Release Notes

Release 13.1

NVIDIA Corporation

Jan 11, 2026

Contents

Chapter 1. Overview

CUDA Toolkit 13.1 Update 1 - Release Notes

Welcome to the release notes for NVIDIA® CUDA® Toolkit 13.1. This release includes enhancements and fixes across the CUDA Toolkit and its libraries.

This documentation is organized into two main sections:

- **General CUDA**

  Focuses on the core CUDA infrastructure including component versions, driver compatibility, compiler/runtime features, issues, and deprecations.

- **CUDA Libraries**

  Covers the specialized computational libraries with their feature updates, performance improvements, API changes, and version history across CUDA 13.x releases.

Chapter 2. General CUDA

2.1. CUDA Toolkit Major Components

Note: Starting with CUDA 11, individual components within the CUDA Toolkit (for example: compiler, libraries, tools) are versioned independently.

For CUDA 13.1 Update 1, the table below indicates the versions:

Table 1: CUDA 13.1 Update 1 Component Versions

Component Name		Version Information	Supported Architectures	Supported Platforms
CUDA C++ Core Compute Libraries	Thrust	3.1.4	x86_64, arm64-sbsa	Linux, Windows
	CUB	3.1.4		
	libcu++	3.1.4		
	Cooperative Groups	13.1.115		
CUDA Application Compiler (crt)		13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA Compilation Optimizer (ctadvisor)		13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA Runtime (cudart)		13.1.80	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA culibos		13.1.115	x86_64, arm64-sbsa	Linux
CUDA cuobjdump		13.1.115	x86_64, arm64-sbsa	Linux, Windows
CUPTI		13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA cuxxfilt (demangler)		13.1.115	x86_64, arm64-sbsa	Linux, Windows

continues on next page

Table 1 – continued from previous page

Component Name	Version Information	Supported Architectures	Supported Platforms
CUDA Documentation	13.1.115	x86_64	Linux, Windows
CUDA GDB	13.1.115	x86_64, arm64-sbsa	Linux, WSL
CUDA Nsight Eclipse Plugin	13.1.115	x86_64	Linux
CUDA NVCC	13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA nvdisasm	13.1.115	x86_64, arm64-sbsa	Linux, Windows
CUDA NVML Headers	13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA nvprune	13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA NVRTC	13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA NVTX	13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA OpenCL	13.1.115	x86_64	Linux, Windows
CUDA Profiler API	13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA Sandbox dev	13.1.115	x86_64, arm64-sbsa	Linux, WSL
CUDA Compute Sanitizer API	13.1.118	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA TILE-IR AS	13.1.80	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA cuBLAS	13.2.1.1	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA cuFFT	12.1.0.78	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA cuFile	1.16.1.26	x86_64, arm64-sbsa	Linux
CUDA cuRAND	10.4.1.81	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA cuSOLVER	12.0.9.81	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA cuSPARSE	12.7.3.1	x86_64, arm64-sbsa	Linux, Windows, WSL

continues on next page

Table 1 – continued from previous page

Component Name	Version Information	Supported Architectures	Supported Platforms
CUDA NPP	13.0.3.3	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA nvFatbin	13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA nvJitLink	13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA nvJPEG	13.0.3.75	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA nvptxcompiler	13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
CUDA nvvm	13.1.115	x86_64, arm64-sbsa	Linux, Windows, WSL
Nsight Compute	2025.4.1.2	x86_64, arm64-sbsa	Linux, Windows, WSL (Windows 11)
Nsight Systems	2025.5.2.266	x86_64, arm64-sbsa	Linux, Windows, WSL
Nsight Visual Studio Edition (VSE)	2025.4.0.25287	x86_64 (Windows)	Windows
nvidia_fs[1]	2.27.3	x86_64, arm64-sbsa	Linux
Visual Studio Integration	13.1.115	x86_64 (Windows)	Windows
NVIDIA Linux Driver	590.48.01	x86_64, arm64-sbsa	Linux

2.2. CUDA Driver

Running a CUDA application requires the system with at least one CUDA capable GPU and a driver that is compatible with the CUDA Toolkit. See Table 3. For more information various GPU products that are CUDA capable, visit https://developer.nvidia.com/cuda-gpus.

Each release of the CUDA Toolkit requires a minimum version of the CUDA driver. The CUDA driver is backward compatible, meaning that applications compiled against a particular version of the CUDA will continue to work on subsequent (later) driver releases.

More information on compatibility can be found at https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html#cuda-compatibility-and-upgrades.

[1] Only available on select Linux distros

Note: Starting with CUDA 11.0, the toolkit components are individually versioned, and the toolkit itself is versioned as shown in the table below.

The minimum required driver version for CUDA minor version compatibility is shown below. CUDA minor version compatibility is described in detail in https://docs.nvidia.com/deploy/cuda-compatibility/index.html

Table 2: CUDA Toolkit and Minimum Required Driver Version for CUDA Minor Version Compatibility

CTK Version	Driver Range for Minor Version Compatibility	
	Min	**Max**
13.x	>= 580	N/A
12.x	>= 525	< 580
11.x	>= 450	< 525

* Using a Minimum Required Version that is **different** from Toolkit Driver Version could be allowed in compatibility mode – please read the CUDA Compatibility Guide for details.

** Starting with CUDA 13.1, the Windows display driver is **no longer** bundled with the CUDA Toolkit package. Users must download and install the appropriate NVIDIA driver separately from the official driver download page.

For more information on supported driver versions, see the CUDA Compatibility Guide for drivers.

*** CUDA 11.0 was released with an earlier driver version, but by upgrading to Tesla Recommended Drivers 450.80.02 (Linux) / 452.39 (Windows), minor version compatibility is possible across the CUDA 11.x family of toolkits.

The version of the development NVIDIA GPU Driver packaged in each CUDA Toolkit release is shown below.

Table 3: CUDA Toolkit and Corresponding Driver Versions

CUDA Toolkit	Toolkit Driver Version	
	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 13.1 Update 1	>=590.48.01	N/A
CUDA 13.1 GA	>=590.44.01	N/A
CUDA 13.0 Update 2	>=580.95.05	N/A
CUDA 13.0 Update 1	>=580.82.07	N/A
CUDA 13.0 GA	>=580.65.06	N/A
CUDA 12.9 Update 1	>=575.57.08	>=576.57
CUDA 12.9 GA	>=575.51.03	>=576.02
CUDA 12.8 Update 1	>=570.124.06	>=572.61
CUDA 12.8 GA	>=570.26	>=570.65

continues on next page

Table 3 – continued from previous page

CUDA Toolkit	**Toolkit Driver Version**	
CUDA 12.6 Update 3	>=560.35.05	>=561.17
CUDA 12.6 Update 2	>=560.35.03	>=560.94
CUDA 12.6 Update 1	>=560.35.03	>=560.94
CUDA 12.6 GA	>=560.28.03	>=560.76
CUDA 12.5 Update 1	>=555.42.06	>=555.85
CUDA 12.5 GA	>=555.42.02	>=555.85
CUDA 12.4 Update 1	>=550.54.15	>=551.78
CUDA 12.4 GA	>=550.54.14	>=551.61
CUDA 12.3 Update 1	>=545.23.08	>=546.12
CUDA 12.3 GA	>=545.23.06	>=545.84
CUDA 12.2 Update 2	>=535.104.05	>=537.13
CUDA 12.2 Update 1	>=535.86.09	>=536.67
CUDA 12.2 GA	>=535.54.03	>=536.25
CUDA 12.1 Update 1	>=530.30.02	>=531.14
CUDA 12.1 GA	>=530.30.02	>=531.14
CUDA 12.0 Update 1	>=525.85.12	>=528.33
CUDA 12.0 GA	>=525.60.13	>=527.41
CUDA 11.8 GA	>=520.61.05	>=520.06
CUDA 11.7 Update 1	>=515.48.07	>=516.31
CUDA 11.7 GA	>=515.43.04	>=516.01
CUDA 11.6 Update 2	>=510.47.03	>=511.65
CUDA 11.6 Update 1	>=510.47.03	>=511.65
CUDA 11.6 GA	>=510.39.01	>=511.23
CUDA 11.5 Update 2	>=495.29.05	>=496.13
CUDA 11.5 Update 1	>=495.29.05	>=496.13
CUDA 11.5 GA	>=495.29.05	>=496.04
CUDA 11.4 Update 4	>=470.82.01	>=472.50
CUDA 11.4 Update 3	>=470.82.01	>=472.50
CUDA 11.4 Update 2	>=470.57.02	>=471.41
CUDA 11.4 Update 1	>=470.57.02	>=471.41
CUDA 11.4.0 GA	>=470.42.01	>=471.11
CUDA 11.3.1 Update 1	>=465.19.01	>=465.89

continues on next page

Table 3 – continued from previous page

CUDA Toolkit	Toolkit Driver Version	
CUDA 11.3.0 GA	>=465.19.01	>=465.89
CUDA 11.2.2 Update 2	>=460.32.03	>=461.33
CUDA 11.2.1 Update 1	>=460.32.03	>=461.09
CUDA 11.2.0 GA	>=460.27.03	>=460.82
CUDA 11.1.1 Update 1	>=455.32	>=456.81
CUDA 11.1 GA	>=455.23	>=456.38
CUDA 11.0.3 Update 1	>= 450.51.06	>= 451.82
CUDA 11.0.2 GA	>= 450.51.05	>= 451.48
CUDA 11.0.1 RC	>= 450.36.06	>= 451.22
CUDA 10.2.89	>= 440.33	>= 441.22
CUDA 10.1 (10.1.105 general release, and updates)	>= 418.39	>= 418.96
CUDA 10.0.130	>= 410.48	>= 411.31
CUDA 9.2 (9.2.148 Update 1)	>= 396.37	>= 398.26
CUDA 9.2 (9.2.88)	>= 396.26	>= 397.44
CUDA 9.1 (9.1.85)	>= 390.46	>= 391.29
CUDA 9.0 (9.0.76)	>= 384.81	>= 385.54
CUDA 8.0 (8.0.61 GA2)	>= 375.26	>= 376.51
CUDA 8.0 (8.0.44)	>= 367.48	>= 369.30
CUDA 7.5 (7.5.16)	>= 352.31	>= 353.66
CUDA 7.0 (7.0.28)	>= 346.46	>= 347.62

- CUDA Toolkit driver bundling (pre-CUDA 13.1):
  - The CUDA Toolkit previously included an NVIDIA display driver for convenience.
  - This bundled driver was intended only for development purposes.
  - It is not recommended for production use, especially with Tesla GPUs.
- Recommended driver for Tesla GPUs:
  - For production environments using Tesla GPUs, download the latest certified driver from the official NVIDIA Driver Downloads site:

    https://www.nvidia.com/drivers
- Optional driver installation during Toolkit setup:
  - During CUDA Toolkit installation, users may choose to skip driver installation:
    - On Windows: via interactive or silent install options.
    - On Linux: by skipping driver meta packages.
- Change in CUDA 13.1 (Windows-specific):

  - Starting with CUDA 13.1, the Windows display driver is **no longer bundled** with the CUDA Toolkit.
  - Windows users must **manually download and install** the appropriate driver from the official NVIDIA site.
- Driver compatibility notes:
  - Some compatibility tables may list “N/A” for Windows driver versions.
  - Users must still ensure the installed driver meets or exceeds the minimum required version for the CUDA Toolkit.
  - For details, refer to the official CUDA Compatibility Guide for Drivers:

    https://docs.nvidia.com/deploy/cuda-compatibility/index.html

2.3. New Features

- No new features were introduced in CUDA 13.1 Update 1. For features introduced in CUDA 13.1, see the CUDA 13.1 release notes.

2.3.1. CUDA Developer Tools

For details on new features, improvements, and bug fixes, see the changelogs for:

- Nsight Systems.
- Nsight Visual Studio Edition.
- CUPTI.
- Nsight Compute.
- Compute Sanitizer.
- CUDA-C++-Programming-Guide.

2.4. Resolved Issues

2.4.1. CUDA Compiler

- Fixed an issue where CUDA 13.0–13.1 could generate illegal instructions when compiling a kernel with FFMA2 enabled.
- Fixed an issue where cuFFTDx FFT computations could produce incorrect results when built with NVCC 12.8 or NVCC 13.
- Fixed a bug where NVCC 13.0 could silently corrupt data in CuTe GEMM on GB202 GPUs.

2.5. Known Issues

2.5.1. General CUDA

- Certain Linux kernels with KASLR enabled have a known issue in HMM initialization, causing CUDA initialization to fail. This issue is indicated by the following debug message:

  ```
  [64689.125237] nvidia-uvm: uvm_pmm_gpu.c:3176 devmem_alloc_
  ↪pagemap[pid:92821] request_free_mem_region() err -34
  ```

  Fixes to this issue are being handled in upstream kernels. In the meantime, you can use one of the following workarounds:

  - Option 1: Disable KASLR (Preferred option)

    If using GRUB, edit /etc/default/grub and add `nokaslr` to `GRUB_CMDLINE_LINUX_DEFAULT`:

    ```
    GRUB_CMDLINE_LINUX_DEFAULT="quiet splash nokaslr"
    ```

    Then, update GRUB and reboot:

    ```
    sudo update-grub
    sudo reboot
    ```

  - Option 2: Disable HMM for UVM

    1. Create or edit /etc/modprobe.d/uvm.conf.
    2. Add or update the following line:

       ```
       options nvidia_uvm uvm_disable_hmm=1
       ```

    3. Unload and reload the `nvidia_uvm` kernel module or reboot the system:

       ```
       sudo modprobe -r nvidia_uvm
       sudo modprobe nvidia_uvm
       ```

2.5.2. CUDA Compiler

- The Tile-IR AS compiler currently supports only Blackwell-class devices and has limited low-precision support. These limitations will be removed in a future release.

2.6. Deprecated or Dropped Features

2.6.1. General CUDA

- CUDA 13.0 deprecates the following legacy vector types:
  - double4
  - long4
  - ulong4
  - longlong4
  - ulonglong4

  These types are being replaced by new aligned variants:
  - *_16a and *_32a (e.g., double4_16a, double4_32a)

  Deprecation warnings can be managed as follows:
  - Globally silenced by defining __NV_NO_VECTOR_DEPRECATION_DIAG
  - Locally suppressed using the macro pair __NV_SILENCE_HOST_DEPRECATION_BEGIN / __NV_SILENCE_HOST_DEPRECATION_END

  These legacy types are planned for removal in CUDA 14.0.
- The **CUDA installer** for Windows **no longer** bundles the display driver. Users must install the display driver separately, either before or after installing the CUDA Toolkit.
- **Multi-device launch APIs** and related references for Cooperative Groups **have been removed**. These APIs were previously marked as deprecated in CUDA 12.x:
  - cudaLaunchCooperativeKernelMultiDevice has been removed from cuda_runtime_api.h.
  - The accompanying parameter struct cudaLaunchParam has been removed from driver_types.h.
  - this_multi_grid and multi_grid_group have been removed from cooperative_groups.h.
- **Changes to cudaDeviceProperties structure**:

  In CUDA 13.0, several deprecated fields have been removed from the cudaDeviceProperties structure. To ensure forward compatibility, use the recommended replacement APIs listed below:

  Removed Fields and Their Replacements

Removed Field	Replacement API
clockRate	`cudaDeviceGetAttribute(cudaDevAttrClockRate)`
deviceOverlap	Use the `asyncEngineCount` field
kernelExecTimeoutEnabled	`cudaDeviceGetAttribute(cudaDevAttrKernelExecTimeout)`
computeMode	`cudaDeviceGetAttribute(cudaDevAttrComputeMode)`
maxTexture1DLinear	`cudaDeviceGetTexture1DLinearMaxWidth()`
memoryClockRate	`cudaDeviceGetAttribute(cudaDevAttrMemoryClockRate)`
singleToDoublePrecisionPerfRatio	`cudaDeviceGetAttribute(cudaDevAttrSingleToDoublePrecisionPerfRat`
cooperativeMultiDeviceLaunch	*No replacement available*

Removed cudaDeviceAttr Types (No Replacement Available)

- `cudaDevAttrCooperativeMultiDeviceLaunch`
- `cudaDevAttrMaxTimelineSemaphoreInteropSupported`

- The following legacy header files related to deprecated texture and surface references have been removed from the CUDA 13.0 runtime:
  - cuda_surface_types.h
  - cuda_texture_types.h
  - surface_functions.h
  - texture_fetch_functions.h

2.6.2. Deprecated Architectures

- Architecture support for Maxwell, Pascal, and Volta is considered feature-complete. Offline compilation and library support for these architectures have been removed in CUDA Toolkit 13.0 major version release. The use of CUDA Toolkits through the 12.x series to build applications for these architectures will continue to be supported, but newer toolkits will be unable to target these architectures.

2.6.3. Deprecated or Dropped Operating Systems

- Support for Ubuntu 20.04 has been dropped starting with this release. Users are advised to migrate to Ubuntu 22.04 LTS or later.

2.6.4. Deprecated or Dropped CUDA Toolchains

CUDA Tools

- As of CUDA 13.1, support for Nsight Eclipse Edition plugins is deprecated, and will be dropped in a future CUDA release.

Chapter 3. CUDA Libraries

This section covers CUDA Libraries release notes for 13.x releases.

Note: Documentation will be updated to accurately reflect supported C++ standard libraries for CUDA Math Libraries.

3.1. cuBLAS Library

3.1.1. cuBLAS: Release 13.1 Update 1

- **Known Issues**
  - The cuBLASLt Grouped GEMM API ignores groups with `k = 0`, which can lead to incorrect results. As a workaround, initialize output matrices `D` with `beta*C` for all groups, and then compute Grouped GEMM as `D += A*B` so the result for groups with `k = 0` is computed properly. This issue applies to the experimental cuBLASLt Grouped GEMM API introduced in CUDA 13.1. [*CUB-9529*]
  - Complex FP64 GEMM routines using fixed-point emulation can produce incorrect results when matrix dimensions are large enough that `m*n > 2^31` due to integer overflow in an address calculation. [*5720478*]
- **Resolved issues**
  - Fixed an issue where fixed point emulation with 7 mantissa bits or less could trigger unspecified launch failures. [*5692684*]
  - Fixed an issue where `cublasLtMatmul` with FP8 arguments and `CUBLASLT_MATMUL_MATRIX_SCALE_SCALAR_32F` scaling mode (default) incorrectly required scaling factor addresses to be 16-byte aligned. This issue existed since cuBLAS 12.9. [*5728938*]

3.1.2. cuBLAS: Release 13.1

- **New Features**
  - Introduced experimental support for grouped GEMM in cuBLASLt. Users can create a matrix with grouped layout using cublasLtGroupedMatrixLayoutCreate or cublasLtGroupedMatrixLayoutInit, where matrix shapes are passed as device arrays. cublasLtMatmul now accepts matrices with grouped layout, in which case matrices are passed as a device array of pointers, where each pointer is a separate matrix that represents a group with its own shapes. Initial support covers A/B types FP8 (E4M3/E5M2), FP16, and BF16, with C/D types FP16, BF16, and FP32; column-major only, default epilogue, 16-byte alignment; requires GPUs with compute capability 10.x or 11.0.

    In addition, the following experimental features were added as part of grouped GEMM:

    - Per-batch tensor-wide scaling for FP8 inputs, enabled by the new cublasLtMatmulDescAttributes_t entry CUBLASLT_MATMUL_MATRIX_SCALE_PER_BATCH_SCALAR_32F.
    - Per-batch device-side alpha and beta, enabled by the new cublasLtMatmulDescAttributes_t entries CUBLASLT_MATMUL_DESC_ALPHA_BATCH_STRIDE and CUBLASLT_MATMUL_DESC_BETA_BATCH_STRIDE.
  - Improved performance on NVIDIA DGX Spark for CFP32 GEMMs. [*5514146*]
  - Added SM121 DriveOS support.
  - Improved performance on Blackwell (sm_100 and sm_103) via heuristics tuning for FP32 GEMMs whose shapes satisfy M, N >> K. [*CUB-8572*]
  - Improved performance of FP16, FP32, and CFP32 GEMMs on Blackwell Thor.
- **Resolved Issues**
  - Fixed missing memory initialization in cublasCreate() that could result in emulation environment variables being ignored. [*CUB-9302*]
  - Removed unnecessary overhead related to loading kernels on GPUs with compute capability 10.3. [*5547886*]
  - Fixed FP8 matmuls potentially failing to launch on multi-device Blackwell GeForce systems. [*CUB-9487*]
  - Added stricter checks for in-place matmul to prevent invalid use cases (C == D is allowed if and only if Cdesc == Ddesc). As a side effect, users are no longer able to use D as a dummy pointer for C when using CUBLASLT_POINTER_MODE_DEVICE with beta = 0. However, a distinct dummy pointer may still be passed. The stricter checking was added in CUDA Toolkit 13.0 Update 2. [*5471880*]
  - Fixed cublasLtMatmul with INT8 inputs, INT32 accumulation, and INT32 outputs potentially returning CUBLAS_STATUS_NOT_SUPPORTED when dimension N is larger than 65,536 or when batch count is larger than 1. [*5541380*]
  - Added validation for batched matmul to reject invalid configurations where the batch counts differ (Adesc batch count != Bdesc batch count). [*5645772*]
- **Known Issues**
  - The Grouped GEMM cuBLASLt API ignores groups with k = 0, which can lead to incorrect results. As a workaround, initialize each output matrix D with beta * C for all groups before the call, then compute Grouped GEMM as D += A * B so that the result for groups with

k = 0 is preserved. This issue applies to the experimental Grouped GEMM cuBLASLt API released in CUDA 13.1. [*CUB-9529*]

3.1.3. cuBLAS: Release 13.0 Update 2

- **New Features**
  - Enabled opt-in fixed-point emulation for FP64 matmuls (D/ZGEMM) which improves performance and power-efficiency. The implementation follows the Ozaki-1 Scheme and leverages an automatic dynamic precision framework to ensure FP64-level accuracy. See here for more details on fixed-point emulation along with the table of supported compute-capabilities and the CUDA library samples for example usages.
  - Improved performance on NVIDIA DGX Spark for FP16/BF16 and FP8 GEMMs.
  - Added support for BF16x9 FP32 emulation to `cublas[SC]syr[2]k` and `cublasCher[2]k` routines. With the math mode set to `CUBLAS_FP32_EMULATED_BF16X9_MATH`, for large enough problems, cuBLAS will automatically dispatch SYRK and HERK to BF16x9-accelerated algorithms.
- **Resolved Issues**
  - Fixed undefined behavior caused by dereferencing a `nullptr` when passing an uninitialized matrix layout descriptor for `Cdesc` in `cublasLtMatmul`. [*CUB-8911*]
  - Improved performance of `cublas[SCDZ]syr[2]k` and `cublas[CZ]her[2]k` on Hopper GPUs when dimension `N` is large. [*CUB-8293*, *5384826*]
- **Known Issues**
  - `cublasLtMatmul` with INT8 inputs, INT32 accumulation, and INT32 outputs might return `CUBLAS_STATUS_NOT_SUPPORTED` when dimension `N` is larger than 65,536 or when the batch count is larger than 1. The issue has existed since CUDA Toolkit 13.0 Update 1 and will be fixed in a later release. [*5541380*]

3.1.4. cuBLAS: Release 13.0 Update 1

- **New Features**
  - Improved performance:
    - Block-scaled FP4 GEMMs on NVIDIA Blackwell and Blackwell Ultra GPUs
    - SYMV on NVIDIA Blackwell GPUs [*5171345*]
    - `cublasLtMatmul` for small cases when run concurrently with other CUDA kernels [*5238629*]
    - TF32 GEMMs on Thor GPUs [*5313616*]
    - Programmatic Dependent Launch (PDL) is now supported in some cuBLAS kernels for architectures `sm_90` and above, decreasing kernel launch latencies when executed alongside other PDL kernels.
- **Resolved Issues**
  - Fixed an issue where some `cublasSsyrkx` kernels produced incorrect results when `beta = 0` on NVIDIA Blackwell GPUs. [*CUB-8846*]

- Resolved issues in cublasLtMatmul with INT8 inputs, INT32 accumulation, and INT32 outputs where:
  - cublasLtMatmul could have produced incorrect results when A and B matrices used regular ordering (*CUBLASLT_ORDER_COL* or *CUBLASLT_ORDER_ROW*). [*CUB-8874*]
  - cublasLtMatmul could have been run with unsupported configurations of alpha/beta, which must be 0 or 1. [*CUB-8873*]

3.1.5. cuBLAS: Release 13.0

- **New Features**
  - The cublasGemmEx, cublasGemmBatchedEx, and cublasGemmStridedBatchedEx functions now accept CUBLAS_GEMM_AUTOTUNE as a valid value for the algo parameter. When this option is used, the library benchmarks a selection of available algorithms internally and chooses the optimal one based on the given problem configuration. The selected algorithm is cached within the current cublasHandle_t, so subsequent calls with the same problem descriptor will reuse the cached configuration for improved performance.

    This is an experimental feature. Users are encouraged to transition to the cuBLASLt API, which provides fine-grained control over algorithm selection through the heuristics API and includes support for additional data types such as FP8 and block-scaled formats, as well as kernel fusion. (see autotuning example in cuBLASLt).
  - Improved performance of BLAS Level 3 non-GEMM kernels (SYRK, HERK, TRMM, SYMM, HEMM) for FP32 and CF32 precisions on NVIDIA Blackwell GPUs.
  - This release adds support of SM110 GPUs for arm64-sbsa on Linux.
- **Known Issues**
  - cublasLtMatmul previously ignored user-specified auxiliary (Aux) data types for ReLU epilogues and defaulted to using a bitmask. The correct behavior is now enforced: an error is returned if an invalid Aux data type is specified for ReLU epilogues. *[CUB-7984]*
- **Deprecations**
  - The experimental feature for atomic synchronization along the rows (CUBLASLT_MATMUL_DESC_ATOMIC_SYNC_NUM_CHUNKS_D_ROWS) and columns (CUBLASLT_MATMUL_DESC_ATOMIC_SYNC_NUM_CHUNKS_D_COLS) of the output matrix which was deprecated in 12.8 has now been **removed**.
  - Starting with this release, cuBLAS will return CUBLAS_STATUS_NOT_SUPPORTED if any of the following descriptor attributes are set but the corresponding scale is not supported:
    - CUBLASLT_MATMUL_DESC_A_SCALE_POINTER
    - CUBLASLT_MATMUL_DESC_B_SCALE_POINTER
    - CUBLASLT_MATMUL_DESC_D_SCALE_POINTER
    - CUBLASLT_MATMUL_DESC_D_OUT_SCALE_POINTER
    - CUBLASLT_MATMUL_DESC_EPILOGUE_AUX_SCALE_POINTER
  - Previously, this restriction applied only to non-narrow precision matmuls. It now also applies to narrow precision matmuls when a scale is set for a non-narrow precision tensor.

3.2. cuFFT Library

3.2.1. cuFFT: Release 13.1

- **New Features**
  - Improved performance for transforms whose sizes are powers of 2, 3, 5, and 7 on Blackwell GPUs, in both single and double precision.
  - Improved performance for selected power-of-two sizes in 2D and 3D transforms, in both single and double precision.
  - Introduced an experimental cuFFT device API that provides host functions to query or generate device function code and exposes database metadata through a C++ header for use with the cuFFTDx library.
- **Resolved Issues**
  - Fixed a correctness issue, identified in CUDA 13.0, that affected a very specific subset of kernels: half- and bfloat16-precision strided R2C and C2R FFTs of size 1.

3.2.2. cuFFT: Release 13.0 Update 1

- **Known Issues**
  - In CUDA 13.0, a correctness issue affects a specific subset of kernels, namely half and bfloat precision size 1 strided R2C and C2R kernels. A fix will be included in a future CUDA release.

3.2.3. cuFFT: Release 13.0

- **New Features**
  - Added new error codes:
    - `CUFFT_MISSING_DEPENDENCY`
    - `CUFFT_NVRTC_FAILURE`
    - `CUFFT_NVJITLINK_FAILURE`
    - `CUFFT_NVSHMEM_FAILURE`
  - Introduced `CUFFT_PLAN_NULL`, a value that can be assigned to a `cufftHandle` to indicate a null handle. It is safe to call `cufftDestroy` on a null handle.
  - Improved performance for single-precision C2C multi-dimensional FFTs and large power-of-2 FFTs.
- **Known Issues**
  - An issue identified in CUDA 13.0 affects the correctness of a specific subset of cuFFT kernels, specifically half-precision and bfloat16 size-1 strided R2C and C2R transforms. A fix will be included in a future CUDA release.
- **Deprecations**

- Removed support for Maxwell, Pascal, and Volta GPUs, corresponding to compute capabilities earlier than Turing.
- Removed legacy cuFFT error codes:
  - `CUFFT_INCOMPLETE_PARAMETER_LIST`
  - `CUFFT_PARSE_ERROR`
  - `CUFFT_LICENSE_ERROR`
- Removed the `libcufft_static_nocallback.a` static library. Users should link against `libcufft_static.a` instead, as both are functionally equivalent.

3.3. cuSOLVER Library

3.3.1. cuSOLVER: Release 13.1

- **Resolved Issues**
  - Fixed a bug that prevented users from changing the algorithm for `cusolverDnXsyevBatched` by using `cusolverDnSetAdvOptions`.[*5539844*]

3.3.2. cuSOLVER: Release 13.0 Update 1

- **Resolved Issues**
  - Fixed a race condition in *cusolverDnXgeev* that could occur when using multiple host threads with either separate handles per thread or a shared handle, which caused execution to abort and returned CUSOLVER_STATUS_INTERNAL_ERROR.

3.3.3. cuSOLVER: Release 13.0

- **New Features**
  - cuSOLVER offers a new math mode to leverage improved performance of emulated FP32 arithmetic on Nvidia Blackwell GPUs.

    To enable and control this feature, the following new APIs have been added:

    - `cusolverDnSetMathMode()`
    - `cusolverDnGetMathMode()`
    - `cusolverDnSetEmulationStrategy()`
    - `cusolverDnGetEmulationStrategy()`
  - Performance improvements for `cusolverDnXsyevBatched()` have been made by introducing an internal algorithm switch on Blackwell GPUs for matrices of size `n <= 32`.

    To revert to the previous algorithm for all problem sizes, use cusolverDnSetAdvOptions().

    For more details, refer to the cusolverDnXsyevBatched() documentation.

- **Deprecations**
  - cuSOLVERMg is deprecated and may be removed in an upcoming major release. Users are encouraged to use cuSOLVERMp for multi-GPU functionality across both single and multi-node environments. To disable the deprecation warning, add the compiler flag -DDISABLE_CUSOLVERMG_DEPRECATED.
  - cuSOLVERSp and cuSOLVERRf are fully deprecated and may be removed in an upcoming major release. Users are encouraged to use the cuDSS library for better performance and ongoing support.

    For help with the transition, refer to the cuDSS samples or CUDA samples for migrating from cuSOLVERSp to cuDSS.

    To disable the deprecation warning, add the compiler flag: -DDISABLE_CUSOLVER_DEPRECATED.
- **Resolved Issues**
  - The supported input matrix size for cusolverDnXsyevd, cusolverDnXsyevdx, cusolverDnXsyevBatched, cusolverDn<t>syevd, and cusolverDn<t>syevdx is no longer limited to n <= 32768.

    This update also applies to routines that share the same internal implementation: cusolverDnXgesvdr, cusolverDnXgesvdp, cusolverDn<t>sygvd, cusolverDn<t>sygvdx, and cusolverDn<t>gesvdaStridedBatched.

3.4. cuSPARSE Library

3.4.1. cuSPARSE: Release 13.1 Update 1

- **New Features**
  - Added a new cusparseSpMVOp_bufferSize API that returns the size of the workspace buffer required for SpMVOp computations. Users provide this buffer when creating cusparseSpMVOpDescr_t, removing internal memory allocations.
  - Improved SpMVOp performance on B200. [*CUSPARSE-2931*] [*CUSPARSE-2932*] [*CUSPARSE-2933*]
- **Resolved Issues**
  - Fixed an accuracy issue in mixed-precision CSR/COO SpMM computations. [*CUSPARSE-2349*]
  - Fixed an issue in CSR SpMM computations when the input dense matrix has a high number of columns. [*CUSPARSE-2301*]

3.4.2. cuSPARSE: Release 13.1

- **New Features**
  - Introduced an experimental Sparse Matrix-Vector Multiplication (SpMVOp) API that provides improved performance compared with the existing generic CsrMV API. This API supports CSR format with 32-bit indices, double precision, and user-defined epilogues.
  - The nvJitLink shared library is now loaded dynamically at runtime.
  - Improved `cusparseXcsrsort` with reduced memory usage and higher performance. [*CUSPARSE-2630*]
- **Known Issues**
  - When using 32-bit indexing, `cusparseSpSV` and `cusparseSpSM` may crash if the number of nonzero elements (nnz) approaches `2^31 - 1`.[*CUSPARSE-2211*]
- **Resolved Issues**
  - Fixed potential issues when input and output pointers are not 16-byte aligned in `cusparseCsr2cscEx2`, `cusparseSparseToDense`, and CSR/COO `cusparseSpMM`. *[CUSPARSE-2380]*
  - Fixed a determinism issue in CSR `cusparseSpMM` ALG3. [*CUSPARSE-2612*]
  - All routines now support matrices with up to `2^31 - 1` nonzero elements (nnz) when using 32-bit indexing, with the exception of `cusparseSpSV` and `cusparseSpSM`. [*CUSPARSE-2153*]
  - Fixed a potential race condition that could occur when dynamically loading driver APIs. [*CUSPARSE-2764*]

3.4.3. cuSPARSE: Release 13.0 Update 1

- **New Features**
  - Added support for the BSR format in the generic SpMV API *(CUSPARSE-2518)*.
- **Deprecation**
  - Deprecated the legacy BSR SpMV API (replaced by the generic SpMV API).
- **Resolved Issues**
  - Enabled all generic APIs to support zero-dimension matrices/vectors (m, n, k = 0) *(CUSPARSE-2378)*.
  - Enabled all generic APIs to support small-dimension matrices/vectors (small m, n, or k) *(CUSPARSE-2379)*.
  - Fixed incorrect results in mixed-precision CSR/COO SpMV computations *(CUSPARSE-2349)*.

3.4.4. cuSPARSE: Release 13.0

- **New Features**
  - Added support for 64-bit index matrices in SpGEMM computation. *(CUSPARSE-2365)*
- **Known Issues**
  - cuSPARSE logging APIs can crash on Windows.
  - `CUSPARSE_SPMM_CSR_ALG3` does not return deterministic results as stated in the documentation.
- **Deprecation**
  - Dropped support for pre-Turing architectures (Maxwell, Volta, and Pascal).
- **Resolved Issues**
  - Fixed a bug in `cusparseSparseToDense_bufferSize` that caused it to request up to 16× more memory than required. *[CUSPARSE-2352]*
  - Fixed unwanted 16-byte alignment requirements on the external buffer. Most routines will now work with any alignment. In the generic API, only `cusparseSpGEMM` routines are still affected. *[CUSPARSE-2352]*
  - Fixed incorrect results from `cusparseCsr2cscEx2` when any of the input matrix dimensions are zero, such as when `m = 0` or `n = 0`. *[CUSPARSE-2319]*
  - Fixed incorrect results from CSR SpMV when any of the input matrix dimensions are zero, such as when `m = 0` or `n = 0`. *[CUSPARSE-1800]*

3.5. Math Library

3.5.1. CUDA Math: Release 13.0

- **New Features**
  - Single and double precision math functions received targeted performance and accuracy improvements through algorithmic simplifications, reduced branching, and tighter approximations.
    - `atan2f`, `atan2`: Up to 10% faster with minor improvements in accuracy.
    - `sinhf`, `coshf`, `acoshf`, `asinhf`, `asinh`: Up to 50% speedups with minor improvements in accuracy.
    - `cbrtf`, `rcbrtf`: 15% faster with minor improvements in accuracy.
    - `erfinvf`, `erfcinvf`, `normcdfinvf`: Minor accuracy improvements, performance neutral.
    - `ldexpf`, `ldexp`: Up to 3x faster in single precision and 30% faster in double precision, with no accuracy loss.
    - `modff`, `modf`: Up to 50% faster in single precision and 10% faster in double precision, with no accuracy loss.

3.6. nvJPEG Library

3.6.1. nvJPEG: Release 13.1

Resolved Issues

- nvJPEG's lossless JPEG 92 (lj92) implementation can now correctly handle lj92 files that contain a comment marker in the header. *[5484797]*

3.6.2. nvJPEG: Release 13.0 Update 1

Resolved Issues

- Fixed a race condition in certain cases during progressive encoding (*5307748*).
- Fixed an uninitialized read when encoding images as 4:1:0 JPEG bitstreams (*5308008*).

3.6.3. nvJPEG: Release 13.0

- **Deprecations**
  - Removed the `nvjpegEncoderParamsCopyHuffmanTables` API.

Resolved Issues

- nvJPEG is now more robust and no longer crashes or exhibits undefined behavior when decoding malformed or truncated bitstreams. *[5168024, 5133845, 5143450]*
- `nvjpegEncodeYUV` now avoids reading outside of allocated device memory in certain cases. *[5133826]*
- Optimized memory usage when encoding RGB inputs using the hardware encoder.
- Fixed issues related to rounding in various transform, sampling, and conversion steps, improving image quality for both encoder and decoder. *[5064901, 3976092]*
- Various bug fixes for improved security.

3.7. NPP Library

3.7.1. NPP: Release 13.1 Update 1

- **Resolved Issues**
  - Reduced nvJPEG Encoder initialization time on Thor. [*5533951*]

3.7.2. NPP: Release 13.1

- **Resolved Issues**
  - Fixed an issue in `nppiCFAToRGB_8u_C1C3R()` affecting SSIM validation for `NPPI_BAYER_GBRG` patterns. *[5192648]*

3.7.3. NPP: Release 13.0

- **Deprecations**
  - **Removal of Legacy Non-Context APIs**

    All legacy NPP APIs without the *_Ctx* suffix have been deprecated and are now removed starting with this release. Developers should transition to the context-aware (*_Ctx*) versions to ensure continued support and compatibility with the latest CUDA releases.

  - **Deprecation of ``nppGetStreamContext()``**

    The `nppGetStreamContext()` API has been deprecated and removed. Developers are strongly encouraged to adopt application-managed stream contexts by explicitly managing the `NppStreamContext` structure. For guidance, refer to the NPP Documentation – General Conventions and the usage demonstrated in the StreamContexts example.

- **Resolved Issues**
  - Fixed an issue in `nppiFloodFillRange_8u_C1IR_Ctx` where the flood fill operation did not correctly fill the full target area. *[5141474]*
  - Resolved a bug in the `nppiDebayer()` API that affected proper reconstruction of color data during Bayer pattern conversion. *[5138782]*

Chapter 4. Notices

4.1. Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

4.2. OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

4.3. Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright