCm using either the cu las/cu lasLt li raries or hip las/hip lasLt li raries, respectively. How oes one know which implementation is the fastest an shoul e chosen? That's what Tuna leOp provi es. Certain operators have een implemente using multiple strategies as Tuna le Operators. At runtime, all strategies are profile an the fastest is selecte for all su sequent operations.

See the ocumentation for information on how to use it.

Stream Sanitizer (rototy e)

C DA Sanitizer is a prototype tool for etecting synchronization errors etween streams in PyTorch. See the ocumentation for information on how to use it.

‹ Previous

Next ›