

Accelerate / BNNSFlags / useClientPointer

Type Property

useClientPointer

A flag that instructs the filter to use pointers to data you provide at creation time.

iOS 10.0+ | iPadOS 10.0+ | Mac Catalyst | macOS 10.12+ | tvOS 10.0+ | visionOS | watchOS 3.0+

```
static var useClientPointer: BNSFlags { get }
```

Discussion

Set this flag to instruct the filter to keep the pointers you provide at creation time and to work directly from this data. In this case, the you must ensure these pointers remain valid through the entire lifetime of the filter.

If not set, the filter creation function allocates buffers and keeps an internal copy of the data. In that case, you don't have to keep the pointers valid after the you've created the filter.

The following code returns a convolution layer with `useClientPointer` set:

[illegible]

```

        table_data: nil,
        table_data_type: .float,
        data_scale: 1,
        data_bias: 0)

let biasDescriptor = BNNSNDArrayDescriptor(flags: BNNSNDArrayFlags(0),
        layout: BNNSDataLayoutVector,
        size: (1, 0, 0, 1, 0, 0, 0, 0),
        stride: (0, 0, 0, 0, 0, 0, 0, 0),
        data: biasPtr.baseAddress,
        data_type: .float,
        table_data: nil,
        table_data_type: .float,
        data_scale: 1,
        data_bias: 0)

var layerParameters = BNNSLayerParametersConvolution(i_desc: inDescriptor,
        w_desc: weightsDescriptor,
        o_desc: outDescriptor,
        bias: biasDescriptor,
        activation: .identity,
        x_stride: 1, y_stride: 1,
        x_dilation_stride: 0, y_dilation_stride: 0,
        x_padding: 0, y_padding: 0,
        groups: 1,
        pad: (0, 0, 0, 0))

var filterParameters = BNNSFilterParameters(flags: BNNSFlags.useClientPointers,
        n_threads: 1,
        alloc_memory: nil,
        free_memory: nil)

return BNNSFilterCreateLayerConvolution(&layerParameters, &filterParameters)
}

```

If either `weights` or `bias` mutate before calling `BNNSFilterApplyBatch(: : : : :)`, the filter uses the new values. However, the convolution layer works with its internal copies of `weights` and `bias` if you pass a `flags` of 0, and subsequent changes have no effect.