API Collection

BNNS

Implement and run neural networks for training and inference.

Overview

The Accelerate framework's BNNS library is a collection of functions that you use to construct neural networks for training and inference. BNNS provides routines optimized for high performance and low energy consumption across all Apple platforms.

The BNNSGraph API provides the means to build CPU based neural networks from the mlmodelc file that Xcode compiles from an ML package.

BNNSGraph allows the BNNS library to execute entire networks rather than individual layers. By considering the full model, BNNS can apply graph-level optimizations such as omitting unnecessary copy operations, fusing computational kernels, and avoiding computing redundant information. Furthermore, BNNS can optimize data layouts for constant data — such as convolution weights — and this allows the fastest execution on specific hardware. These optimizations can represent significant performance and energy-efficiency improvements.

Topics

Building graphs in Swift

```
static func makeContext(options: BNNSGraph.CompileOptions, (inout
BNNSGraph.Builder) -> [any BNNSGraph.TensorDescriptor]) throws ->
BNNSGraph.Context
```

Returns a new context that wraps a graph object that the given closure defines.

`struct Builder`

A structure thats provides a closure you can use to define the arguments and operations of a BNNS Graph.

`struct Tensor`

A structure that represents an abstract handle to a tensor that you use within a BNNSGraph `.makeContext` closure.

{} Supporting real-time ML inference on the CPU

Add real-time digital signal processing to apps like Logic Pro X and GarageBand with the BNNS Graph API.

Creating and executing graphs in Swift

`class Context`

A wrapper around a compiled graph object that adds a required modifiable context to support dynamically sized models and set execute-time options.

Compiling a graph object

`struct bnns_graph_t`

The compiled graph object.

`func BNNSGraphCompileFromFile(UnsafePointer<CChar>, UnsafePointer<CChar>?, bnns_graph_compile_options_t) -> bnns_graph_t`

Compiles a source mlmodelc file to a graph object.

Specifying and querying compilation options

`struct bnns_graph_compile_options_t`

The compilation options that BNNS uses when compiling a source mlmodelc file to a graph object.

`func BNNSGraphCompileOptionsMakeDefault() -> bnns_graph_compile_options_t`

Returns an allocated compilation options object with default values.

`func BNNSGraphCompileOptionsDestroy(bnns_graph_compile_options_t)`

Destroys the specified compilation options object.

```
func BNNSGraphCompileOptionsSetOutputPath(bnns_graph_compile_options_t,
UnsafePointer<CChar>?)
```

Sets the option for graph compilation to generate the graph object directly to the specified file.

```
func BNNSGraphCompileOptionsGetOutputPath(bnns_graph_compile_options_t)
-> UnsafePointer<CChar>?
```

Returns the option for the compiled graph's output path.

```
func BNNSGraphCompileOptionsSetOutputFD(bnns_graph_compile_options_t,
Int32)
```

Sets the option for graph compilation to generate the graph object directly to the specified file descriptor.

```
func BNNSGraphCompileOptionsGetOutputFD(bnns_graph_compile_options_t) -
> Int32
```

Returns the option for the compiled graph's output file descriptor.

```
func BNNSGraphCompileOptionsSetTargetSingleThread(bnns_graph_compile
_options_t, Bool)
```

Sets the option for the compiled graph to execute on a single thread.

```
func BNNSGraphCompileOptionsGetTargetSingleThread(bnns_graph_compile
_options_t) -> Bool
```

Returns the option for the compiled graph to execute on a single thread.

```
func BNNSGraphCompileOptionsSetOptimizationPreference(bnns_graph
_compile_options_t, BNNSGraphOptimizationPreference)
```

Sets the option for the compiled graph to optimize for either size or performance.

```
func BNNSGraphCompileOptionsGetOptimizationPreference(bnns_graph
_compile_options_t) -> BNNSGraphOptimizationPreference
```

Returns the option for the compiled graph to optimize for either size or performance.

```
struct BNNSGraphOptimizationPreference
```

Constants that describe the compilation optimization preference.

```
func BNNSGraphCompileOptionsSetGenerateDebugInfo(bnns_graph_compile
_options_t, Bool)
```

Sets the option for the compiled graph to include debugging information.

`func BNNSGraphCompileOptionsGetGenerateDebugInfo(bnns_graph_compile_options_t) -> Bool`

Returns the option for the compiled graph to include debugging information.

`var BNNSTargetSystemGeneric: BNNSTargetSystem`

Specifying a graph's compile-time message callback

`func BNNSGraphCompileOptionsSetMessageLogMask(bnns_graph_compile_options_t, UInt32)`

Sets the mask for compile-time messages.

`func BNNSGraphContextSetMessageLogMask(bnns_graph_context_t, UInt32) -> Int32`

Sets mask for log messages that are logged (either via `os_log` or the user specified callback)

`struct BNNSGraphMessageLevel`

Constants that specify the mask for compile-time messages.

`func BNNSGraphCompileOptionsSetMessageLogCallback(bnns_graph_compile_options_t, bnns_graph_compile_message_fn_t, UnsafeMutablePointer<bnns_user_message_data_t>?)`

Specifies a customized callback function that reports compile-time messages.

`typealias bnns_graph_compile_message_fn_t`

The graph compile-message logging callback function.

`struct bnns_user_message_data_t`

Additional user-defined logging argument for message-logging callbacks.

Querying a graph's properties

`func BNNSGraphGetArgumentIntents(bnns_graph_t, UnsafePointer<CChar>?, Int, UnsafeMutablePointer<BNNSGraphArgumentIntent>) -> Int32`

Extracts the intents of arguments for the given function argument.

`struct BNNSGraphArgumentIntent`

Constants that describe argument intents.

```
func BNNSGraphGetArgumentCount(bnns_graph_t, UnsafePointer<CChar>?) -> Int
```

Returns the number of arguments for the given function argument.

```
func BNNSGraphGetArgumentNames(bnns_graph_t, UnsafePointer<CChar>?, Int, UnsafeMutablePointer<UnsafePointer<CChar>?>) -> Int32
```

Extracts the names of arguments for the given function argument.

```
func BNNSGraphGetFunctionCount(bnns_graph_t) -> Int
```

Returns the number of callable functions in the specified graph.

```
func BNNSGraphGetFunctionNames(bnns_graph_t, Int, UnsafeMutablePointer<UnsafePointer<CChar>?>) -> Int32
```

Extracts the names of callable functions in the graph.

```
func BNNSGraphGetInputCount(bnns_graph_t, UnsafePointer<CChar>?) -> Int
```

Returns the number of input arguments for the given function argument.

```
func BNNSGraphGetInputNames(bnns_graph_t, UnsafePointer<CChar>?, Int, UnsafeMutablePointer<UnsafePointer<CChar>?>) -> Int32
```

Extracts the names of input arguments for the given function argument.

```
func BNNSGraphGetOutputCount(bnns_graph_t, UnsafePointer<CChar>?) -> Int
```

Returns the number of output arguments for the given function argument.

```
func BNNSGraphGetOutputNames(bnns_graph_t, UnsafePointer<CChar>?, Int, UnsafeMutablePointer<UnsafePointer<CChar>?>) -> Int32
```

Extracts the names of output arguments for the given function argument.

```
func BNNSGraphGetArgumentPosition(bnns_graph_t, UnsafePointer<CChar>?, UnsafePointer<CChar>) -> Int
```

Returns the index into the arguments array for the given function argument.

```
func BNNSGraphGetArgumentInterleaveFactors(bnns_graph_t, UnsafePointer<CChar>?, Int, UnsafeMutablePointer<UnsafePointer<UInt16>?>, UnsafeMutablePointer<Int>) -> Int32
```

Returns the interleave factors for arguments, if present

Creating and destroying a context

```
struct bnns_graph_context_t
```

An object that wraps a compiled graph object.

`func BNNSGraphContextMake(bnns_graph_t) -> bnns_graph_context_t`

Returns an allocated and initialized graph context from the specified graph.

`func BNNSGraphContextMakeStreaming(bnns_graph_t, UnsafePointer<CChar>?, Int, UnsafePointer<BNNSTensor>?) -> bnns_graph_context_t`

Returns an allocated and initialized graph context with streaming support from the specified graph.

`func BNNSGraphContextDestroy(bnns_graph_context_t)`

Destroys the specified graph context.

Specifying and querying a context's properties

`func BNNSGraphContextSetStreamingAdvanceCount(bnns_graph_context_t, Int) -> Int32`

Sets the streaming advancement amount for cases with dynamically shaped inputs.

`func BNNSGraphContextSetArgumentType(bnns_graph_context_t, BNNSGraphArgumentType) -> Int32`

Specifies the argument type for a graph context.

`struct BNNSGraphArgumentType`

Constants that specify the argument type for a graph context.

`func BNNSGraphContextSetDynamicShapes(bnns_graph_context_t, UnsafePointer<CChar>?, Int, UnsafeMutablePointer<bnns_graph_shape_t>) -> Int32`

Specifies the dynamic shapes for a graph and, if possible, infers, the output shapes.

`struct bnns_graph_shape_t`

The specification of the shape of an argument.

`func BNNSGraphContextSetBatchSize(bnns_graph_context_t, UnsafePointer<CChar>?, UInt64) -> Int32`

Sets the batch size for a graph.

`func BNNSGraphContextEnableNanAndInfChecks(bnns_graph_context_t, Bool)`

Specifies that the context checks intermediate tensors for NaNs and infinities.

`func BNNSGraphContextGetWorkspaceSize(bnns_graph_context_t, UnsafePointer<CChar>?) -> Int`

Returns the minimum size, in bytes, of the workspace that graph context execution requires.

`func BNNSGraphContextSetStreamingAdvanceCount(bnns_graph_context_t, Int) -> Int32`

Sets the streaming advancement amount for cases with dynamically shaped inputs.

Specifying a context's execute-time message callback

`func BNNSGraphContextSetMessageLogCallback(bnns_graph_context_t, bnns_graph_execute_message_fn_t, UnsafeMutablePointer<bnns_user_message_data_t>?) -> Int32`

Specifies a customized callback function that reports execution-time messages.

`typealias bnns_graph_execute_message_fn_t`

The graph execute-message logging callback function.

`struct BNNSGraphMessageLevel`

Constants that specify the mask for compile-time messages.

`struct bnns_user_message_data_t`

Additional user-defined logging argument for message-logging callbacks.

Specifying a context's allocation callbacks

`func BNNSGraphContextSetWorkspaceAllocationCallback(bnns_graph_context_t, bnns_graph_realloc_fn_t?, bnns_graph_free_all_fn_t?, Int, UnsafeMutableRawPointer?) -> Int32`

Sets the allocation and deallocation callbacks for internal workspace.

`func BNNSGraphContextSetOutputAllocationCallback(bnns_graph_context_t, bnns_graph_realloc_fn_t?, bnns_graph_free_all_fn_t?, Int, UnsafeMutableRawPointer?) -> Int32`

Sets the allocation and deallocation callbacks for function outputs.

`typealias bnns_graph_realloc_fn_t`

The workspace and output allocation function.

`typealias bnns_graph_free_all_fn_t`

The workspace and output deallocation function.

Specifying and querying a tensor's properties

`struct` **BNNSTensor**

A structure that describes the shape, stride, data type, and, optionally, the memory location of an n-dimensional array.

`func` **BNNSTensorGetAllocationSize**`(UnsafePointer<BNNSTensor>) -> Int`

Returns the minimum allocation size, in bytes, of the specified tensor.

`func` **BNNSGraphContextGetTensor**`(bnns_graph_context_t, UnsafePointer<CChar>?, UnsafePointer<CChar>, Bool, UnsafeMutablePointer<BNNSTensor>) -> Int32`

Sets the properties of a tensor for the specified function argument.

`func` **BNNSGraphTensorFillStrides**`(bnns_graph_t, UnsafePointer<CChar>?, UnsafePointer<CChar>, UnsafeMutablePointer<BNNSTensor>) -> Int32`

Sets the stride of the specifed tensor for compatibility with the given model's input or output argument based on its current shape.

Executing a graph

`func` **BNNSGraphContextExecute**`(bnns_graph_context_t, UnsafePointer<CChar>?, Int, UnsafeMutablePointer<bnns_graph_argument_t>, Int, UnsafeMutablePointer<CChar>?) -> Int32`

Executes the specified function with the given context.

`struct` **BNNSGraphArgumentType**

Constants that specify the argument type for a graph context.

Enumerations

`enum` **BNNS**

An enumeration that acts as a namespace for Swift overlays to BNNS.

`enum` **BNNSGraph**

An enumeration that acts as a namespace for the Swift overlays to BNNS Graph.

Structures

`struct` BNNSDataType

BNNS Data Types.

`struct` BNNSSparsityParameters

`struct` BNNSSparsityType

`struct` BNNSTargetSystem

`struct` bnns_graph_argument_t

Describes data associated with an input or output argument

~~`struct` BNNSImageStackDescriptor~~ Deprecated

~~`struct` BNNSVectorDescriptor~~ Deprecated

Protocols

`protocol` BNNSScalar

Macros

`var` BNNS_MAX_TENSOR_DIMENSION: Int32

Deprecated symbols

Classic BNNS API

See Also

Neural Networks

Training a neural network to recognize digits

Build a simple neural network and train it to recognize randomly generated numbers.