Structure

BNNSGraph.Builder.Tensor

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

iOS 26.0+ | iPadOS 26.0+ | Mac Catalyst | macOS 26.0+ | tvOS 26.0+ | visionOS 26.0+ | watchOS 26.0+

```
struct Tensor<T> where T : BNNSScalar
```

Topics

Operators

`static func .! (BNNSGraph.Builder.Tensor<Bool>) -> BNNSGraph.Builder.Tensor<Bool>`

Adds an element-wise logical not operation to the current graph.

`static func .!= (BNNSGraph.Builder.Tensor<T>, BNNSGraph.Builder.Tensor<T>) -> BNNSGraph.Builder.Tensor<Bool>`

Adds an element-wise inequality operation to the current graph.

`static func .& (BNNSGraph.Builder.Tensor<Bool>, BNNSGraph.Builder.Tensor<Bool>) -> BNNSGraph.Builder.Tensor<Bool>`

Adds an element-wise logical and operation to the current graph.

`static func .== (BNNSGraph.Builder.Tensor<T>, BNNSGraph.Builder.Tensor<T>) -> BNNSGraph.Builder.Tensor<Bool>`

Adds an element-wise equality operation to the current graph.

```
static func .| (BNNSGraph.Builder.Tensor<Bool>, BNNSGraph.Builder.
Tensor<Bool>) -> BNNSGraph.Builder.Tensor<Bool>
```

Adds an element-wise logical or operation to the current graph.

```
static func .< (BNNSGraph.Builder.Tensor<T>, BNNSGraph.Builder.Tensor<T
>) -> BNNSGraph.Builder.Tensor<Bool>
```

Adds an element-wise less than comparison operation to the current graph.

```
static func .^ (BNNSGraph.Builder.Tensor<Bool>, BNNSGraph.Builder.
Tensor<Bool>) -> BNNSGraph.Builder.Tensor<Bool>
```

Adds an element-wise logical xor operation to the current graph.

```
static func .> (BNNSGraph.Builder.Tensor<T>, BNNSGraph.Builder.Tensor<T
>) -> BNNSGraph.Builder.Tensor<Bool>
```

Adds an element-wise greater than comparison operation to the current graph.

```
static func .<= (BNNSGraph.Builder.Tensor<T>, BNNSGraph.Builder.Tensor<
T>) -> BNNSGraph.Builder.Tensor<Bool>
```

Adds an element-wise less than or equal comparison operation to the current graph.

```
static func .>= (BNNSGraph.Builder.Tensor<T>, BNNSGraph.Builder.Tensor<
T>) -> BNNSGraph.Builder.Tensor<Bool>
```

Adds an element-wise greater than or equal comparison operation to the current graph.

Instance Properties

```
var dataType: BNNSDataType?
```

The data type of the tensor.

```
var description: String
```

A textual representation of this instance.

```
var rank: Int?
```

The rank of the tensor.

```
var shape: [Int]?
```

The shape of the tensor.

```
var stride: [Int]?
```

The stride of the tensor.

Instance Methods

```
func abs() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise absolute operation to the current graph.

```
func acos() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise acos operation to the current graph.

```
func acosh() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise atan operation to the current graph.

```
func argMax(axis: Int, keepDimension: Bool) -> BNNSGraph.Builder.Tensor
<Int32>
```

Adds an argmax operation to the current graph.

```
func argMin(axis: Int, keepDimension: Bool) -> BNNSGraph.Builder.Tensor
<Int32>
```

Adds an argmin operation to the current graph.

```
func argSort(axis: Int, sortOrder: BNNSGraph.Builder.SortOrder) ->
BNNSGraph.Builder.Tensor<Int32>
```

Adds an argsort operation to the current graph.

```
func asin() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise asin operation to the current graph.

```
func asinh() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise asinh operation to the current graph.

```
func atan() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise atan operation to the current graph.

```
func atanh() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise atanh operation to the current graph.

```
func batchNorm(mean: some BNNSGraph.Builder.OperationParameter<T>,
variance: some BNNSGraph.Builder.OperationParameter<T>, epsilon: Float)
-> BNNSGraph.Builder.Tensor<T>
```

Adds a batch normalization operation to the current graph.

```
func batchNorm(mean: some BNNSGraph.Builder.OperationParameter<T>,
variance: some BNNSGraph.Builder.OperationParameter<T>, weight: some
BNNSGraph.Builder.OperationParameter<T>, bias: some BNNSGraph.Builder.
OperationParameter<T>, epsilon: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds a batch normalization operation to the current graph.

```
func bidirectionalLSTM(initialHiddenStates: BNNSGraph.Builder.Tensor<T
>, initialCellStates: BNNSGraph.Builder.Tensor<T>, inputHiddenWeight:
BNNSGraph.Builder.Tensor<T>, hiddenHiddenWeight: BNNSGraph.Builder.
Tensor<T>, bias: BNNSGraph.Builder.Tensor<T>, inputHiddenWeightBack:
BNNSGraph.Builder.Tensor<T>, hiddenHiddenWeightBack: BNNSGraph.Builder.
Tensor<T>, biasBack: BNNSGraph.Builder.Tensor<T>, activation: BNNSGraph
.Builder.Activation, recurrentActivation: BNNSGraph.Builder.Activation,
cellActivation: BNNSGraph.Builder.Activation, outputSequence: Bool) ->
(output: BNNSGraph.Builder.Tensor<T>, hiddenStates: BNNSGraph.Builder.
Tensor<T>, memoryStates: BNNSGraph.Builder.Tensor<T>)
```

Adds a bidirectional LSTM operation to the current graph.

```
func cast<U>(to: U.Type) -> BNNSGraph.Builder.Tensor<U>
```

Adds a cast operation to the current graph.

```
func ceil() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise ceiling operation to the current graph.

```
func channelNorm(epsilon: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds a channel normalization operation to the current graph.

```
func channelNorm(weight: some BNNSGraph.Builder.OperationParameter<T>,
bias: some BNNSGraph.Builder.OperationParameter<T>, epsilon: Float) ->
BNNSGraph.Builder.Tensor<T>
```

Adds a channel normalization operation to the current graph.

```
func clampedReLU(alpha: Float, beta: Float) -> BNNSGraph.Builder.Tensor
<T>
```

Adds a Clamped Rectified Linear Unit (ReLU) activation operation to the current graph.

```
func clip(to: ClosedRange<Float>) -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise clip operation to the current graph.

```
func conv(weight: some BNNSGraph.Builder.OperationParameter<T>, strides
: [Int], bias: some BNNSGraph.Builder.OperationParameter<T>, padding:
BNNSGraph.Builder.ConvolutionPadding, dilations: [Int]?, groupCount:
Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a convolution operation to the current graph.

```
func conv(weight: some BNNSGraph.Builder.OperationParameter<T>, strides
: [Int], padding: BNNSGraph.Builder.ConvolutionPadding, dilations: [Int
]?, groupCount: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a convolution operation to the current graph.

```
func convTranspose(weight: some BNNSGraph.Builder.OperationParameter<T
>, strides: [Int], bias: some BNNSGraph.Builder.OperationParameter<T>,
padding: BNNSGraph.Builder.ConvolutionPadding, outputPaddingValues: [
Int]?, dilations: [Int]?, groupCount: Int) -> BNNSGraph.Builder.Tensor<
T>
```

Adds a transposed convolution operation to the current graph.

```
func convTranspose(weight: some BNNSGraph.Builder.OperationParameter<T
>, strides: [Int], padding: BNNSGraph.Builder.ConvolutionPadding,
outputPaddingValues: [Int]?, dilations: [Int]?, groupCount: Int) ->
BNNSGraph.Builder.Tensor<T>
```

Adds a transposed convolution operation to the current graph.

```
func cos() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise cos operation to the current graph.

```
func cosh() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise cosh operation to the current graph.

```
func cumulativeSum(axis: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a cumulative sum operation to the the graph.

```
func elu(alpha: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds an Exponential Linear Unit (ELU) activation operation to the current graph.

```
func erf() -> BNNSGraph.Builder.Tensor<T>
```

Adds an Error Function (erf) activation operation to the current graph.

```
func exp() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise exp operation to the current graph.

```
func exp2() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise exp operation to the current graph.

```
func floor() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise floor operation to the current graph.

```
func fma(y: some BNNSGraph.Builder.OperationParameter<T>, z: some
BNNSGraph.Builder.OperationParameter<T>) -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise fused multiply-add operation to the current graph.

```
func gather(indices: some BNNSGraph.Builder.OperationParameter<Int32>,
axis: Int, batchDimensionCount: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a gather operation to the current graph.

```
func gatherAlongAxis(indices: some BNNSGraph.Builder.OperationParameter
<Int32>, axis: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a gather-along-axis operation to the current graph.

```
func gatherND(indices: some BNNSGraph.Builder.OperationParameter<Int32
>, batchDimensionCount: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a gather-nd operation to the current graph.

```
func gelu() -> BNNSGraph.Builder.Tensor<T>
```

Adds a Gaussian Error Linear Unit (GELU) activation operation to the current graph.

```
func geluSigmoidApproximation() -> BNNSGraph.Builder.Tensor<T>
```

Adds a Gaussian Error Linear Unit (GELU) sigmoid approximation activation operation to the current graph.

```
func geluTanhApproximation() -> BNNSGraph.Builder.Tensor<T>
```

Adds a Gaussian Error Linear Unit (GELU) tanh approximation activation operation to the current graph.

```
func gru(initialHiddenStates: BNNSGraph.Builder.Tensor<T>, inputHidden
Weight: BNNSGraph.Builder.Tensor<T>, hiddenHiddenWeight: BNNSGraph.
Builder.Tensor<T>, bias: BNNSGraph.Builder.Tensor<T>, inputBias:
BNNSGraph.Builder.Tensor<T>, direction: BNNSGraph.Builder.Direction,
activation: BNNSGraph.Builder.Activation, recurrentActivation:
BNNSGraph.Builder.Activation, applyResetGateAfterMatMul: Bool, output
Sequence: Bool) -> (output: BNNSGraph.Builder.Tensor<T>, hiddenStates:
BNNSGraph.Builder.Tensor<T>)
```

Adds a GRU operation to the current graph.

```
func hardSigmoid(alpha: Float, beta: Float) -> BNNSGraph.Builder.Tensor
<T>
```

Adds a hard sigmoid activation operation to the current graph.

```
func hardSwish() -> BNNSGraph.Builder.Tensor<T>
```

Adds a hard swish activation operation to the current graph.

```
func instanceNorm(epsilon: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds a instance normalization operation to the current graph.

```
func instanceNorm(weight: some BNNSGraph.Builder.OperationParameter<T>,
bias: some BNNSGraph.Builder.OperationParameter<T>, epsilon: Float) ->
BNNSGraph.Builder.Tensor<T>
```

Adds a instance normalization operation to the current graph.

```
func l1Norm(axes: [Int], keepDimensions: Bool) -> BNNSGraph.Builder.
Tensor<T>
```

Adds a sum-of-absolutes reduction operation along the given axis operation to the current graph.

```
func l2Norm(axes: [Int], keepDimensions: Bool) -> BNNSGraph.Builder.
Tensor<T>
```

Adds a Euclidean-norm reduction operation along the given axis operation to the current graph.

```
func l2Norm(epsilon: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds an L2 spatial normalization operation to the current graph.

```
func layerNorm(axes: [Int], epsilon: Float) -> BNNSGraph.Builder.Tensor
<T>
```

Adds a layer normalization operation to the current graph.

```
func layerNorm(weight: some BNNSGraph.Builder.OperationParameter<T>,
bias: some BNNSGraph.Builder.OperationParameter<T>, axes: [Int],
epsilon: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds a layer normalization operation to the current graph.

```
func leakyReLU(alpha: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds a Leaky Rectified Linear Unit (ReLU) activation operation to the current graph.

```
func linear(weight: some BNNSGraph.Builder.OperationParameter<T>) ->
BNNSGraph.Builder.Tensor<T>
```

Adds a linear transformation operation to the current graph.

```
func linear(weight: some BNNSGraph.Builder.OperationParameter<T>, bias: some BNNSGraph.Builder.OperationParameter<T>) -> BNNSGraph.Builder.Tensor<T>
```

Adds a linear transformation operation to the current graph.

```
func log(epsilon: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise log operation to the current graph.

```
func logSoftmax(axis: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a log-softmax along the given axis operation to the current graph.

```
func logSum(axes: [Int], keepDimensions: Bool) -> BNNSGraph.Builder.Tensor<T>
```

Adds a log-sum reduction operation along the given axis operation to the current graph.

```
func logSumExp(axes: [Int], keepDimensions: Bool) -> BNNSGraph.Builder.Tensor<T>
```

Adds a log-sum-exp reduction operation along the given axis operation to the current graph.

```
func lstm(initialHiddenStates: BNNSGraph.Builder.Tensor<T>, initialCellStates: BNNSGraph.Builder.Tensor<T>, inputHiddenWeight: BNNSGraph.Builder.Tensor<T>, hiddenHiddenWeight: BNNSGraph.Builder.Tensor<T>, bias: BNNSGraph.Builder.Tensor<T>, direction: BNNSGraph.Builder.Direction, activation: BNNSGraph.Builder.Activation, recurrentActivation: BNNSGraph.Builder.Activation, cellActivation: BNNSGraph.Builder.Activation, outputSequence: Bool) -> (output: BNNSGraph.Builder.Tensor<T>, hiddenStates: BNNSGraph.Builder.Tensor<T>, memoryStates: BNNSGraph.Builder.Tensor<T>)
```

Adds an LSTM operation to the current graph.

```
func matmul(transpose: Bool, other: some BNNSGraph.Builder.OperationParameter<T>, transposeOther: Bool) -> BNNSGraph.Builder.Tensor<T>
```

Adds a matrix-matrix multiplication operation to the current graph.

```
func matmul(transpose: Bool, other: some BNNSGraph.Builder.OperationParameter<T>, transposeOther: Bool, bias: some BNNSGraph.Builder.OperationParameter<T>) -> BNNSGraph.Builder.Tensor<T>
```

Adds a matrix-matrix multiplication operation to the current graph.

```
func max(y: some BNNSGraph.Builder.OperationParameter<T>) -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise maximum operation to the current graph.

```
func maximum(axes: [Int], keepDimensions: Bool) -> BNNSGraph.Builder.
Tensor<T>
```

Adds a maximum reduction operation along the given axis operation to the current graph.

```
func mean(axes: [Int], keepDimensions: Bool) -> BNNSGraph.Builder.
Tensor<T>
```

Adds a mean reduction operation along the given axis operation to the current graph.

```
func min(y: some BNNSGraph.Builder.OperationParameter<T>) -> BNNSGraph.
Builder.Tensor<T>
```

Adds an element-wise minimum operation to the current graph.

```
func minimum(axes: [Int], keepDimensions: Bool) -> BNNSGraph.Builder.
Tensor<T>
```

Adds a minimum reduction operation along the given axis operation to the current graph.

```
func pad(BNNSGraph.Builder.Padding, padding: [Int]) -> BNNSGraph.
Builder.Tensor<T>
```

Adds a padding operation to the graph

```
func pooling(BNNSGraph.Builder.PoolingFunction, kernelSize: [Int],
strides: [Int], padding: BNNSGraph.Builder.PoolingPadding, ceilingMode:
BNNSGraph.Builder.CeilingMode) -> BNNSGraph.Builder.Tensor<T>
```

Adds a pooling operation to the current graph.

```
func pow(y: some BNNSGraph.Builder.OperationParameter<T>) -> BNNSGraph.
Builder.Tensor<T>
```

Adds an element-wise power operation to the current graph.

```
func prelu(alpha: [Float]) -> BNNSGraph.Builder.Tensor<T>
```

Adds a Parametric ReLU (PReLU) activation operation to the current graph.

```
func product(axes: [Int], keepDimensions: Bool) -> BNNSGraph.Builder.
Tensor<T>
```

Adds a product reduction operation along the given axis operation to the current graph.

```
func reciprocal(epsilon: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise reciprocal operation to the current graph.

```
func relu() -> BNNSGraph.Builder.Tensor<T>
```

Adds a Rectified Linear Unit (ReLU) activation operation to the current graph.

```
func relu6() -> BNNSGraph.Builder.Tensor<T>
```

Adds a Rectified Linear Unit 6 (ReLU6) activation operation to the current graph.

```
func reshape(to: some BNNSGraph.Builder.OperationParameter<Int32>) ->
BNNSGraph.Builder.Tensor<T>
```

Adds a dynamic reshape operation to the current graph.

```
func reshape(to: [Int]) -> BNNSGraph.Builder.Tensor<T>
```

Adds a reshape operation to the current graph.

```
func rmsNorm(scale: some BNNSGraph.Builder.OperationParameter<T>,
epsilon: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds an RMS spatial normalization operation to the current graph.

```
func rnn(initialHiddenStates: BNNSGraph.Builder.Tensor<T>, inputHidden
Weight: BNNSGraph.Builder.Tensor<T>, hiddenHiddenWeight: BNNSGraph.
Builder.Tensor<T>, bias: BNNSGraph.Builder.Tensor<T>, direction:
BNNSGraph.Builder.Direction, activation: BNNSGraph.Builder.Activation,
outputSequence: Bool) -> (output: BNNSGraph.Builder.Tensor<T>, hidden
States: BNNSGraph.Builder.Tensor<T>)
```

Adds an RNN operation to the current graph.

```
func round() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise round operation to the current graph.

```
func rsqrt(epsilon: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise rsqrt operation to the current graph.

```
func scaledTanh(alpha: Float, beta: Float) -> BNNSGraph.Builder.Tensor<
T>
```

Adds a scaled tanh activation operation to the current graph.

```
func scatter(updates: BNNSGraph.Builder.Tensor<T>, indices: some
BNNSGraph.Builder.OperationParameter<Int32>, mode: BNNSGraph.Builder.
ScatterMode, axis: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a scatter operation to the current graph.

```
func scatterAlongAxis(updates: BNNSGraph.Builder.Tensor<T>, indices:
some BNNSGraph.Builder.OperationParameter<Int32>, mode: BNNSGraph.
Builder.ScatterMode, axis: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a scatter-along-axis operation to the current graph

```
func scatterND(updates: BNNSGraph.Builder.Tensor<T>, indices: some
BNNSGraph.Builder.OperationParameter<Int32>, mode: BNNSGraph.Builder.
ScatterMode) -> BNNSGraph.Builder.Tensor<T>
```

Adds a scatter-nd operation to the current graph.

```
func select<U>(BNNSGraph.Builder.Tensor<U>, BNNSGraph.Builder.Tensor<U
>) -> BNNSGraph.Builder.Tensor<U>
```

Adds a tensor-tensor select operation to the current graph.

```
func select<U>(BNNSGraph.Builder.Tensor<U>, any BNNSGraph.Builder.
OperationParameter<U>) -> BNNSGraph.Builder.Tensor<U>
```

Adds a tensor-scalar select operation to the current graph.

```
func select<U>(any BNNSGraph.Builder.OperationParameter<U>, BNNSGraph.
Builder.Tensor<U>) -> BNNSGraph.Builder.Tensor<U>
```

Adds a scalar-tensor select operation to the current graph.

```
func sigmoid() -> BNNSGraph.Builder.Tensor<T>
```

Adds a sigmoid activation operation to the current graph.

```
func silu() -> BNNSGraph.Builder.Tensor<T>
```

Adds a Sigmoid Linear Unit (SiLU) activation operation to the current graph.

```
func sin() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise sin operation to the current graph.

```
func sinh() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise sinh operation to the current graph.

```
func softmax(axis: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a softmax along the given axis operation to the current graph.

```
func softplus(alpha: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds a softplus activation operation to the current graph.

```
func softsign() -> BNNSGraph.Builder.Tensor<T>
```

Adds a softsign activation operation to the current graph.

```
func sqrt() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise sqrt operation to the current graph.

```
func squeeze(axis: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds a squeeze operation in the graph.

```
func sum(axes: [Int], keepDimensions: Bool) -> BNNSGraph.Builder.Tensor
<T>
```

Adds a sum reduction operation along the given axis operation to the current graph.

```
func sumOfSquares(axes: [Int], keepDimensions: Bool) -> BNNSGraph.
Builder.Tensor<T>
```

Adds a sum-of-squares reduction operation along the given axis operation to the current graph.

```
func tan() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise tan operation to the current graph.

```
func tanh() -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise hyperbolic tangent operation to the current graph.

```
func tensorShape() -> BNNSGraph.Builder.Tensor<Int32>
```

Adds a shape operation to the current graph.

```
func threshold(to: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds an element-wise round operation to the current graph.

```
func thresholdedReLU(alpha: Float) -> BNNSGraph.Builder.Tensor<T>
```

Adds a Thresholded Leaky Rectified Linear Unit (ReLU) activation operation to the current graph.

```
func topK(Int, axis: Int, findLargest: Bool) -> (values: BNNSGraph.
Builder.Tensor<T>, indices: BNNSGraph.Builder.Tensor<Int32>)
```

Adds a top-k operation to the current graph.

```
func transpose(axes: [Int]) -> BNNSGraph.Builder.Tensor<T>
```

Adds a transpose operation to the current graph.

```
func unsqueeze(axis: Int) -> BNNSGraph.Builder.Tensor<T>
```

Adds an unsqueeze operation in the graph.

Subscripts

```
subscript(any BNNSGraph.Builder.SliceIndex...) -> BNNSGraph.Builder.
Tensor<T>
```

Adds a slice operation to the current graph.

Default Implementations

OperationParameter Implementations

Relationships

Conforms To

`BNNSGraph.Builder.OperationParameter`

Conforms when `T` conforms to `BNNSScalar`.

`BNNSGraph.TensorDescriptor`
`Copyable`
`CustomStringConvertible`

See Also

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.

{} 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.