GRAPHCORE

Poplar® SDK v2.1.0 Release notes

Version 2.1.0

Graphcore Ltd

Jul 01, 2021

CONTENTS

CHAPTER

ONE

LEGAL NOTICE

The information included in Release notes is only for use with or for Graphcore products.
Use of the information included in Release notes is at your own risk, and subject to the terms of the Graphcore end-user license agreement.
All Graphcore products including hardware and software, described in Release notes, are subject to continuous changes at any time, without notice.

Graphcore ® and Poplar ® are registered trademarks of Graphcore Ltd.

AI-Float™, Colossus™, Exchange Memory™, In-Processor-Memory™, IPU-Core™, IPU-Exchange™, IPU-Fabric™, IPU-Link™, IPU-M2000™, IPU-Machine™, IPU-POD™, IPU-Tile™, PopART™, PopLibs™, PopVision™, PopTorch™, Streaming Memory™ and Virtual-IPU™ are trademarks of Graphcore Ltd.
All other trademarks are the property of their respective owners.

CHAPTER

TWO

SCOPE OF THIS DOCUMENT

This document contains the Release notes for the Poplar SDK 2.1.0 for Graphcore's IPU product family. The software deliverables covered by this document are the following:

Driver & Utilities Driver and associated utilities needed by the Graphcore IPU.

PopART The Poplar Advanced Run Time is a flexible ONNX-compatible runtime supporting both training & inference.

PopTorch The PopTorch library provides a set of extensions for PyTorch to enable it to run on the Graphcore IPU hardware.

Poplar A graph programming framework for the IPU.

PopDist/PopRun Poplar Distributed Configuration Library (PopDist) is a library for configuring and coordinating distributed execution of (large-scale) machine learning applications.

TensorFlow An implementation of the TensorFlow framework for the Graphcore IPU.

CHAPTER

THREE

PACKAGE CONTENTS

The downloaded unified Poplar SDK will contain the following packages:

3.1 Ubuntu 18.04

Package	Version
Driver & Utilities	1.0.52
PopART	2.1.0+145366
PopTorch	2.1.0+18411
Poplar	2.1.0+145407
PopDist/PopRun	2.1.0
TensorFlow 1	Graphcore TensorFlow 2.1.0
TensorFlow 2	Graphcore TensorFlow 2.1.0

3.2 CentOS 7.6

Package	Version
Driver & Utilities	1.0.52
PopART	2.1.0+145366
PopTorch	2.1.0+18411
Poplar	2.1.0+145407
PopDist/PopRun	2.1.0
TensorFlow 1	Graphcore TensorFlow 2.1.0
TensorFlow 2	Graphcore TensorFlow 2.1.0

Note: See *Appendix A* (page 36) for TensorFlow additional requirements.

CHAPTER

FOUR

PRODUCT SUPPORT AND COMPATIBILITY MATRIX

SUPPORTED

These products are actively worked on: they will receive new features, general updates and security updates.
Notice of deprecation will be sent in advance for supported products.

DEPRECATED

These products will only receive security updates.
These products are expected to work with the indicated products however correctness is not guaranteed.
It is advised **not** to upgrade to this software version, unless strictly necessary.
In the future, these products can move to a **Not Supported** state, without further notice.
Support level will reflect the deprecated status.

NOT SUPPORTED

These products are not expected to work with this release.
No support will be provided.

Important: Deprecated products can be moved to a **Not supported** status without further notice.

4.1 IPU-M2000 System Software compatibility matrix

IPU-Machine model	Version	Support level	Notes
IPU-M2000 300-0024	2.1.0	Supported	N/A

4.2 IPU PCIe Hardware Support level

Model	Revision	ICU Firmware version	Driver version	Support level	Notes
C2 300-0004	All revisions	1.4.14	1.0.52	Supported	•

Note: Use Firmware revision in accordance with IPU revision.

Important: For Firmware revision, compatibility is only enforced for patch versions.

4.3 Driver Support level

OS	Support level	Supported Kernel Version	Notes
CentOS 7.4/7.5	Supported	3.10	CentOS LTS kernel.
CentOS 7.6	Supported	3.10	CentOS LTS kernel.
Microsoft Windows	Supported	Windows Server 2019	
Ubuntu 18.04	Supported	5.4	Ubuntu LTS kernel.

Warning:

It is strongly recommended to update the kernel module of the driver to the version included with this 2.1.0 release.
This is to avoid incompatibilities with the non-kernel components of this SDK

4.4 SDK 2.1.0 Support level

OS	Support level	Notes
Microsoft Windows	Not Supported	
CentOS 7.6	Supported	In some specific instances we encountered a less than optimal model compilation time. Investigations are ongoing to address the problem.
Ubuntu 18.04	Supported	

4.5 Supported toolchain

4.5.1 Ubuntu 18.04

Tool	Support level	Version
GCC/G++	Supported	7.2.0
libstdc++	Supported	6.0.24
libc	Supported	2.27
binutils	Supported	2.30

4.5.2 CentOS 7.6

Tool	Support level	Version
GCC/G++	Supported	7.3.1
libstdc++	Supported	6.0.24
libc	Supported	2.17
binutils	Supported	2.28

4.6 Supported tools

Tool	Support level	Version	
Python	Supported	3.6	
Boost library	Deprecated	1.70	Older versions of boost are no longer supported.

CHAPTER

FIVE

LIST OF CHANGES

The following sections will list changes in version 2.1.0, as well as older releases, for all products contained in the Poplar SDK.

There are three main sections, divided by argument:

Changelogs *Changelogs* (page 8) section lists important bug fixes and relevant functionality that has been added. Minor fixes or features will not be listed.

Known issues *Known Issues* (page 28) section will list all important issues known to date. This section will list issues that will impact Poplar functionality.

Compatibility changes *Compatibilities changes* (page 32) section will capture any change that needs to apply existing code, to remain compatible with this version of the SDK.

CHAPTER

SIX

CHANGELOGS

Product	Changelog
Driver & Utilities	*Changelog Driver & Utilities* (page 8)
PopART	*Changelog PopART* (page 11)
PopTorch	*Changelog PopTorch* (page 17)
Poplar	*Changelog Poplar* (page 18)
Poplar Libraries	*Changelog Poplar Libraries* (page 21)
PopDist/PopRun	*Changelog PopRun/PopDist* (page 24)
Libpva Library	*Changelog Libpva Library* (page 25)
TensorFlow	*Changelog TensorFlow* (page 25)

6.1 Driver & Utilities Changelog

6.1.1 Kernel Module

1.0.52

- T38126: Improved error handling when IPU cards are disconnected.
- T36583: Avoid clearing PL DDR on docker restart.
- T36158: Add a mechanism for measuring IPU utilisation and mark count monitoring.
- T34827: Detect attachBuffer() failures.
- T30346: Display AER error counts in gc-hosttraffictest on M2000.

1.0.51

- T36583: Partial support to avoid clearing PL DDR on docker restart
- T30346: Partial support to display AER error counts in gc-hosttraffictest on M2000
- T34827: Fixed error when detaching buffers in the PCIe driver.
- T36158: Add a mechanism for measuring IPU utilisation and mark count monitoring.
- T38126: Improved error handling when IPU cards are disconnected.

6.1.2 Low level libraries and tools

2.1.0+145407

- T38422: Fixed boost exception on application shutdown.
- T40362: Modification of ICU comms to avoid race condition during intensive IPU sync conditions.
- T39479: IPU utilisation, power and temperature sensor data added to PVTI.
- T39908: Avoid parsing invalid driver version field when running gc-monitor over fabric.
- T39773: Set correct PHY equalisation settings.
- T35524: Add [parent.device] style for MultiIPU id's in GCDA logging statements.
- T39393: Improve tile parity reset speed for C200 and M2000 based systems.
- T39068: Fix `gc-info` memory dump when using debug server.
- T38926: Fixed `gc-inventory` error message and exit code when Fabric device discovery fails.
- T39329: Make gc-gwlinkstraffictest timeout configurable.
- T38738: Fixed `gc-inventory` JSON output when no devices are found.
- T32643: Reduce docker image size of ipuof server.
- T38981: Introduce non-blocking CQ event mode.
- T39375: Fixed a busy loop in IPUoF server when client application was terminated during QP initialisation.
- T39164: Fixed segfault during IPUoF device discovery.
- T37392: re-create IPUoF-server QPs for each run.
- T38090: Add support for runtime enabling and disabling of graphs and series in PVTI.
- T39608: Add C++ example program to gcipuinfo library.
- T25931: Support reading from the middle of a stream by limiting the bytes read when reading binaries.
- T36157: Report Poplar MultiIPU application code, data and stack sizes per IPU.
- T36158: Add a mechanism for measuring IPU utilisation and mark count monitoring.
- T34827: Detect attachBuffer() failures.
- T30346: Display AER error counts in gc-hosttraffictest on M2000.
- T35916: Add gcipuinfo library.
- T36683: Add documentation for the gateway links test.
- T28235: Add more IPU-Machine specific details to GCDA and command line tools documentation.
- T32019: Removed tools zip file inside the same tools zip file.
- T35962: Permit loading of first kilobyte of memory when using the secondary IPU bootloader.
- T31166: Added documentation for `gc-exchangewritetest`.
- T32949: Enhanced `gc-iputraffictest` features (MultiIPU, iterations, sync).
- T35327: Added additional libs to support external ICUComms linkage.
- T35012: Document gc-info—tile-overview.
- T34666: Added libpvti headers to command line tools install.
- T38727: Device attributes are now generated from a common JSON description file.
- T38629: Harmonise and improve device attribute labels. The previous attribute labels are still supported for backwards compatibility.

- T36630: Device attributes are now specified in a common header file.
- T37935: Change default sync method for C2 and C200 from hybrid to polling.
- T38378: Fix parity reset for first memory element.
- T36687: Add `GCDA_SWAP_NLC_MODE` environment variable for internal testing.
- T38203: Fix IPUDebug::initIPURegs initialisation of worker registers.
- T36694: Add support for recording IPU telemetry for the system analyzer.
- T37772: "IPU" index attribute on IPU-Machines is now 0-3.
- T37753: Fix Python checkForSOCErrors function.
- T33826: Added `isIPUMachineGateway()` convenience method.
- T35781: segmentation fault fixed.
- T35490: Improve ICU attach failure reporting.
- T37275: Fixed `gc-infoshow-insn`.
- T37104: Updated `gc-infoipu-arch` to stop it attaching to the IPU.
- T36628: IPU architecture information resolved during device discovery instead of attach.
- T35955: Rename some options/fields for better clarity.
- T35954: In gc-monitor, display tile memory usage per IPU in terms of percentage of total tile-memory.
- T28548: Add support for all Mk2 sync zones.
- T30698: Use shared contiguous buffer on MultiIPU devices.
- T31712: Enhanced support for contiguous buffers.
- T33010: Improved binary load performance via via parallelism via the bootloader.
- T36330: Support bootloading tile binaries less than 16K.
- T36023: Build system enhancement to locate IPU test binaries.
- T34815: Support staged IPU reset across multiple applications.
- T35543: Fixed clearing of XB debug state registers.
- T33617: Improve GCDA and IPUoF error reporting.
- T35778: Enhance bootloader to prevent tile column data leak through buffers.
- T35651: Ensure mirrorBuffer has completed before starting next bootloader transfer.
- T34986: Improved the interface to the tile overview state functions.

2.0.1

- T32426: Support added for power and sensor monitoring on fabric systems.
- T32535: Fixed bug with post_send with IBV_SEND_INLINE.
- T35781: Fixed sensor sampling segfault.
- T35916: Add gcipuinfo library
- T35954: In gc-monitor, display tile memory usage per IPU in terms of percentage of total tile-memory.
- T35955: Rename some options/fields for better clarity.
- T36158: Add a mechanism for measuring IPU utilisation and mark count monitoring.
- T36619: Fixed race in Fabric get_device_info.
- T36630: Device attributes are now specified in a common header file

- T38126: Improved error handling when IPU cards are disconnected.
- T38629: Harmonise and improve device attribute labels. The previous attribute labels are still supported for backwards compatibility.
- T38727: Device attributes are now generated from a common JSON description file

6.2 PopART Changelog

6.2.1 2.1.0+145366

New features

- Add explicit accumulation and step loop
- Generalize the pruning algorithm
- Refactor HostLoad and HostStore for use with explicit loops
- Implement LoopScanOut pattern (ONNX scan out support)
- Support Recompute checkpoints on the final forward pipeline stage
- Add experimental option `scaled_optimizer_state` to Adam optimizer. Improves numerical stability of FLOAT16 `accl1_type`.
- Allow CastGradOp in path of `SerializeMatMul` transform
- Support setting activations for `LSTMOp`
- Support gradient clipping for Adam and Lamb
- Add an `isDirectViewChain` method for finding chains of inplace view changing ops.
- Add additional checks and error messages to serialised `matmuls`. Behaviour is unchanged but the error messaging and checking is clearer.
- Ability to use sequence_lens tensor in LSTM ops. See https://github.com/onnx/onnx/blob/master/docs/Changelog.md#LSTM-14 for details.
- Added Host Store / Load Ops with transform, this is a pre-cursor to overlapped IO.
- Add new variant of SGD optimiser that has separate gradient accumulation and velocity tensors
- Add enum `SGDAccumulatorAndMomentum` for selecting whether to use combined or separate gradient accumulation and velocity tensors
- Add ability to create Session directly from an Ir
- Add ability to pass forward tensors required in the backwards pass by adding them as outputs of the forward graph
- Refactoring of grad creation logic for subgraph ops
- Make it possible for some graph inputs to not have an associated gradient tensor in backwards pass
- Add `RandomSetup` transform
  - 1) support for outlining dropouts
  - 2) support for random ops in `LoopOp` and `CallOp`
- Add mocking support
- Add a TensorFlow variant of the `RMSProp` optimizer
- Allow BatchNormalization to be in inference mode during training
- Support `uint8` streams and `uint8` casts

- Add the `ReduceMedian` op
- Redesign AutomaticLossScale transform so that the loss scale update factor is persistent between runs
- Support Adaptive and Adam optimizers with auto loss scaling (Preview)
- Support auto loss scaling with sharding, pipelining, gradient accumulation, and replicated graphs (Preview)
- Implement constant folding in ONNX passes
- Add ONNX constant folding for shape op
- Add ONNX pattern to remove PopART gemm
- Support ONNX batch norm shape inference for 5 outputs
- Add BitwiseNot, Fmod, ReduceMedian, Remainder, Round ONNX shape inference
- Add a CTC beam search decoder Op to do CTC inference
- Implemented bitwise operators
- Use a `snap::Graph` rather than `poplar::Graph`
- Expose Poplar callbacks to end users

Bug Fixes

- Fix logic in the unwinding method of the Expand Op
- Fix nested-loop compile time bug
- Fix compile time bugs related to slow inplacing pass
- Fix elementwise inplace reshaping of RTS tensors
- Rewrite pattern to fix SubtractArg1GradOp
- Fix `modifiedRegionsByOps`
- Update aliasing info after matmul serialization
- Add missing `constexpr` for detach
- Explicit main loops fixes and additional tests
- Fix randomsetup non-determinism
- Add missing synthetic data check for host IO
- Stop removeScope from mangling tensor names
- Fix lower case data type name
- Adjust opx modify check in irlowering (`opxModifyChecking`) to support integer types
- Traverse only aliases with non-empty region overlap to the modified tensor
- Add new required topo cons each time the optimizer changes
- Fix case of missing metaShape propagation (RTS)
- Fix type errors with FLOAT16 optimizer state and non-constant lossScaling
- Fix recursive loop in MultiConvBaseOp::getPads
- Do not save `accum__` tensor in serialised executable, this fixes an error when trying to use gradient accumulation and saving a serialised executable.
- Fix for map::at error in lstm sdk tests by ensuring `seq_lens` tensor is found correctly.
- Sort tensors in `Tensors::getOfType`, this fixed an non-determinism across different OSes
- Fixed dynamicslice shape inference

- Correct NLL loss for when ignoring all indices to ensure parity with PyTorch.
- Fix spurious CMake warning about non-existant POPLAR_INSTALL_DIR
- Correctly use default copy/move semantics for OptimizerValue
- Remove source of non-determinism in storage of topological constraints
- Avoid spdlog formatting exception when stack trace contains '{}'
- Disallow inplacing on an output of a recomputation
- Moved POPART_PRINT_TENSORS code that prints outputs to after the op in the Poplar sequence
- Fix mapping issue for tensorLocationSettingsOverride
- Avoid applying PostNRepl pattern if output is graph output
- Fix for test with callbacks that sometimes fail when multithreaded callbacks are enabled
- Fix for dealing with missing edge gradients with SmoothL1Loss
- Removed use of 'final' in DropoutOp classes to make subclassing possible
- Made it so createOp returns `DerivedOp*` instead of `Op*`
- Set default prefetch buffering depth to 1
- If doing implicit recomputation, require that ops with a path from the loss are scheduled after those that have a path to the loss
- Do not alias input for `IdentityLossOp` with no reduction
- Fix ONNX gemm performance regression
- Set default options for test device when simulate target
- Add missing input to loss scale update op.
- Throw error in Ir if using `AnchorReturnType` other than ALL or SUM with explicit main loops
- Fix casting from 16-bit floating point to 8-bit (un)signed integer to behave like NumPy's casting
- Remove leakage of private dependencies in the public headers

Optimizations

- Use Poprithms to do inplacing, which makes the transform orders of magnitude faster
- Accelerate scheduler in non-optimal case by ignoring certain annotations
- Accelerate scheduler by using transitive closure optimizations to insert additional edges
- Add unit test for `modifiedRegionsByOps`
- Improve prune speed and graph traversal utility
- Improve implicit tensor and graph scope handling
- Schedule pipeline Restore operations as late as possible
- Set the global seed on each replica to the same and offset random operations by the replica index on the device
- Add patterns to avoid keeping additional copies of tensors due to outlining
- Calculate SoftmaxGrad from the output activation to avoid recomputing
- Add tests for negative padding
- Refactor `Op::setup` methods
- Speed up `annotateAccumulateOuterFragment` by returning optimizer prefixes as const

- Make TaskId type cheaper to construct
- Change `Tensor::anyAlias` to use Poprithms backend
- Various small refactorings (removal of `nEdgesToLoss` and `Op::readyToCreateGradients`)
- Testing and refactoring of `autodiff`
- Made host-reduce transformation test more robust
- If CastOp has same input type as output type, convert to identity
- Release the GIL before entering `Session::run`
- Enable Poplar multithreaded IO by default
- Optimise PopART gather memory usage with slice planning

Logging and documentation

- Add logging and timing to simplify triage of compile time bottlenecks
- Remove or lower noisy warnings
- Change default log level to WARN
- Moved/ Removed the examples in PopART to public_examples where relevant. Otherwise removed unnecessary examples.
- Added Graphcore opset functions to the Python api docs
- Update Python docs and add missing docstrings where appropriate.
- Add developer notes for transforms
- Added stream operators for `std::tuple` and `std::pair`
- Remove logging on prefetch failure
- Added Trompeloeil instructions to README.md
- Make `SessionOptions` comments Doxygen-friendly
- Support weight-specific optimizer tensors for automatic loss scaling
- Add an optional user progress logger for graph compilation

6.2.2 2.0.1

New features

- Change optimizer to use globalReplicationFactor when distributed
- Add error when trying replicated tensor sharding and global replication
- Copy partialType in serializematmuls
- Allow Offline device with distributed replication
- Ensure enableEngineCaching is set before loading cache and handle loading errors
- Allow Session to load binaries from serialized files
- Update compileAndExport to use same file format as the executable cache
- Fuse cache files and support multiple cache entries
- Add ability to PopART to build in a standalone Conda environment
- Add DepthToSpace as a custom op

- Add backwards op for atan2
- Re-enable `synthetic_data_test` that requires HW
- Add enhanced debug information to PopART
- Make multiconv op attributes optional
- Add support for biased convolutions in PopART MultiConv
- Add numerous doxygen comments and comment improvements
- Add SessionOption `accumulationAndReplicationReductionType`, which controls the reduction type of accumulated gradients and reduction of replicas
- Stop using reduction option of loss operations to determine how loss gradients are reduced if `accumulationReductionType` is used
- Add reverse operation in the aiGraphcore domain
- Support input/output/anchor tensors of type int8 (including synthetic data)
- Support stashing/restoring/IPUCopying an int8 input tensor when pipelining
- Infer tensor to update in VarUpdateOps from input Tensors
- Check early for CMake versions in the found dependencies
- Improve model compile time by planning convolutions and matmuls in parallel
- Use a more poplar tensor-expression efficient implementation of ResizeOp on non-neg integers
- Add Connectionist Temporal Classification (CTC) loss
- Add support for setting Poplar options for LSTM operations
- Improve the debug information that is recorded in PopART
- Improve the PopART user guide
- Add ability to copy inputs/outputs to subgraphs in a just-in-time manner
- Improve error message when unknown tensors are used in the builder
- Add Sequence slice op working with packed sequences
- Add custom op shape inference that doesn't require onnx
- Add environment variable, `POPART_TRACE_TENSORS`, as an alternative to `PrintTensorOp`
- Support user added CallOps being used with pipelining
- Improve error messages in the case of optimizer compatibility errors
- Change PriTaskDependency to allow delaying picking tensor creators and express more complex forms of dependencies
- Add LAMB optimizer off-chip test with mean reduction
- Add support for ONNX ScanOp (forward-only)
- Add global batch size test with batch serialisation
- Implement operator support for AllReduce & ReduceScatter
- Add transformation to unroll and decompose LoopOps
- Add dead code elimination support as post-IR optimisation
- Add range-based indexing for remote buffers when used in LoopOps
- Add transform to merge adjacent LoopOps together
- Allow pipelining to resolve correct IPU from CallOps
- Add new warnings and errors for incompatible user settings

- Improve IR transform order to run critical transforms before outlining
- Improve replicated tensor sharding (RTS) optimizers for pipelining
- Add reshape to correct output shape for binary/mul gradoppattern outputs
- Avoid double scoping TensorIds when creating new outputs for operations within subgraphs
- Improve pre/post loss and RemoteLoad/RemoteStore scheduling
- Improve catching all types of (indirectly or directly) unmodifiable tensors for inplacing
- Ensure input and output data type of replicated allgather remains identical
- Remove extra outline attributes from ops that do not need them
- Add custom reshape with one input and one attribute
- Migrate DepthToSpace and SpaceToDepth to onnxpasses
- Add C code comments to Python docs for cases where C and Python functionality are the same.
- Add support for negative axes in reduce ops
- Move graph compiled logging to info level
- Add logging feature: Compile time breakdown into main components
- Add ONNX $\rightarrow$ ONNX transformations.
- Add Mod and Remainder Operators
- Run deserialized executable on a different device
- Make loadEngineAndConnectStreams public
- Add isAttached function to DeviceInfo
- Allow resetting device associated with session
- Enable loading OfflineIpu executables into Ipu device
- Add assert operation to PopART using Poplar's Abort and AbortOnCondition program
- Added hashing for all session option values (improved serialization)
- Defer cached engine loading to `Session::prepareDevice`

Bug Fixes

- Fix bug where it could fail to retrieve subgraph virtual graph IDs
- Improve unwinding and fix cases of circular tensor dependencies
- Fix batch axis pickup on weights for batch serialisation
- Fixes for engine caching of optimizer state data
- Make instancenorm, convolutions, LSTM/GRU re-setupable
- Fix aliasing (fwd/bwd region mapping) for elementwise broadcasting
- Fix replicated tensor sharding (RTS) for elementwise broadcasting
- Fix batch serialization in the case where graph replication factor is greater than zero
- Fix incorrect computation in the 'atan2' and 'fmod' gradient operators
- Fix the scaling of the output of the IdentityLossGradOp when the 'reduction' option is set to popart.ReductionType.Mean
- Fix an outlining bug related to pruning

6.3 PopTorch Changelog

6.3.1 2.1.0+18411

- Support for `torch.unbind`
- Add option to set poptorch.Options using options specified in a config file.
- Add `mode=poptorch.DataLoaderMode.AsyncRebatched`
- Support for PopART name scopes via `poptorch.NameScope`
- Add mixed precision automatic casting
- Support for `torch.cross`
- Support for `torch.functional.one_hot`
- Support for `torch.int8 data types`
- Support for `torch.median`
- Support for `torch.index_select`
- Support for `torch.scatter_add`
- Add `poptorch.Options.Precision.enableFloatingPointExceptions` to control floating point exception behaviour
- Support for inplace changes to inputs.
- Add option to log the number of IPU cycles used in executing the main graph
- Support for `torch.nn.GRU`
- Add automatic loss scaling option which can be enabled via `poptorch.Options.Training.setAutomaticLossScaling`. (Preview)
- Add `poptorch.BlockFunction` decorating for assigning an existing function to a block.
- Add mechanism for inspecting arbitrary tensors
- Add custom operator for CTC beam search decoding: `poptorch.ctc_beam_search_decoder`
- Add a separate tensor variant (now default) to the SGD optimiser.
- Add a TensorFlow variant to the RMSProp optimiser.
- Use of SGD via PyTorch's or PopTorch's API now results in use of the new separate tensor variant by default. To revert to the previous default variant, use `poptorch.optim.SGD` with `use_combined_accum=True`.

6.3.2 2.0.0

- Added support for the following activation functions:
  - torch.nn.acosh
  - torch.nn.asinh
  - torch.nn.atanh
  - torch.nn.Hardshrink
  - torch.nn.SiLU
  - torch.nn.Softplus
  - torch.nn.Softshrink
  - torch.nn.Threshold

- Add support for the following random sampling operations:
  - torch.bernoulli
  - torch.distributions.Bernoulli
- Add experimental support for torch.nn.CTCLoss
- Added Adam optimizer
- Added support for `torch.nn.AdaptiveAvgPool1d`, `torch.nn.AdaptiveAvgPool3d`
- Migrated to PyTorch version 1.7.1
- Add support for `aten::index`, `aten::index_put_`
- Add support for `torch.zeros_like`, `torch.ones_like`
- Allow the user to specify which Optimizer attributes are constant or not.
- Allow the user to specify `mode=poptorch.DataLoaderMode.Async` in `poptorch.DataLoader` constructor instead of explicitly creating an AsynchronousDataAccessor
- Add support for `torch.nn.EmbeddingBag`
- Added support for `torch.clamp_max` and `torch.clamp_min`
- Add support for `torch.min(tensor, dim=., keepdim=.)` and `torch.max(tensor, dim=., keepdim=.)` overloads.
- Add support for `poptorch.isRunningOnIpu`. This function returns True when executing on IPU and False when executing the model outside IPU scope.
- Add support for `torch.amax` and `torch.amin`
- Add support for attributes in custom ops.
- Add support for precompilation and reloading exported executables (`poptorch.PoplarExecutor.compileAndExport` and `poptorch.load`)
- Add support for slices with variable start index (slice size must be constant).
- Add `ipuHardwareVersion` function to read the version of the IPU hardware present on the system.
- Changed default targetd Ipu version for the model and offline compilation to 2.
- Changed `accumulationReductionType(reduction)` option to now apply to replication reduction as well
- Add environment variable `POPTORCH_CACHE_DIR`

6.4 Poplar Changelog

6.4.1 2.1.0+145407

New features

- Many compile time improvements across graph construction, engine compilation and profiling
- Made exchange code relocatable, so it will be copied into the executable region instead of the model going out of memory
- Added support for replication when using the IPUModel
- Multi-IPU support for a single remote buffer
- New improved bootloader for faster binary load times
- Extended stream callback interface that support waiting and resuming

Bugs fixes

- Fixed a quadratic compile time behaviour when applying element constraints
- Print the correct link register in crash dumps
- Fixed an issue where autoReport would fail silently
- Fixed a codegen issue where excessive sync instructions would be generated when profiling
- Better error handling when setting the print stream on an engine
- Fixed out of bound access when using replica subset
- Fixed an overflow of the cycle counter
- Fixed a data race that could occur when a stream callback calls prefetch and complete simultaneously
- Fixed a double move that could happen during data flow analysis
- Fixed some compilation warnings when compiled with clang
- Allow the Poplar SDK to be installed in a different tree structure and still find the libraries
- Skip execution profiling when all replicas are marked as not to be profiled
- Made an error during code lowering less cryptic
- Fixed an assertion failure when an unconnected data stream is used
- Fixed a crash when using the experimental profiler format
- Added a missing WriteUndef that was causing tensors to remain always live
- Clear remote buffers during device prepare
- Sorted some graph state because it was causing compilation to not be deterministic across machines
- Variables that require init were allocated in the uninitialisable region when they should not
- Fixed an issue that was preventing executables larger than 32GB from being loaded
- Added a missing implicit sync during profiling that was causing the profile to appear mangled
- Fixed latency spikes seen on single IPU models
- Removed an incorrect assertion that was firing for some broadcast streams
- Corrected calculation of whether a stream is exclusive on CPU if there are multiple copies
- Avoid initialisation of already allocated remote buffers
- Fixed a crash in Poplar when using NUMA aware parallel callbacks

Other improvements

- Support If programs for integer types other than BOOL
- Reduce memory allocation overheads
- Updated assembly programming guide for Mk2 (and renamed to Vertex Programming Guide)
- Document IPU specific builtins.
- Improved sub-graph replication efficiency for sparse layers
- More support for merging host copies on the device
- Exposed WORKER_SCRATCH_SIZE for customer vertices
- Report all variables involved in exchanges
- Check for and report SOC errors by default

- Better validation and error reporting for sync misconfigurations
- Enable gateway mode when a target is created from a device with one
- Added ability to determine if a IPU is connected to a IPU machine gateway
- Support for multiple memory configurations with IPU-M2000s
- Optimise runtime by performing some computation at compile time
- Better log output on host sync timeout shows what each tile's state is
- Better liveness analysis when profiling
- Added PVTI tracepoints to Poplar compilation and more to the runtime
- Optimised some instrumentation codegen
- Added more Repeat program options to Poplar
- Profiler now collects the engine options used during compilation
- Enabled write combining by default when on gateway systems
- Optimised the copy library by extending the range supported by the fastest vertices
- Optimised the host exchange codegen
- Added documentation of the exchange planning options
- Optimised device memory usage by the service tables when on gateway systems
- Optimised speed of error polling code in the case there is no error
- Made the timeout after which we check for exceptions / SOC errors configurable
- Add date and time information to the profiler
- Allow write combining on Mk1 devices
- Improved logging for Poplar's HSP messages
- Improved data flow analysis allows deducing the sync ID instead of querying the host for it
- Include CPU usage in the logging for compilation
- Added some basic host memory plotting to PVTI during compilation
- Improved the documentation for loops and repeats
- Added a new Vertex type: MultiVertex for sharing vertex state between instances
- Improved how the sub-graph replicator handles constants

6.4.2 2.0.1

New features

- None.

Bugs fixes

- Fixed a memory leak that occurred on each `Engine::run` invocation

Other improvements

- Improved the latency when polling for device exceptions

6.5 Poplar Libraries Changelog

6.5.1 2.1.0

New features

- Many improvements to the time spent in graph construction
- CTC loss inference using a beam search
- Distributed batch norm

Bug fixes

- Deduplicated repeated information in the documentation
- Fixed a bank conflict that could occur in the broadcast vertices
- Fixed an issue where a call to varianceToInvStdDev is slower than equivalent map expression
- Fixed a possible over-write in the cast vertices
- Fixed a bug in the calculation of the RELU in the LSTM layer
- Fixed cases in SLIC codelet where data was being over read
- Fixed a bug in the convolution planner that meant it was not pruning as much of the search space as it could do
- Fixed a liveness issue that caused the input of some convolutions to remain live
- Added missing aXPlusbY vertices
- Fixed a liveness issue where intermediates tensors remain live in reductions
- Correctly account for kernel dilation when computing input range in convolution planner
- Fixed an issue where `poputil::duplicate` with GATHER_AND_PRESERVE_TILE_ORDER_AND_ALIASES does not copy correctly
- Do not map scale tensor in ScaledAdd
- Changed the behaviour when casting to 8-bit types to match the C++ standard
- Fixed a crash that could occur during convolution option validation
- Fixed an overwrite by the BroadcastVectorInnerInPlace vertex
- Fixed some issues when dividing in codelets above a certain range
- Added a missing reduction vertex
- Don't use doubles in the implementations of `sin()` and `cos()`
- Remove unnecessary variables from the public interface that was causing ABI imcompatibilities

Other improvements

- Added scaledAdd specialisations where input types can be both half and float
- Better test coverage for elementwise vertices
- Added the swish non-linearity
- Added the hard sigmoid non-linearity
- Added non-inplace variants for non-linearities
- Added multi-convolution variant of `weightsTransposeChansFlipXY`
- Made the PopLibs RNN activations configurable
- Improved the elementwise library to produce better output groupings
- Add support for (u)int8 dynamic slicing
- Added PVTI tracepoints to PopLibs for graph construction
- Specialisations for inner broadcast vertices with small region sizes
- Use MultiVertex for the TopK vertices and some of the elementwise vertices
- Implement efficient dot products when using the triangle solve library
- Added support for half partials when using the VMAC convolution vertices
- Improved the efficiency of the VMAC vertices
- Added the option to gather the output of a convolution to improve efficiency of following layers
- Added cube root support to popops
- Added support for an outer stride to the reduction vertices
- Added support to the LSTM library for sequence size to be a runtime variable
- Reduced the amount of vertex state for broadcast elementwise ops
- Optimised map expressions that include division
- Added a new operator to cast, pad and normalise image channels
- Added support for processing the WU matmuls in the LSTM/RNN/GRU layers in mini batches
- Better tile mapping for vertices used for loss calculations
- Better reduction output mapping to avoid subword writes
- Added support for the error function (erf)
- Better worker utilisation for the MultiUpdate and MultiAdd vertices
- Add support to the pooling library to allow for 32-bit partials when given 16-bit input

6.5.2 2.0.0

New features

- Added support for CTC loss in training graphs
- Added support for Faster Transformer using dynamic sequence padding
- Reimplemented TopK and sort using a bitonic sort algorithm
- Extended LSTM and GRU support to better handle layers with large sequences
- Added support for Cholesky decomposition

- Early Access POD128 (2xPOD64) and 2xPOD16 support for replica size 4 IPUs as demonstrated by RN50 and Bert L. Enabled by 3 phase allreduce where phase 2 crosses the gateway links.
- Early Access Grouped collectives API enabling collective operations on a sub-group of all replicas constituting the application e.g. all replicas within a rack (over IPU links) or across racks over gateway links only.

Bug fixes

- Fixed a bug where doing `abs` on an int treating it as a float
- Fixed a bug in the embeddings where the wrong vertex was chosen
- Reduced the probability of overflow because of Block Sparse masking
- Fixed some performance issues with the `sparse_fc_layer` tool
- Fix problems in block sparse matmul documentation
- Fixed an overflow in the transpose vertex that caused a tile exception
- Fixed the operators of SlicePlan when used in an associative container
- Fixed a bug where the wrong vertex was chosen for a fully connected layer that had different input and output type
- Instantiate histogram codelets correctly for absolute input values
- Remove internal API from the public convolution header
- Fixed a bug that caused `popops::scaledAddTo(a, X, b, Y)` to fail when X is integer type
- Fixed a compile time explosion when planning certain convolution
- Updated documentation to fix broken links
- Fixed discrepancies in the 1x1 convolution vertex cycle estimates
- Fix histogram graph construction to allow for multi-dimensional tensors
- Force convolution groups to be 1 when the outer product vertex is used
- Fixed reference to “Unnamed Group” in the Poplibs documentation
- Fixed prearranges using the triangular solver with better tile mappings
- Clarified unclear documentation of the `remapOutputTensor` convolution option
- Fixed an error with the API in the popsparse documentation

Other improvements

- Compile time improvements during graph construction
- Added support for casting between int8 and fp16/fp32
- Add a grain size to created output tensor for convolutions if it cannot be based off the input operand tensors
- Improve calculation of the division of workers in the elementwise operations
- Improved the triangular solver layer with a planner and allocation functions
- Update the planner to specialise the convolution partial types based on the candidate vertex type
- Optimise binary elementwise operations that return a boolean
- Added non-inplace variants for non-linearities
- Added enhanced debug information to poprand
- Moved PopLibs collective operations to GCL (see known issues for API change)

- Updated collectives documentation to reflect this move
- Improved logging during graph constructions
- Improved logging of convolutions
- Added `popops::hasInfOrNan` and `popops::hasInf` operators
- Added mixed precision version of aXMinusbY codelet
- Added support for elementwise equality of 16-bit integer types
- Improved the code generation of the `popops::iota` op
- Added support for reduce `LogAdd` in the reduction library
- Improve documentation on what types are supported by `popops::fill`
- Added an introduction to PopLibs to the README

6.6 PopDist Changelog

6.6.1 2.1.0

New features

- Support offline mode with PopTorch without attaching to device.
- Prevent `poptorch.Options.Distributed` being changed when using PopDist.
- Update to use new TensorFlow `IPUConfig` option configuration API.

6.6.2 2.0.0

New features

- Added documentation
- PopTorch support
- Improved all user error messages
- ipus_per_replica is now optional when calling `getDeviceId`

6.7 PopRun Changelog

6.7.1 2.1.0

New features

- Show full hostnames after the topology table if they cannot fit inside the table.
- Added command-line arguments for additional V-IPU options: `--ipu-link-routing-type`, `--gw-link-routing-type` and `--sync-type`.
- Improved error reporting when user program is missing from the command-line invocation.
- Added support for passing an environment variable to a specific instance by using `--instance-mpi-local-args=<instance-index>:-x VAR=VALUE`.
- Added initial support for the Slurm workload manager. All the resources allocated by Slurm are made available to PopRun.

- Removed dependency on the user locale. Avoids crashing in the case of an incorrectly configured user locale.
- Improved NUMA node binding when using cpusets. Only the NUMA nodes allowed by the current cpuset are used.
- Forward V-IPU timeout argument `--vipu-server-timeout` to IPUoF by internally passing the environment variable `IPUOF_VIPU_API_TIMEOUT`.
- Improved SSH error reporting. Instead of hanging on authentication issues, a clear error is reported.
- Automatically enable the gateway mode target option when using V-IPU.
- Added support for running programs in the current working directory without a `./` prefix for consistency with `mpirun`.
- Automatically enable NUMA awareness when there is more than one instance per host.
- Support passing `--mpi-local-args` and `--mpi-global-args` multiple times by merging the values.
- Verify the final state of partition after creation/reset. An error is reported if the partition was not created/reset correctly.
- Get V-IPU server address from local V-IPU configuration if not specified as command-line argument.
- Set the target options based on values reported by the V-IPU server.

6.8 Libpva Library Changelog

6.8.1 2.0.0

New features

- Preview version of the PopVision Analyser Library for programmatic access to the `profile.pop` report. API provide for C++ and Python applications.

Bug fixes

None

6.9 TensorFlow Changelog

6.9.1 2.1.0

New features

- Added `tensorflow.python.ipu.config.IPUConfig`, a new IPU system options configuration API designed for usability which will eventually replace the old API.
- Improved the `ON_DEMAND` IPU connection type to wait for available IPUs in the system.
- Improved support and performance of recomputation checkpoints in pipelined models including `RecomputationCheckpoint` Keras layer
- Added an option to accumulate pipeline results, with an ability to control the accumulation data type, in order to improve performance.
- Added support for configurable activations in IPU Keras RNN layers.
- Added PVTI integration into the XLA compiler.
- XLA compile time optimisations.

- Add experimental support for distributed batch normalisation.
- Support for Keras `Upsample2D` for nearest-neighbour and bilinear interpolations.
- Improved integration with PopVision Graph Analyzer tool.
- Performance optimisations for convolution neural networks.
- Implementation of CTC beam search including `ipu.keras.layers.CTCInferenceLayer` and `ipu.keras.layers.CTCPredictionsLayer` Keras layers.
- Support for resetting the IPU system configuration within a Python process to allow for different configurations between training and inference.
- Support for setting the same seed for the hardware random number on all model replicas.
- Support for on-device assert operations.
- New IPU specific operations: `tf.python.ipu.statistics_ops.histogram`, `tf.python.ipu.statistics_ops.histogram_update`, `tf.python.ipu.image_ops.normalise_image`, `tf.python.ipu.nn_ops.hard_sigmoid`, `tf.python.ipu.nn_ops.swish`, `tf.python.ipu.slicing_ops.sequence_slice`, `tf.python.ipu.slicing_ops.sequence_slice_pack` and `tf.python.ipu.slicing_ops.sequence_slice_unpack`.
- Improved performance of `tf.math.erf`, reduction operations, `tf.python.ipu.rand_ops.dropout` and the `ipu.keras.layers.Dropout` Keras layer.
- Provided TensorFlow 2 version of `ipu.keras.optimizers.GradientAccumulationOptimizer` and `ipu.keras.optimizers.MapGradientOptimizer` optimizers.
- Added automatic checking of tensor sizes before placing them IO tiles in order to avoid out-of-memory errors.
- Improved Poplar tensor allocation, including conditional operations, to improve model performance.
- Improved documentation for distribution strategies.
- Added CPU feature guard to give a meaningful error message when a built TensorFlow wheel is not compatible with the CPU architecture in the system.
- Add support for setting prefetch depth with estimators.

Bug fixes

- Fixed numerical issues where random number generator operations were fused incorrectly.
- Support for 8-bit integer for IPU Keras Models.
- Fixed a crash where if a model variable was used in different pipeline stages it could cause a data type mismatch error.

6.9.2 2.0.1

New features

- None.

Bug fixes

- Clear the executor state after failed engine execution. —

CHAPTER

SEVEN

KNOWN ISSUES

The following section will detail known issues in v2.1.0.
Each product will be detailed separately.

7.1 Driver & Utilities known issues

7.1.1 1.0.52

None.

7.1.2 1.0.51

- `gc-monitor` help incorrectly documents the `ipu-util` option (listed as `ipu-util-curr`, whereas the correct option is `ipu-util`). Upon entering the incorrect value, `gc-monitor` will provide an error message with the correct value.

7.2 PopART known issues

7.2.1 2.1.0+145366

- Using a conv op with the value of padding greater than the convolution kernel size will result in an error when training. Use a pad op instead for the excess padding.

7.2.2 2.0.1

None.

7.3 PopTorch known issues

7.3.1 2.1.0+18411

None.

7.3.2 2.0.0

None.

7.4 Poplar known issues

7.4.1 2.1.0+145407

None.

7.4.2 2.0.1

- It is not safe to create a DeviceManager while already attached to a Device; this limitation is not currently documented.

7.5 Poplar Libraries known issues

7.5.1 2.1.0

None.

7.5.2 2.0.0

None.

7.6 PopDist known issues

7.6.1 2.1.0

None.

7.6.2 2.0.0

None.

7.7 PopRun known issues

7.7.1 2.1.0

None.

7.7.2 2.0.0

None.

7.8 Libpva Library known issues

7.8.1 2.0.0

None.

7.9 TensorFlow known issues

7.9.1 2.1.0

- IPU Keras API ignores the `shuffle` and `verbose` arguments.
- IPU Keras per-output losses are not averaged, instead last loss is returned.

7.9.2 2.0.1

None

CHAPTER

EIGHT

COMPATIBILITY CHANGES

The following section will detail compatibility changes in v2.1.0

Product	Paragraph
Driver & Utilities	*Driver & Utilities compatibility changes* (page 32)
PopART	*PopART compatibility changes* (page 32)
PopTorch	*PopTorch compatibility changes* (page 33)
Poplar	*Poplar compatibility changes* (page 33)
Poplar Libraries	*Poplar Libraries compatibility changes* (page 34)
PopDist/PopRun	*PopRun/PopDist compatibility changes* (page 34)
Libpva Library	*Libpva Library compatibility changes* (page 35)
TensorFlow	*TensorFlow compatibility changes* (page 35)

8.1 Driver & Utilities Compatibility changes

8.1.1 1.0.52

None.

8.1.2 1.0.51

None.

8.2 PopART Compatibility changes

8.2.1 2.1.0+145366

- [API] Replace the poprithms anneal scheduler with the poprithms shift scheduler
- [API] Remove Undefined default from TensorLocation, TensorStorage and introduce `OptionalTensorLocation`
- [API] Remove deprecated `SessionOption` `accumulationReductionType
- [API] Remove access to the tile mapping from the public API
- [API] Deprecate grouped matmul `SessionOption`

- [API] Deprecate loss grad op output scaling behaviour when `replicatedGraphCount` > 1 and reduction is `ReductionType::Mean`
- [API] Expose the automatic loss scaling hyperparameters to the user via `SessionOptions`
- [API] Remove deprecated functions that took `debugPrefix`

8.2.2 2.0.1

- [API] Deprecate accumulationReductionType SessionOption
- [API] Remove previously deprecated LegacyOpFactoryFunction
- [API] Deprecate `Opx::debugPrefix`

8.3 PopTorch Compatibility changes

8.3.1 2.1.0+18411

- Removed `Options.Popart` which was deprecated in v2.0 and replaced with `Options._Popart`
- Removed `MultiConvPartialsType` which was deprecated in v2.0
- Deprecated `poptorch.Options.Training.accumulationReductionType` in favour of `poptorch.Options.Training.accumulationAndReplicationReductionType`
- Deprecated `runningVarianceAlwaysFloat` in favour of `runningStatisticsAlwaysFloat` in `poptorch.Options.Precision`, as this new option computes both the running mean and variance in FP32 when this option is set to `True`.

8.3.2 2.0.0

- Deprecated `Options.Popart`, `Options._Popart` may be used experimentally.

8.4 Poplar Compatibility changes

8.4.1 2.1.0+145407

- Removed deprecated APIs and engine options from Poplar
- Deprecated use of V1/V2 profile format
- The following APIs have been deprecated:
  - `poplar::ProfileValue`
  - `poplar::program::Sequence` variadic constructor.
  - `poplar::Engine::getGraphProfile`, `poplar::Engine::getExecutionProfile`, `poplar::Engine::getProfile` and `poplar::Engine::resetExecutionProfile` in favour of the PVA library instead.
  - `poplar::Graph::trace`, PVTI has used to track graph construction time instead
- The following engine options have been deprecated:
  - `target.maxStreamCallbackThreadsPerNumaNode`, use the `streamCallbacks.*` options instead
  - `profile.format` when format is `v1` and `experimental`

8.4.2 2.0.1

None.

8.5 Poplar Libraries Compatibility changes

8.5.1 2.1.0

- The following methods have been deprecated:
  - `poplin::preplanConvolutions` and `poplin::preplanMatMuls`, use `poplin::preplan` instead
  - The fields `dataType`, `batchSize`, `timeSteps`, `layerSizes` in `popnn::gru::GruParams` and `popnn::lstm::LstmParams`, these fields now exist in the `RnnParams` struct.
  - `poplin::rnn::RnnParams::timeSteps`, use `maxTimeSteps` instead
  - The GRU and auGRU overloads that take a `realTimeSteps` parameter
  - The `popops::reduce` overload that takes a `ComputeSet`, use `popops::reduceMany` instead

8.5.2 2.0.0

- The following APIs have been deprecated:
  - `popops::reduceScatter(. . . , popops::Operation, . . . )`
  - `popops::allReduce(. . . , popops::Operation, . . . )`
  - `poplin::createTriangularSolveInputLHS(. . . , std::size_t blockSize, . . . )`
  - `poplin::createTriangularSolveInputRHS(. . . , std::size_t blockSize, . . . )`
  - `poplin::triangularSolve(. . . , std::size_t blockSize, . . . )`
  - `poplin::getTriangularSolveMatMulPrePlanParameters(. . . , std::size_t blockSize, . . . )`

8.6 PopDist Compatibility changes

8.6.1 2.1.0

None.

8.7 PopRun Compatibility changes

8.7.1 2.1.0

None.

8.8 Libpva Library Compatibility changes

8.8.1 2.0.0

None.

8.9 TensorFlow Compatibility changes

8.9.1 2.1.0

- `replication_factor` does not need to be specified for `IPUInfeedQueue` and `IPUOutfeedQueue`.
- See the `API changes` section in the TensorFlow documentation for full details.

8.9.2 2.0.1

None.

CHAPTER

NINE

APPENDIX

9.1 Appendix A : Additional requirements

9.1.1 PopVision Graph Analyser

- To be able to view profiling reports generated by SDK v2.1.0, PopVision Graph Analyser v2.4 and PopVision System Analyser v1.2 are required.

9.1.2 TensorFlow

To correctly execute TensorFlow code please ensure:

Intel platforms

- Use Python 3.6 as minimum version
- A CPU compatible with the AVX-512 instruction set is needed.

AMD plaforms

- Use Python 3.6 as minimum version
- A CPU compatible with the Znver1 instruction set is needed.