Using Metal 2 for Compute

Session 608

Anna Tikhonova, GPU Software Engineer

Metal 2 Ecosystem

Metal API and language

GPU Tools

MetalKit

Metal Performance Shaders

Metal 2

Metal 2 Ecosystem

Metal API and language

GPU Tools

MetalKit

Metal Performance Shaders

Metal 2

Metal Performance Shaders (MPS)

GPU accelerated primitives

- Image Processing
- Linear Algebra
- Machine Learning — Inference

Optimized for iOS

What's New in Metal, Part 2	WWDC 2016
What's New in Metal, Part 2	WWDC 2015

Metal Performance Shaders (MPS)

NEW

GPU accelerated primitives

- Image Processing
- Linear Algebra
- Machine Learning — Inference

Optimized for iOS and macOS

What's New in Metal, Part 2	WWDC 2016
What's New in Metal, Part 2	WWDC 2015

Image Processing

Image Processing

Primitives available in iOS 10

Convolution

Gaussian Blur

Box, Tent

Sobel

Morphology

Lanczos Resampling

Histogram

Equalization and Specification

Median

Thresholding

Transpose

Image Integral

Color Conversion

Gaussian Pyramid

Image Processing

New primitives

NEW

Image Keypoints

Bilinear Rescale

Image Statistics

Element-wise Arithmetic Operations

- With broadcasting

Linear Algebra

Linear Algebra

New primitives

NEW

Matrix-Matrix Multiplication

Matrix-Vector Multiplication

Triangular Matrix Factorization and Linear Solvers

Data Representations

MPSVector

- Interprets data in MTLBuffer as a 1-dimensional array

Data Representations

MPSVector

- Interprets data in MTLBuffer as a 1-dimensional array

MPSMatrix

- Interprets data in MTLBuffer as a rectangular array
- Row-major order

Data Representations

MPSVector

- Interprets data in MTLBuffer as a 1-dimensional array

MPSMatrix

- Interprets data in MTLBuffer as a rectangular array
- Row-major order

MPSTemporaryMatrix

- Allocated from MTLHeap
- Use for most of your intermediate matrices

MPSVector and MPSMatrix

Input types

Single Precision Floating-Point

Half Precision Floating-Point

16-bit Signed Integer

8-bit Signed Integer

MPSVector

Code example

Create a vector of size N

```
// Create a Metal buffer of length N
let buffer = device.makeBuffer(length: N * MemoryLayout<Float32>.size)

// Create a vector descriptor
let descriptor = MPSVectorDescriptor(length: N, dataType: .float32)

// Create a vector with descriptor
let vector = MPSVector(buffer: buffer, descriptor: descriptor)
```

MPSVector

Code example

Create a vector of size N

```swift
// Create a Metal buffer of length N
let buffer = device.makeBuffer(length: N * MemoryLayout<Float32>.size)

// Create a vector descriptor
let descriptor = MPSVectorDescriptor(length: N, dataType: .float32)

// Create a vector with descriptor
let vector = MPSVector(buffer: buffer, descriptor: descriptor)
```

MPSVector

Code example

Create a vector of size N

```
// Create a Metal buffer of length N
let buffer = device.makeBuffer(length: N * MemoryLayout<Float32>.size)

// Create a vector descriptor
let descriptor = MPSVectorDescriptor(length: N, dataType: .float32)

// Create a vector with descriptor
let vector = MPSVector(buffer: buffer, descriptor: descriptor)
```

MPSVector

Code example

Create a vector of size N

```swift
// Create a Metal buffer of length N
let buffer = device.makeBuffer(length: N * MemoryLayout<Float32>.size)

// Create a vector descriptor
let descriptor = MPSVectorDescriptor(length: N, dataType: .float32)

// Create a vector with descriptor
let vector = MPSVector(buffer: buffer, descriptor: descriptor)
```

MPSMatrix

Code example

Create a matrix with M rows and N columns

```
// Get the recommended bytes per row value to use for sizing a Metal buffer
let bytesPerRow = MPSMatrixDescriptor.rowBytes(forColumns: N, dataType: .float32)

// Create a Metal buffer with the recommended bytes per row
let buffer = device.makeBuffer(length: M * bytesPerRow)

// Create a matrix descriptor
let descriptor = MPSMatrixDescriptor(rows: M, columns: N, rowBytes: bytesPerRow,
                                     dataType: .float32)

// Create a matrix with descriptor
let matrix = MPSMatrix(buffer: buffer, descriptor: descriptor)
```

MPSMatrix

Code example

Create a matrix with M rows and N columns

```swift
// Get the recommended bytes per row value to use for sizing a Metal buffer
let bytesPerRow = MPSMatrixDescriptor.rowBytes(forColumns: N, dataType: .float32)

// Create a Metal buffer with the recommended bytes per row
let buffer = device.makeBuffer(length: M * bytesPerRow)

// Create a matrix descriptor
let descriptor = MPSMatrixDescriptor(rows: M, columns: N, rowBytes: bytesPerRow,
                                     dataType: .float32)

// Create a matrix with descriptor
let matrix = MPSMatrix(buffer: buffer, descriptor: descriptor)
```

MPSMatrix

Code example

Create a matrix with M rows and N columns

```
// Get the recommended bytes per row value to use for sizing a Metal buffer
let bytesPerRow = MPSMatrixDescriptor.rowBytes(forColumns: N, dataType: .float32)

// Create a Metal buffer with the recommended bytes per row
let buffer = device.makeBuffer(length: M * bytesPerRow)

// Create a matrix descriptor
let descriptor = MPSMatrixDescriptor(rows: M, columns: N, rowBytes: bytesPerRow,
                                     dataType: .float32)

// Create a matrix with descriptor
let matrix = MPSMatrix(buffer: buffer, descriptor: descriptor)
```

MPSMatrix

Code example

Create a matrix with M rows and N columns

```
// Get the recommended bytes per row value to use for sizing a Metal buffer
let bytesPerRow = MPSMatrixDescriptor.rowBytes(forColumns: N, dataType: .float32)

// Create a Metal buffer with the recommended bytes per row
let buffer = device.makeBuffer(length: M * bytesPerRow)

// Create a matrix descriptor
let descriptor = MPSMatrixDescriptor(rows: M, columns: N, rowBytes: bytesPerRow,
                                     dataType: .float32)

// Create a matrix with descriptor
let matrix = MPSMatrix(buffer: buffer, descriptor: descriptor)
```

Primitives

Matrix-Matrix and Matrix-Vector Multiplication

- API modeled after standard BLAS GEMM and GEMV interfaces

Triangular Matrix Factorization and Linear Solvers

- API modeled after standard LAPACK decomposition and solve interfaces

```swift
// Example: Matrix-Matrix Multiply: C = A B

// Create matrices A, B and C
let A = MPSMatrix(buffer: ABuffer,
                  descriptor: MPSMatrixDescriptor(rows: M, columns: K,
                                                  rowBytes: ARowBytes, dataType: .float32))
let B = MPSMatrix(buffer: BBuffer,
                  descriptor: MPSMatrixDescriptor(rows: K, columns: N,
                                                  rowBytes: BRowBytes, dataType: .float32))
let C = MPSMatrix(buffer: CBuffer,
                  descriptor: MPSMatrixDescriptor(rows: M, columns: N,
                                                  rowBytes: CRowBytes, dataType: .float32))
```

```swift
// Example: Matrix-Matrix Multiply: C = A B

// Perform Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Create a Matrix-Matrix Multiplication kernel
let mmKernel = MPSMatrixMultiplication(device: device, resultRows: M,
                                       resultColumns: N, interiorColumns: K)

// Encode kernel to the command buffer
mmKernel.encode(commandBuffer: commandBuffer, leftMatrix: A,
                rightMatrix: B, resultMatrix: C)

// Tell GPU to start doing the work
commandBuffer.commit()
```

```swift
// Example: Matrix-Matrix Multiply: C = A B

// Perform Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Create a Matrix-Matrix Multiplication kernel
let mmKernel = MPSMatrixMultiplication(device: device, resultRows: M,
                                       resultColumns: N, interiorColumns: K)

// Encode kernel to the command buffer
mmKernel.encode(commandBuffer: commandBuffer, leftMatrix: A,
                rightMatrix: B, resultMatrix: C)

// Tell GPU to start doing the work
commandBuffer.commit()
```

```swift
// Example: Matrix-Matrix Multiply: C = A B

// Perform Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Create a Matrix-Matrix Multiplication kernel
let mmKernel = MPSMatrixMultiplication(device: device, resultRows: M,
                                        resultColumns: N, interiorColumns: K)

// Encode kernel to the command buffer
mmKernel.encode(commandBuffer: commandBuffer, leftMatrix: A,
                rightMatrix: B, resultMatrix: C)

// Tell GPU to start doing the work
commandBuffer.commit()
```

```swift
// Example: Matrix-Matrix Multiply: C = A B

// Perform Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Create a Matrix-Matrix Multiplication kernel
let mmKernel = MPSMatrixMultiplication(device: device, resultRows: M,
                                       resultColumns: N, interiorColumns: K)

// Encode kernel to the command buffer
mmKernel.encode(commandBuffer: commandBuffer, leftMatrix: A,
                rightMatrix: B, resultMatrix: C)

// Tell GPU to start doing the work
commandBuffer.commit()
```

Sample Code

MPSMatrixMultiplication

https://developer.apple.com/library/content/samplecode/MPSMatrixMultiplicationSample

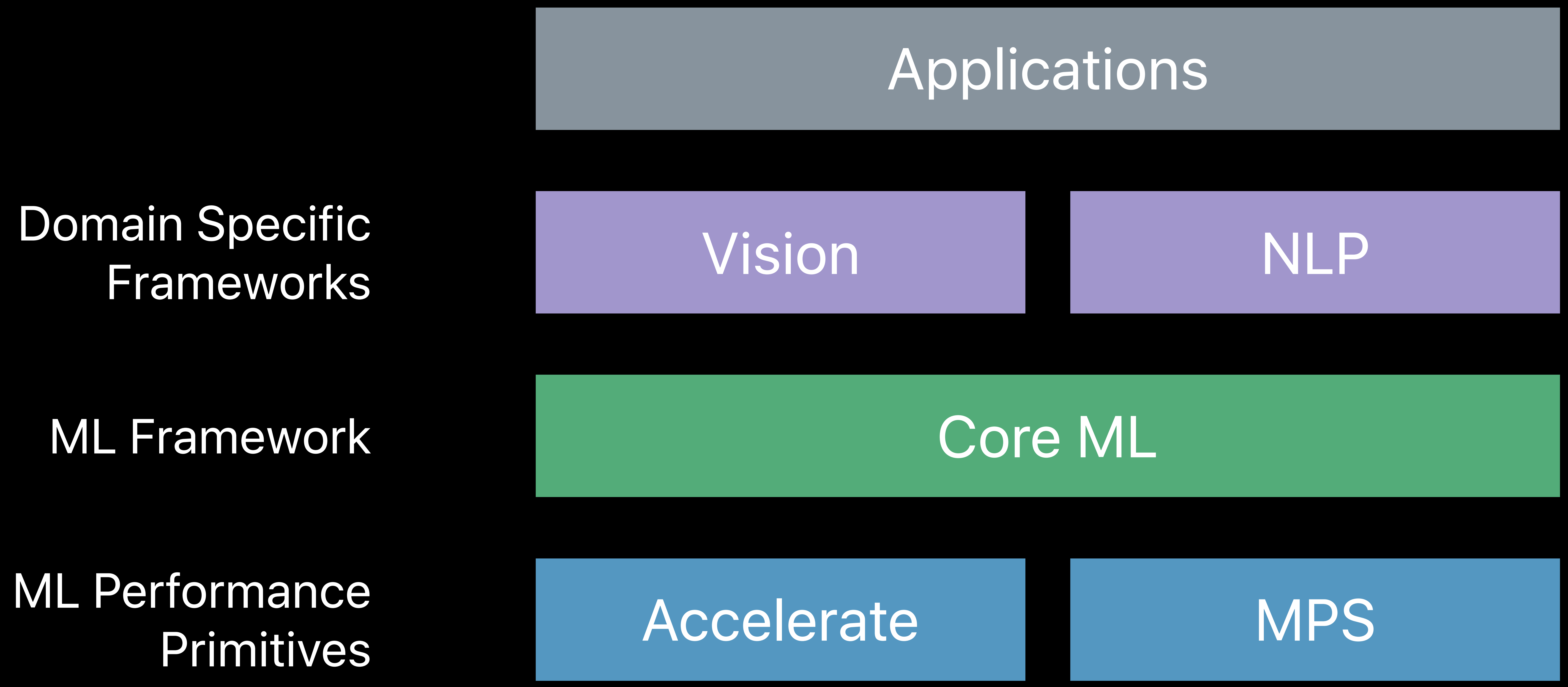
Triangular Matrix Factorization and Linear Solvers

Coming soon

Machine Learning

Machine Learning at Apple

Architecture



What Is Deep Learning?

panda

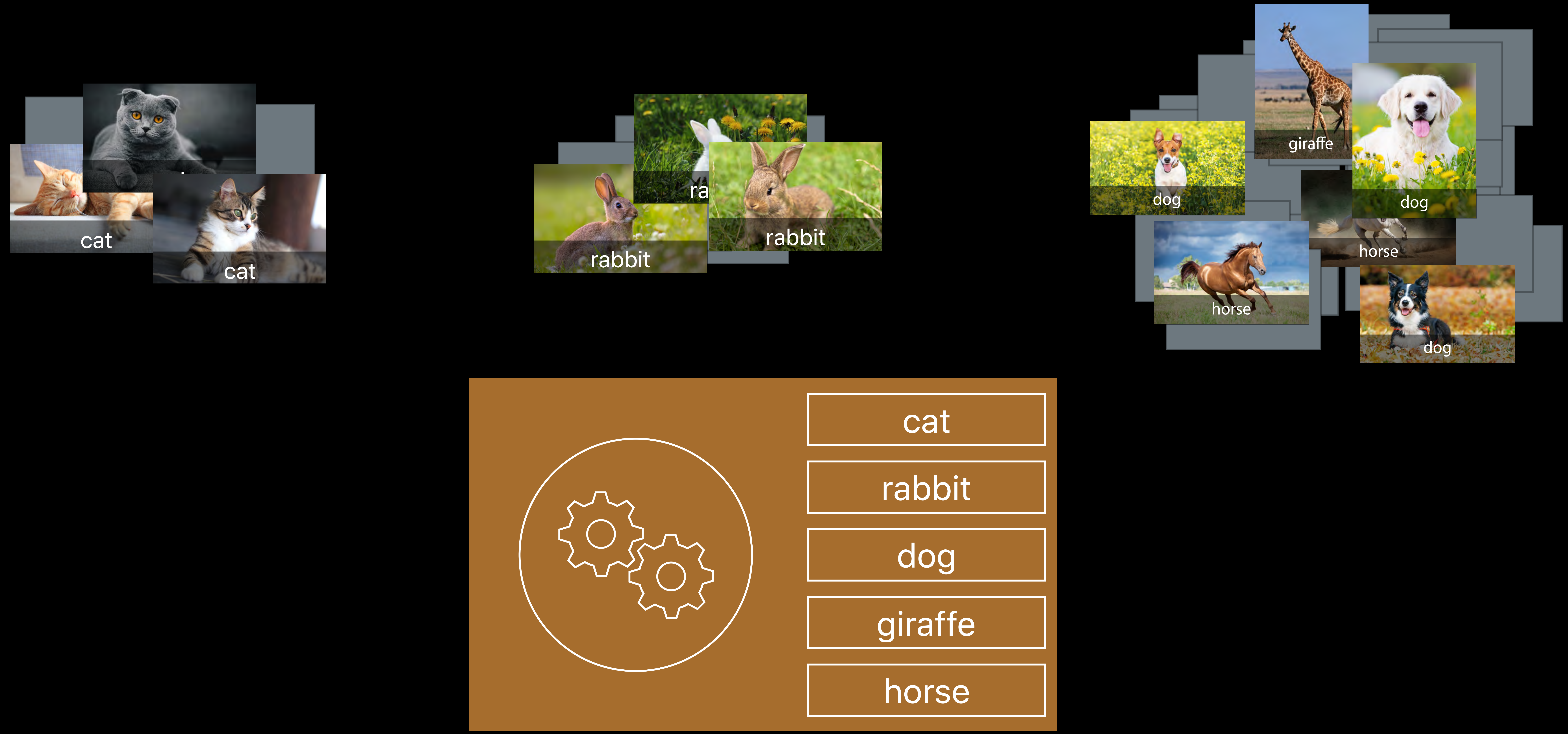
soup
corn
book
stone
balloon
tomato
newspaper
sunrise
beach
bracelet
girl
bench
volcano
sunset
sun hat
arrow
bread
rock
grass
desk
orange
car
photo
sugar
swing
giraffe
tennis
Alice
bell
pencil
sign
pony
raccoon
penguin
Maria
sunglasses
raincoat
house
umbrella
chessboard
window
milk
fork
chair
elephant
cheese
rose
key
basketball
tape
cabinet
rainbow

Training and Inference

cat
rabbit
dog
giraffe
horse

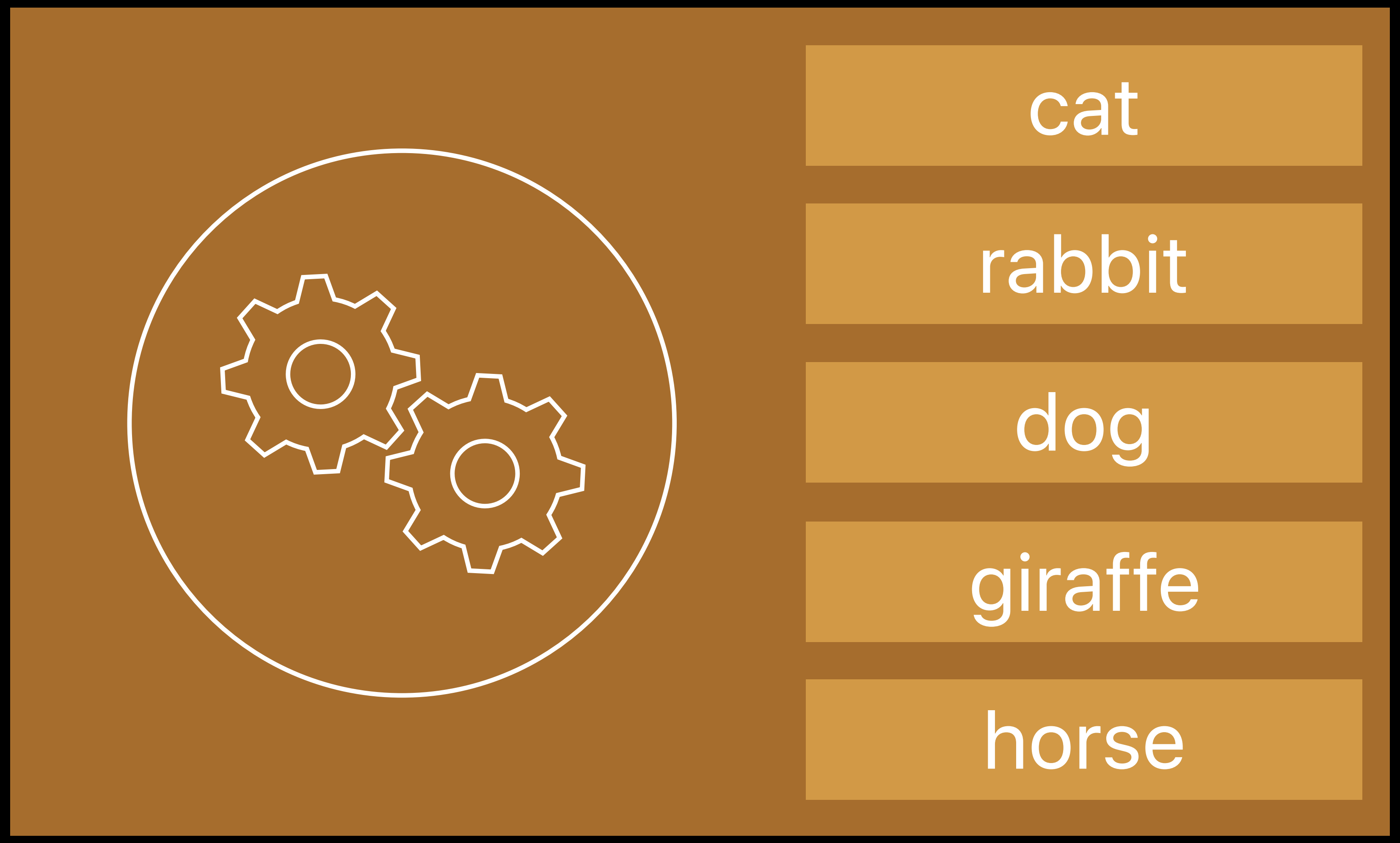
Training to Classify Images

Training



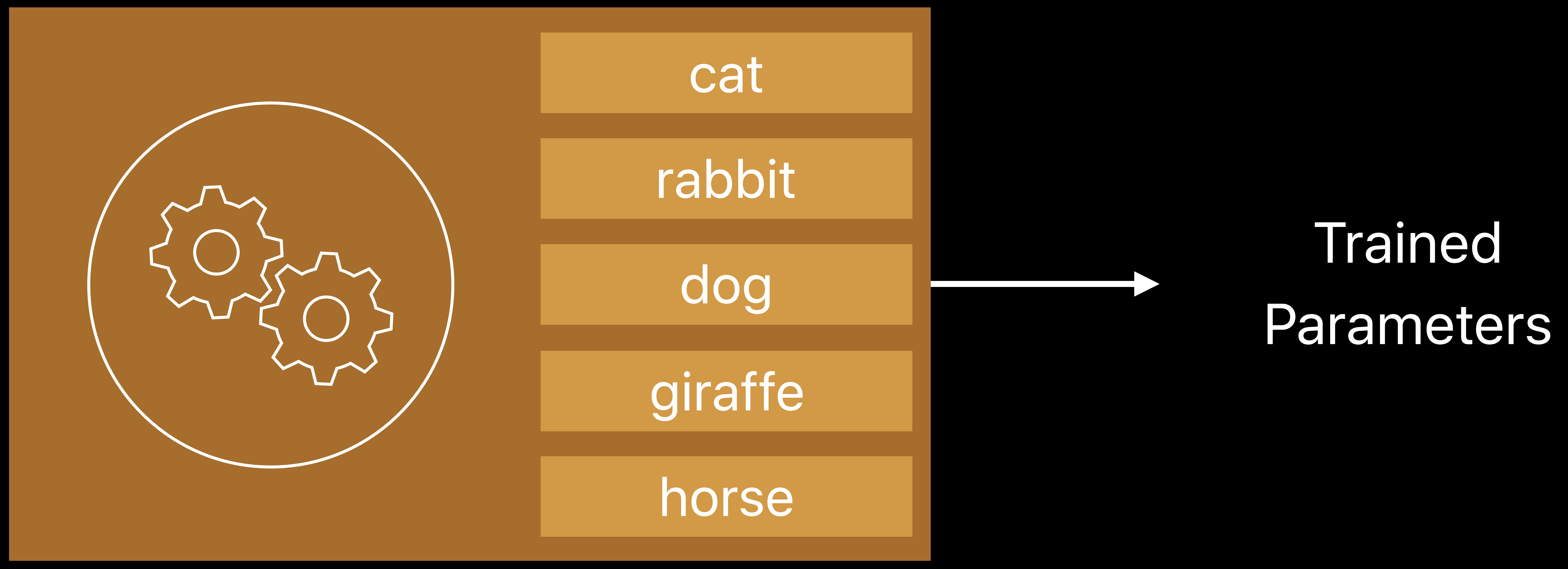
Training to Classify Images

Training



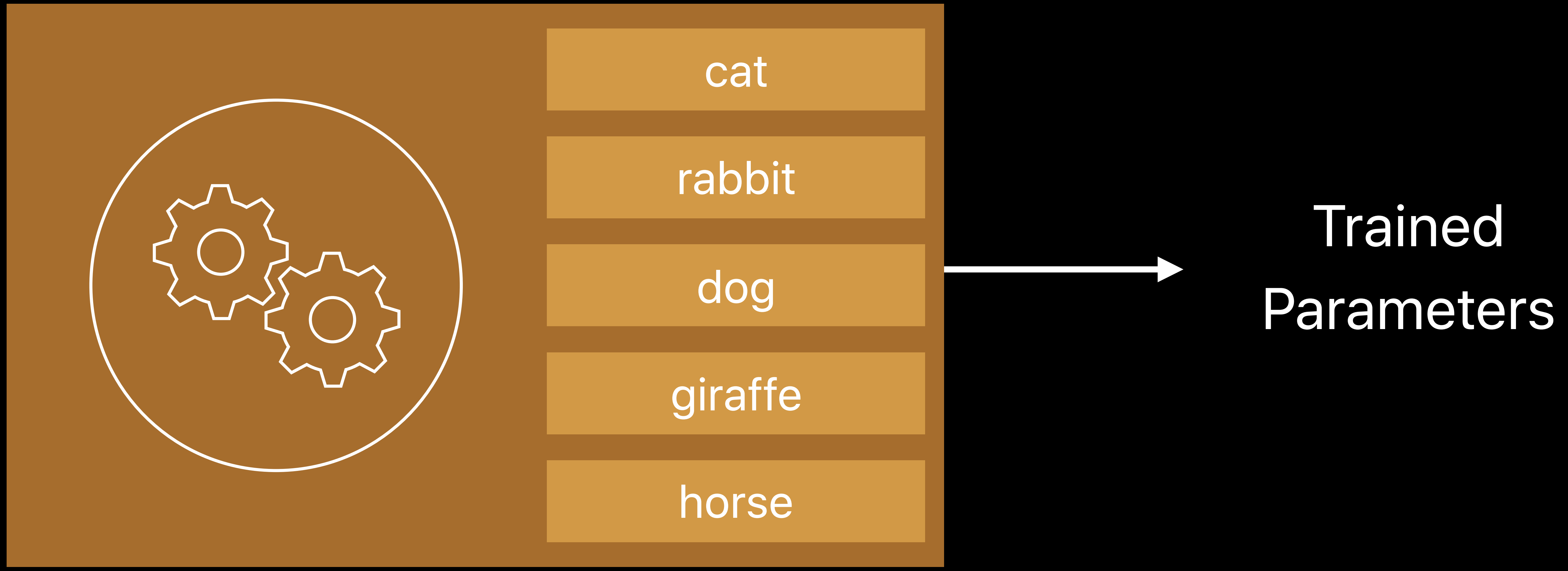
Training to Classify Images

Training



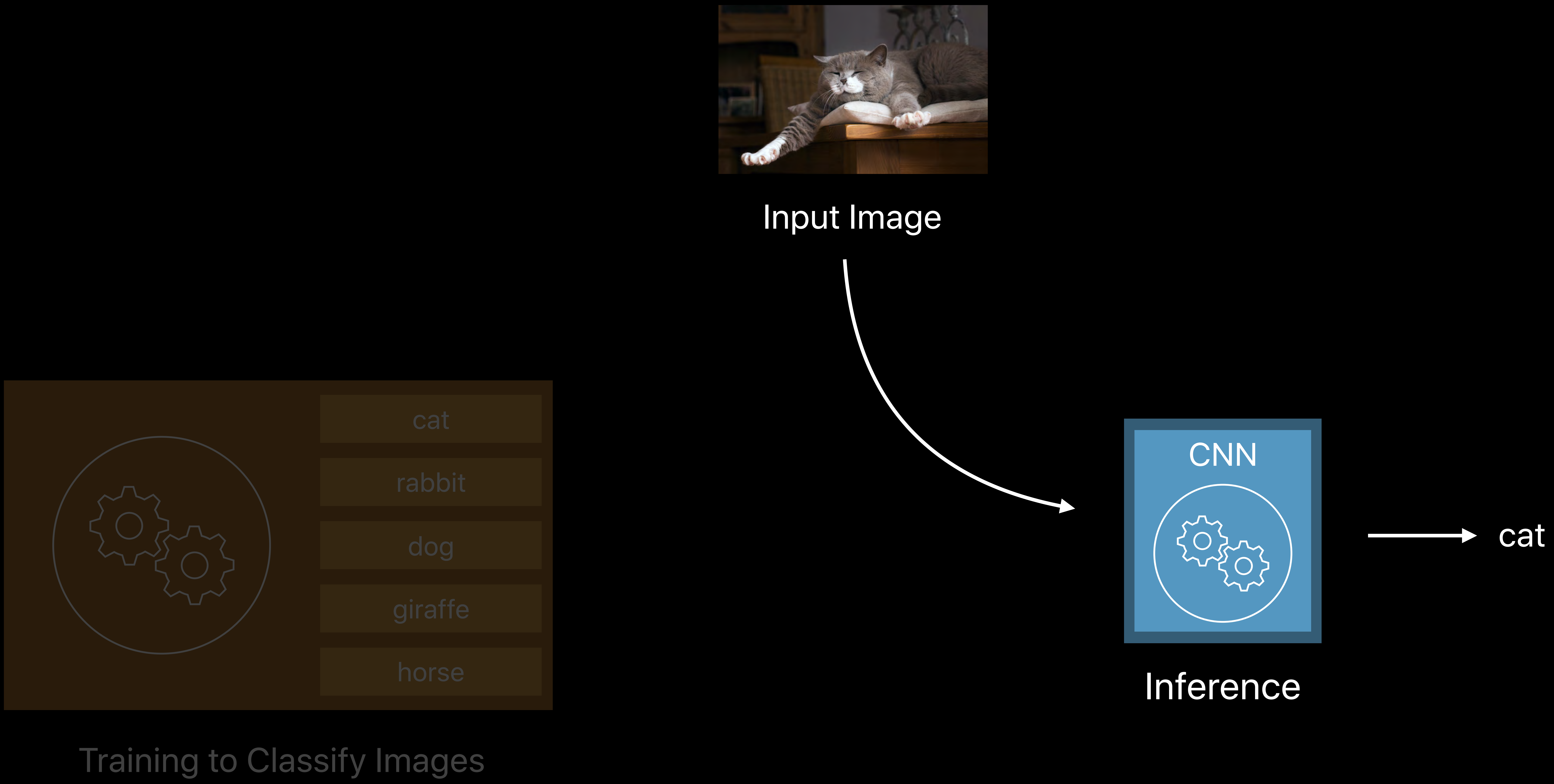
Training to Classify Images

Inference



Training to Classify Images

Inference


Input Image
CNN
cat
Inference
cat
rabbit
dog
giraffe
horse
Training to Classify Images

Agenda

Recap on Convolutional Neural Networks (CNN)

Agenda

Recap on Convolutional Neural Networks (CNN)

Convolutional Neural Networks — New Primitives

Neural Network Graph API

Recurrent Neural Networks (RNN)

Agenda

Recap on Convolutional Neural Networks (CNN)

Convolutional Neural Networks — New Primitives

Neural Network Graph API

Recurrent Neural Networks (RNN)

What Are Convolutional Neural Networks?

Convolutional Neural Networks

Biologically-inspired, resemble the visual cortex

Convolutional Neural Networks

Biologically-inspired, resemble the visual cortex

Hierarchical representation

- Organized into a hierarchy of layers
- Higher-level features are derived from lower-level features

Convolutional Neural Networks

Biologically-inspired, resemble the visual cortex

Hierarchical representation

- Organized into a hierarchy of layers
- Higher-level features are derived from lower-level features

Think of a “feature” as a filter that filters data for that feature

Convolutional Neural Networks

Primitives available in iOS 10

Convolution

Pooling

- Average
- Max

Normalization

- Cross-Channel
- Local Contrast
- Spatial

Fully-Connected

Softmax

Neuron

- Linear
- ReLU
- Sigmoid
- TanH
- Absolute

Convolutional Neural Networks

Primitives available in iOS 10

Convolution

Pooling

- Average
- Max

Normalization

- Cross-Channel
- Local Contrast
- Spatial

Fully-Connected

Softmax

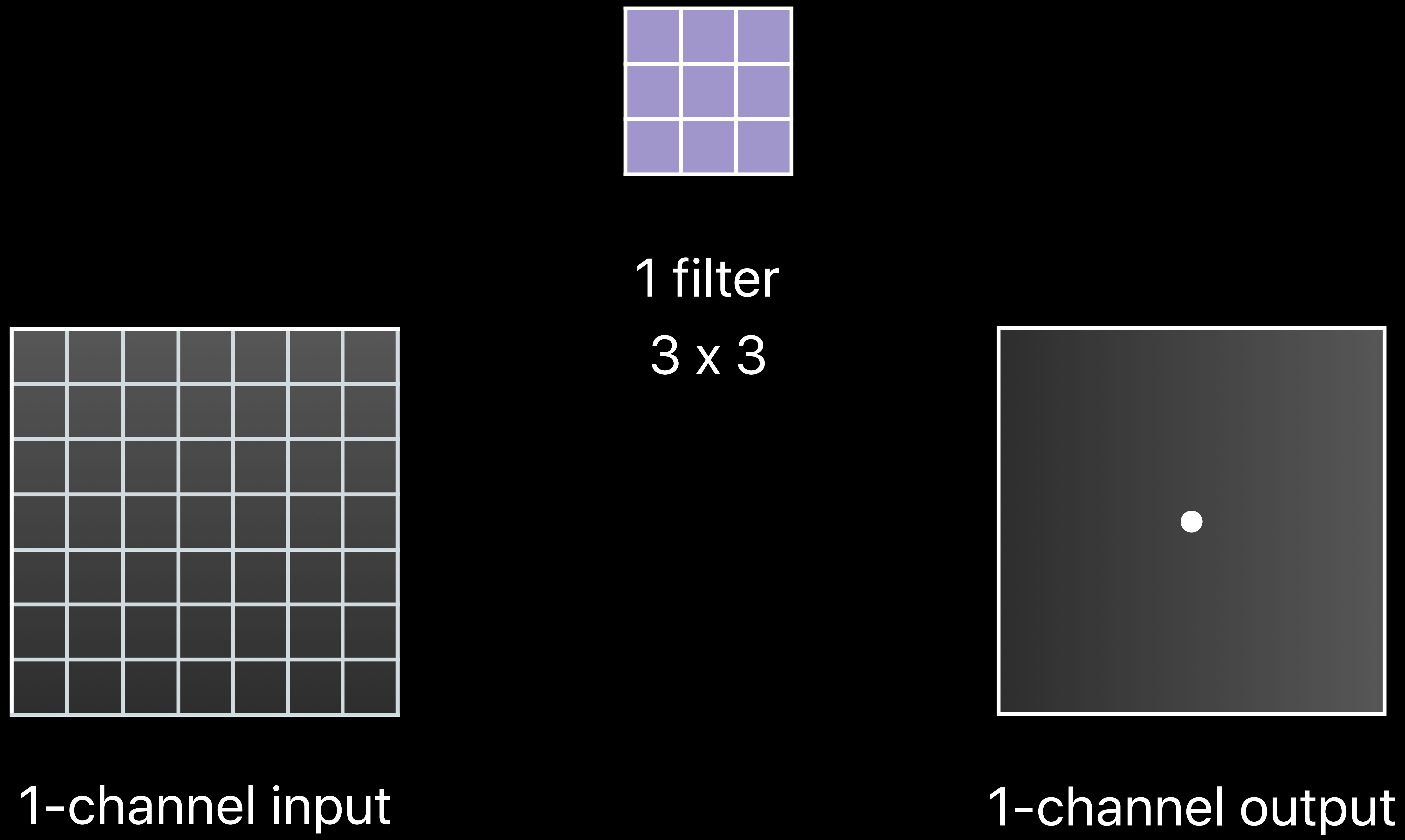
Neuron

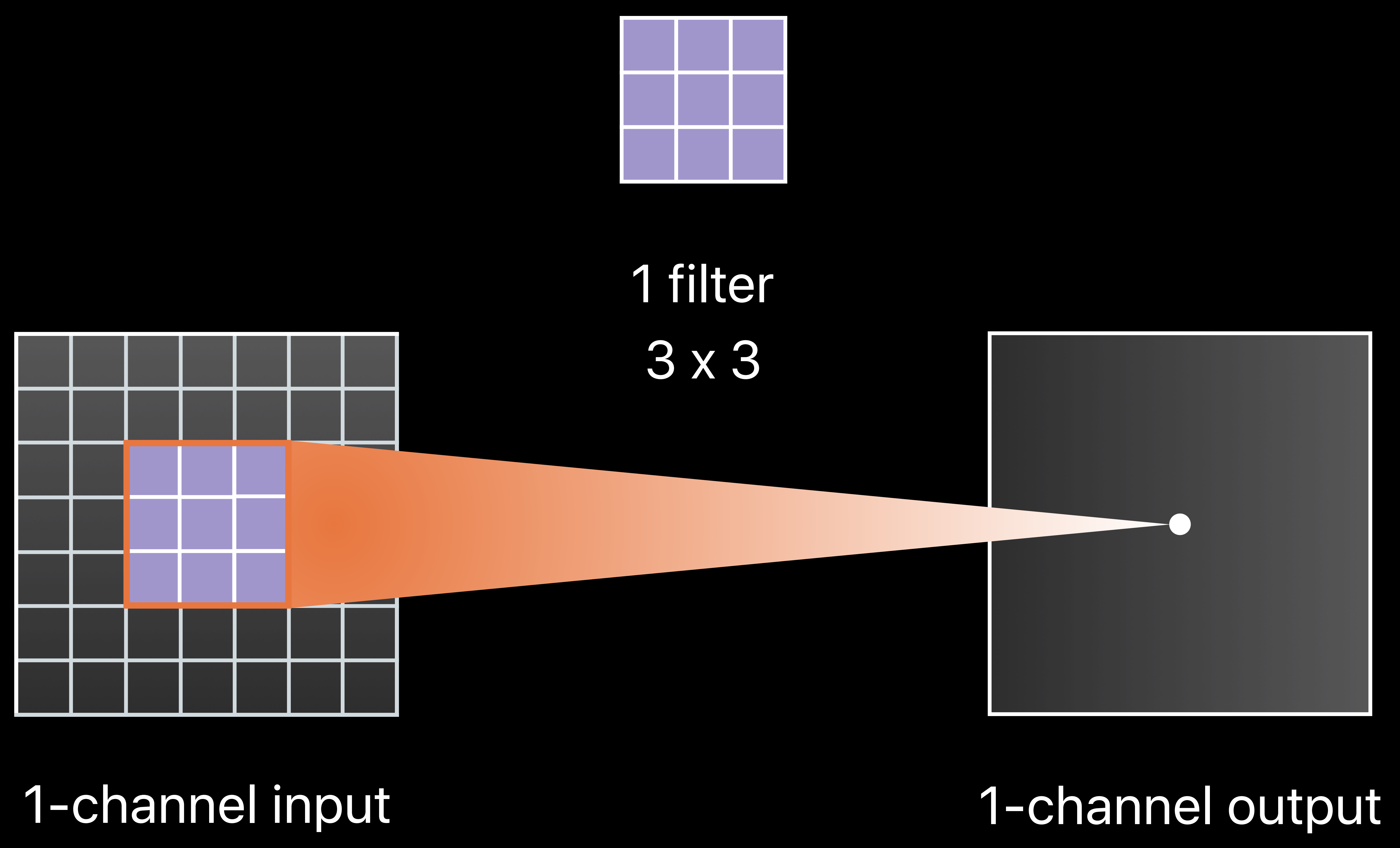
- Linear
- ReLU
- Sigmoid
- TanH
- Absolute

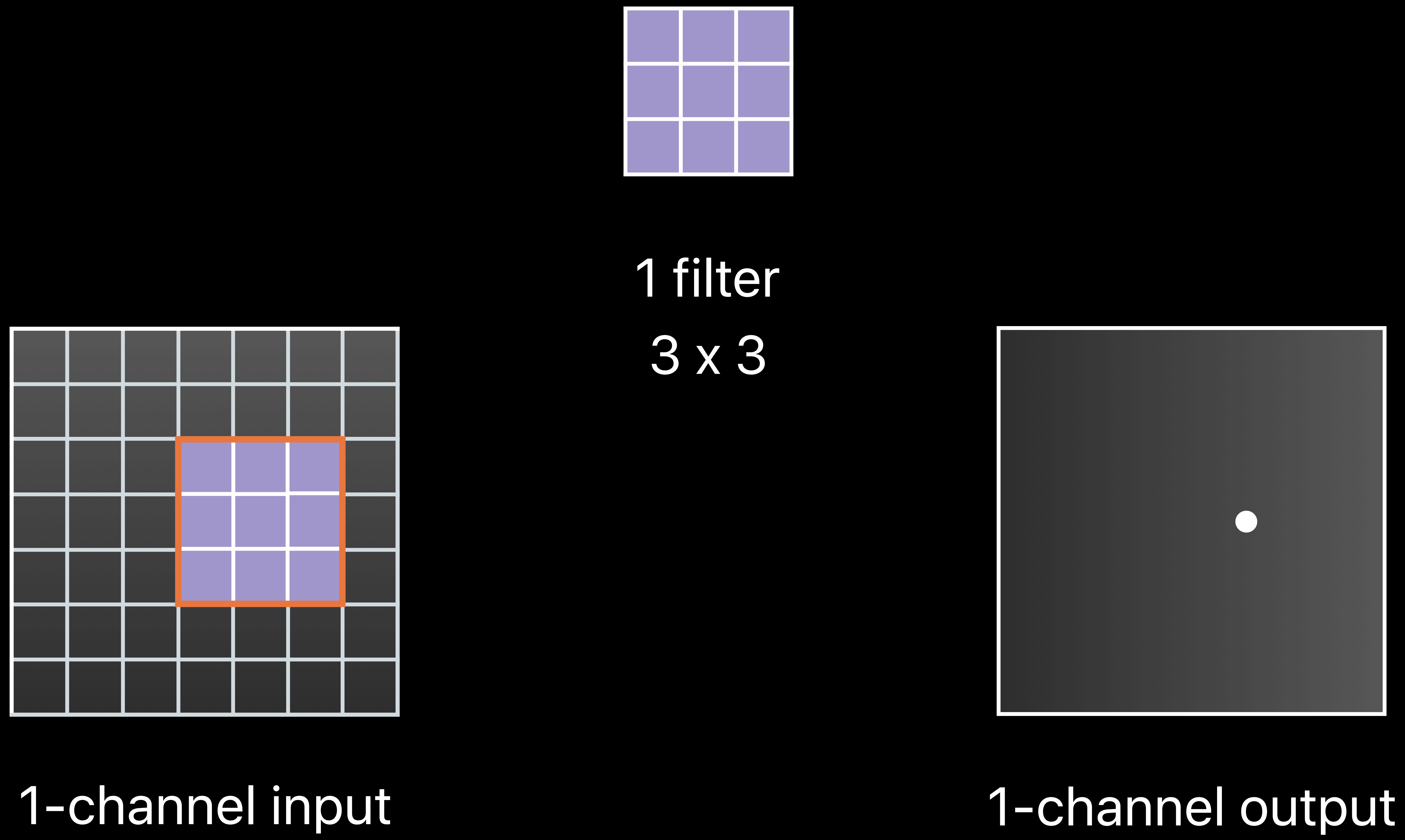
Convolution

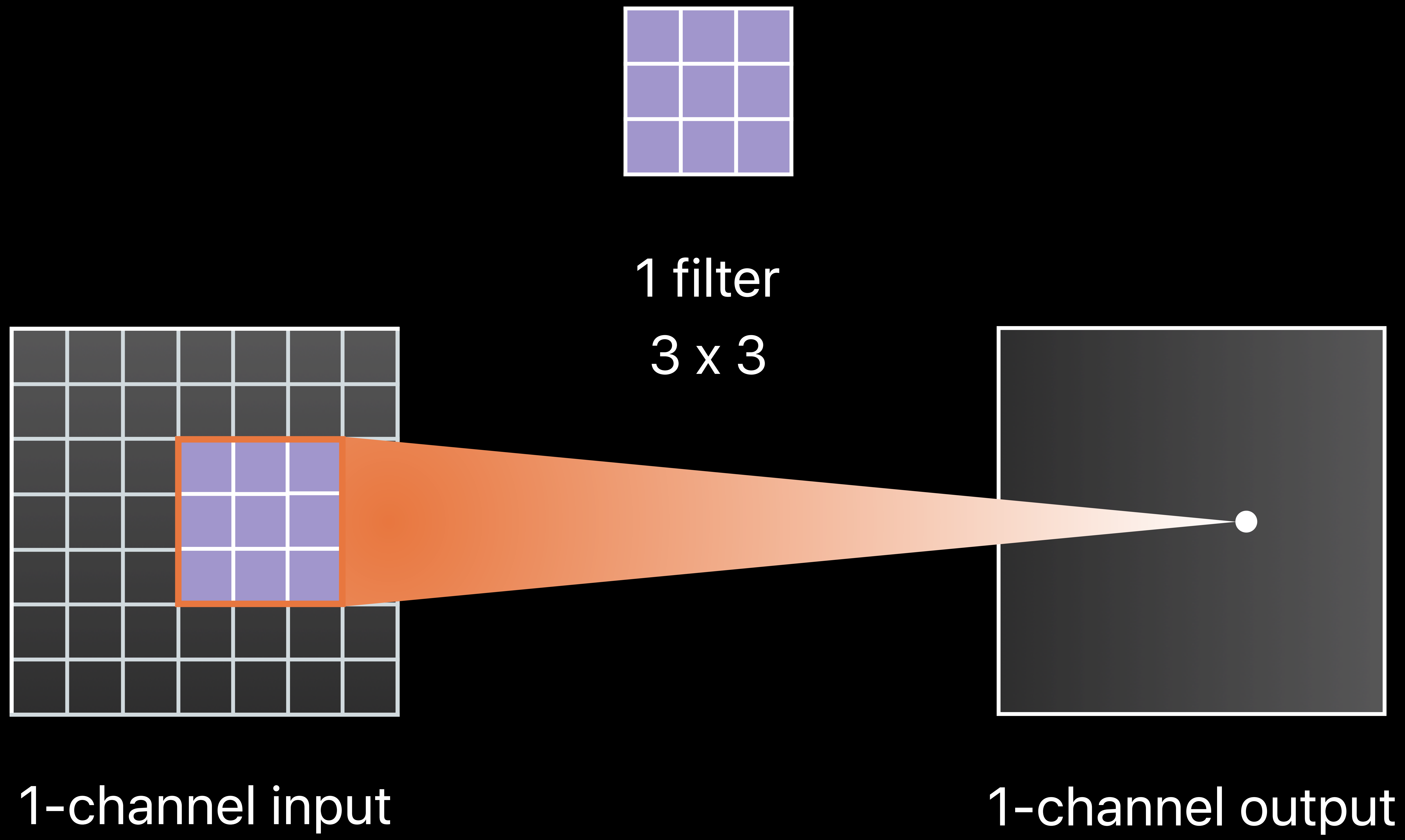
Core building block

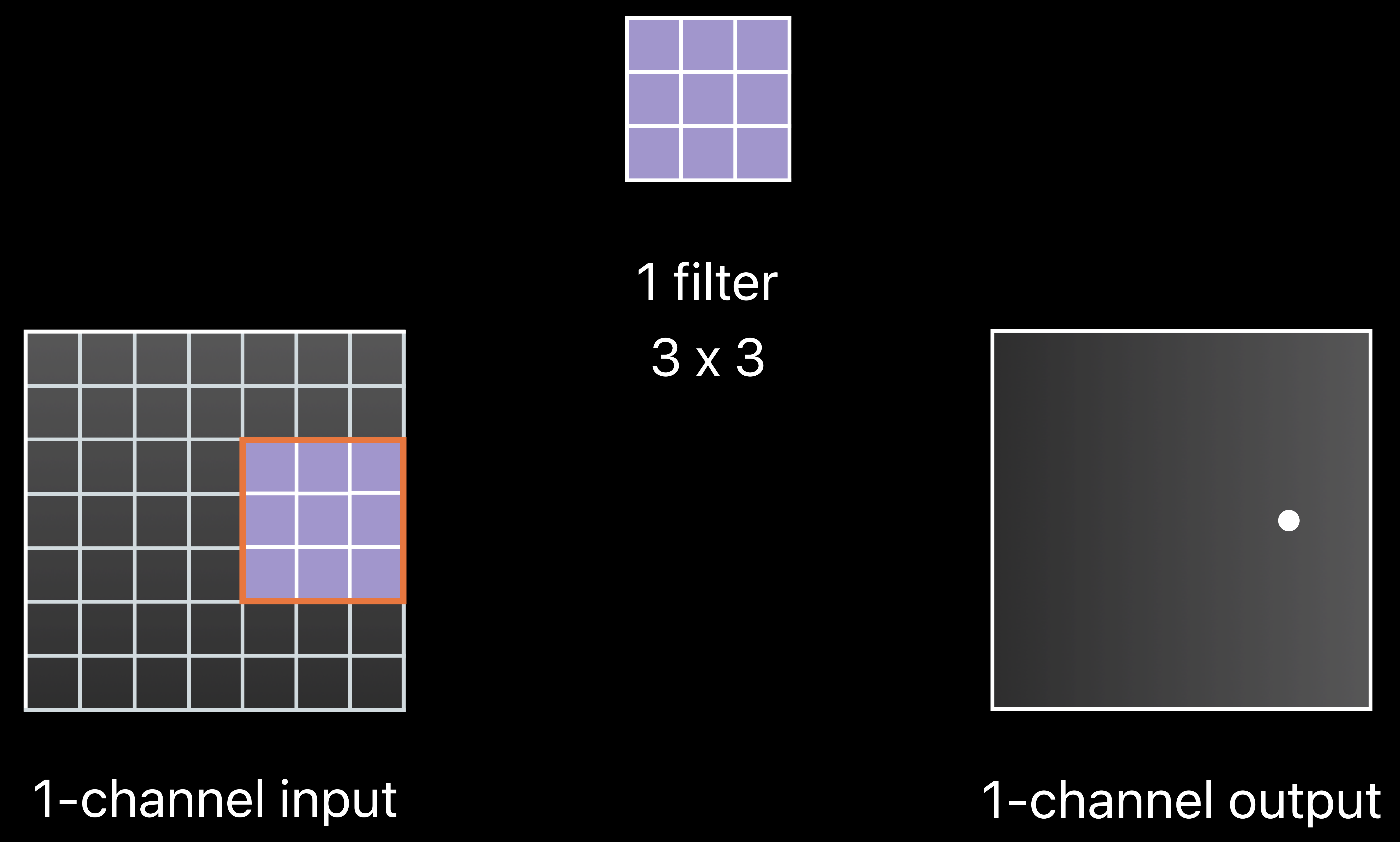
Recognizes features in input

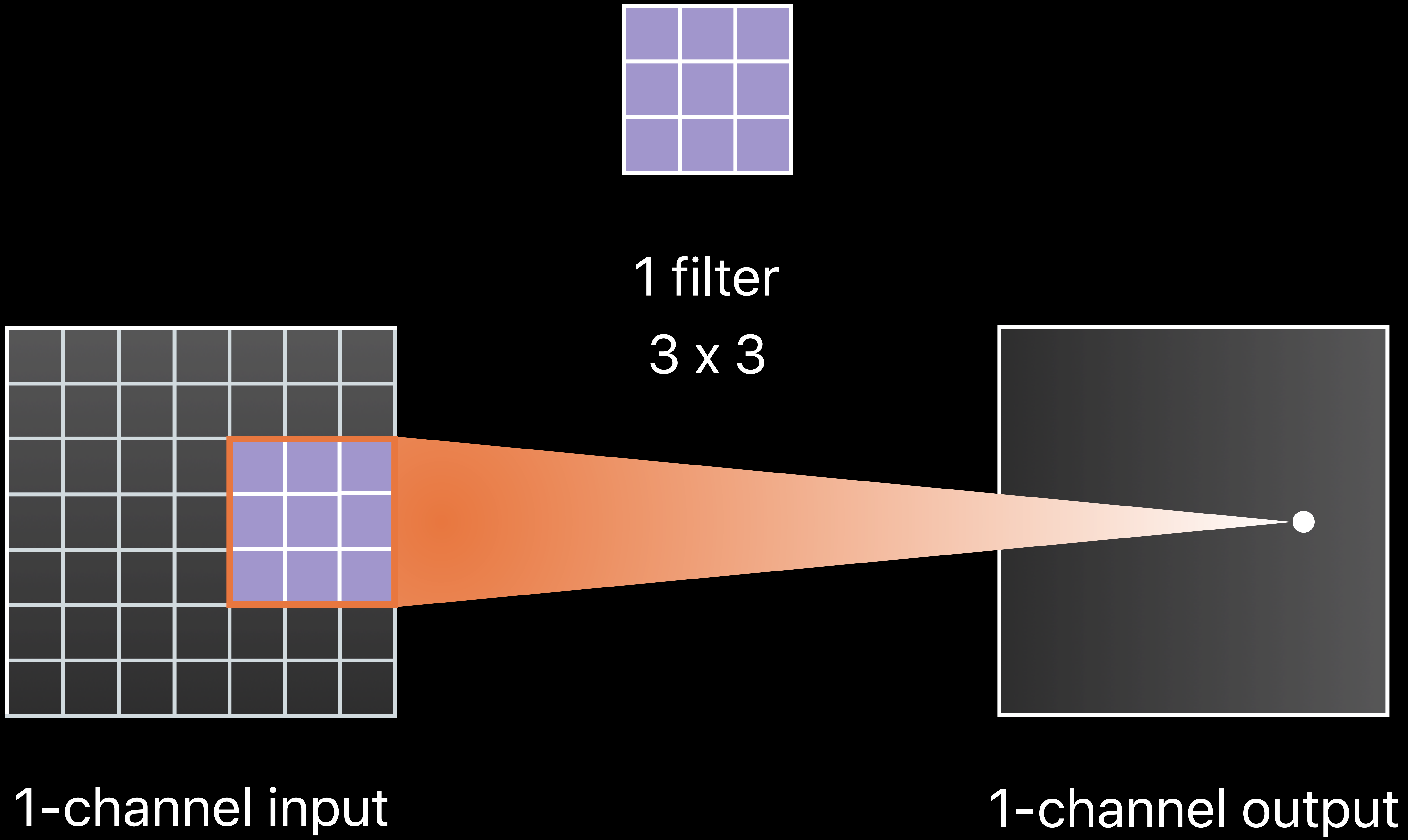

1 filter
3 x 3
1-channel input
1-channel output

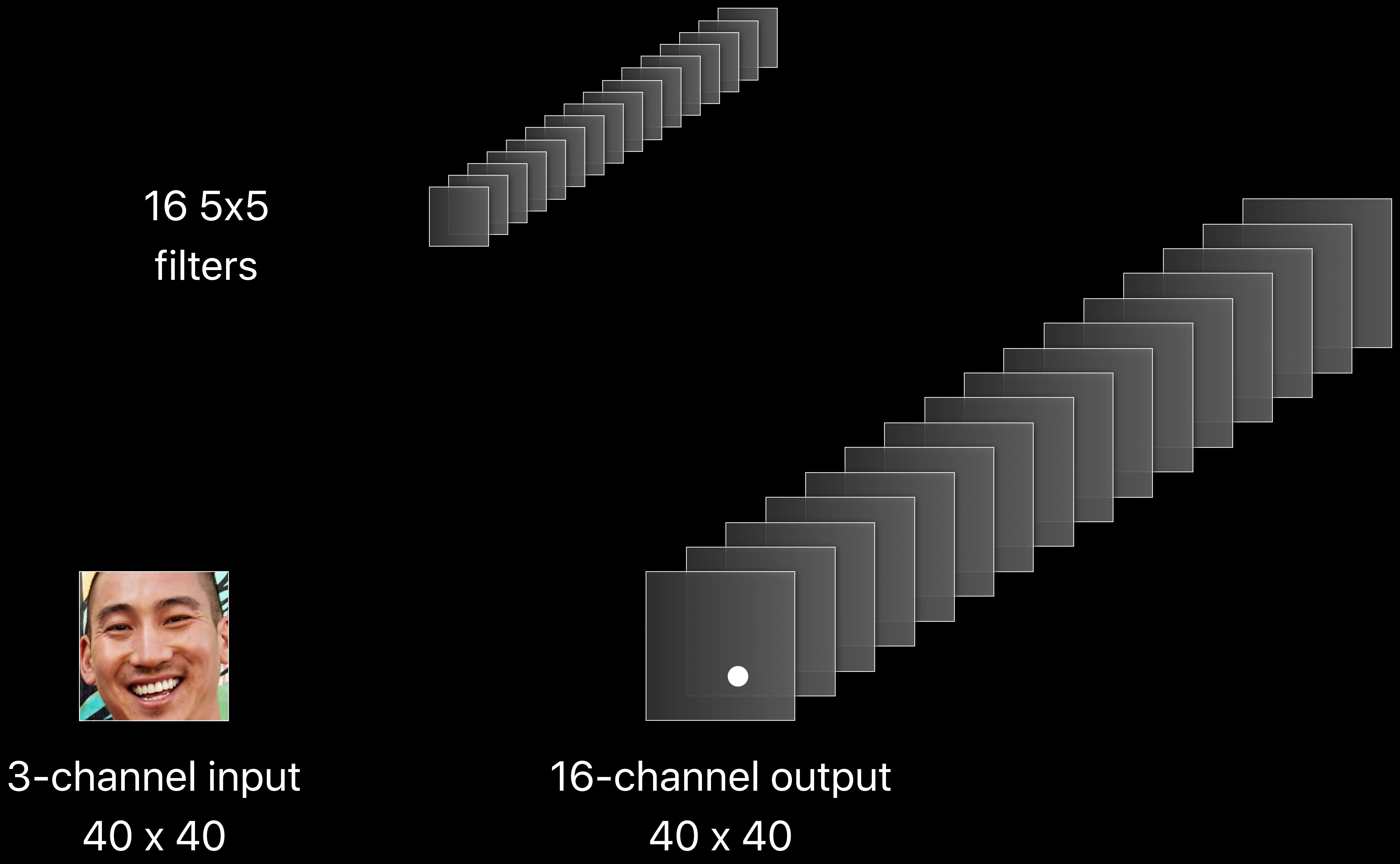

1 filter
3 x 3
1-channel input
1-channel output

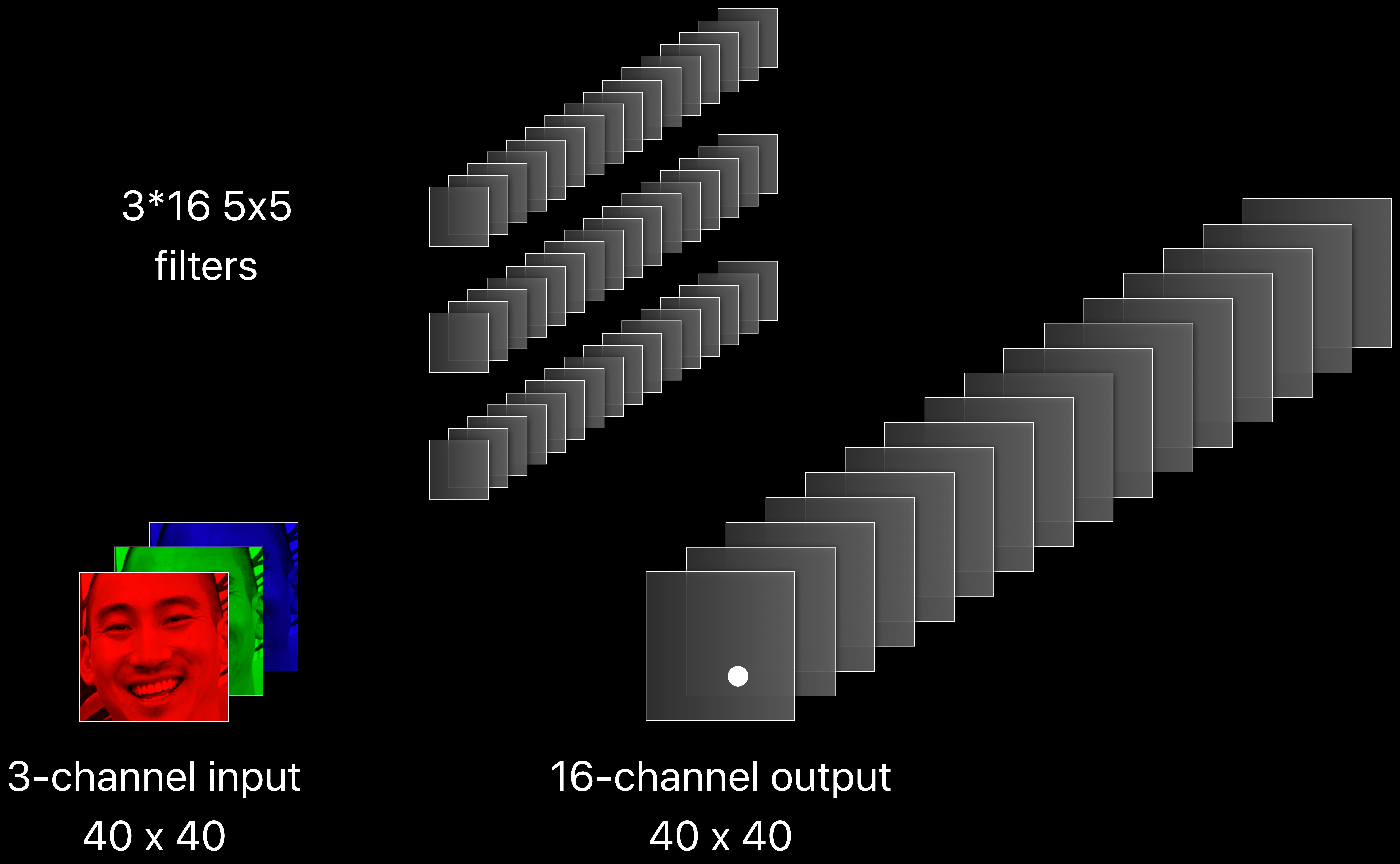

1 filter
3 x 3
1-channel input
1-channel output

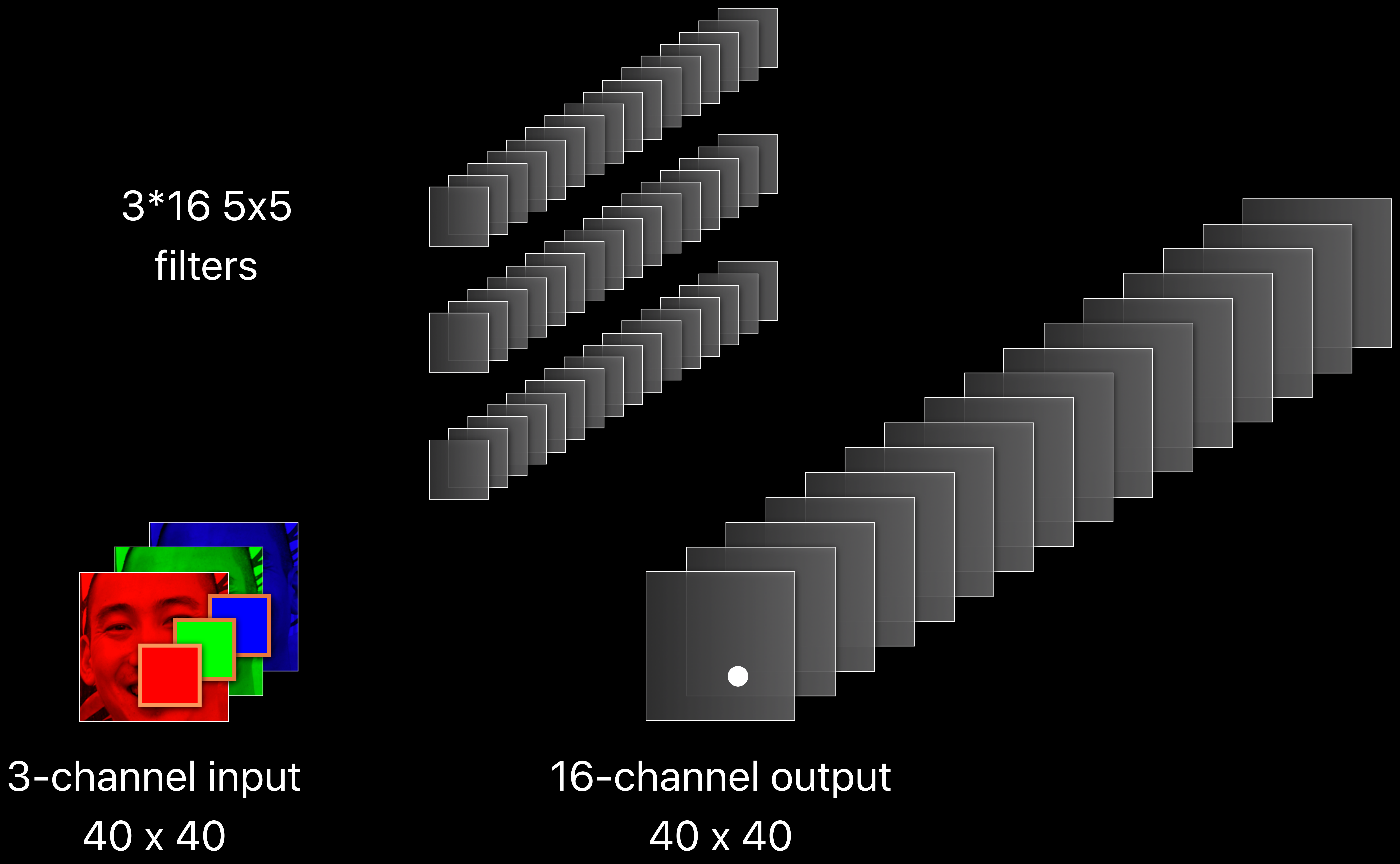

1 filter
3 x 3
1-channel input
1-channel output

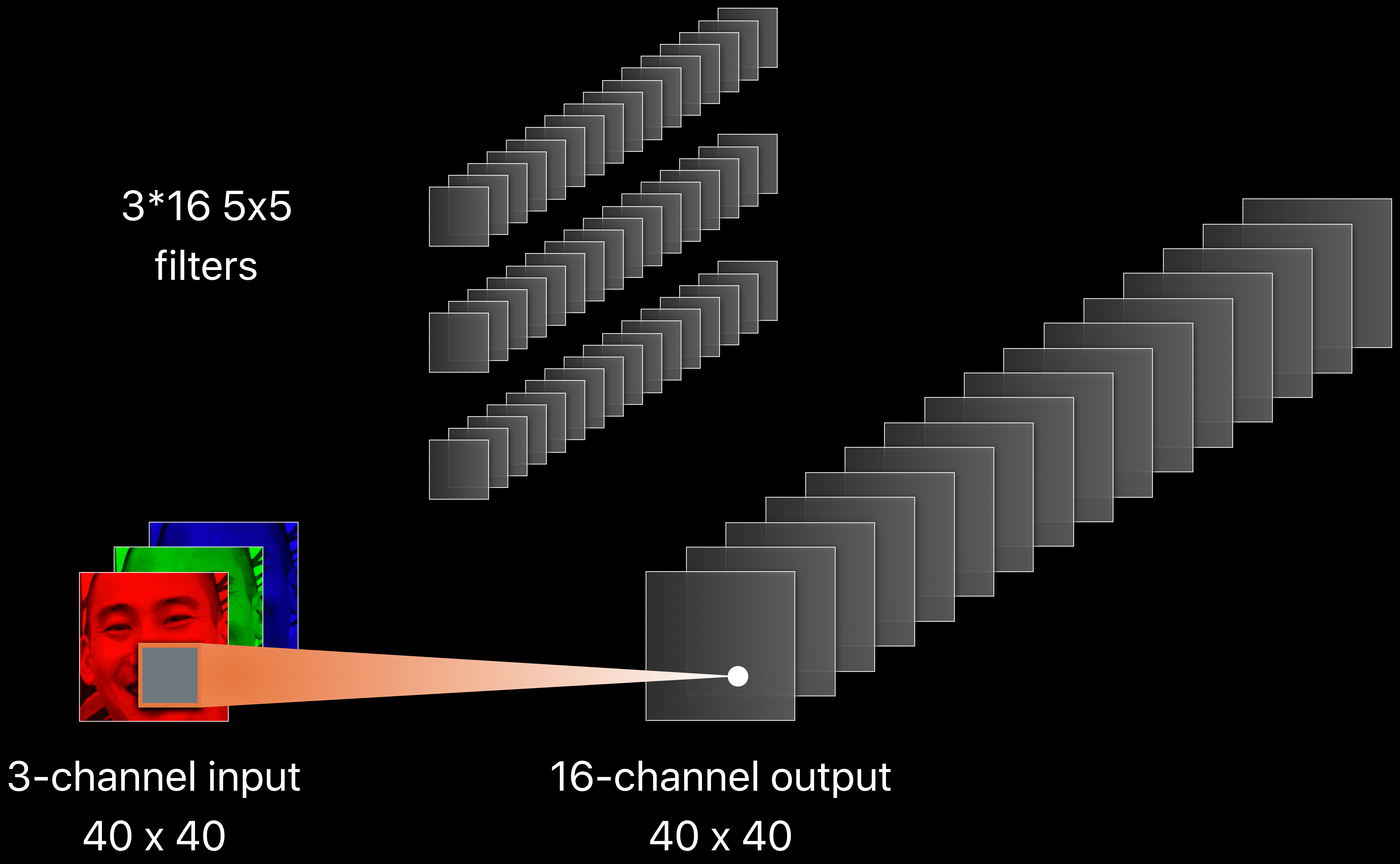

1 filter
3 x 3
1-channel input
1-channel output


1 filter
3 x 3
1-channel input
1-channel output


16 5x5
filters
3-channel input
40 x 40
16-channel output
40 x 40


3*16 5x5
filters
3-channel input
40 x 40
16-channel output
40 x 40


3*16 5x5
filters
3-channel input
40 x 40
16-channel output
40 x 40


3*16 5x5
filters
3-channel input
40 x 40
16-channel output
40 x 40

Agenda

Recap on Convolutional Neural Networks (CNN)

Convolutional Neural Networks — New Primitives

Neural Network Graph API

Recurrent Neural Networks (RNN)

Convolutional Neural Networks

New primitives

NEW

New Convolution weight types

Binary and XNOR Convolution

Sub-Pixel Convolution

Dilated Convolution

Convolution Transpose

L2Norm Pooling

Dilated Max Pooling

Log Softmax

Resampling

- Lanczos, Bilinear

Upsampling

Arithmetic Operators

- Addition, Subtraction, Multiplication, Division

New Neuron layers

- Hard Sigmoid, SoftPlus, SoftSign, ELU

Convolutional Neural Networks

New primitives

NEW

New Convolution weight types

Binary and XNOR Convolution

Sub-Pixel Convolution

Dilated Convolution

Convolution Transpose

L2Norm Pooling

Dilated Max Pooling

Log Softmax

Resampling

- Lanczos, Bilinear

Upsampling

Arithmetic Operators

- Addition, Subtraction, Multiplication, Division

New Neuron layers

- Hard Sigmoid, SoftPlus, SoftSign, ELU

Convolution

Filter weight types

NEW

Single Precision Floating-Point

To reduce memory footprint and improve performance

- Half Precision Floating-Point
- 8-bit Integer
- Binary

Convolution

Primitives

NEW

Standard

Binary and XNOR

Dilated

Sub-Pixel

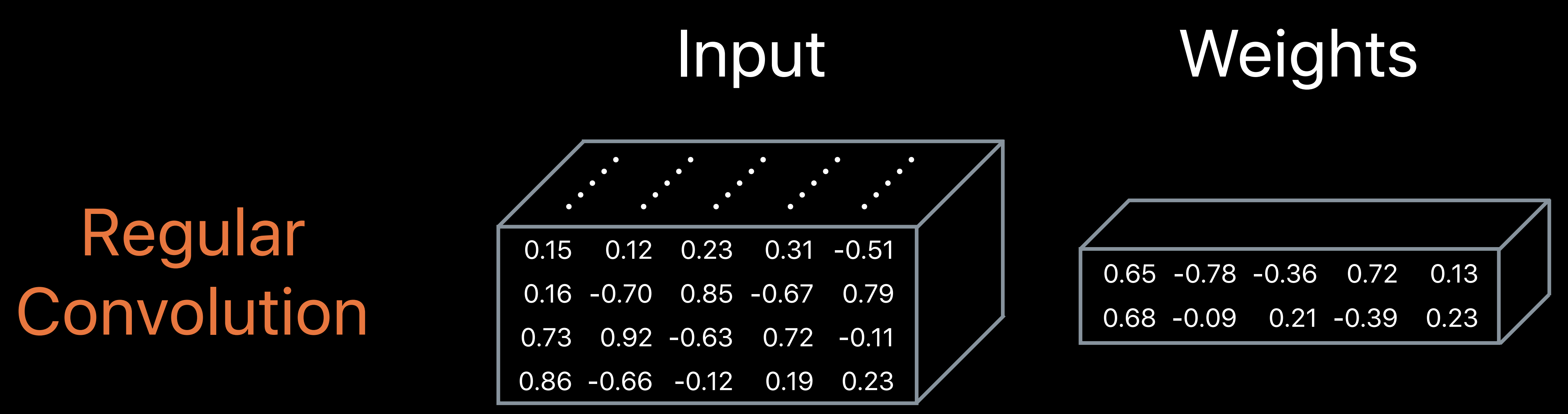
Transpose

Binary and XNOR Convolution

Same operation as regular Convolution

Improved performance

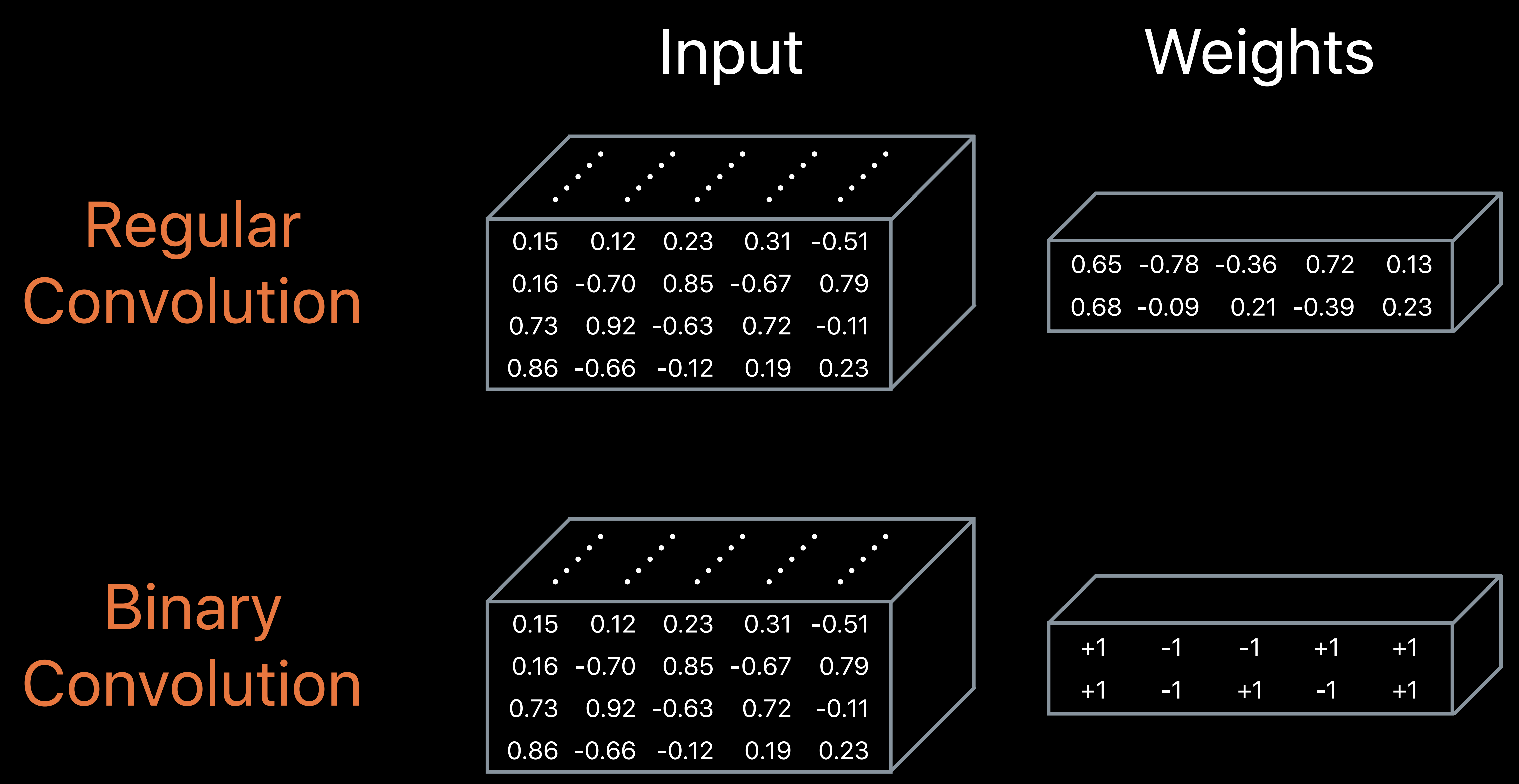
Less memory



Binary and XNOR Convolution

Binary Convolution

- Full-sized input, binary weights



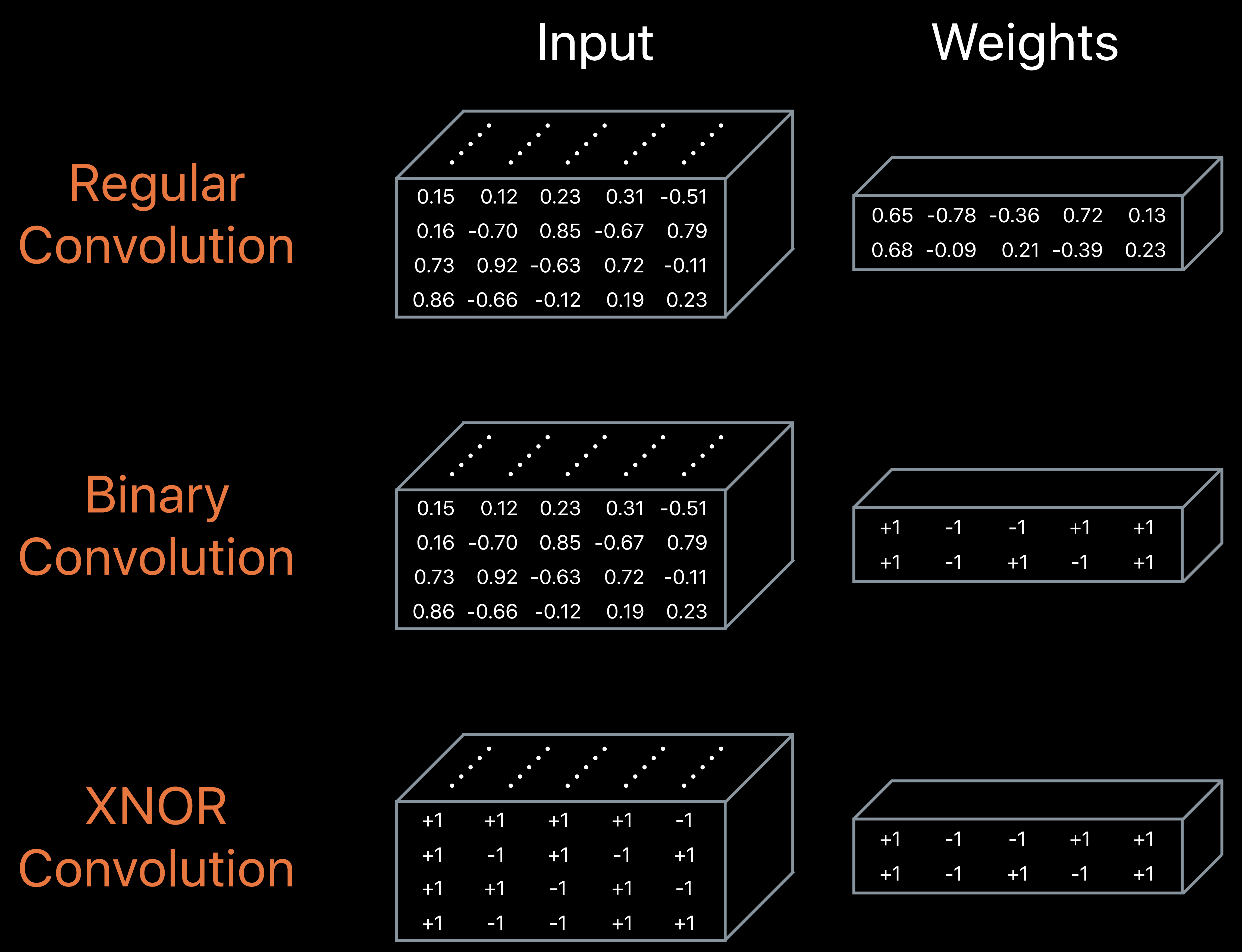
Binary and XNOR Convolution

Binary Convolution

- Full-sized input, binary weights

XNOR Convolution

- Binary input, binary weights

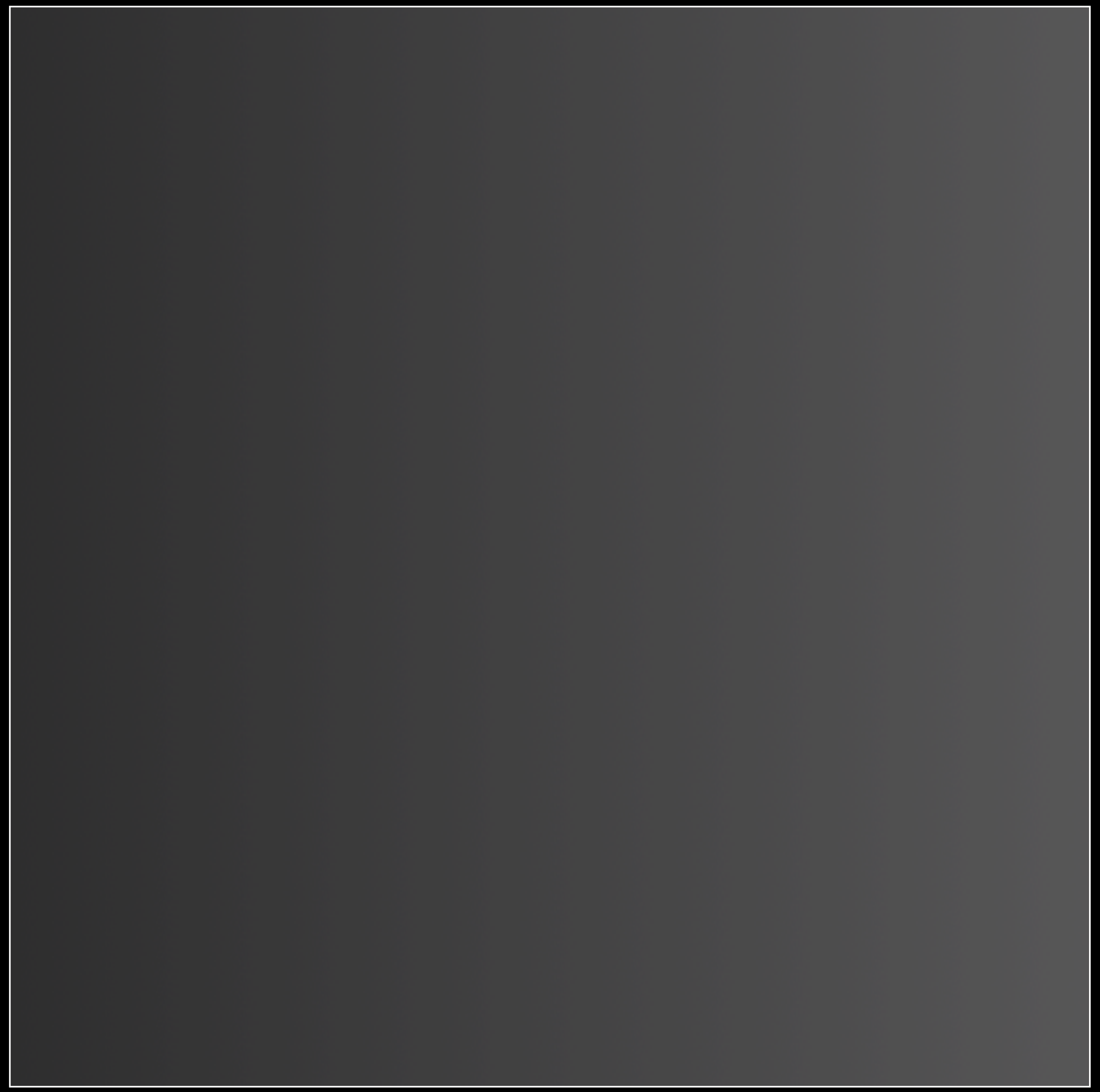


Dilated Convolution

Comparison to regular convolution

Input

Output

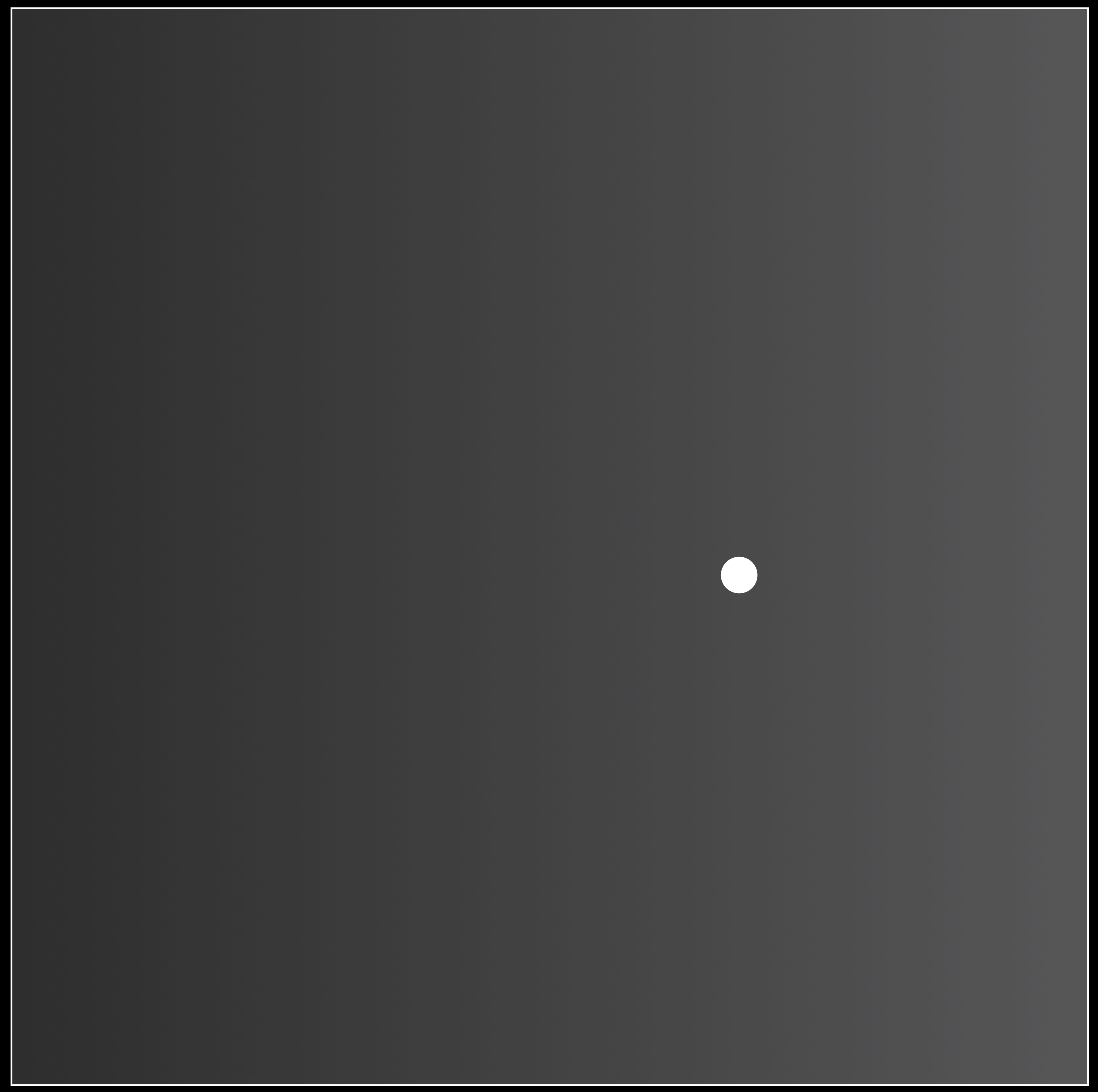


Dilated Convolution

Comparison to regular convolution

Input

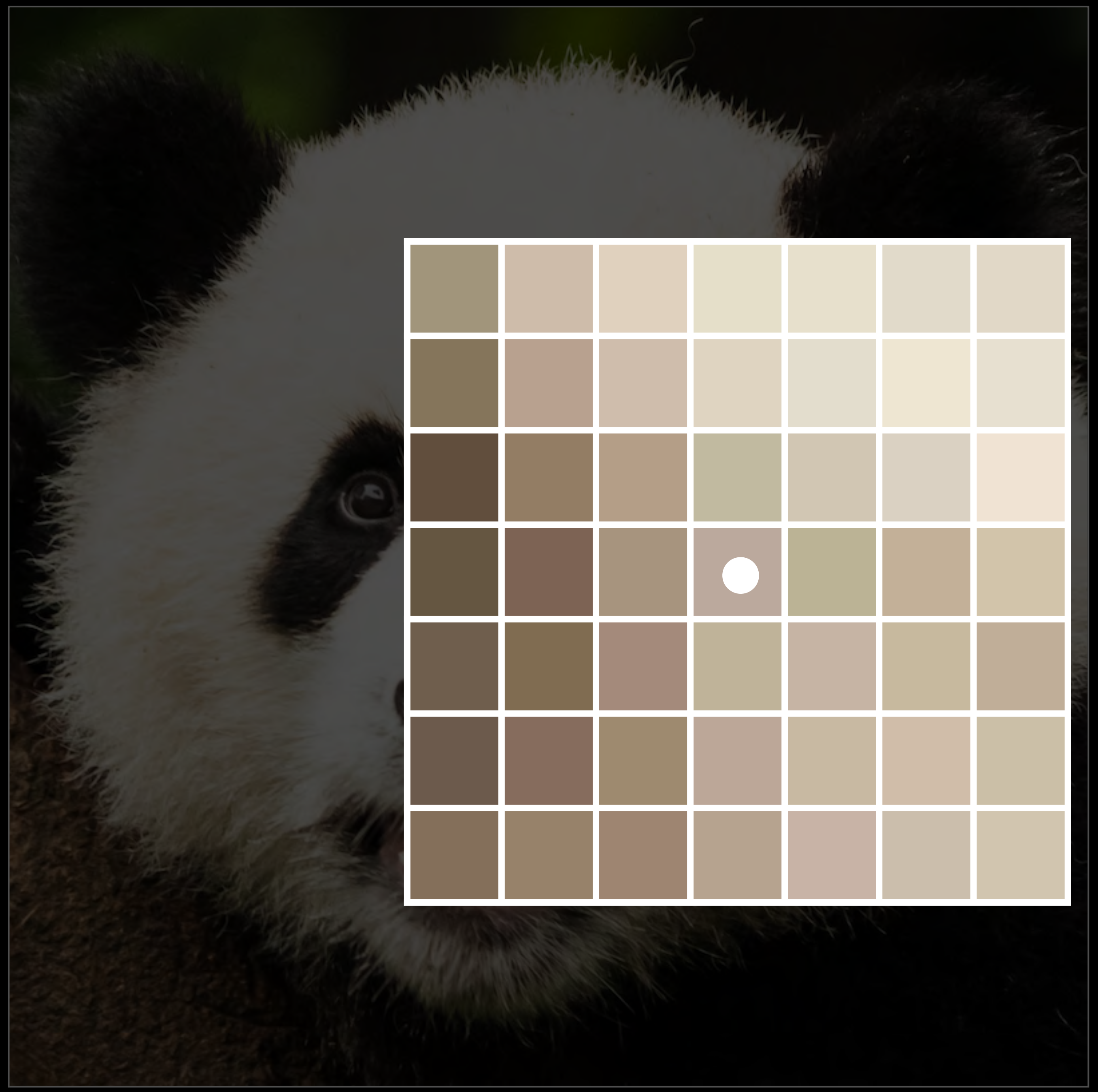
Output



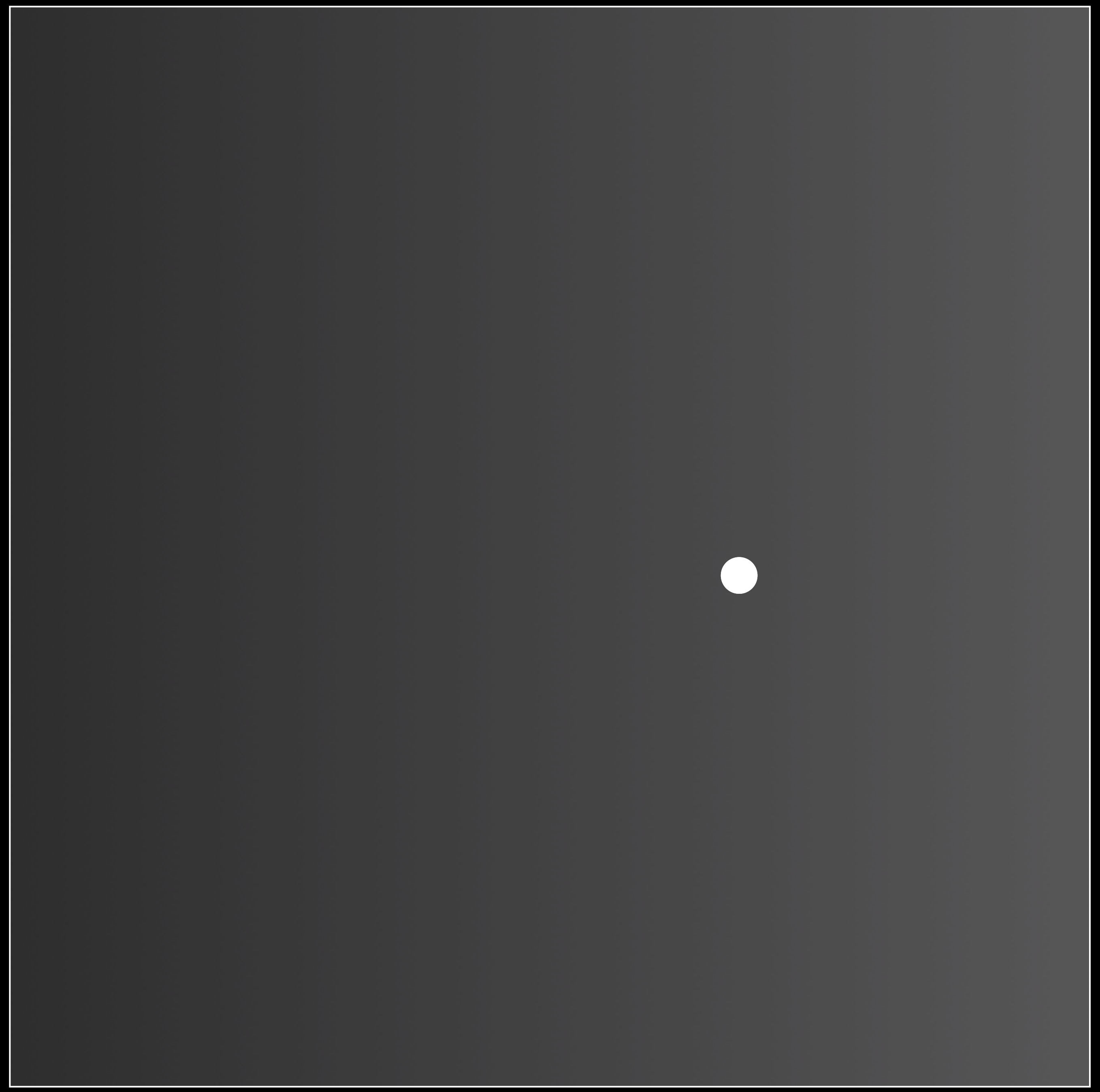
Dilated Convolution

Comparison to regular convolution

Input



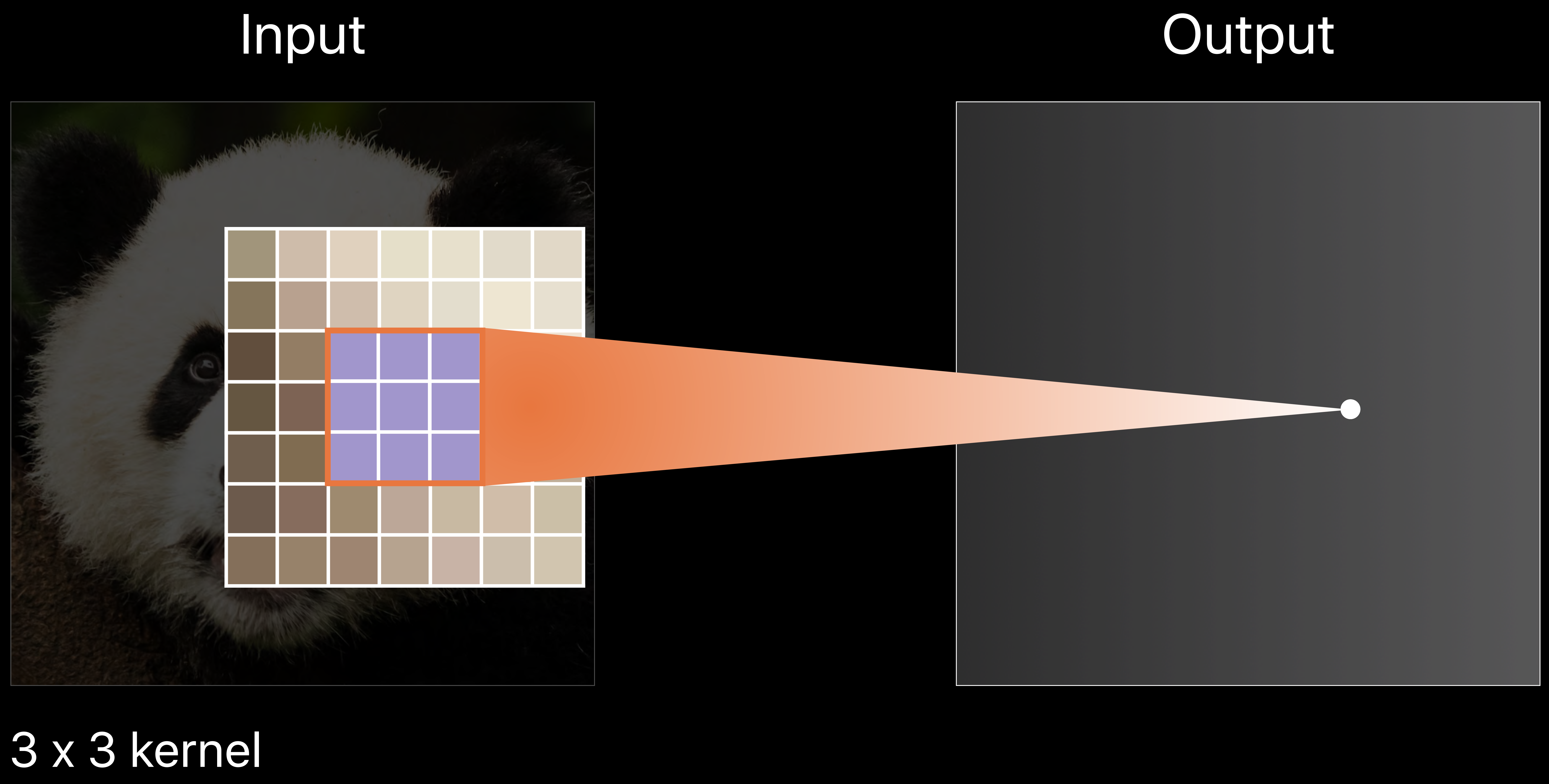
Output



3 x 3 kernel

Dilated Convolution

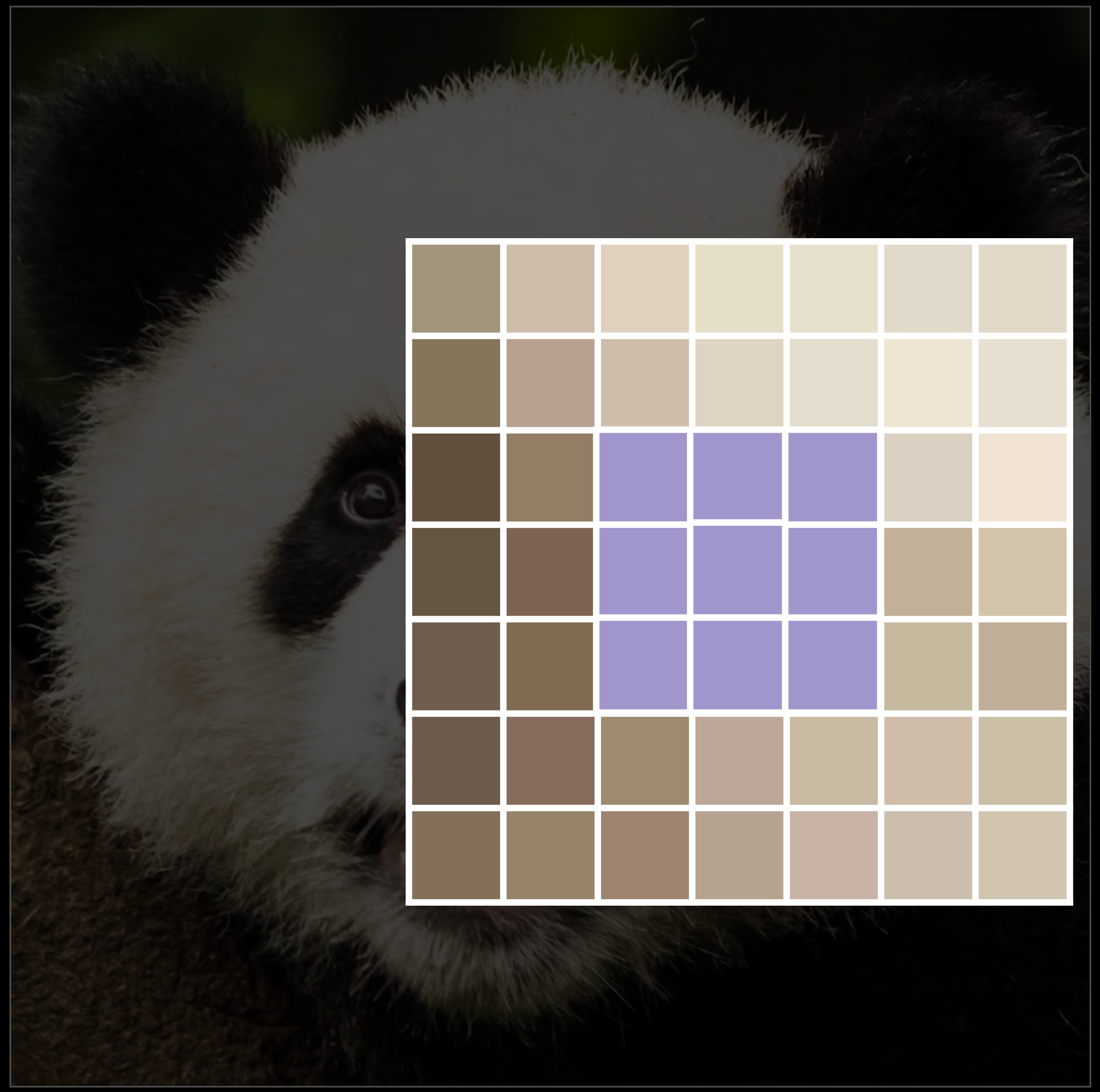
Comparison to regular convolution



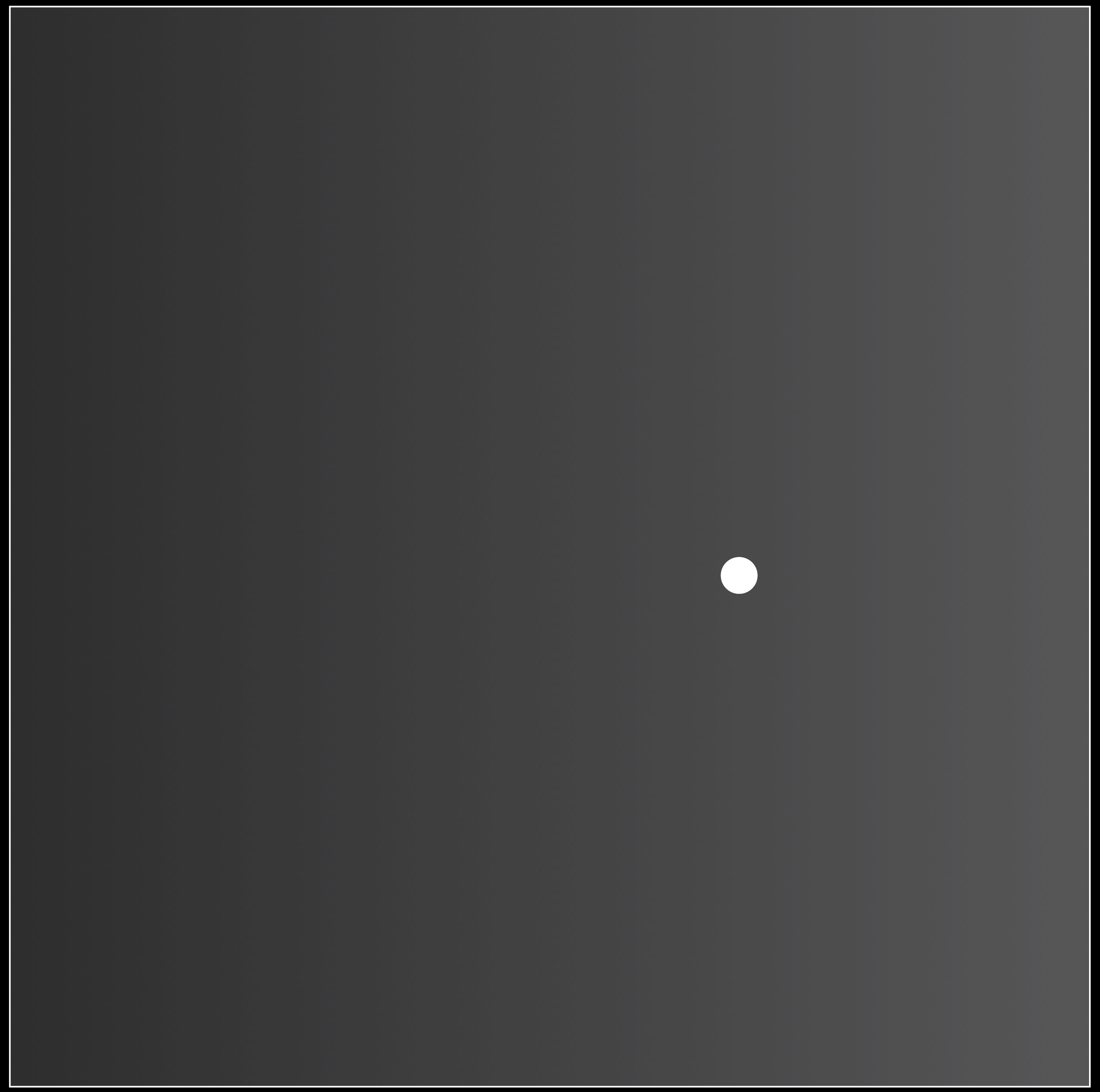
Dilated Convolution

How it works

Input



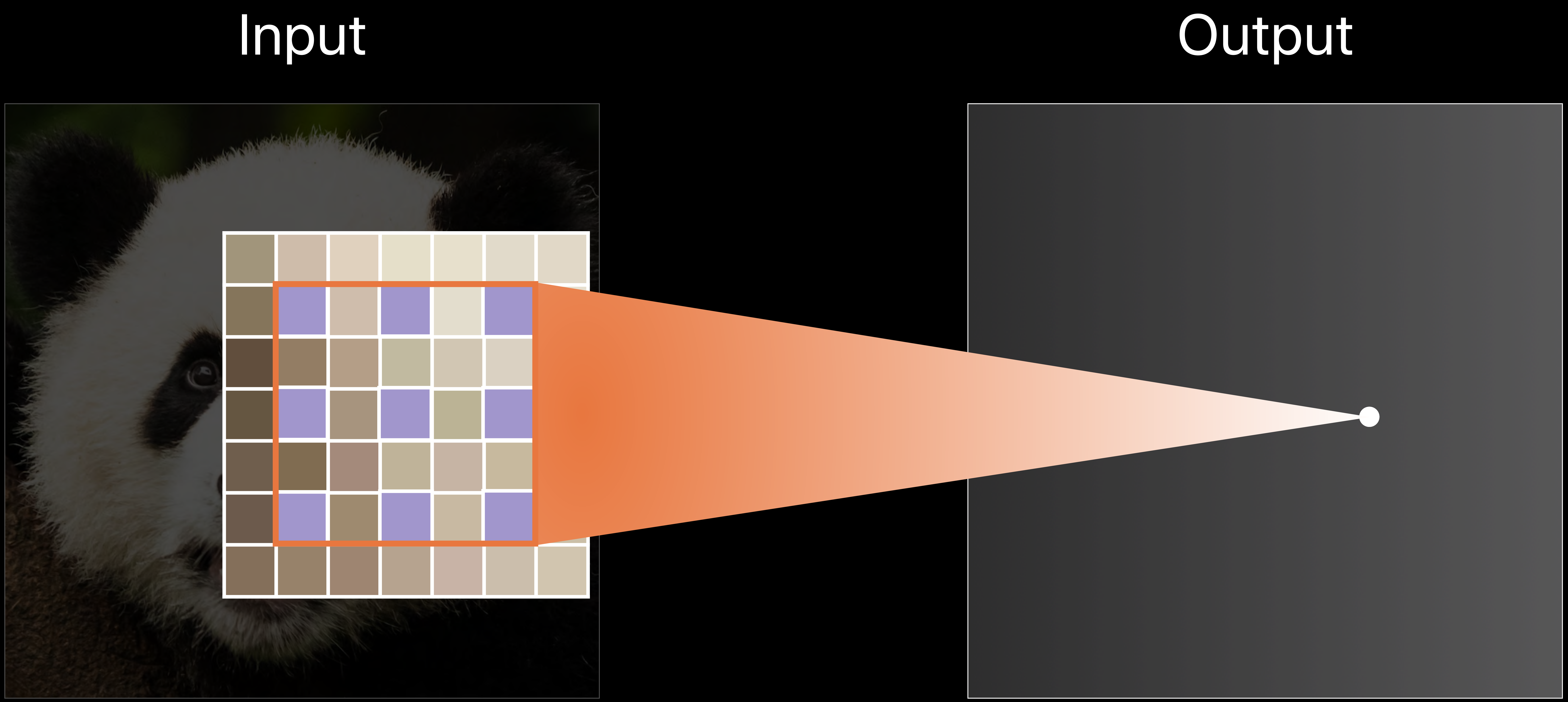
Output



3 x 3 kernel
dilationFactorX = 2
dilationFactorY = 2

Dilated Convolution

How it works



3 x 3 kernel
dilationFactorX = 2
dilationFactorY = 2

Sub-Pixel Convolution and Convolution Transpose

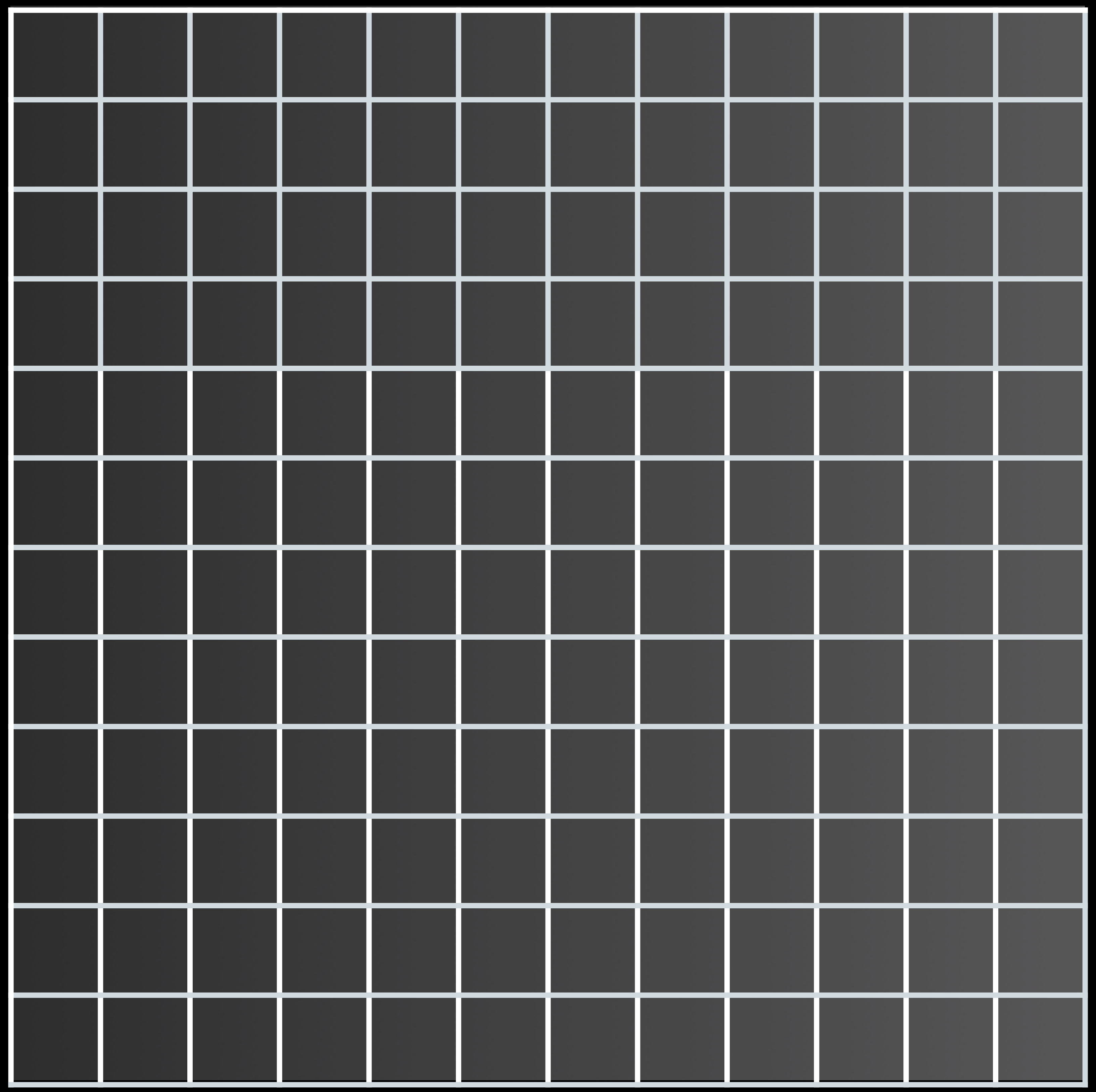
Commonly used for upscaling

Upscaling

Using a box filter

Fixed operation with a constant filter

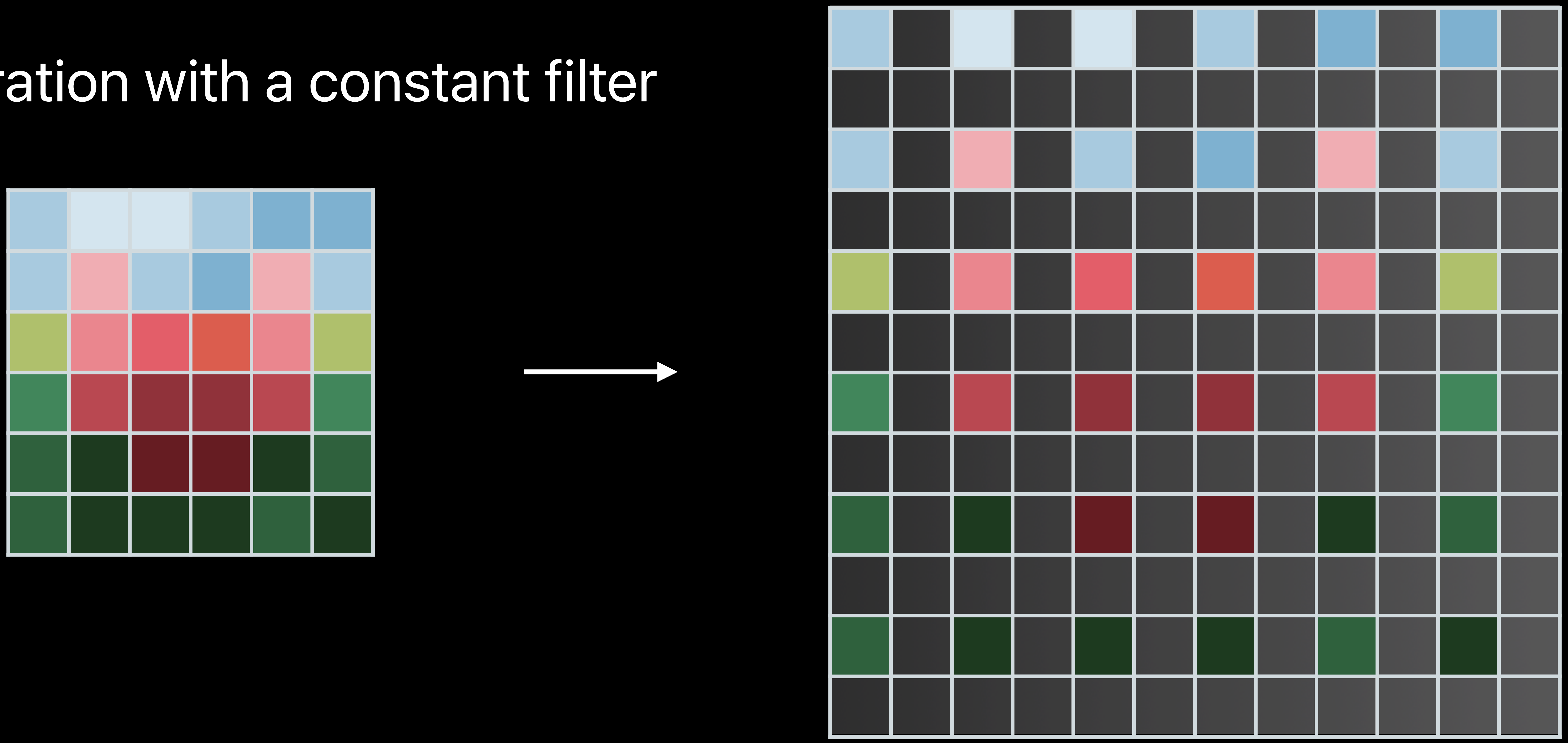
Input
W x H



Output
2W x 2H

Upscaling
Using a box filter

Fixed operation with a constant filter



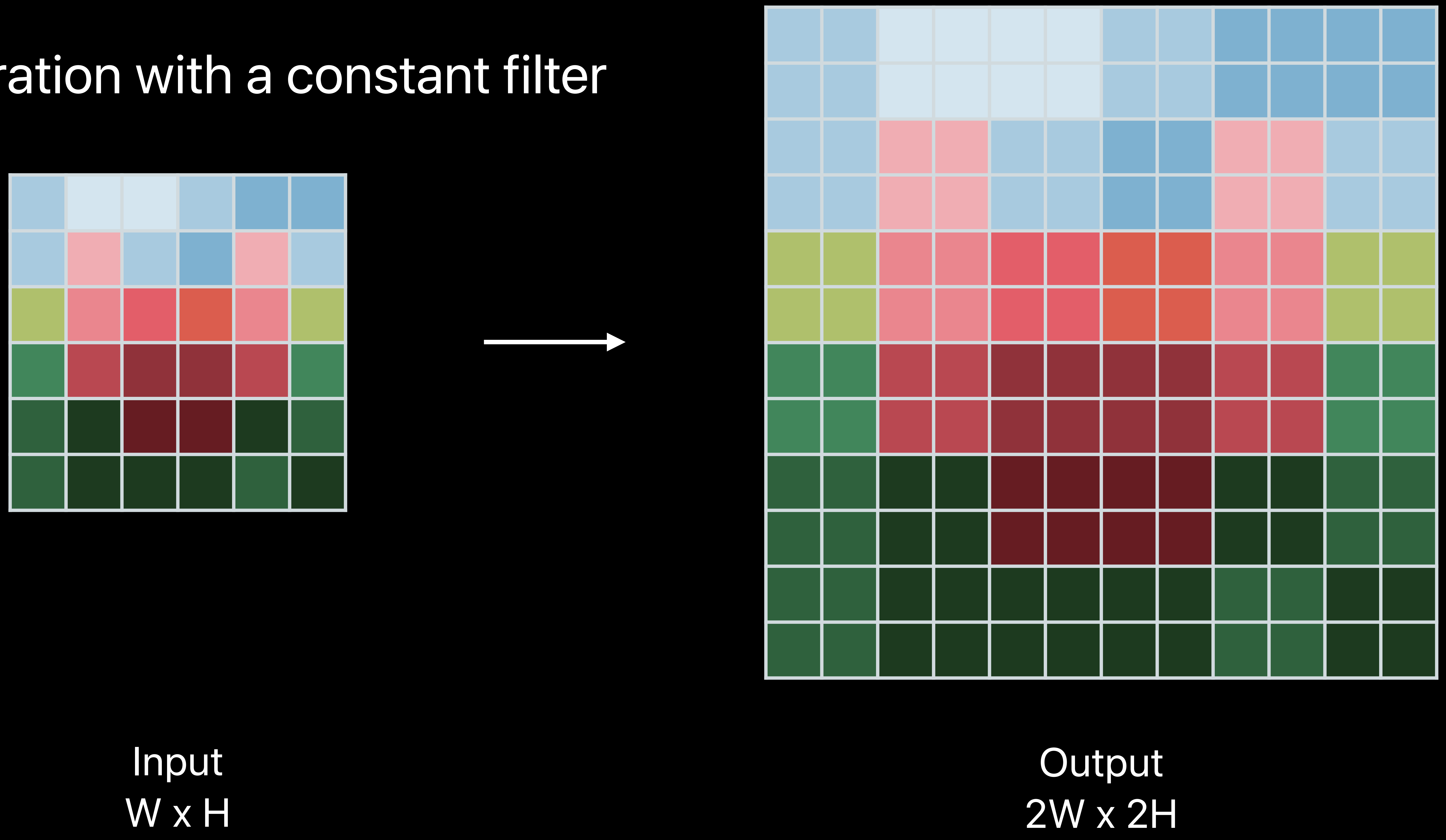
Input
W x H

Output
2W x 2H

Upscaling

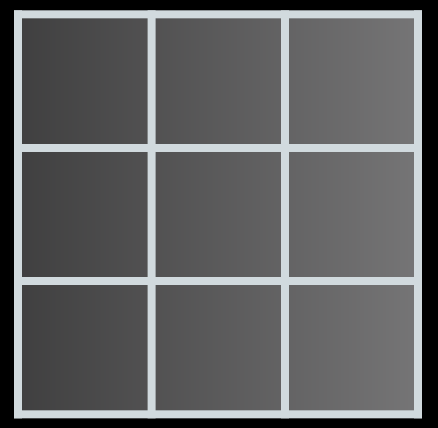
Using a box filter

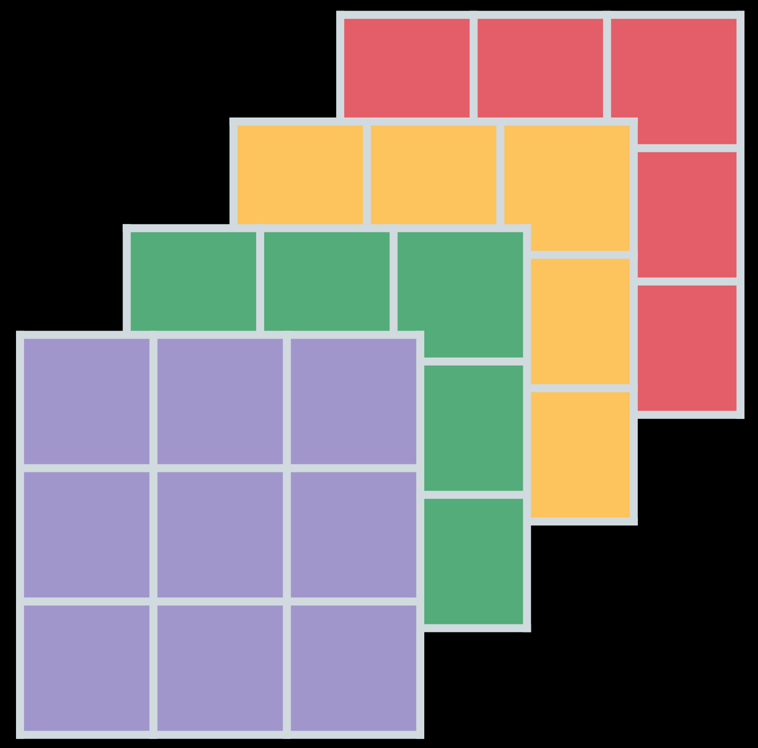
Fixed operation with a constant filter

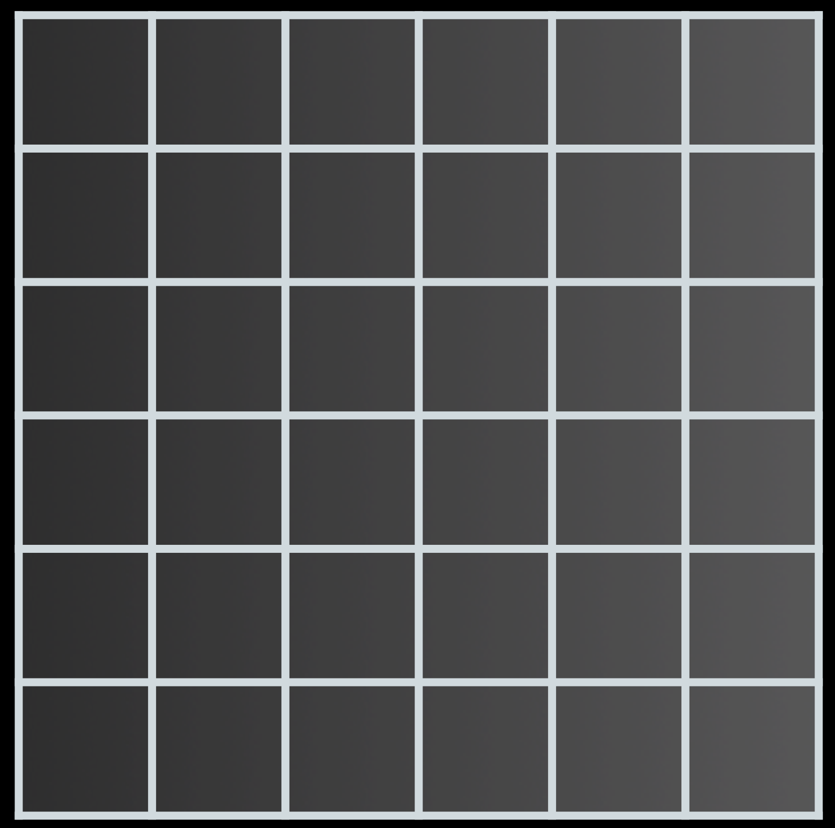


Sub-Pixel Convolution

How it works







Trained Parameters

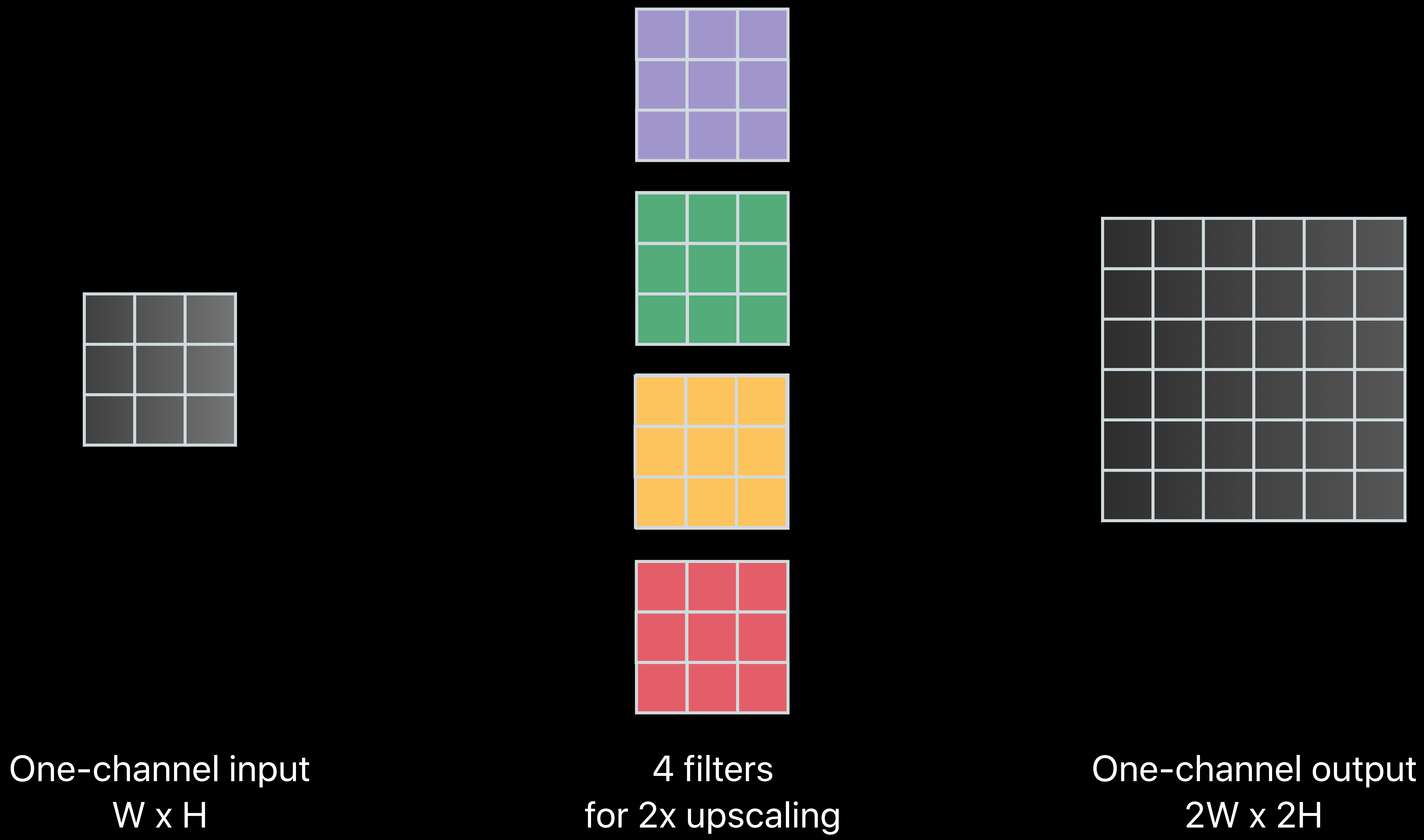
One-channel input
W x H

4 filters
for 2x upscaling

One-channel output
2W x 2H

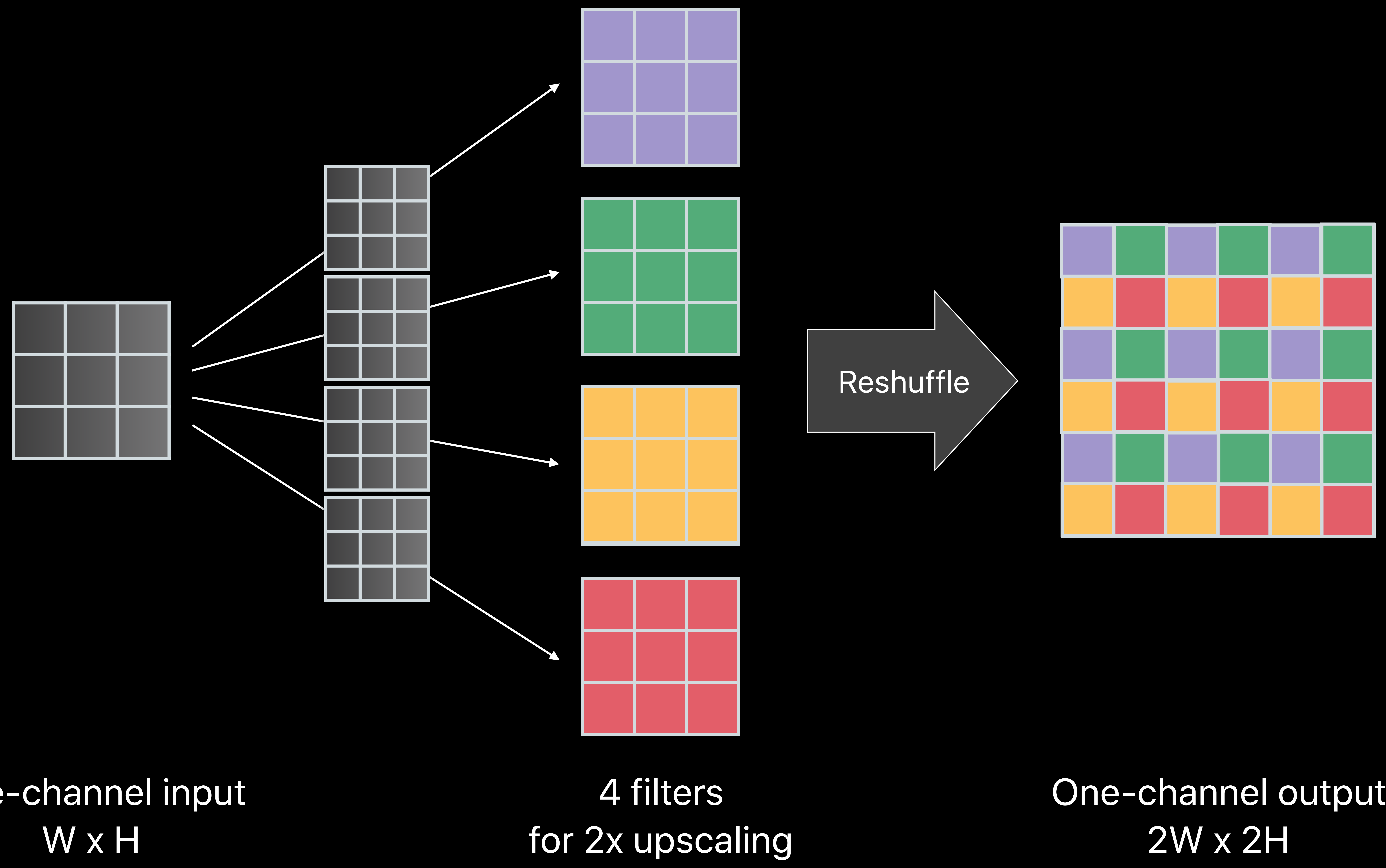
Sub-Pixel Convolution

How it works



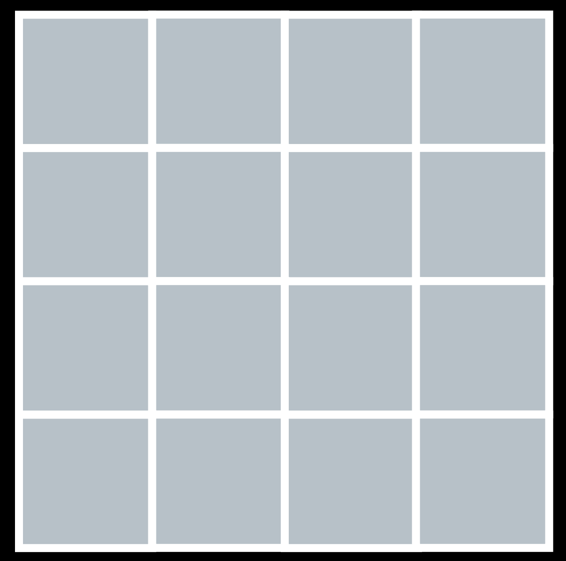
Sub-Pixel Convolution

How it works



Convolution Transpose

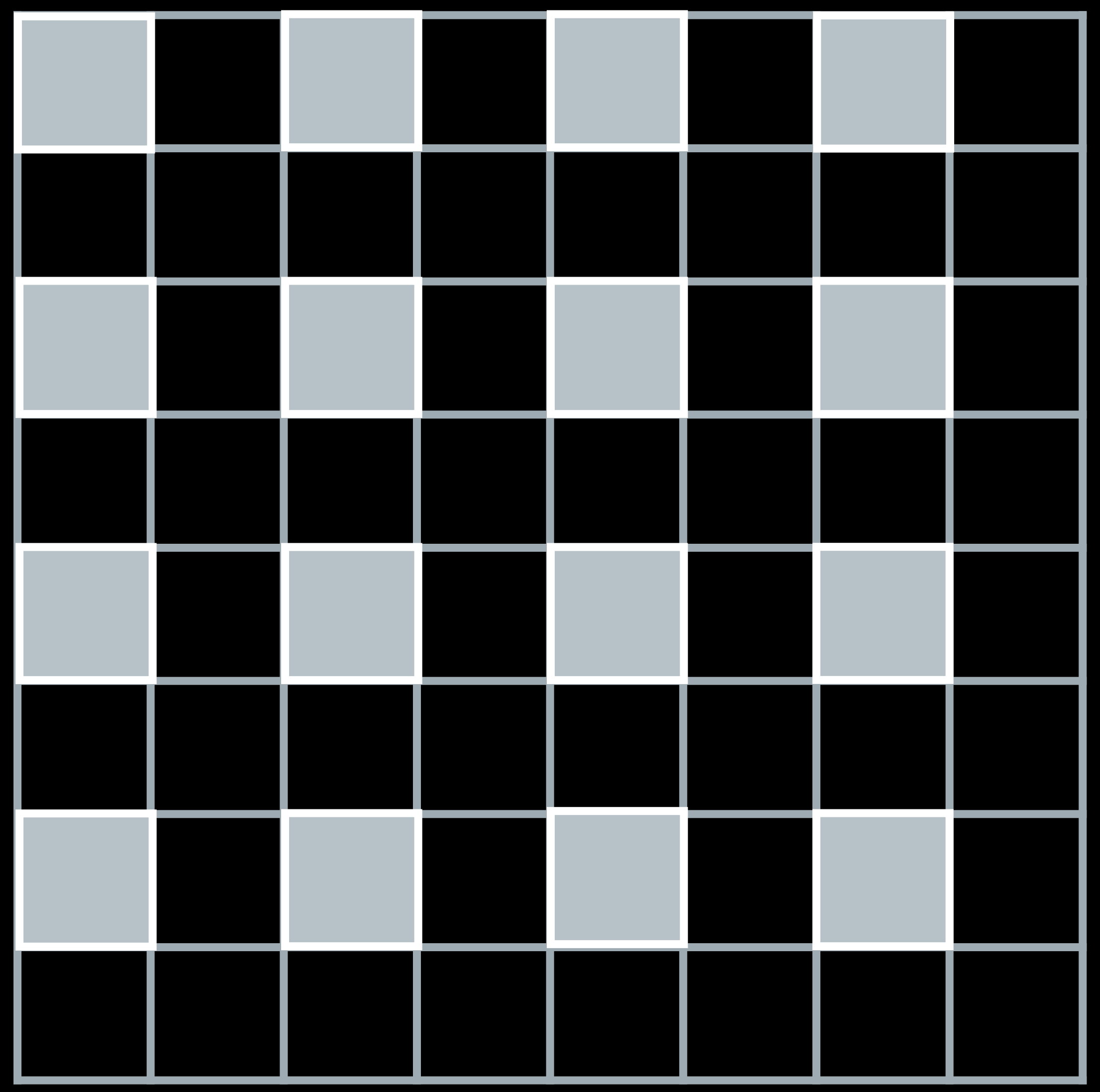
How it works



Input
W x H

Convolution Transpose

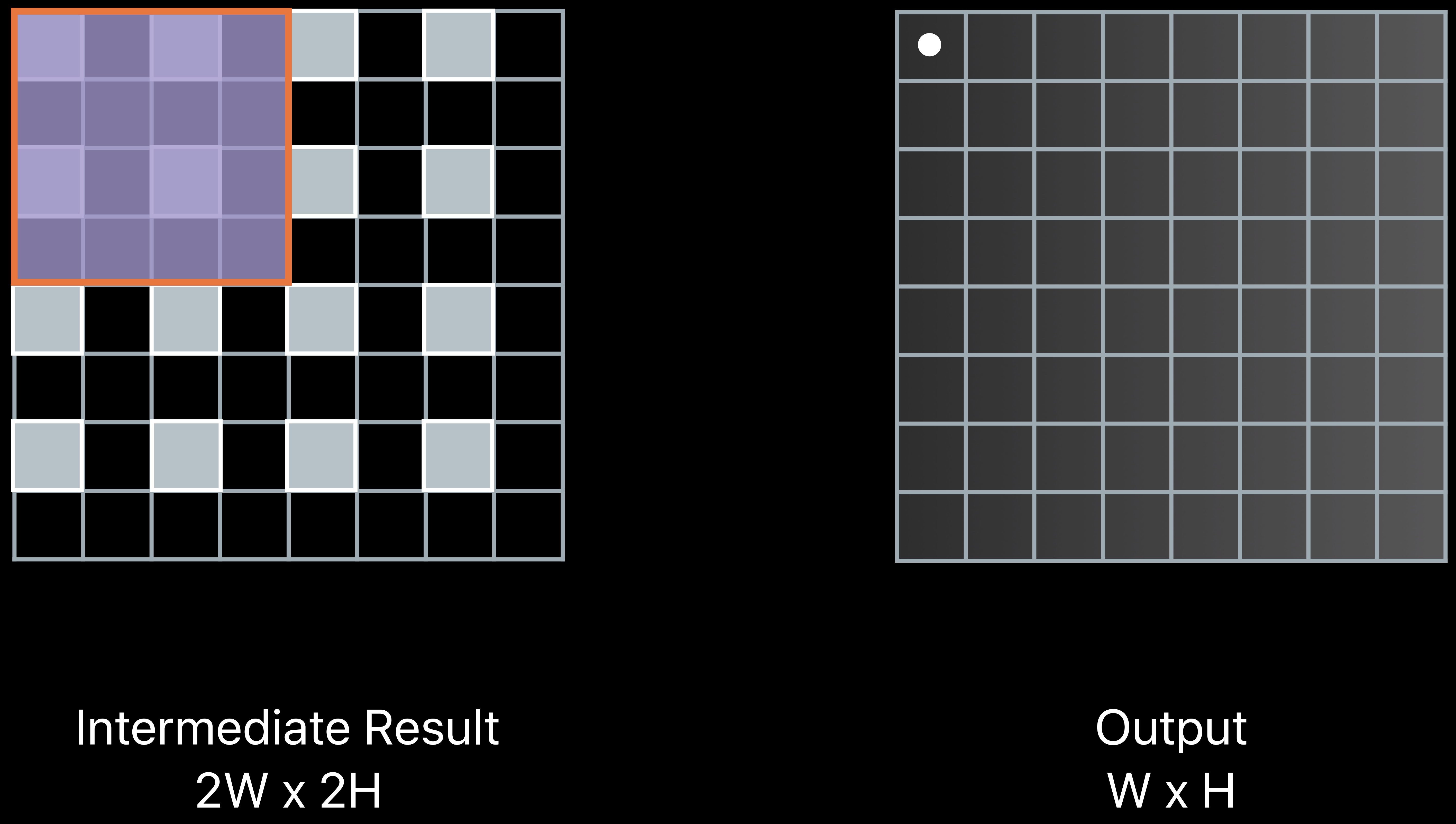
How it works



Input
W x H

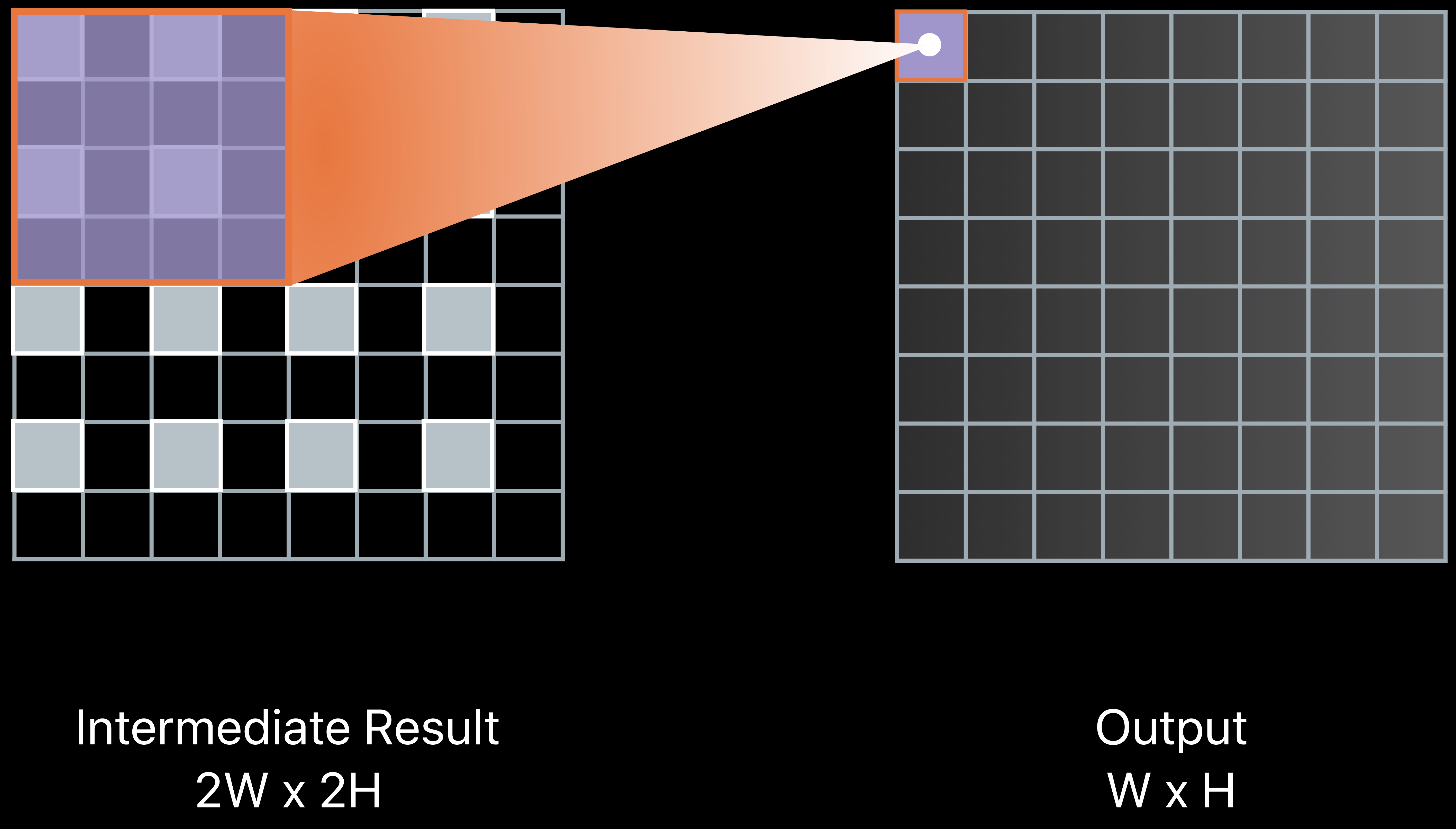
Convolution Transpose

How it works



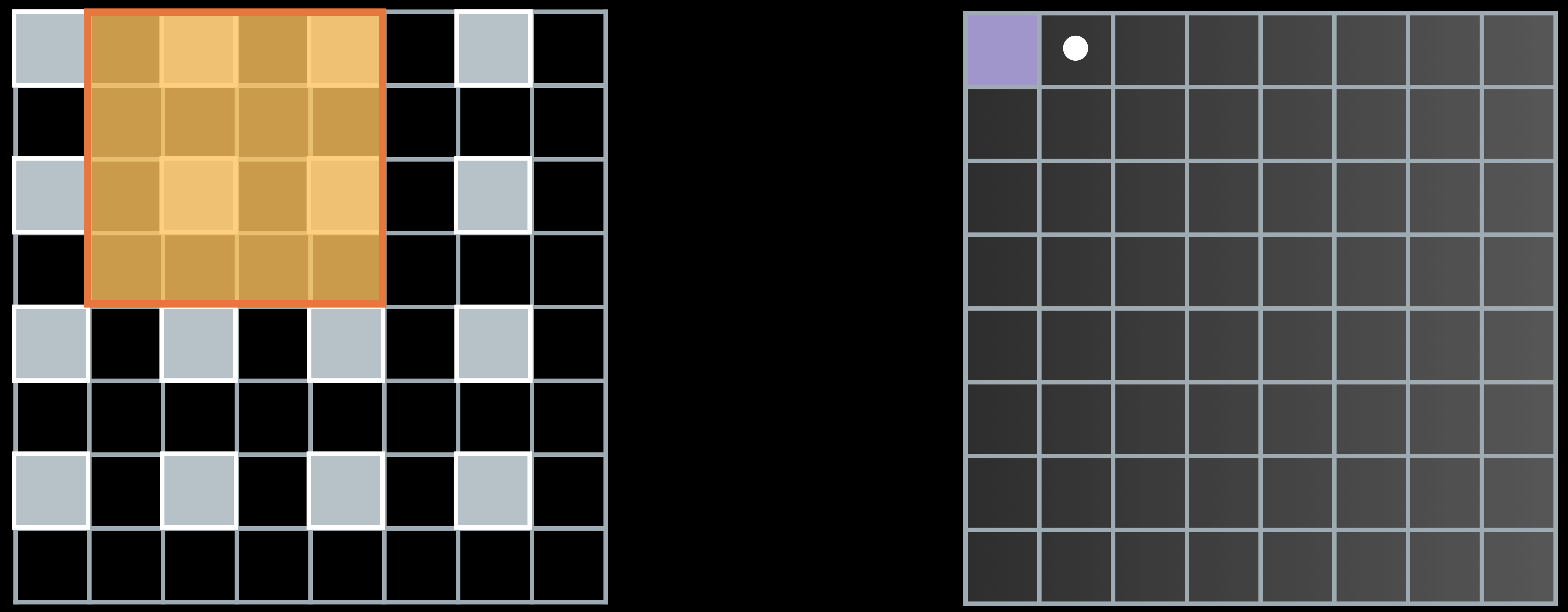
Convolution Transpose

How it works



Convolution Transpose

How it works

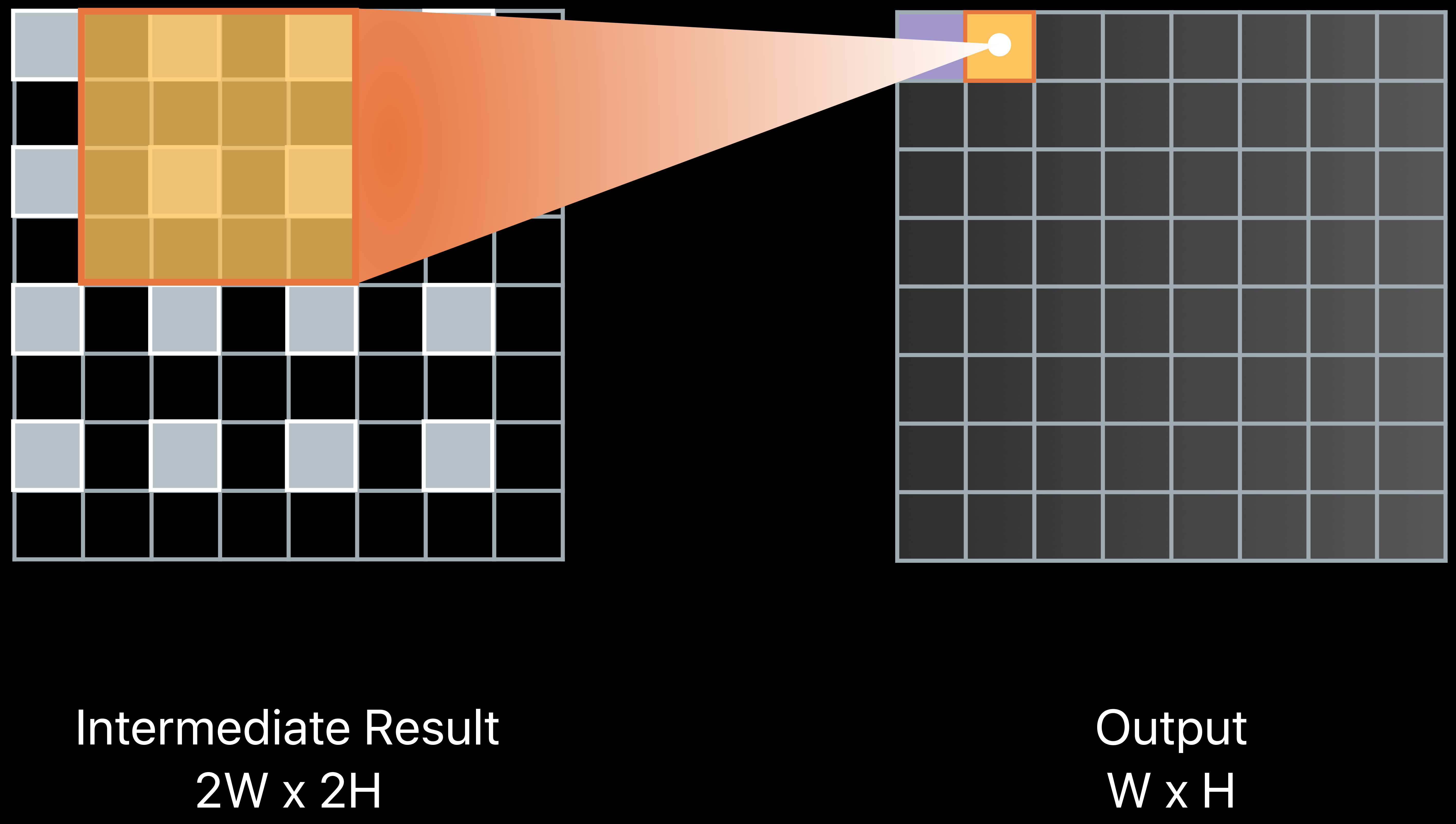


Intermediate Result
2W x 2H

Output
W x H

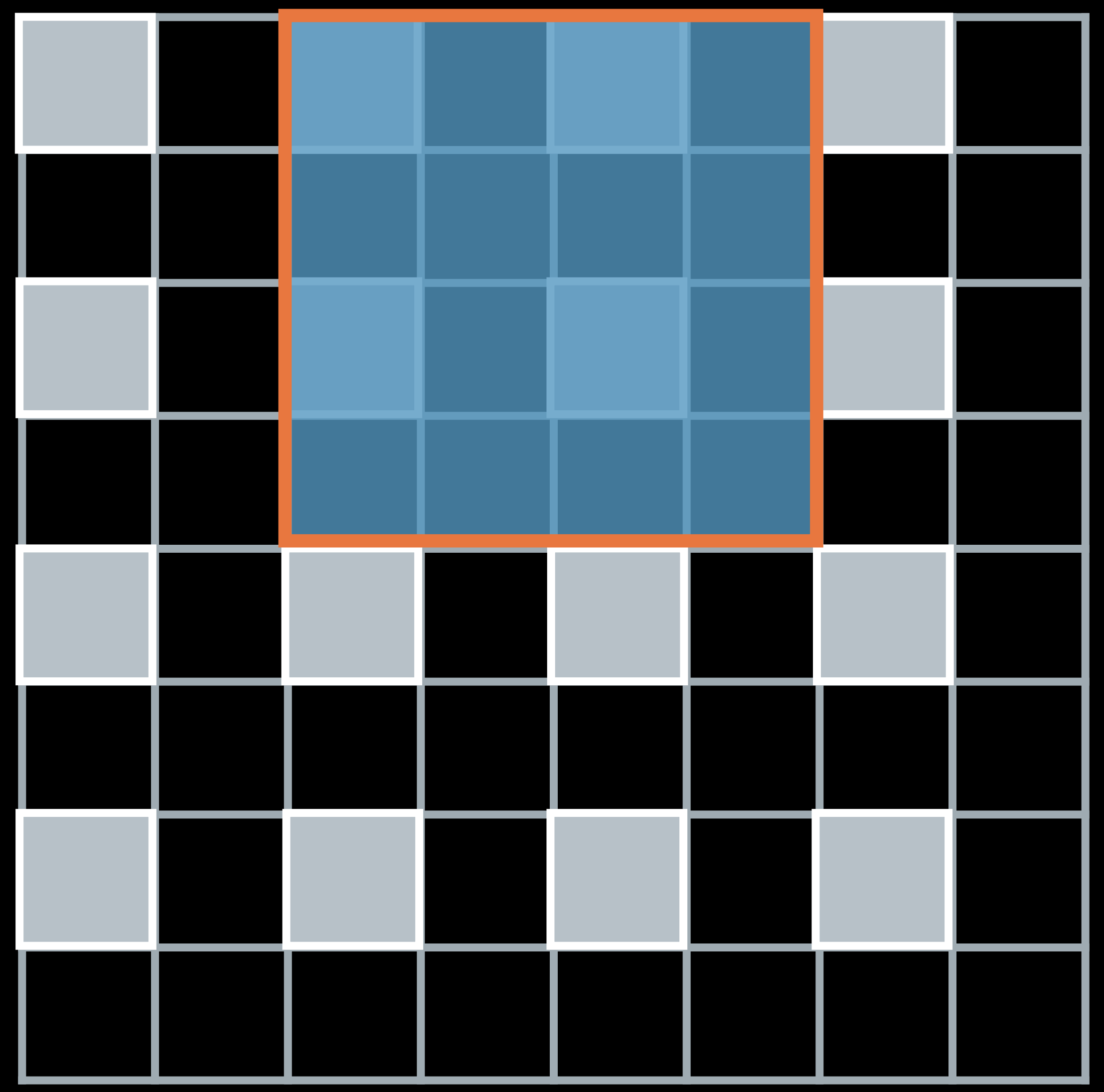
Convolution Transpose

How it works

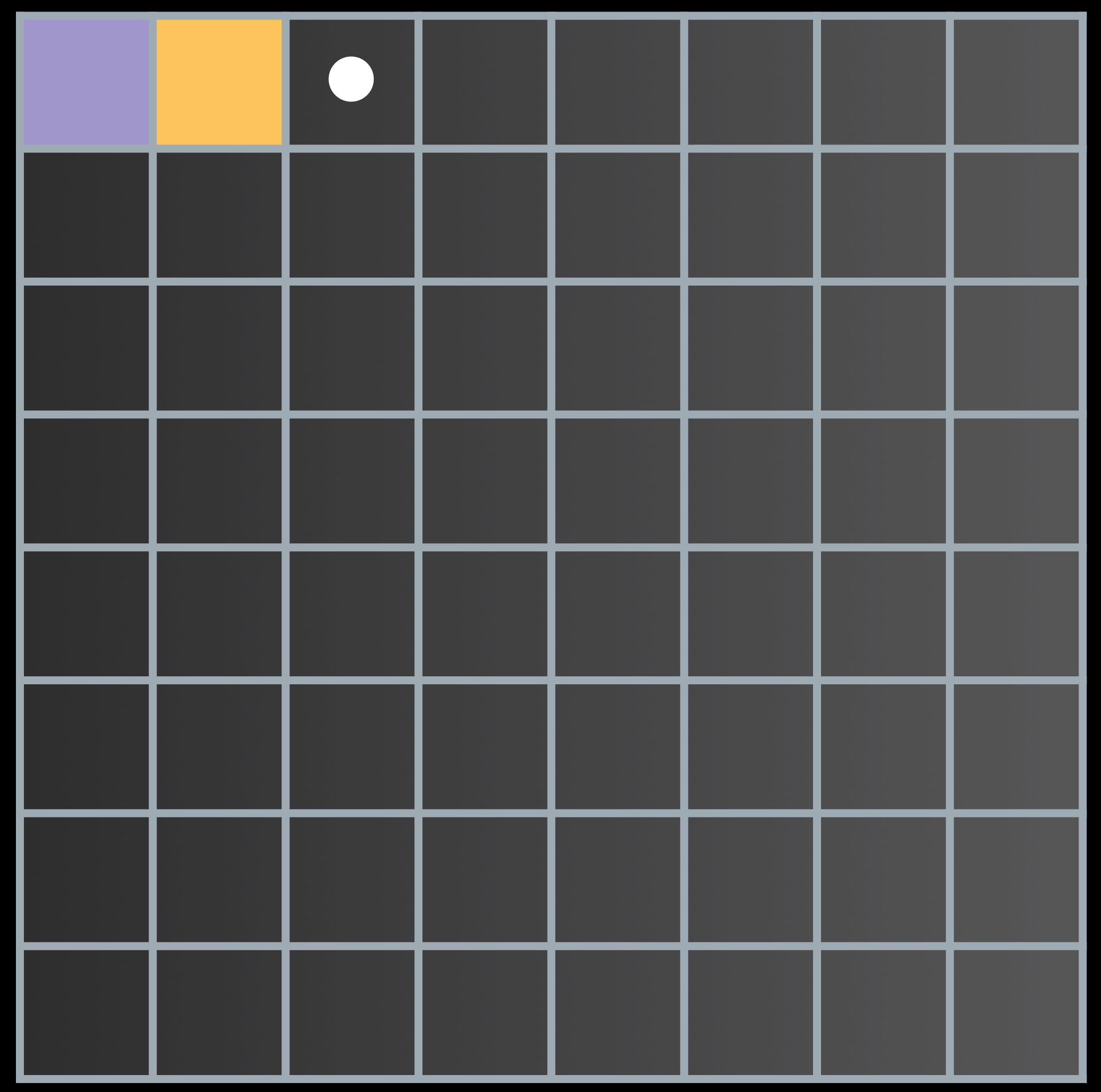


Convolution Transpose

How it works



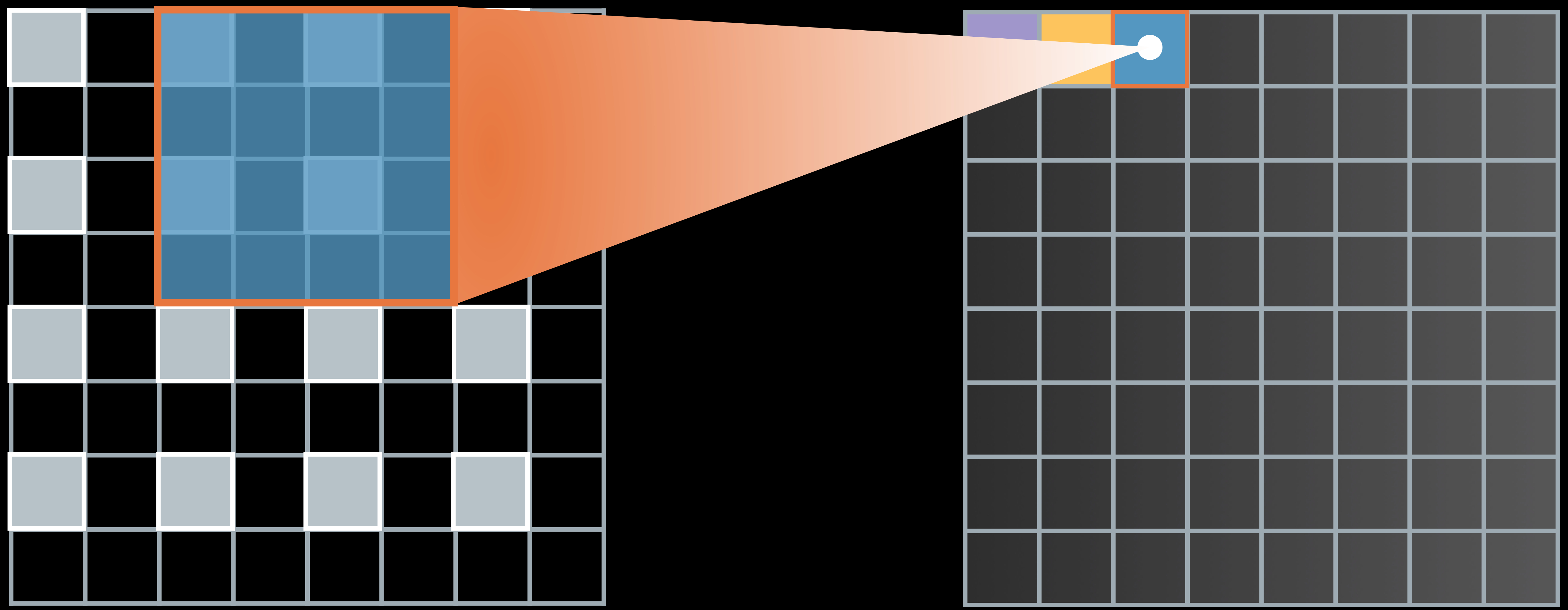
Intermediate Result
2W x 2H



Output
W x H

Convolution Transpose

How it works



Intermediate Result
2W x 2H

Output
W x H

New Convolution Primitives

Example: colorizing black and white images

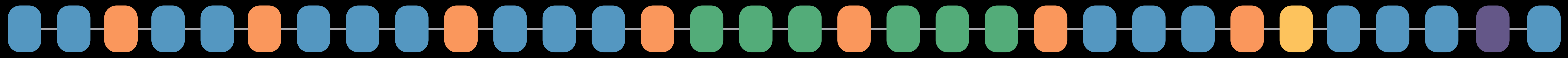
New Convolution Primitives

Example: colorizing black and white images

Input

Output

- Convolution
- Dilated Convolution
- Batch Normalization
- Convolution Transpose
- SoftMax



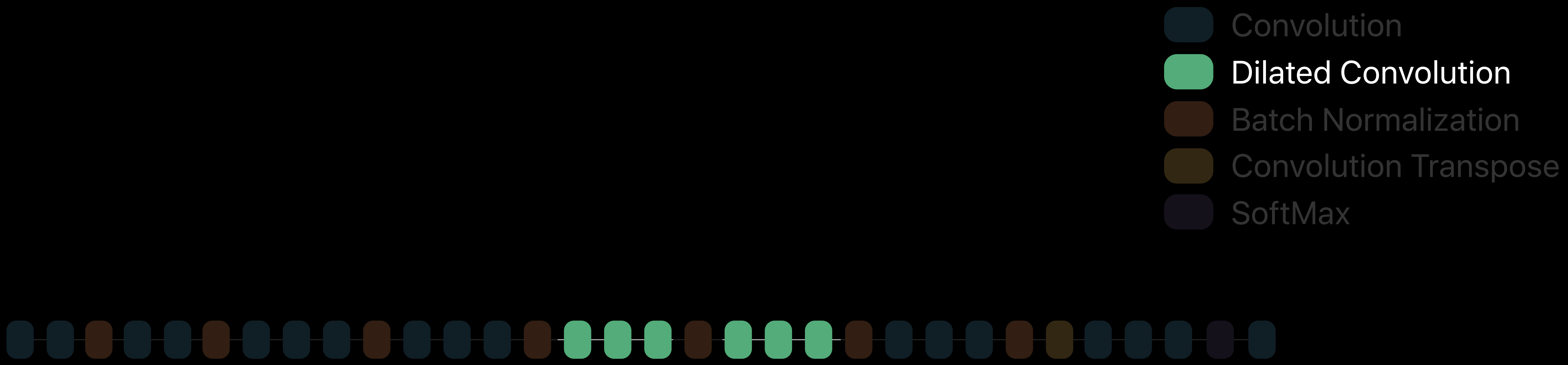
Colorization network*

*Colorful Image Colorization, Richard Zhang, Phillip Isola, Alexei A. Efros, ECCV 2016, http://richzhang.github.io/colorization/

New Convolution Primitives

Example: colorizing black and white images

- Dilated Convolution—integrate wider global context



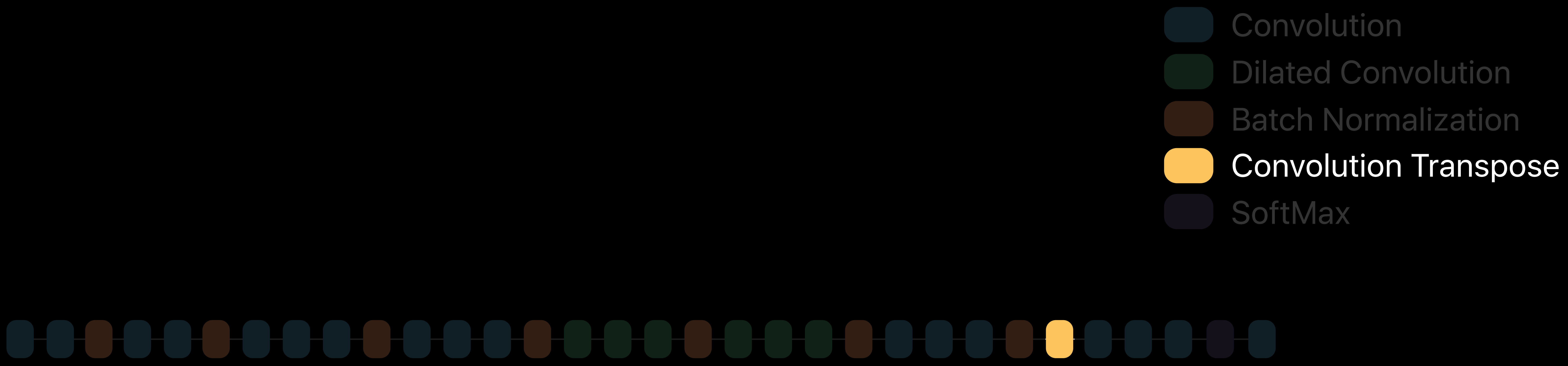
Colorization network*

*Colorful Image Colorization, Richard Zhang, Phillip Isola, Alexei A. Efros, ECCV 2016, http://richzhang.github.io/colorization/

New Convolution Primitives

Example: colorizing black and white images

- Dilated Convolution—integrate wider global context
- Convolution Transpose—upscale output



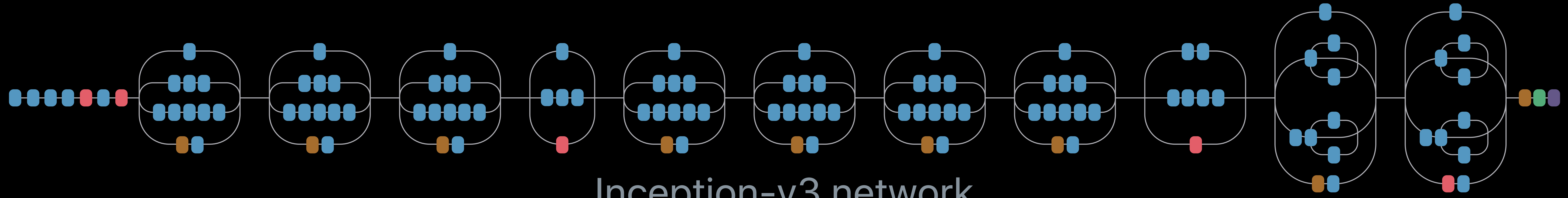
Colorization network*

*Colorful Image Colorization, Richard Zhang, Phillip Isola, Alexei A. Efros, ECCV 2016, http://richzhang.github.io/colorization/

Demo

Image colorization

Performance Improvements in iOS 11

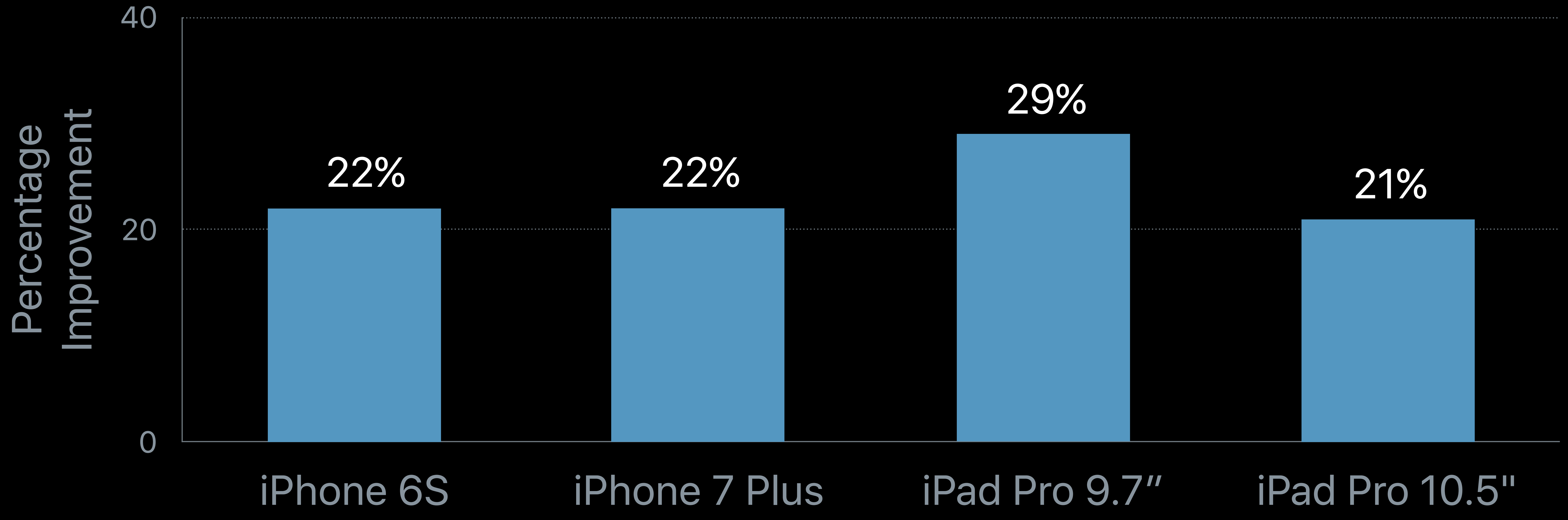


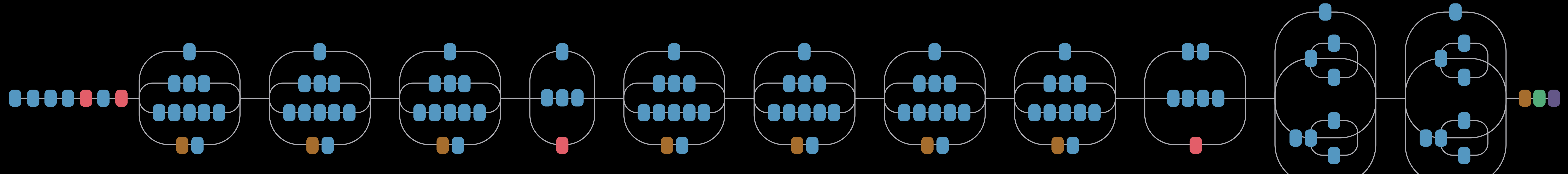
Inception-v3 network

*Rethinking the Inception Architecture for Computer Vision, Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, CVPR 2015, https://arxiv.org/abs/1512.00567

Performance Improvements in iOS 11

Higher is better





Inception-v3 network

*Rethinking the Inception Architecture for Computer Vision, Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, CVPR 2015, https://arxiv.org/abs/1512.00567

Agenda

Recap on Convolutional Neural Networks (CNN)

Convolutional Neural Networks — New Primitives

Neural Network Graph API

Recurrent Neural Networks (RNN)

Neural Network Graph API

Overview

NEW

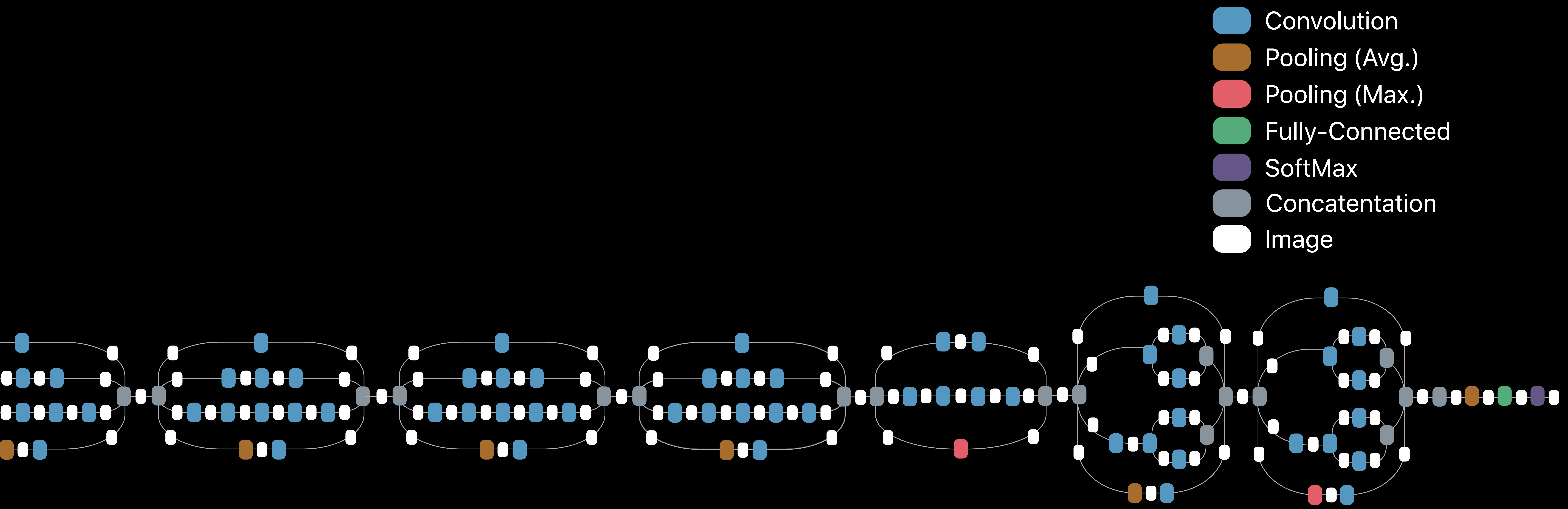
Describe neural network using graph API

Neural Network Graph API

Overview

NEW

Describe neural network using graph API

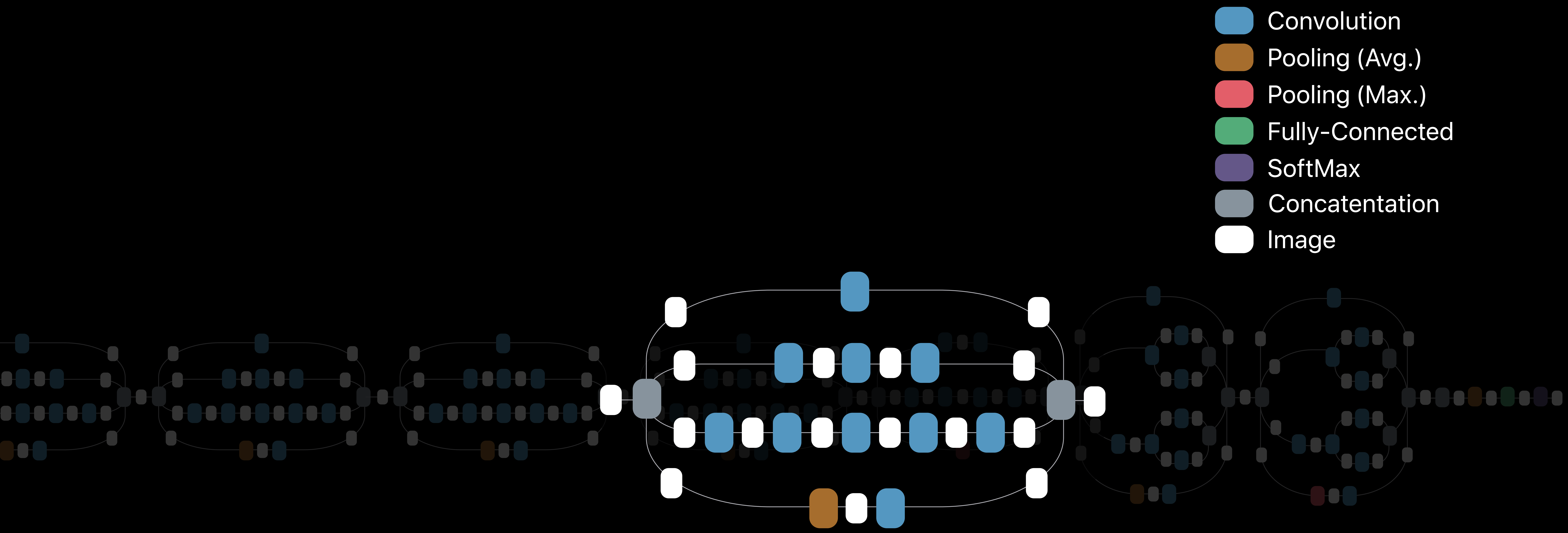


Neural Network Graph API

Overview

NEW

Describe neural network using graph API

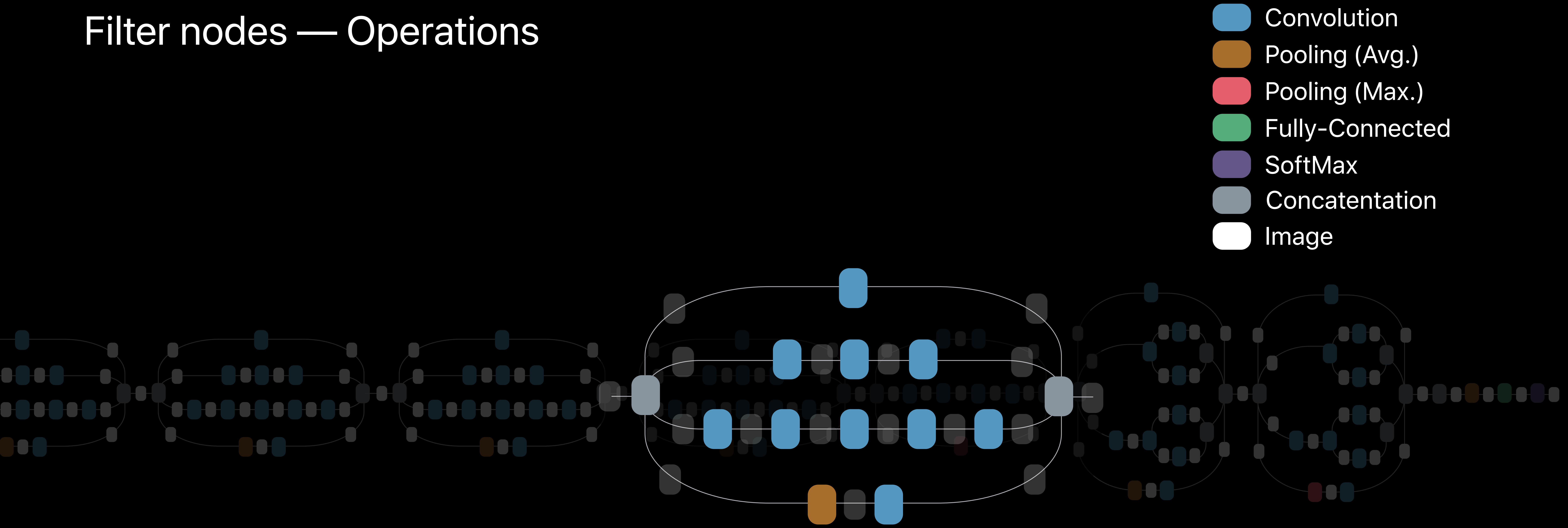


Neural Network Graph API
Overview

NEW

Describe neural network using graph API

Filter nodes — Operations



Neural Network Graph API

Overview

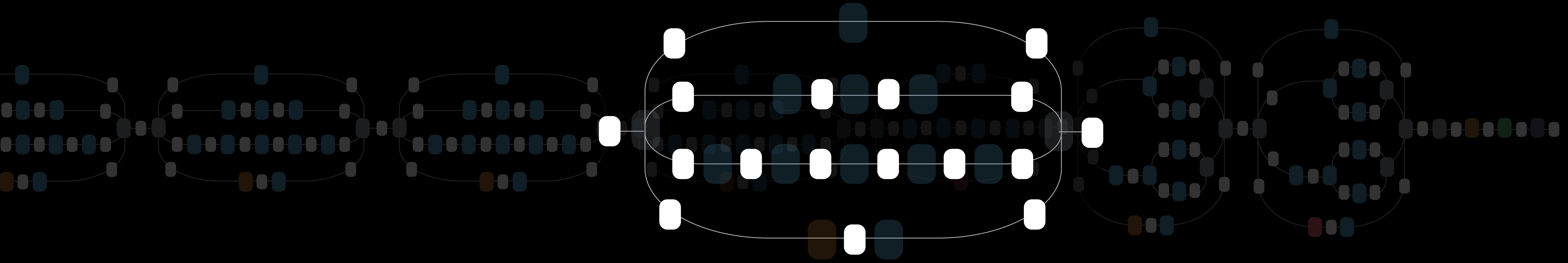
NEW

Describe neural network using graph API

Filter nodes — Operations

Image nodes — Data

Convolution
Pooling (Avg.)
Pooling (Max.)
Fully-Connected
SoftMax
Concatentation
Image



Neural Network Graph API

Ease of use

Compact representation

Neural Network Graph API

Ease of use

Compact representation

Save and restore across platforms (`NSSecureCoding`)

Neural Network Graph API

Ease of use

Compact representation

Save and restore across platforms (`NSSecureCoding`)

Initialize once, reuse

Neural Network Graph API

Ease of use

Compact representation

Save and restore across platforms (`NSSecureCoding`)

Initialize once, reuse

Execute graph on GPU with single call

Neural Network Graph API

Ease of use

Compact representation

Save and restore across platforms (`NSSecureCoding`)

Initialize once, reuse

Execute graph on GPU with single call

No intermediate images to manage, just input/output

Neural Network Graph API

Ease of use

Compact representation

Save and restore across platforms (`NSSecureCoding`)

Initialize once, reuse

Execute graph on GPU with single call

No intermediate images to manage, just input/output

Auto-configuration of image sizes, padding, centering

Neural Network Graph API

Ease of use

Compact representation

Save and restore across platforms (`NSSecureCoding`)

Initialize once, reuse

Execute graph on GPU with single call

No intermediate images to manage, just input/output

Auto-configuration of image sizes, padding, centering

MetalImageRecognition code sample* — 4x less code with NN Graph API

https://developer.apple.com/library/content/samplecode/MetalImageRecognition

Neural Network Graph API

Deliver best performance

Easy to parallelize between CPU and GPU

Neural Network Graph API

Deliver best performance

Easy to parallelize between CPU and GPU

Fuse graph nodes

Neural Network Graph API

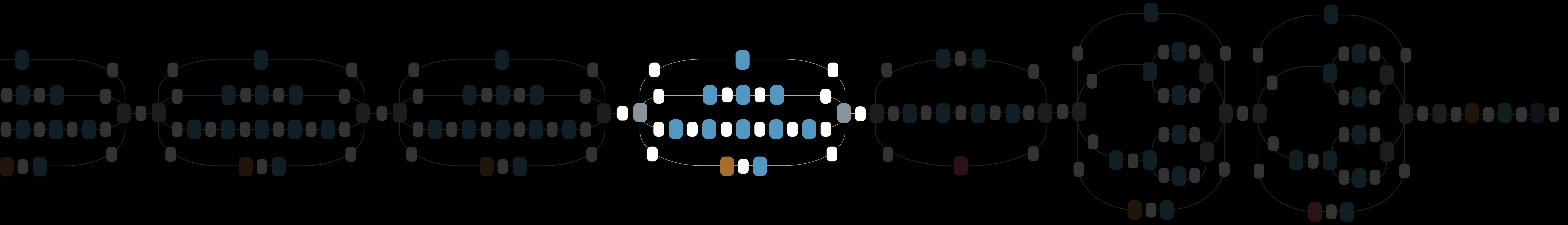
Deliver best performance

NEW

Easy to parallelize between CPU and GPU

Fuse graph nodes

Execute graph nodes concurrently



Neural Network Graph API

Deliver best performance

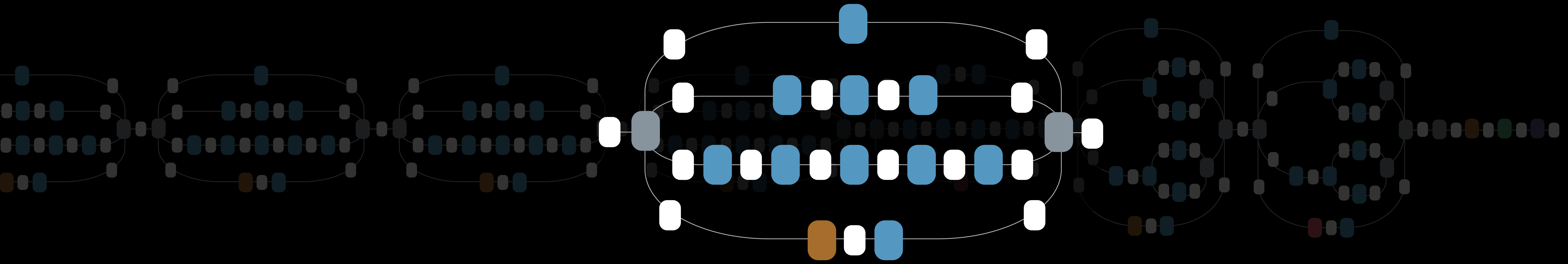
NEW

Easy to parallelize between CPU and GPU

Fuse graph nodes

Execute graph nodes concurrently

- Convolution
- Pooling (Avg.)
- Pooling (Max.)
- Fully-Connected
- SoftMax
- Concatentation
- Image



Neural Network Graph API

Deliver best performance

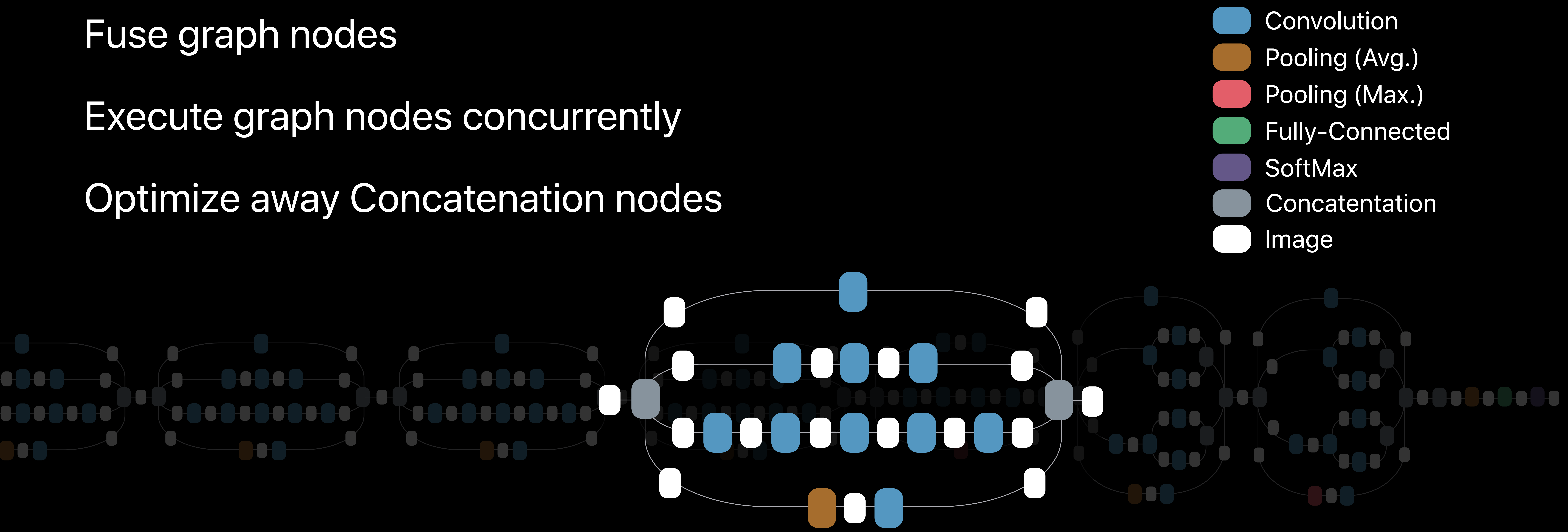
NEW

Easy to parallelize between CPU and GPU

Fuse graph nodes

Execute graph nodes concurrently

Optimize away Concatenation nodes



Neural Network Graph API

Deliver best performance

NEW

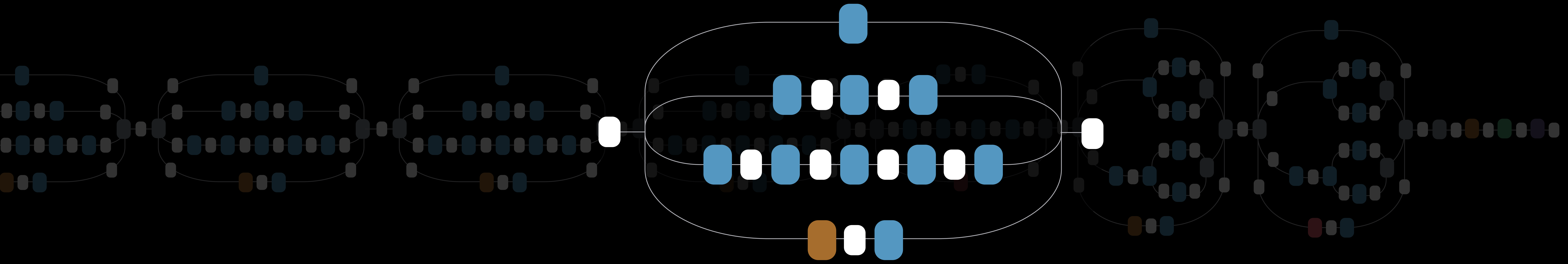
Easy to parallelize between CPU and GPU

Fuse graph nodes

Execute graph nodes concurrently

Optimize away Concatenation nodes

Convolution
Pooling (Avg.)
Pooling (Max.)
Fully-Connected
SoftMax
Concatentation
Image



Filter Nodes

Convolution node

Create a MPSNNConvolutionNode with data source provider

```
let conv1 = MPSCNNConvolutionNode(source: MPSNNImageNode(handle: nil),
                                  weights: MyWeights(file:"conv1.dat"))
```

Filter Nodes

Convolution node

Create a MPSNNConvolutionNode with data source provider

```
let conv1 = MPSCNNConvolutionNode(source: MPSNNImageNode(handle: nil),
                                  weights: MyWeights(file:"conv1.dat"))
```

Filter Nodes

Convolution node

Create a MPSNNConvolutionNode with data source provider

```
let conv1 = MPSCNNConvolutionNode(source: MPSNNImageNode(handle: nil),
                                  weights: MyWeights(file:"conv1.dat"))
```

Feeding Parameters to Convolution Layer

Just-in-time loading and purging of weights data

Minimize memory footprint

```
class MyWeights: NSObject, MPSCNNConvolutionDataSource {
    // Initialize the data source object
    init(file: String) {…}

    public func load() -> Bool {…}
    public func descriptor() -> MPSCNNConvolutionDescriptor {…}
    public func weights() -> UnsafeMutableRawPointer {…}
    public func purge() {…}
}
```

Feeding Parameters to Convolution Layer

Just-in-time loading and purging of weights data

Minimize memory footprint

```
class MyWeights: NSObject, MPSCNNConvolutionDataSource {
    // Initialize the data source object
    init(file: String) {…}

    public func load() -> Bool {…}
    public func descriptor() -> MPSCNNConvolutionDescriptor {…}
    public func weights() -> UnsafeMutableRawPointer {…}
    public func purge() {…}
}
```

Feeding Parameters to Convolution Layer

Just-in-time loading and purging of weights data

Minimize memory footprint

```
class MyWeights: NSObject, MPSCNNConvolutionDataSource {
    // Initialize the data source object
    init(file: String) {…}

    public func load() -> Bool {…}
    public func descriptor() -> MPSCNNConvolutionDescriptor {…}
    public func weights() -> UnsafeMutableRawPointer {…}
    public func purge() {…}
}
```

```
// Example: create a graph
func makeGraph() -> MPSNNImageNode {

conv1

pool1

conv2

pool2

conv3

pool3

conv4

fc1

fc2

}
```


```
// Example: create a graph
func makeGraph() -> MPSNNImageNode {

    let conv1 = MPSCNNConvolutionNode(source: MPSNNImageNode(handle: nil), weights: MyWeights(file:"conv1.dat"))

}
```

```
// Example: create a graph
func makeGraph() -> MPSNNImageNode {

    let conv1 = MPSCNNConvolutionNode(source: MPSNNImageNode(handle: nil), weights: MyWeights(file:"conv1.dat"))

    let pool1 = MPSCNNPoolingMaxNode(source:   conv1.resultImage, filterSize: 2)

}
```

conv1

pool1

conv2

pool2

conv3

pool3

conv4

fc1

fc2


```swift
// Example: create a graph
func makeGraph() -> MPSNNImageNode {

    let conv1 = MPSCNNConvolutionNode(source: MPSNNImageNode(handle: nil), weights: MyWeights(file:"conv1.dat"))

    let pool1 = MPSCNNPoolingMaxNode(source:   conv1.resultImage, filterSize: 2)

    let conv2 = MPSCNNConvolutionNode(source:  pool1.resultImage, weights: MyWeights(file:"conv2.dat"))

    let pool2 = MPSCNNPoolingMaxNode(source:   conv2.resultImage, filterSize: 2)

    let conv3 = MPSCNNConvolutionNode(source:  pool2.resultImage, weights: MyWeights(file:"conv3.dat"))

    let pool3 = MPSCNNPoolingMaxNode(source:   conv3.resultImage, filterSize: 2)

    let conv4 = MPSCNNConvolutionNode(source:  pool3.resultImage, weights: MyWeights(file:"conv4.dat"))

    let fc1 = MPSCNNFullyConnectedNode(source: conv4.resultImage, weights: MyWeights(file:"fc1.dat"))

    let fc2 = MPSCNNFullyConnectedNode(source:   fc1.resultImage, weights: MyWeights(file:"fc2.dat"))

    return                                       fc2.resultImage
}
```

```swift
// Example: create a graph
func makeGraph() -> MPSNNImageNode {

    let conv1 = MPSCNNConvolutionNode(source: MPSNNImageNode(handle: nil), weights: MyWeights(file:"conv1.dat"))

    let pool1 = MPSCNNPoolingMaxNode(source: conv1.resultImage, filterSize: 2)

    let conv2 = MPSCNNConvolutionNode(source: pool1.resultImage, weights: MyWeights(file:"conv2.dat"))

    let pool2 = MPSCNNPoolingMaxNode(source: conv2.resultImage, filterSize: 2)

    let conv3 = MPSCNNConvolutionNode(source: pool2.resultImage, weights: MyWeights(file:"conv3.dat"))

    let pool3 = MPSCNNPoolingMaxNode(source: conv3.resultImage, filterSize: 2)

    let conv4 = MPSCNNConvolutionNode(source: pool3.resultImage, weights: MyWeights(file:"conv4.dat"))

    let fc1 = MPSCNNFullyConnectedNode(source conv4.resultImage, weights: MyWeights(file:"fc1.dat"))

    let fc2 = MPSCNNFullyConnectedNode(source fc1.resultImage, weights: MyWeights(file:"fc2.dat"))

    return fc2.resultImage
}
```

```swift
// Example: execute graph on the GPU
// Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.encode(to: commandBuffer,
                           sourceImages: [input])
// Tell GPU to start executing work and wait until GPU work is done
commandBuffer.commit()
commandBuffer.waitUntilCompleted()
```

```swift
// Example: execute graph on the GPU
// Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.encode(to: commandBuffer,
                           sourceImages: [input])
// Tell GPU to start executing work and wait until GPU work is done
commandBuffer.commit()
commandBuffer.waitUntilCompleted()
```

```swift
// Example: execute graph on the GPU
// Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.encode(to: commandBuffer,
                           sourceImages: [input])
// Tell GPU to start executing work and wait until GPU work is done
commandBuffer.commit()
commandBuffer.waitUntilCompleted()
```

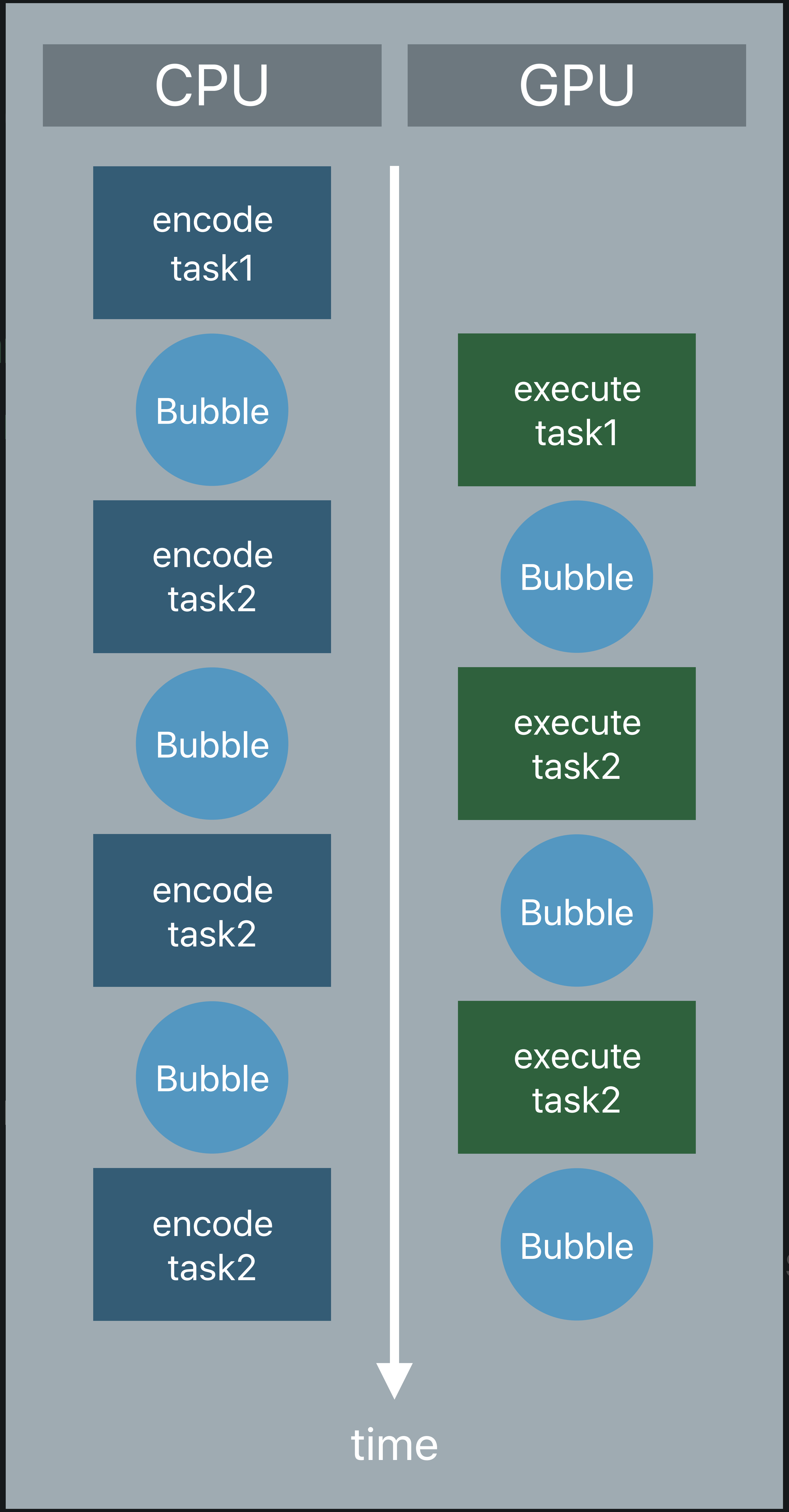
```swift
// Example: execute graph on the GPU
// Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.encode(to: commandBuffer,
                           sourceImages: [input])
// Tell GPU to start executing work and wait until GPU work is done
commandBuffer.commit()
commandBuffer.waitUntilCompleted()
```

```swift
// Example: execute graph on the GPU
// Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.encode(to: commandBuffer,
                           sourceImages: [input])
// Tell GPU to start executing work and wait until GPU work is done
commandBuffer.commit()
commandBuffer.waitUntilCompleted()
```

```swift
// Example: execute graph on the GPU
// Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.encode(to: commandBuffer,
                           sourceImages: [input])
// Tell GPU to start executing work and wait until GPU work is done
commandBuffer.commit()
commandBuffer.waitUntilCompleted()
```

```swift
// Example: execute graph on the GPU
// Metal setup
let device = MTLCreateSystemDefaultDevice()!
let commandQueue = device.makeCommandQueue()
let commandBuffer = commandQueue.makeCommandBuffer()

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.encode(to: commandBuffer,
                           sourceImages: [input])
// Tell GPU to start executing work and wait until GPU work is done
commandBuffer.commit()
commandBuffer.waitUntilCompleted()
```

```
// Example: execute graph on the
// Metal setup
let device = MTLCreateSystemDefau
let commandQueue = device.makeCom
let commandBuffer = commandQueue.

// Initialize graph
let graph = MPSNNGraph(device: de                          ())

// Create input image
let input = MPSImage(texture: tex

// Encode graph
let output = graph?.encode(to: co
                          source
// Tell GPU to start executing wo                          s done
commandBuffer.commit()
commandBuffer.waitUntilCompleted(
```



```swift
// Example: execute graph on the GPU asynchronously
// Metal setup
let device = MTLCreateSystemDefaultDevice()!

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.executeAsync(sourceImages: [input]) {
   resultImage, error in
     // check for error and use resultImage inside closure
}

// Don't wait, encode new GPU task
```

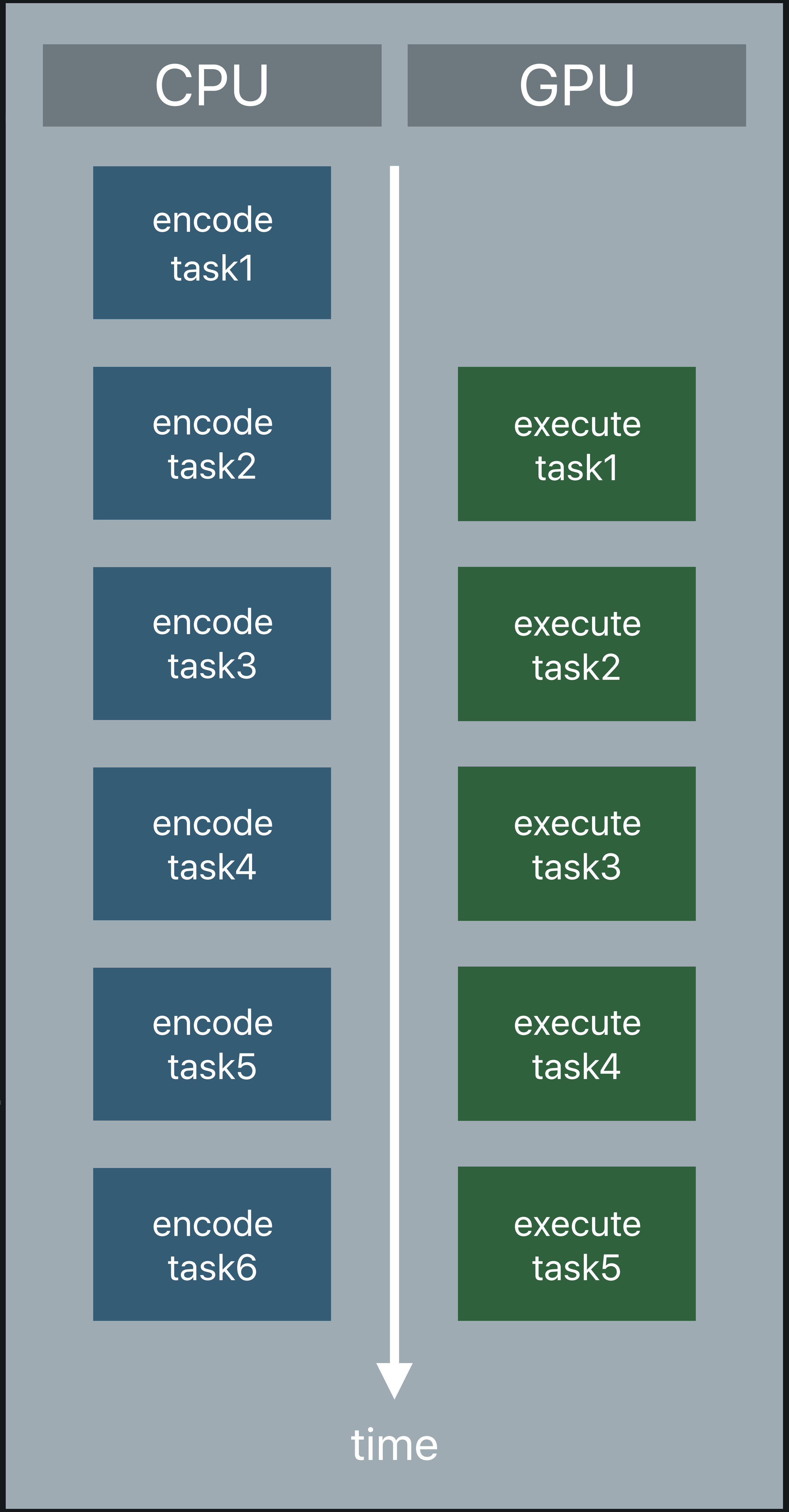
```swift
// Example: execute graph on the GPU asynchronously
// Metal setup
let device = MTLCreateSystemDefaultDevice()!

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.executeAsync(sourceImages: [input]) {
   resultImage, error in
     // check for error and use resultImage inside closure
}

// Don't wait, encode new GPU task
```

```swift
// Example: execute graph on the GPU asynchronously
// Metal setup
let device = MTLCreateSystemDefaultDevice()!

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.executeAsync(sourceImages: [input]) {
   resultImage, error in
     // check for error and use resultImage inside closure
}

// Don't wait, encode new GPU task
```

```swift
// Example: execute graph on the GPU asynchronously
// Metal setup
let device = MTLCreateSystemDefaultDevice()!

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.executeAsync(sourceImages: [input]) {
   resultImage, error in
     // check for error and use resultImage inside closure
}

// Don't wait, encode new GPU task
```

```swift
// Example: execute graph on the GPU asynchronously
// Metal setup
let device = MTLCreateSystemDefaultDevice()!

// Initialize graph
let graph = MPSNNGraph(device: device, resultImage: makeGraph())

// Create input image
let input = MPSImage(texture: texture, …)

// Encode graph
let output = graph?.executeAsync(sourceImages: [input]) {
   resultImage, error in
     // check for error and use resultImage inside closure
}

// Don't wait, encode new GPU task
```

```
// Example: execute graph on the
// Metal setup
let device = MTLCreateSystemDefau

// Initialize graph
let graph = MPSNNGraph(device: de                                    ())

// Create input image
let input = MPSImage(texture: tex

// Encode graph
let output = graph?.executeAsync(
   resultImage, error in
     // check for error and use r
}

// Don't wait, encode new GPU tas
```



Demo

Inception-v3 using Neural Network Graph API

Agenda

Recap on Convolutional Neural Networks (CNN)

Convolutional Neural Networks — New Primitives

Neural Network Graph API

Recurrent Neural Networks (RNN)

What Are Recurrent Neural Networks?

CNN

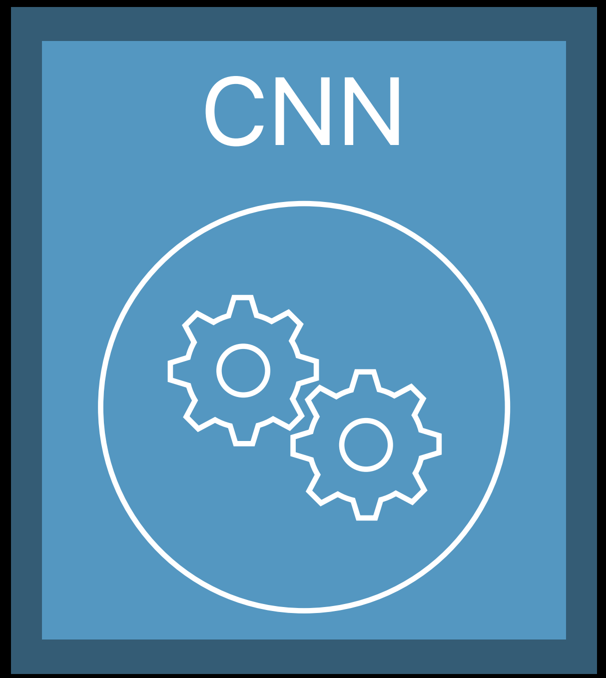
One - to - one

One input

Image

CNN

One - to - one



Inference

dog
grass
...

One input

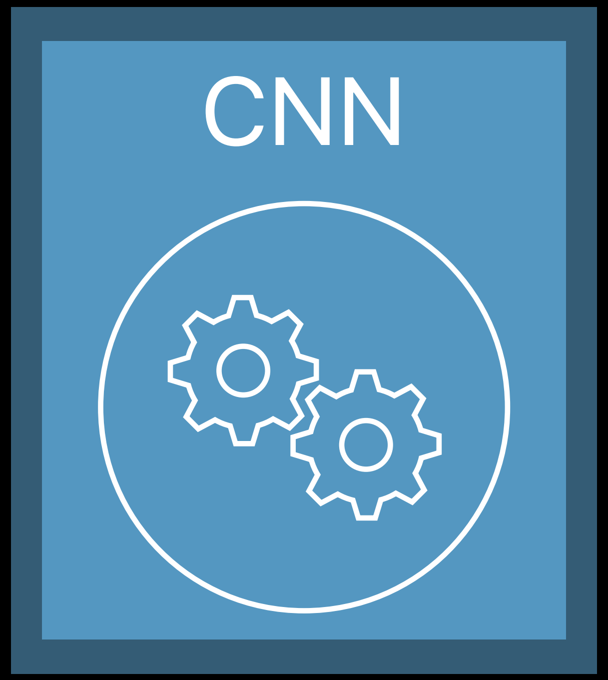
Image

One output

Set of probabilities

RNN

Sequences: one - to - many



Inference

RNN

Sequences: one - to - many

→ CNN → RNN → A black and white dog laying in the grass

Inference

Inference

One input

Set of probabilities

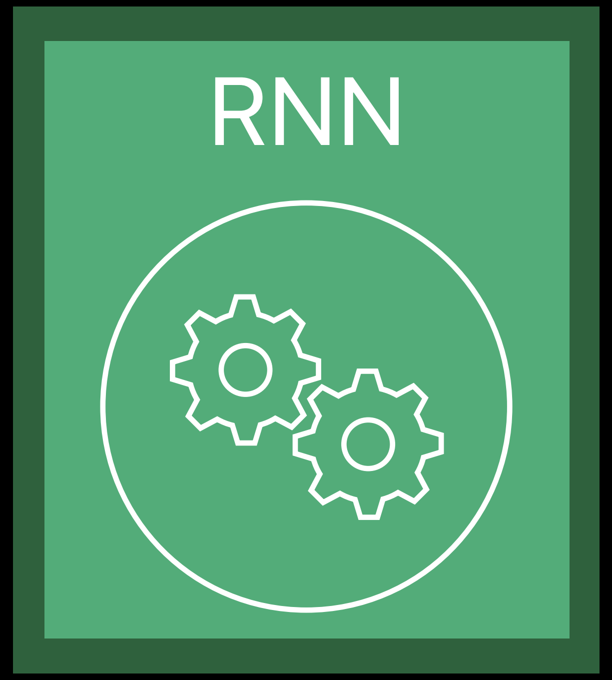
Sequence of outputs

Words / image caption

RNN

Sequences: many - to - many

A black and
white dog
laying in the
grass



Inference

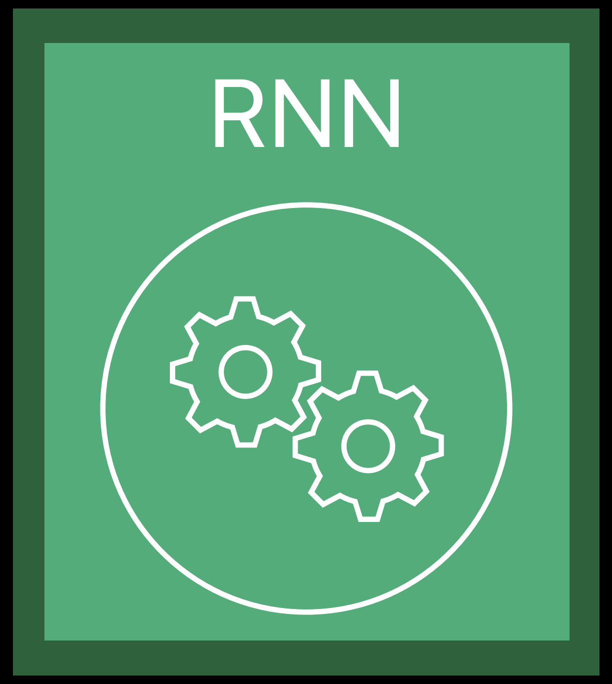
Sequence of inputs

Sentence in English

RNN

Sequences: many - to - many

A black and white dog laying in the grass



Inference

Чёрно-белая собака лежит на траве

Mustan ja valkoisen värinen koira makaa ruohikolla

Sequence of inputs

Sentence in English

Sequence of outputs

Translated sentence

Recurrent Neural Networks

New primitives

NEW

Single Gate

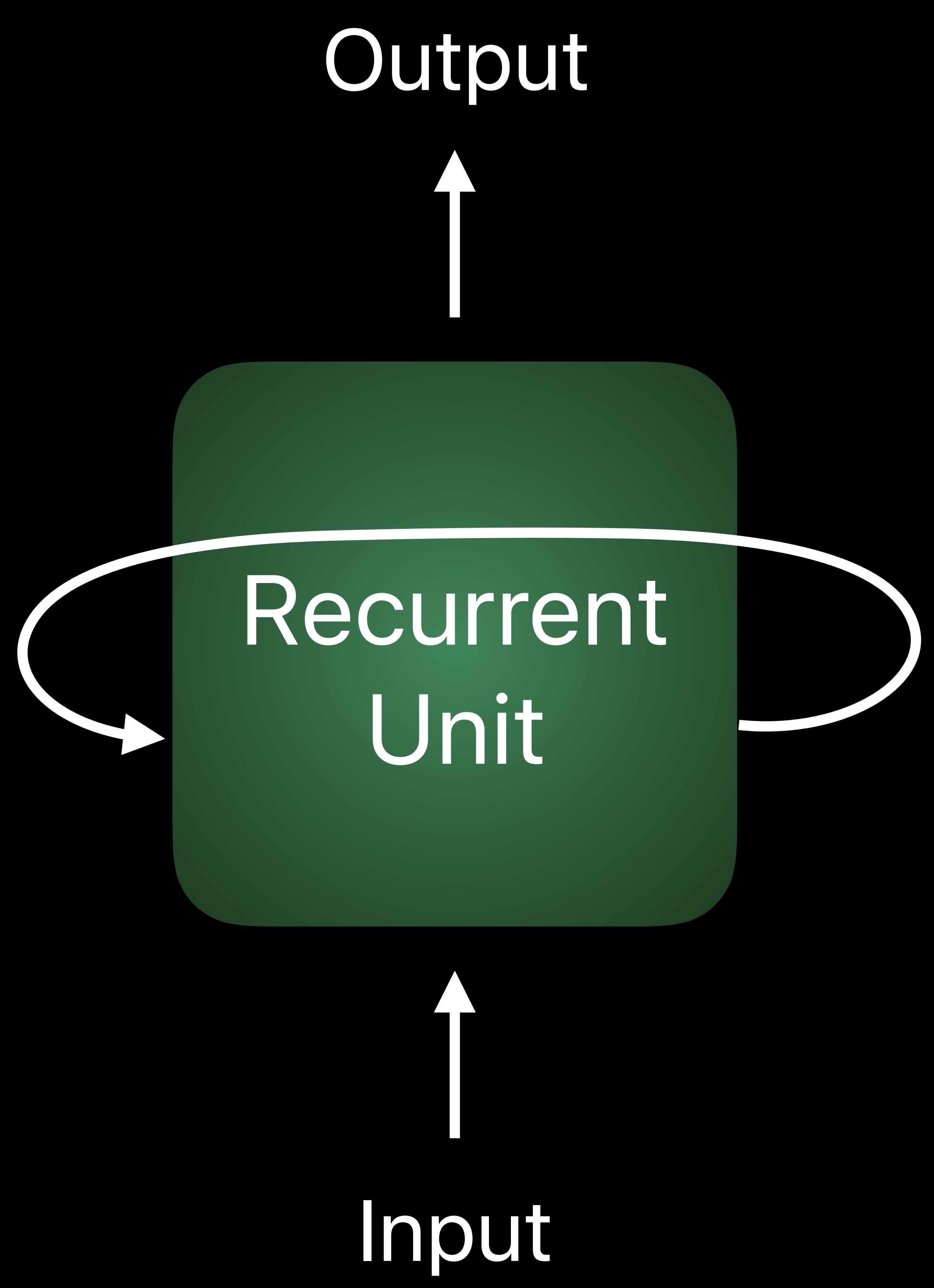
Long Short-Term Memory (LSTM)

Gated Recurrent Unit (GRU)

Minimally Gated Unit (MGU)

Single Gate RNN

Recurrent Unit enables previous output to affect the output of subsequent iterations

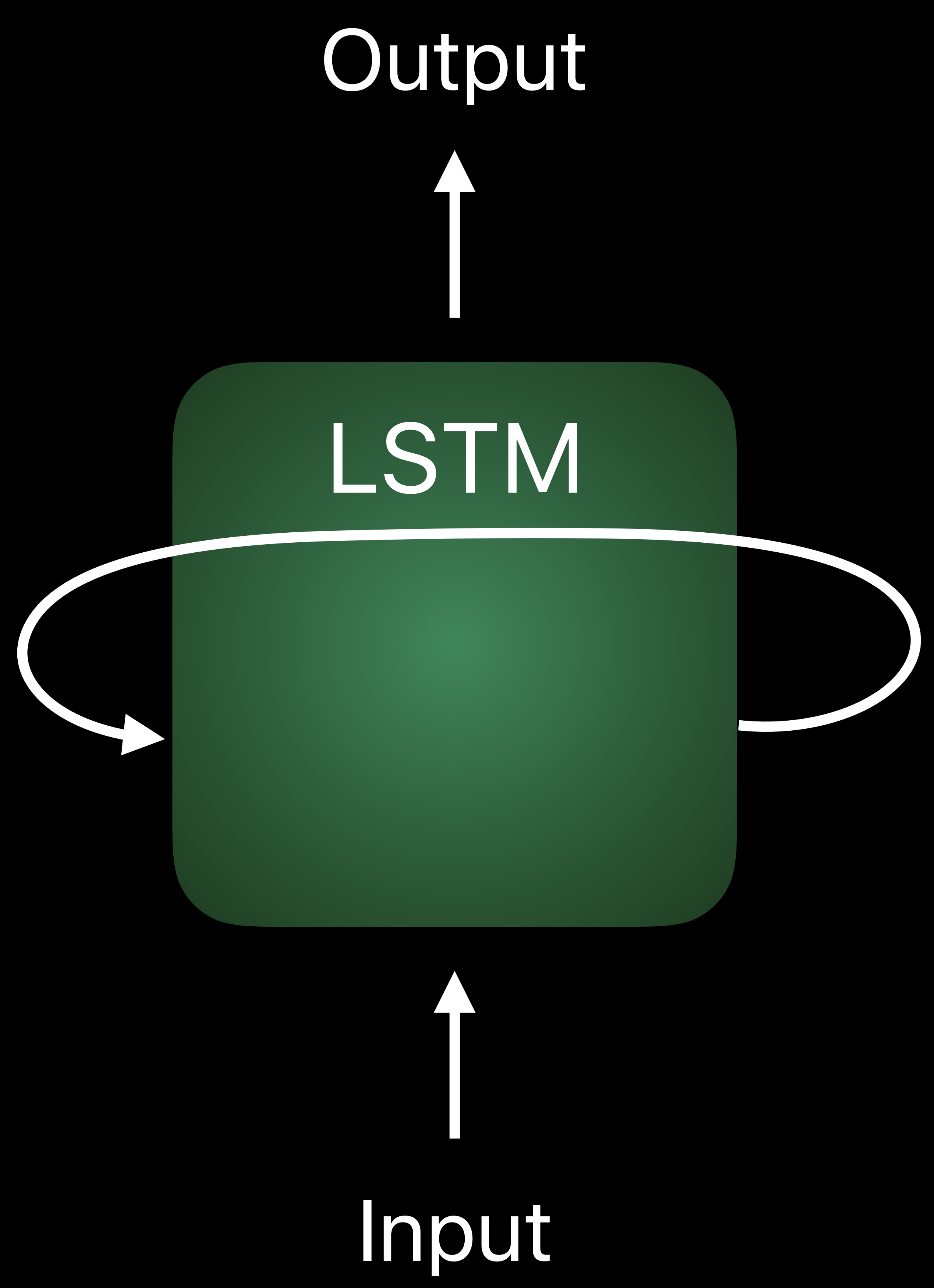


Long Short-Term Memory (LSTM)

Built from Single Gate RNNs

Has an internal Memory Cell

Gates control information flow inside the LSTM and what is stored in the Memory Cell

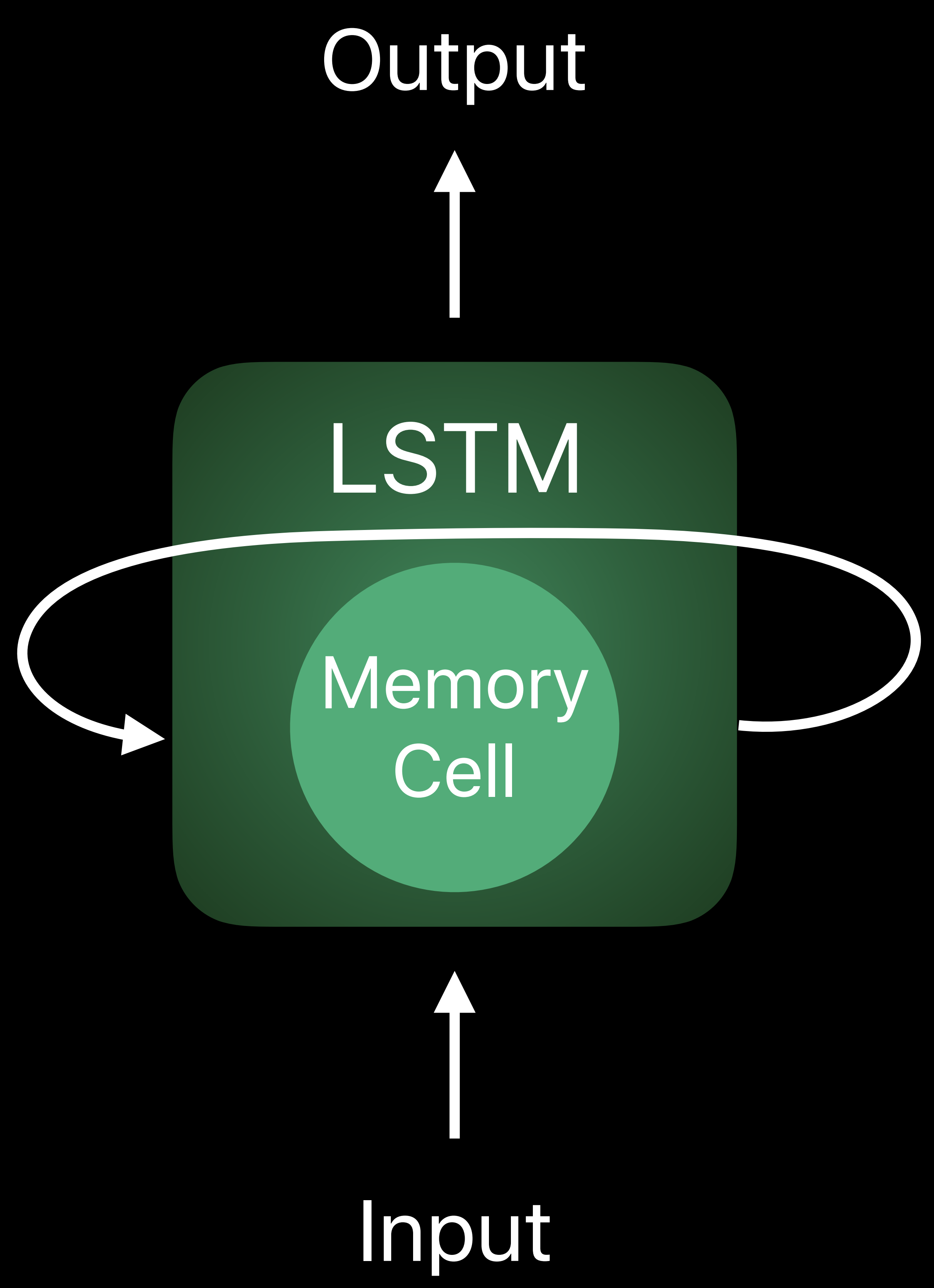


Long Short-Term Memory (LSTM)

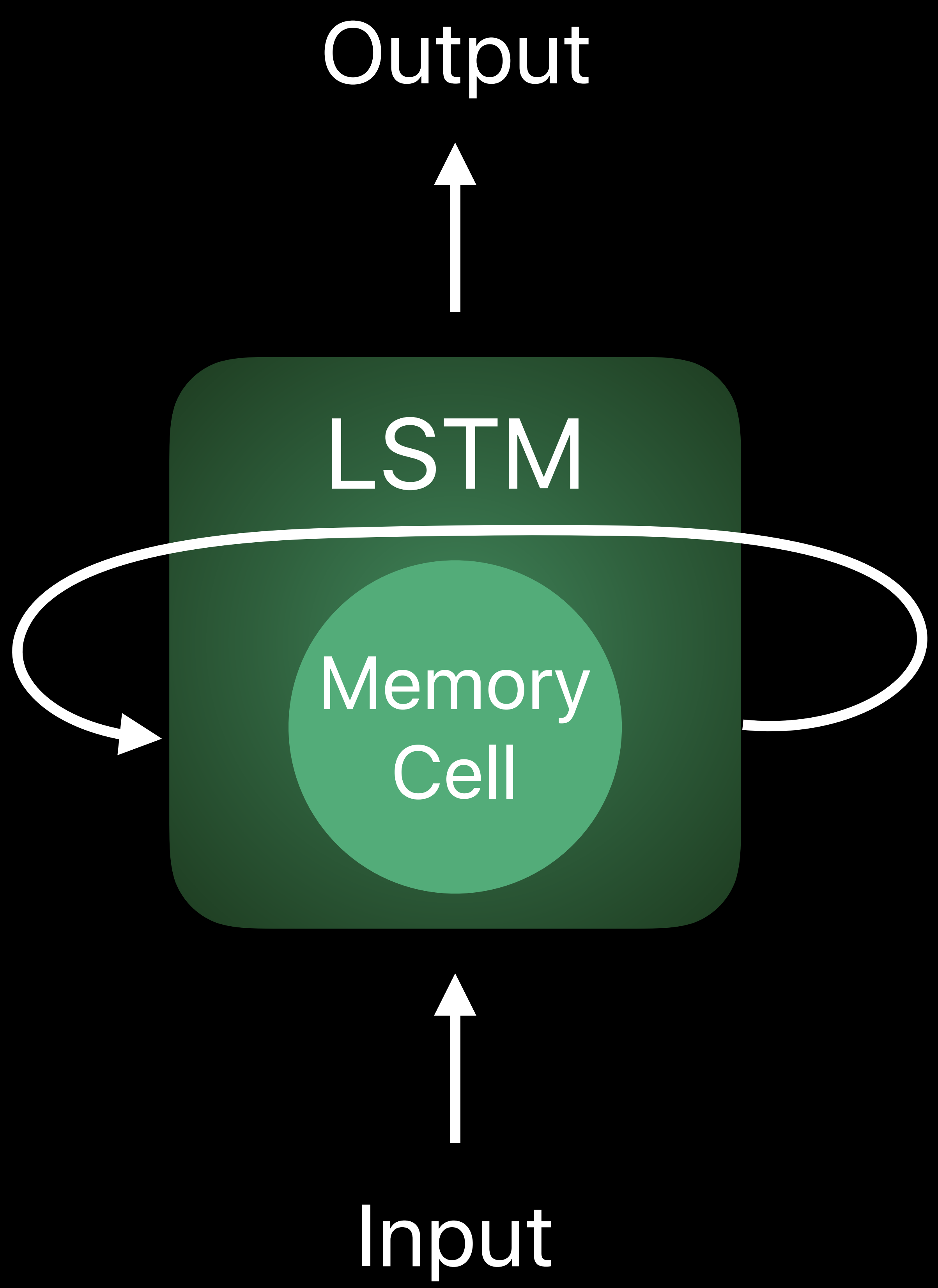
Built from Single Gate RNNs

Has an internal Memory Cell

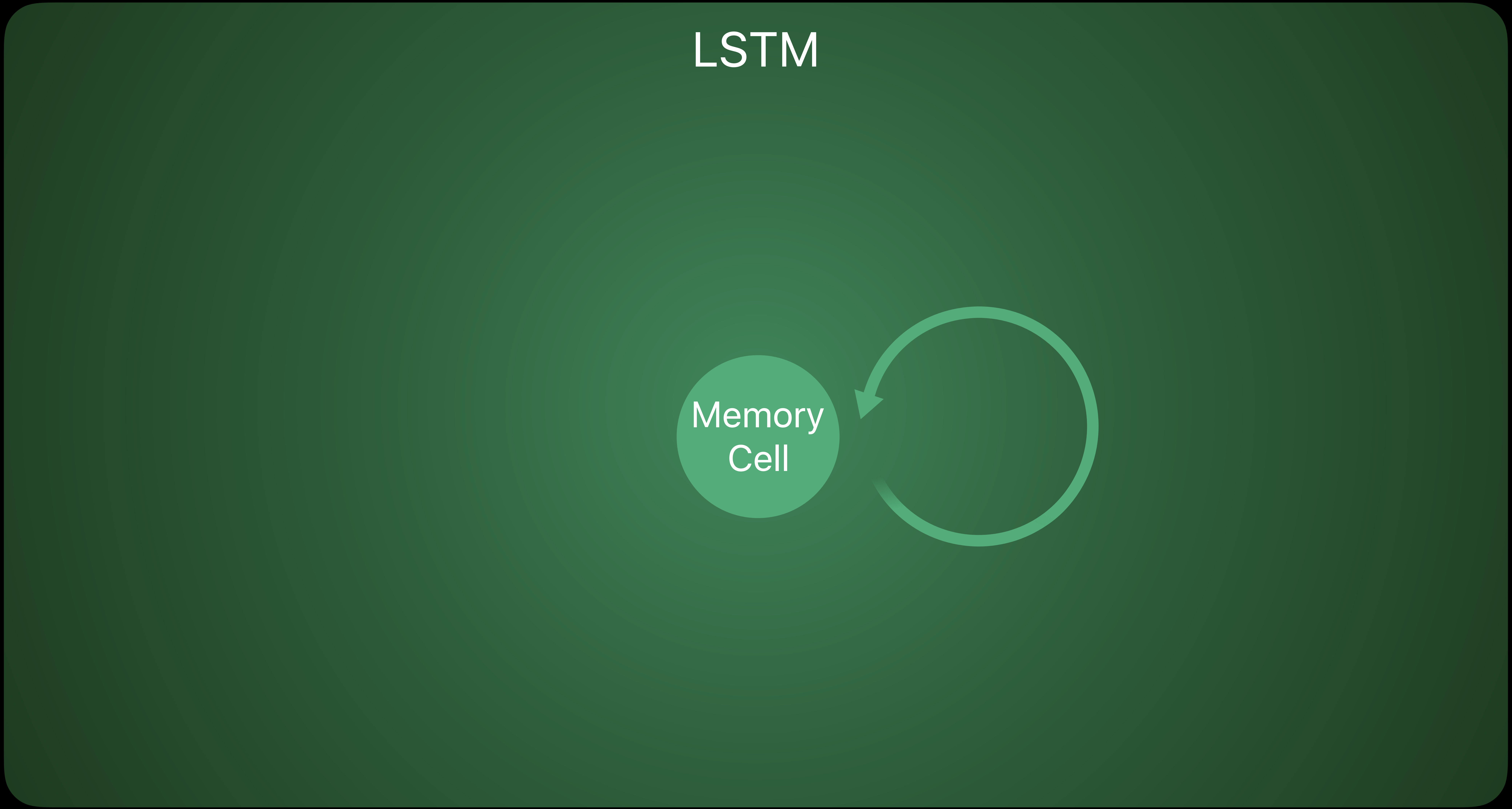
Gates control information flow inside the LSTM and what is stored in the Memory Cell



LSTM
Architecture



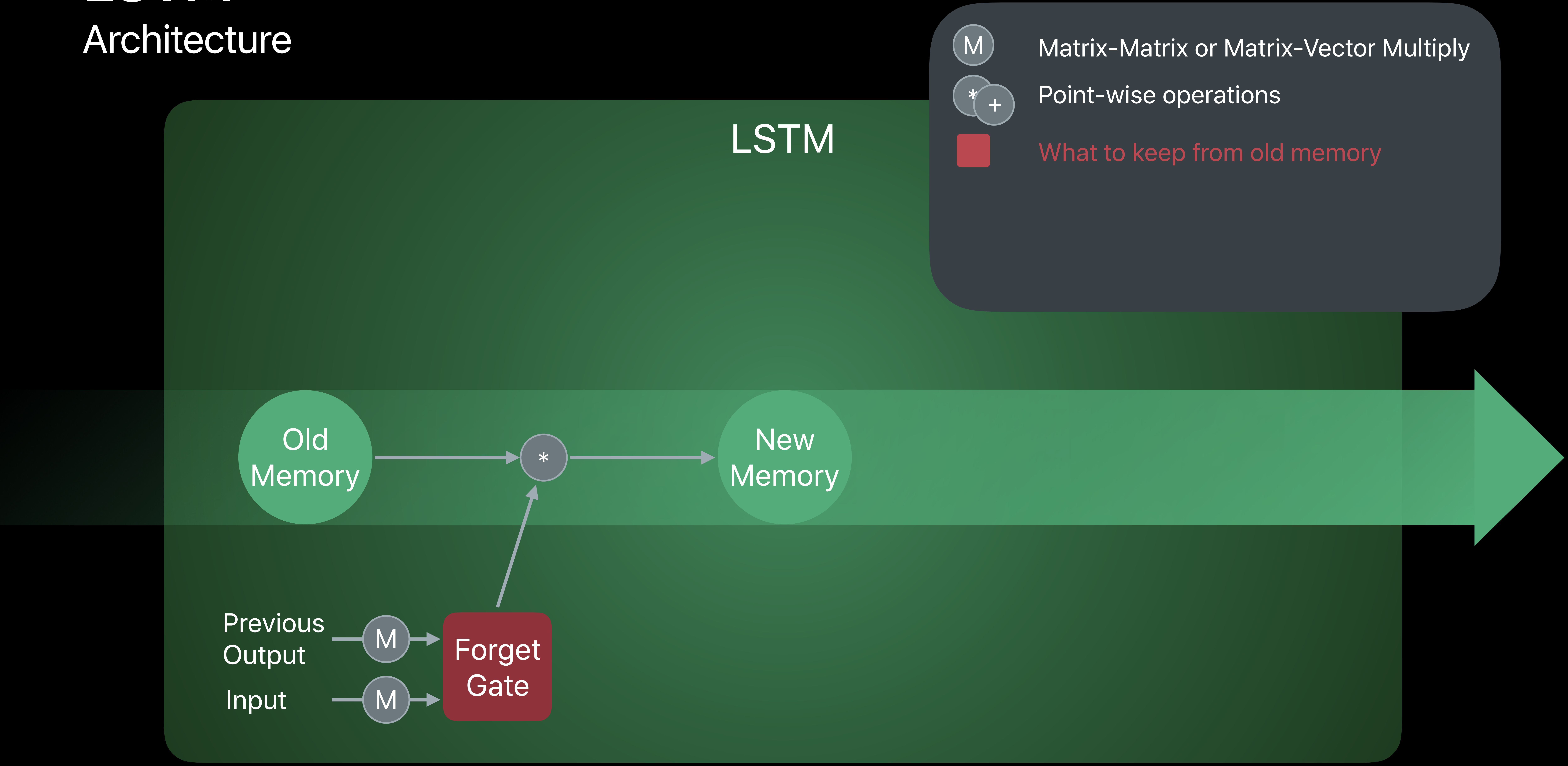
LSTM
Architecture



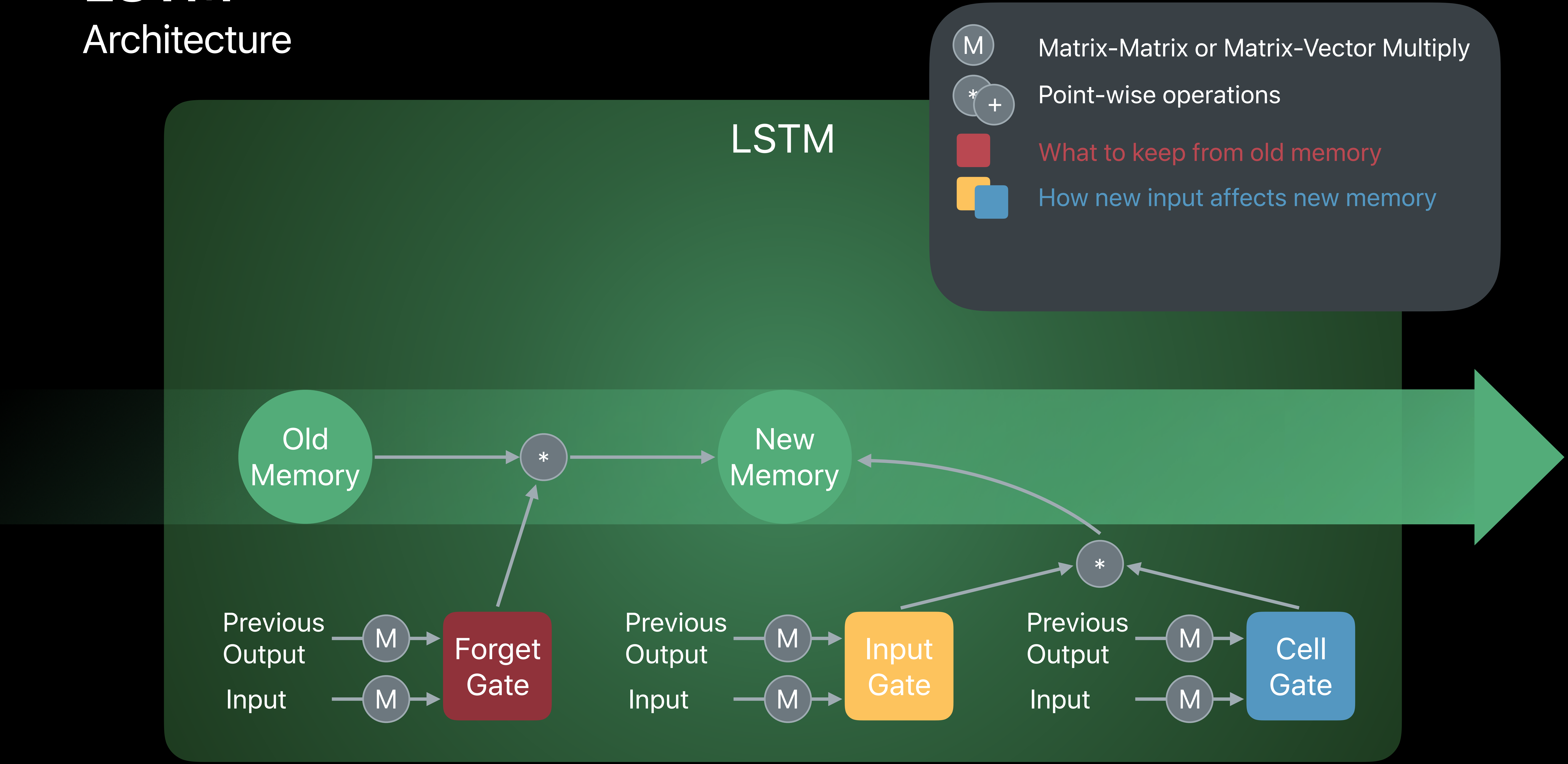
LSTM
Architecture

LSTM

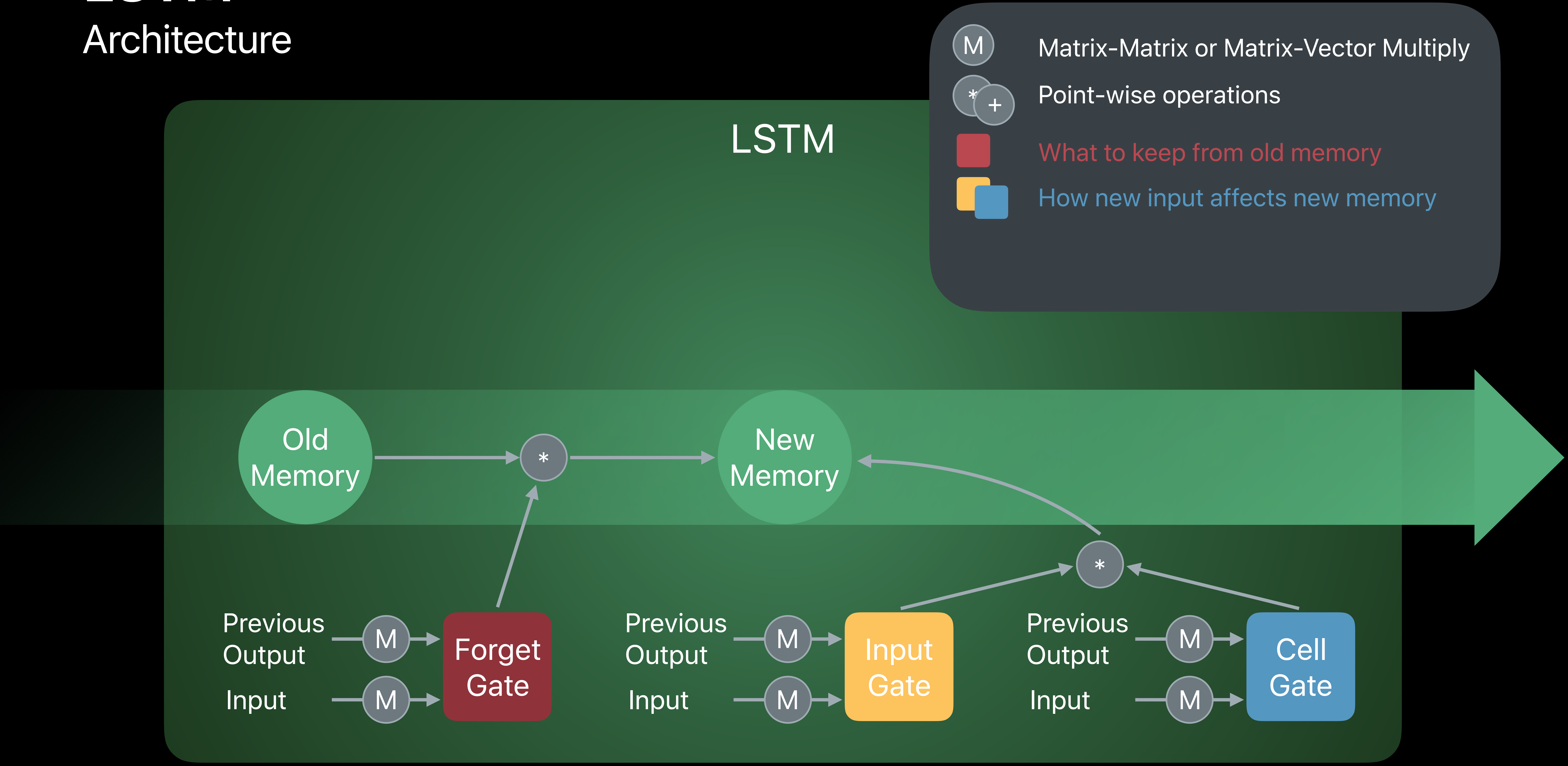
Architecture



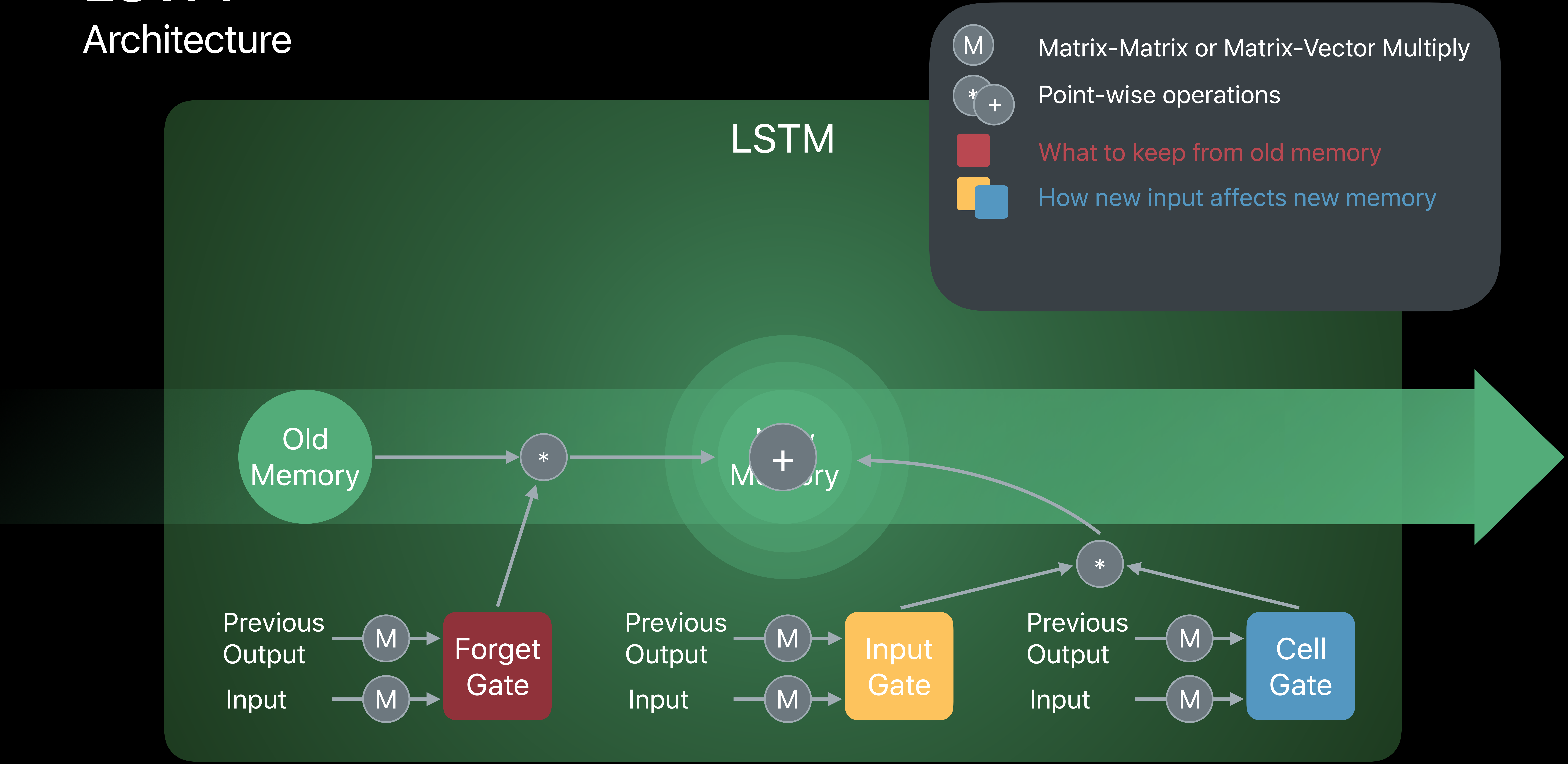
LSTM
Architecture



LSTM
Architecture



LSTM
Architecture



LSTM
Architecture

```swift
// Example: Creating a LSTM RNN

// Create a LSTM layer descriptor
let descriptor = MPSLSTMDescriptor()
descriptor.inputFeatureChannels = inputSize
descriptor.outputFeatureChannels = outputSize

// Create and initialize gate weights with trained parameters, using a data source provider
// for just-in-time loading and purging of weights
descriptor.forgetGateInputWeights = MyWeights(file:"forgetGateWeights.dat"))
descriptor.cellGateInputWeights = MyWeights(file:"cellGateWeights.dat"))
// Initialize the rest of the gates…

// Metal setup
let device = MTLCreateSystemDefaultDevice()! // Also get commandQueue and commandBuffer

// Create a LSTM layer
let layer = MPSRNNMatrixInferenceLayer(device: device, rnnDescriptor: descriptor)
```

```swift
// Example: Creating a LSTM RNN

// Create a LSTM layer descriptor
let descriptor = MPSLSTMDescriptor()
descriptor.inputFeatureChannels = inputSize
descriptor.outputFeatureChannels = outputSize

// Create and initialize gate weights with trained parameters, using a data source provider
// for just-in-time loading and purging of weights
descriptor.forgetGateInputWeights = MyWeights(file:"forgetGateWeights.dat"))
descriptor.cellGateInputWeights = MyWeights(file:"cellGateWeights.dat"))
// Initialize the rest of the gates…

// Metal setup
let device = MTLCreateSystemDefaultDevice()! // Also get commandQueue and commandBuffer

// Create a LSTM layer
let layer = MPSRNNMatrixInferenceLayer(device: device, rnnDescriptor: descriptor)
```

```swift
// Example: Creating a LSTM RNN

// Create a LSTM layer descriptor
let descriptor = MPSLSTMDescriptor()
descriptor.inputFeatureChannels = inputSize
descriptor.outputFeatureChannels = outputSize

// Create and initialize gate weights with trained parameters, using a data source provider
// for just-in-time loading and purging of weights
descriptor.forgetGateInputWeights = MyWeights(file:"forgetGateWeights.dat"))
descriptor.cellGateInputWeights = MyWeights(file:"cellGateWeights.dat"))
// Initialize the rest of the gates…

// Metal setup
let device = MTLCreateSystemDefaultDevice()! // Also get commandQueue and commandBuffer

// Create a LSTM layer
let layer = MPSRNNMatrixInferenceLayer(device: device, rnnDescriptor: descriptor)
```

```swift
// Example: Creating a LSTM RNN

// Create a LSTM layer descriptor
let descriptor = MPSLSTMDescriptor()
descriptor.inputFeatureChannels = inputSize
descriptor.outputFeatureChannels = outputSize

// Create and initialize gate weights with trained parameters, using a data source provider
// for just-in-time loading and purging of weights
descriptor.forgetGateInputWeights = MyWeights(file:"forgetGateWeights.dat"))
descriptor.cellGateInputWeights = MyWeights(file:"cellGateWeights.dat"))
// Initialize the rest of the gates…

// Metal setup
let device = MTLCreateSystemDefaultDevice()! // Also get commandQueue and commandBuffer

// Create a LSTM layer
let layer = MPSRNNMatrixInferenceLayer(device: device, rnnDescriptor: descriptor)
```

```swift
// Example: Running a LSTM RNN on the GPU

// Create input and output data
var inputSequence: [MPSMatrix] = []
var outputSequence: [MPSMatrix] = []
for i in 0..< N {
    // Matrix size is (1, inputSize), inputSize is number of columns
    inputSequence.append(MPSMatrix(…))
    // Matrix size is (1, outputSize), outputSize is number of columns
    outputSequence.append(MPSMatrix(…))
}

// Submit work to GPU
layer.encodeSequence(commandBuffer: commandBuffer,
                     sourceMatrices: inputSequence, destinationMatrices: outputSequence,
                     recurrentInputState: nil, recurrentOutputStates: nil)
// Tell GPU to start executing work
commandBuffer.commit()
```

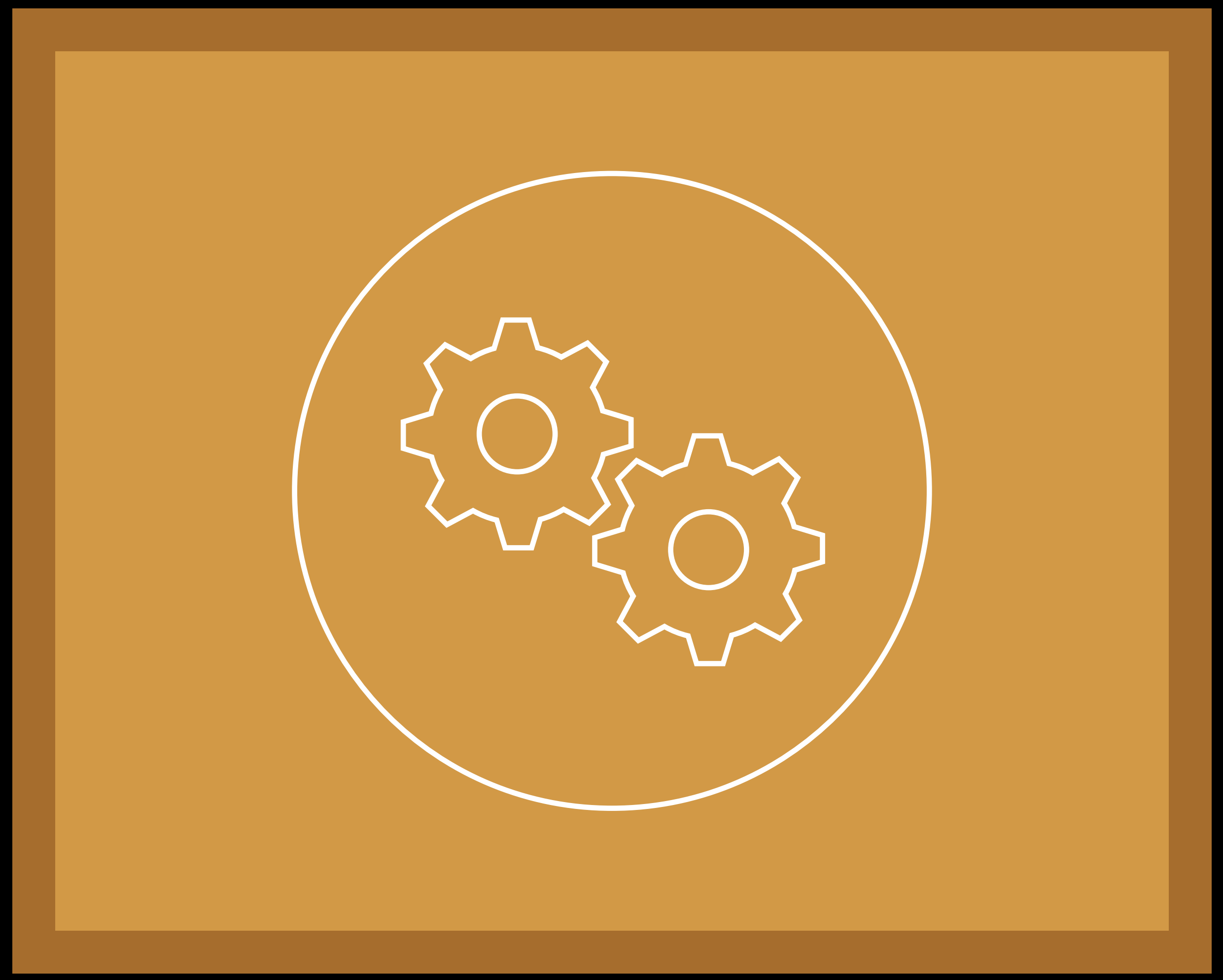
```swift
// Example: Running a LSTM RNN on the GPU

// Create input and output data
var inputSequence: [MPSMatrix] = []
var outputSequence: [MPSMatrix] = []
for i in 0..< N {
   // Matrix size is (1, inputSize), inputSize is number of columns
   inputSequence.append(MPSMatrix(…))
   // Matrix size is (1, outputSize), outputSize is number of columns
   outputSequence.append(MPSMatrix(…))
}

// Submit work to GPU
layer.encodeSequence(commandBuffer: commandBuffer,
                     sourceMatrices: inputSequence, destinationMatrices: outputSequence,
                     recurrentInputState: nil, recurrentOutputStates: nil)
// Tell GPU to start executing work
commandBuffer.commit()
```

```swift
// Example: Running a LSTM RNN on the GPU

// Create input and output data
var inputSequence: [MPSMatrix] = []
var outputSequence: [MPSMatrix] = []
for i in 0..< N {
   // Matrix size is (1, inputSize), inputSize is number of columns
   inputSequence.append(MPSMatrix(…))
   // Matrix size is (1, outputSize), outputSize is number of columns
   outputSequence.append(MPSMatrix(…))
}
```

```swift
// Submit work to GPU
layer.encodeSequence(commandBuffer: commandBuffer,
                     sourceMatrices: inputSequence, destinationMatrices: outputSequence,
                     recurrentInputState: nil, recurrentOutputStates: nil)
// Tell GPU to start executing work
commandBuffer.commit()
```

Example: Image Captioning

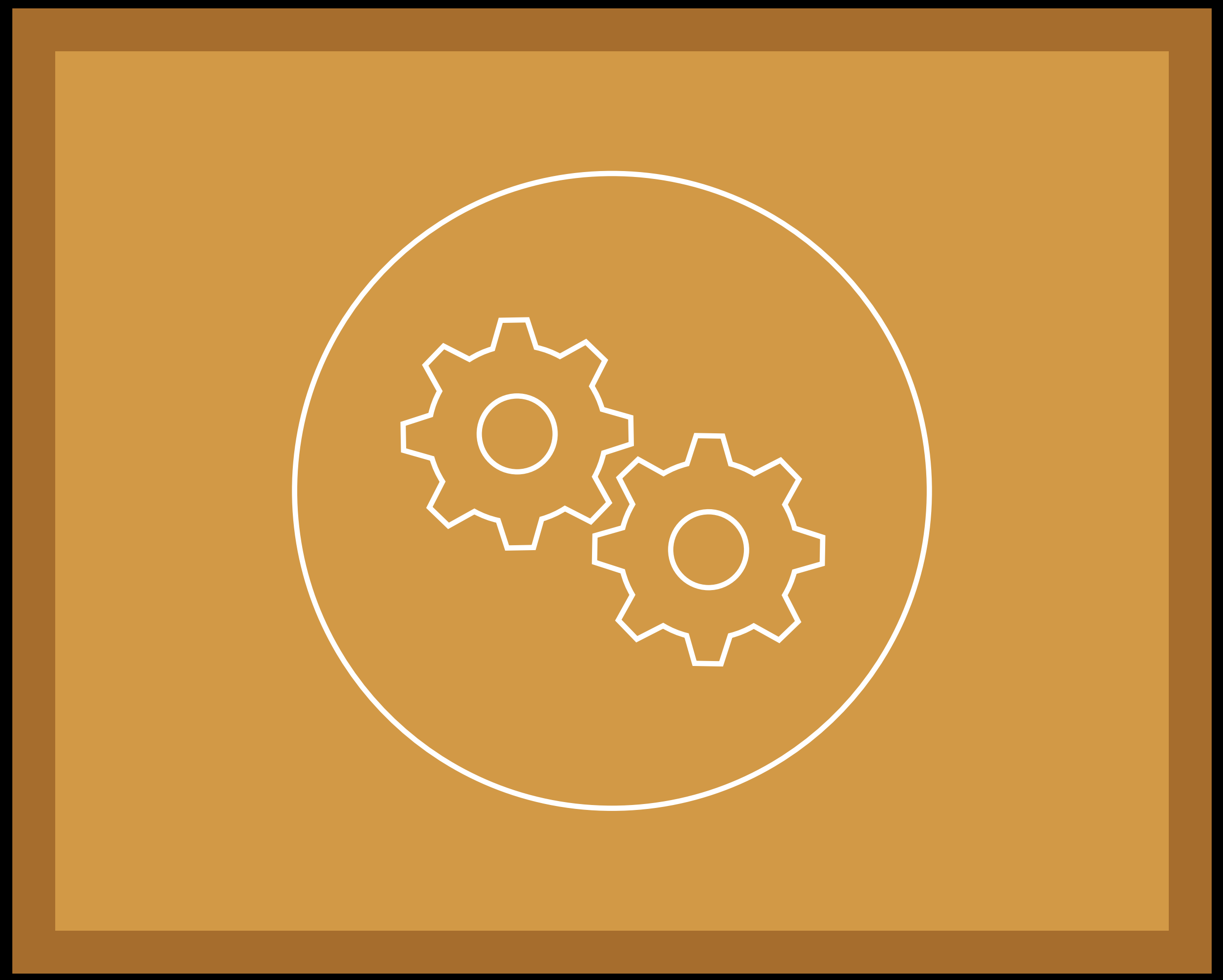
Training



Training to Caption Images

Example: Image Captioning

Training

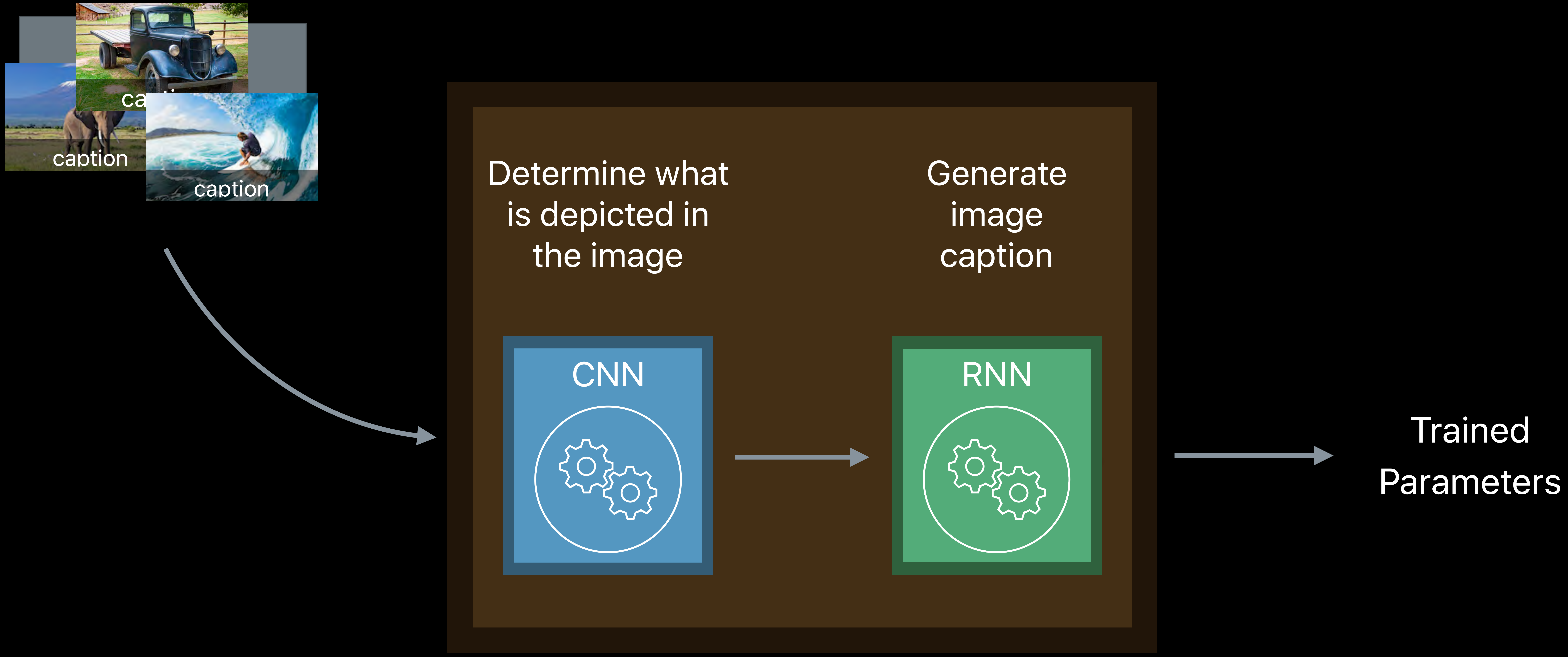


Training to Caption Images

Trained Parameters

Example: Image Captioning

Training



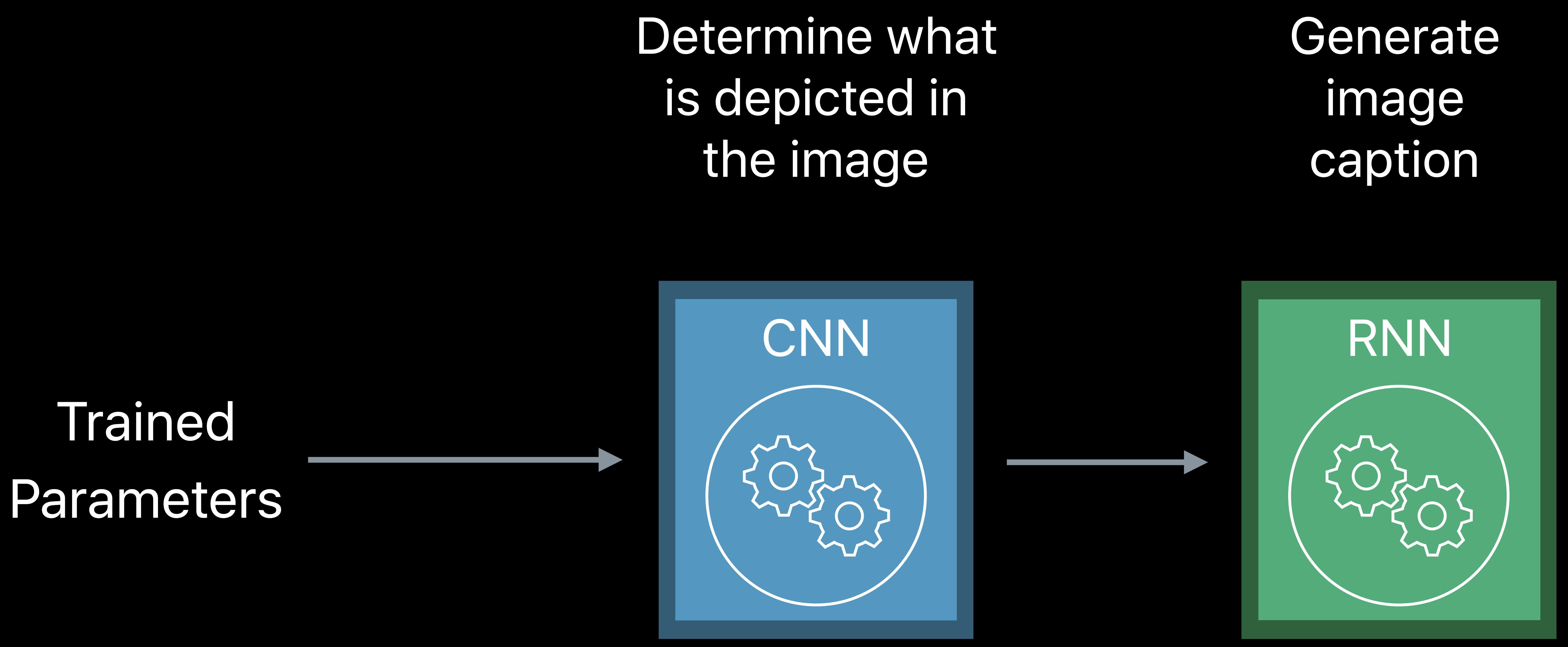
Example: Image Captioning

Inference

Trained
Parameters

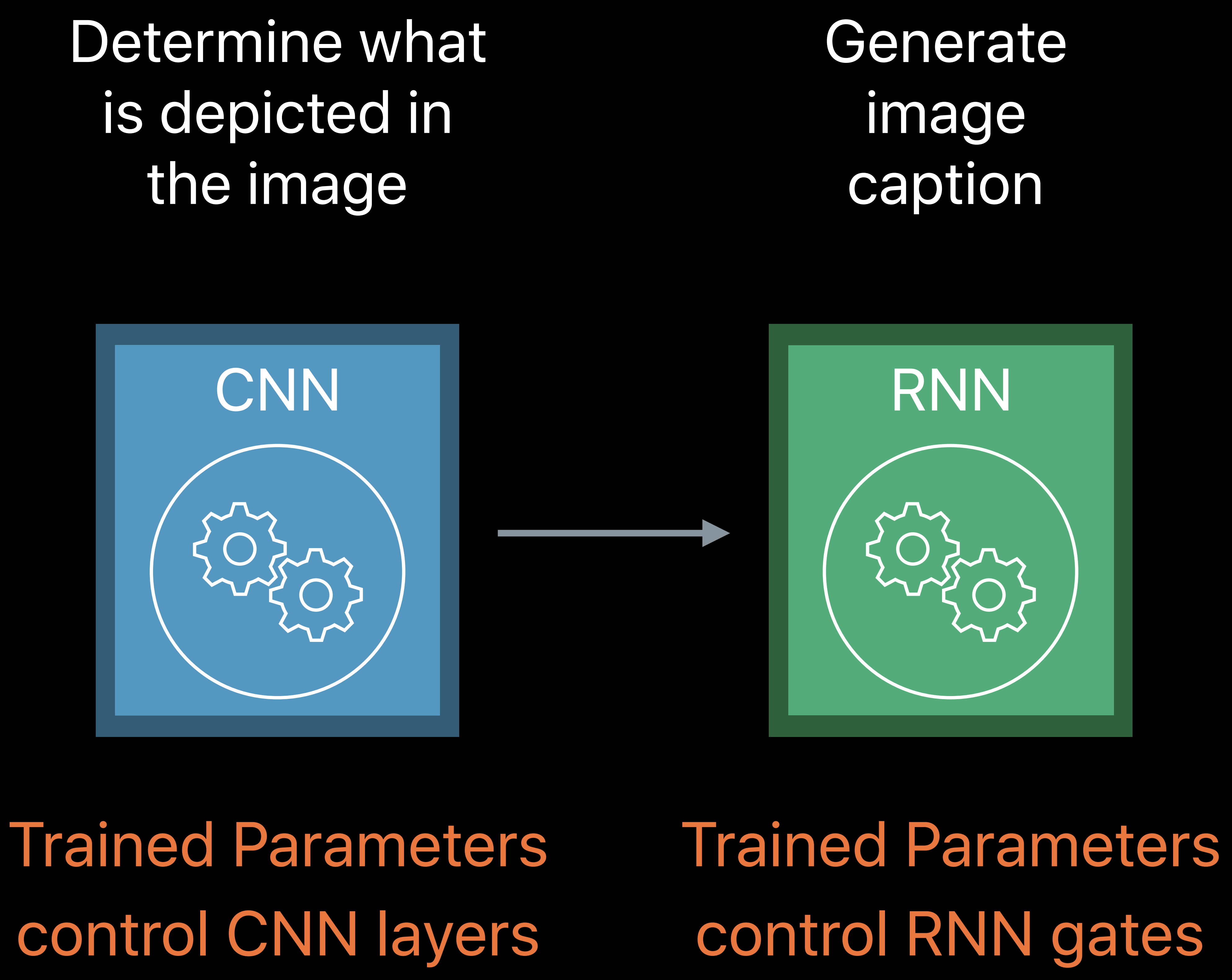
Example: Image Captioning

Inference



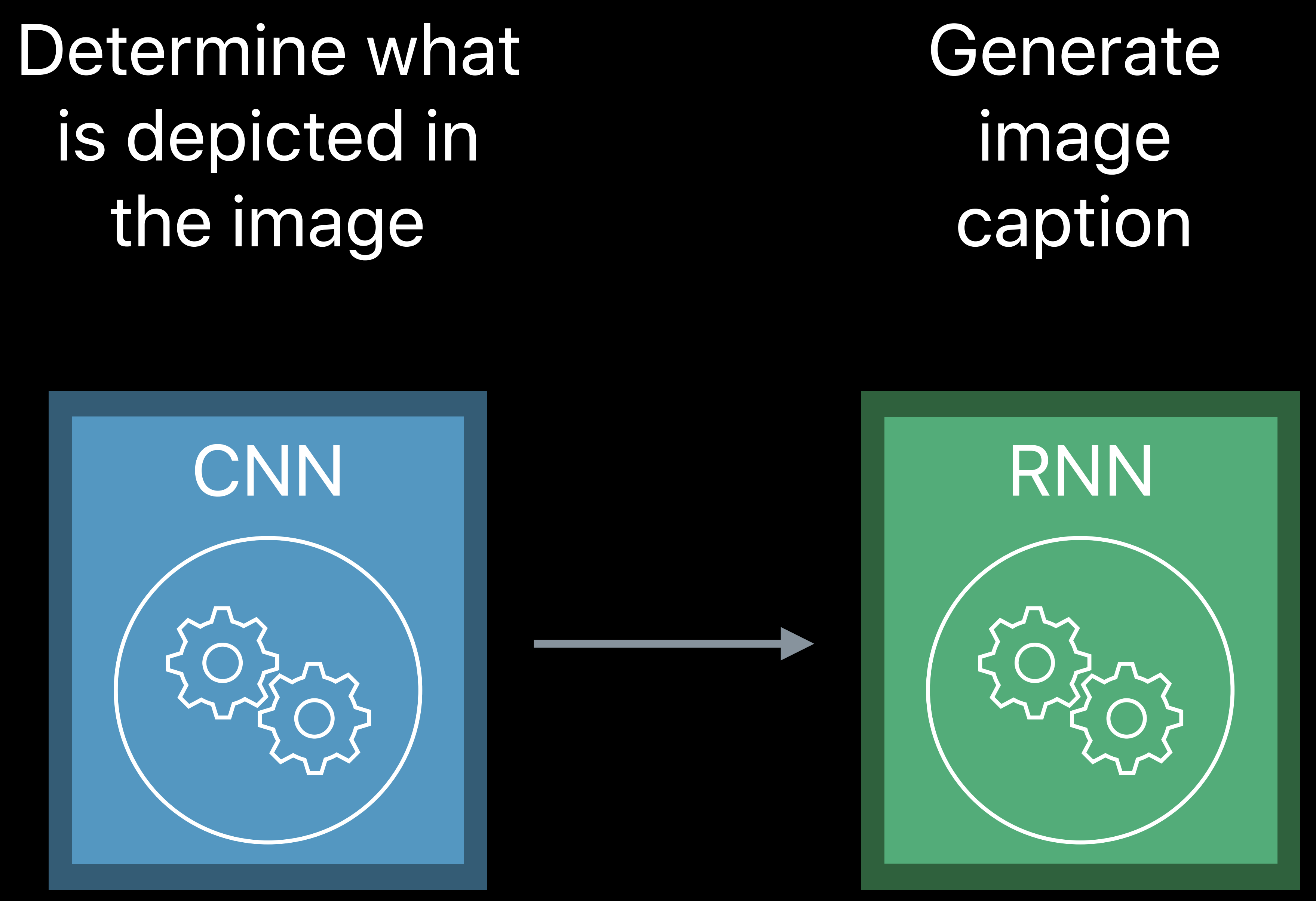
Example: Image Captioning

Inference



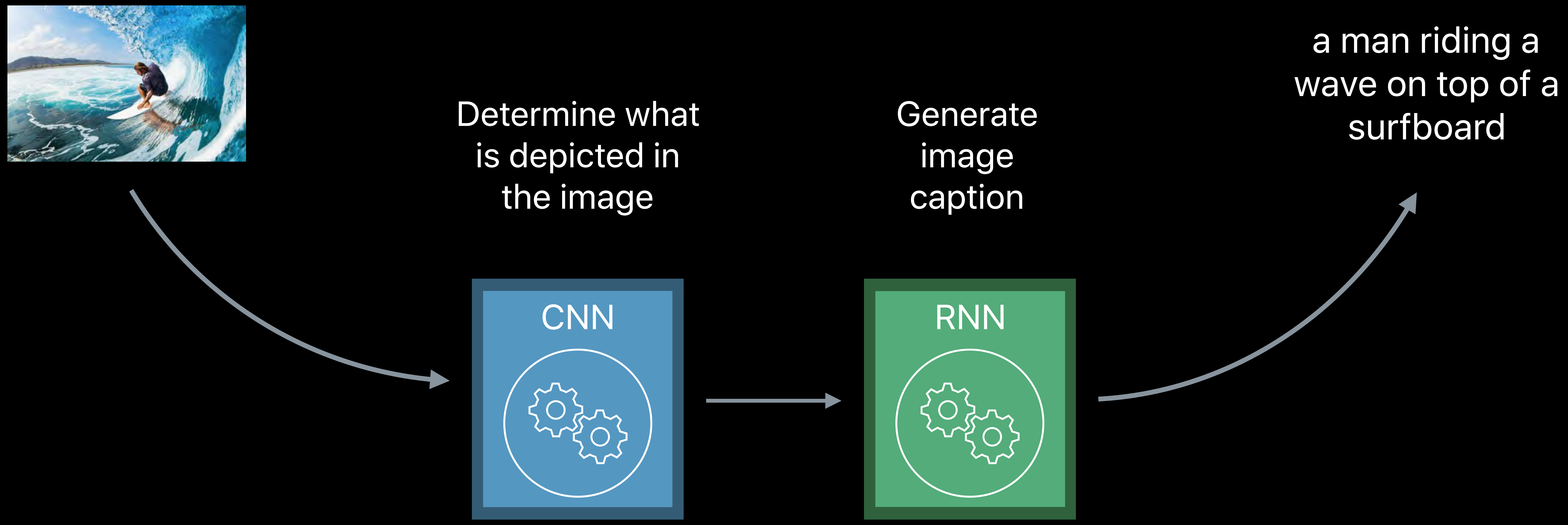
Example: Image Captioning

Inference



Example: Image Captioning

Inference



Example: Image Captioning

Inference

Determine what is depicted in the image

Generate image caption

a man riding a wave on top of a surfboard

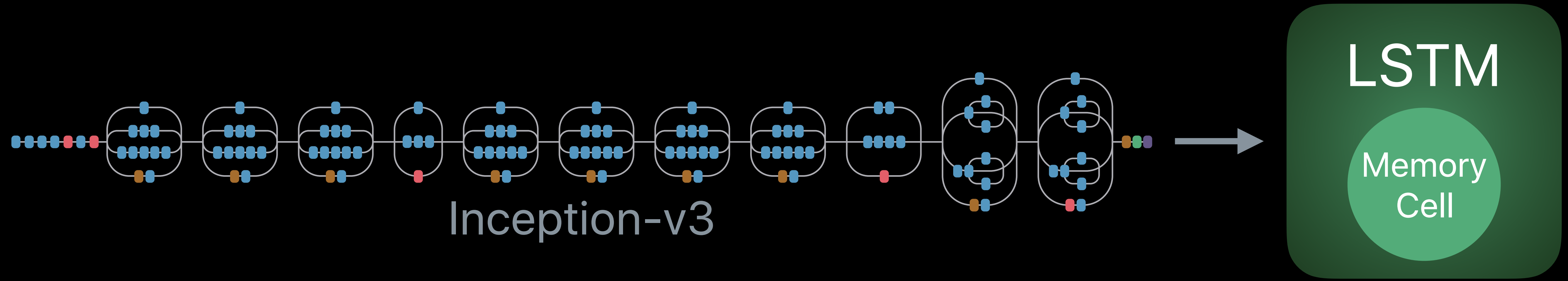
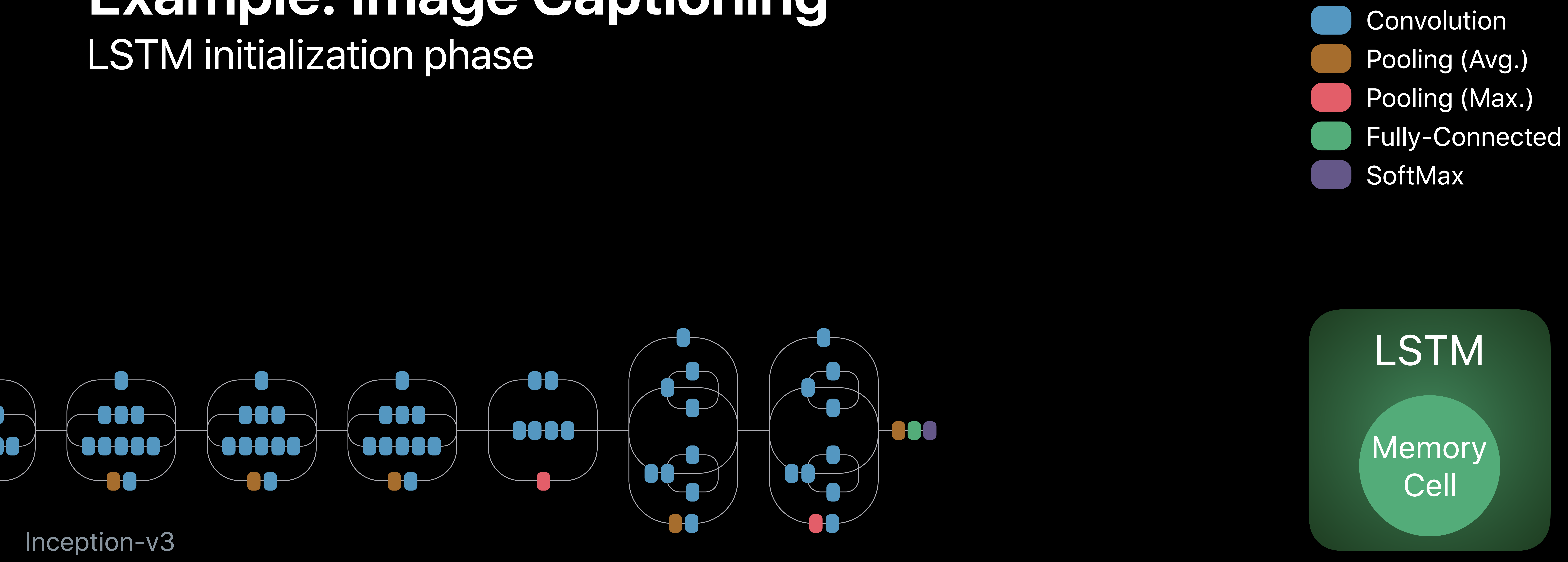


Image Captioning Network*

*Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge, Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan, IEEE Transactions on Pattern Analysis and Machine Intelligence, https://arxiv.org/abs/1609.06647

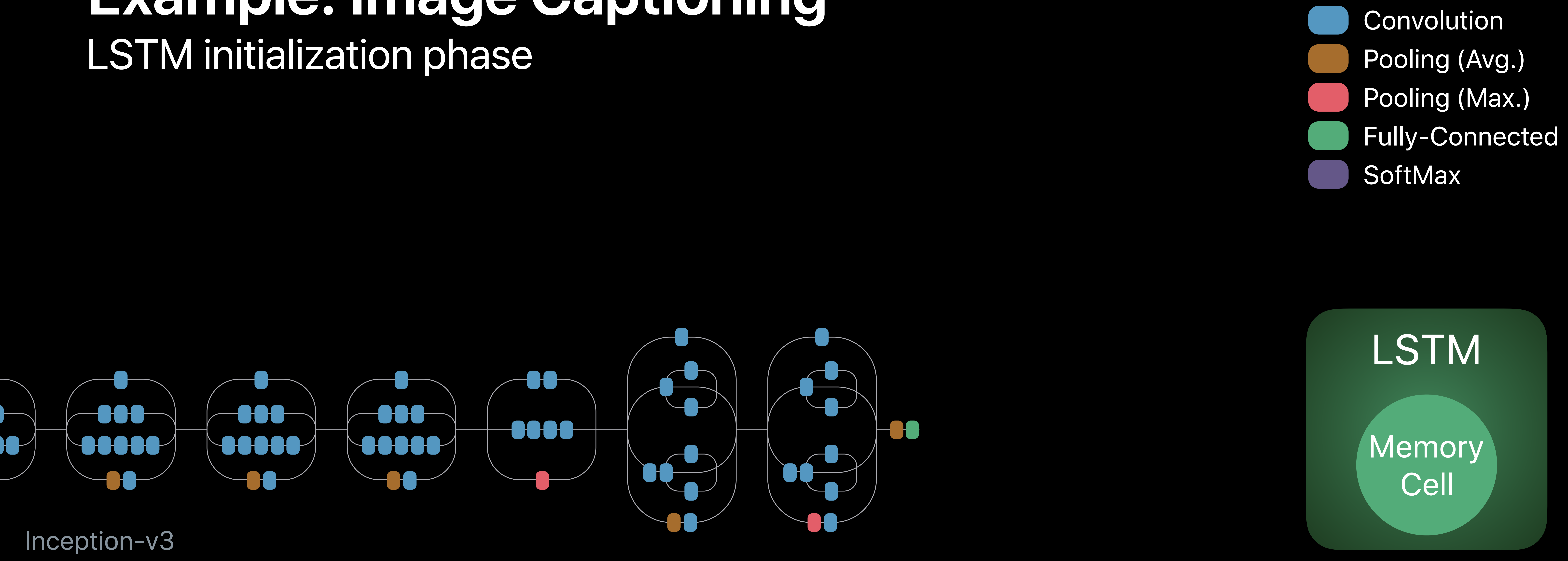
Example: Image Captioning

LSTM initialization phase



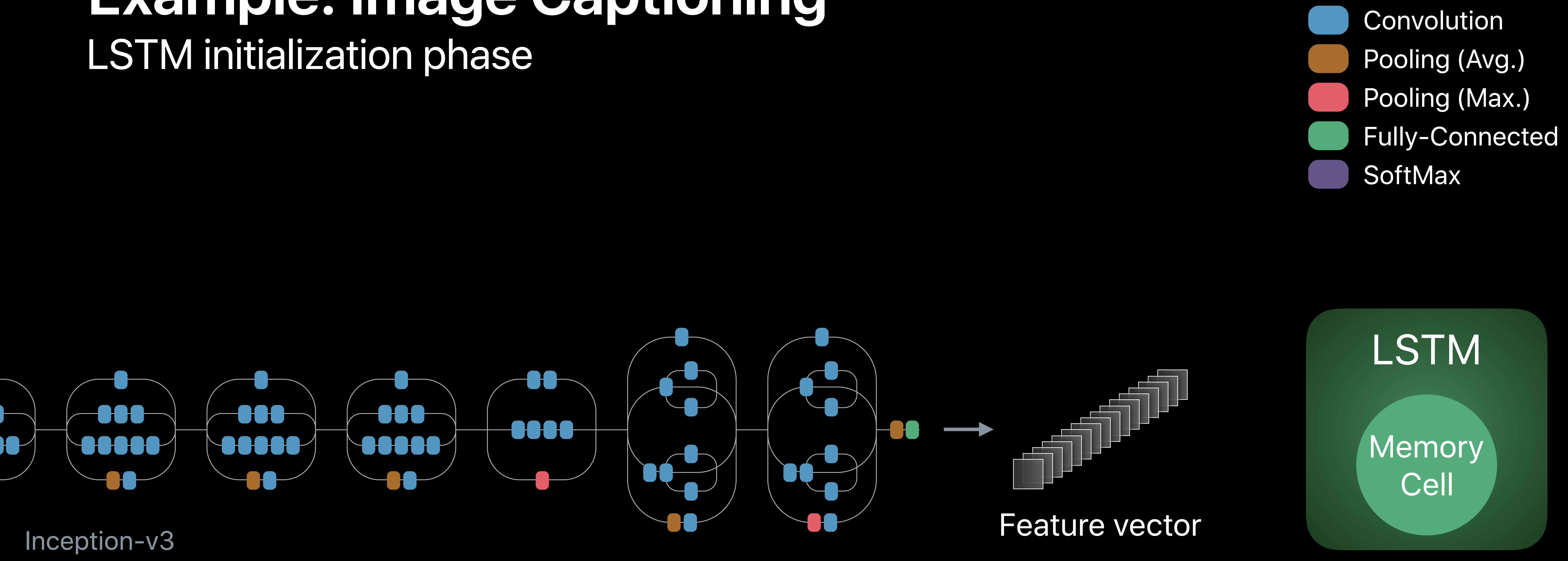
Example: Image Captioning

LSTM initialization phase



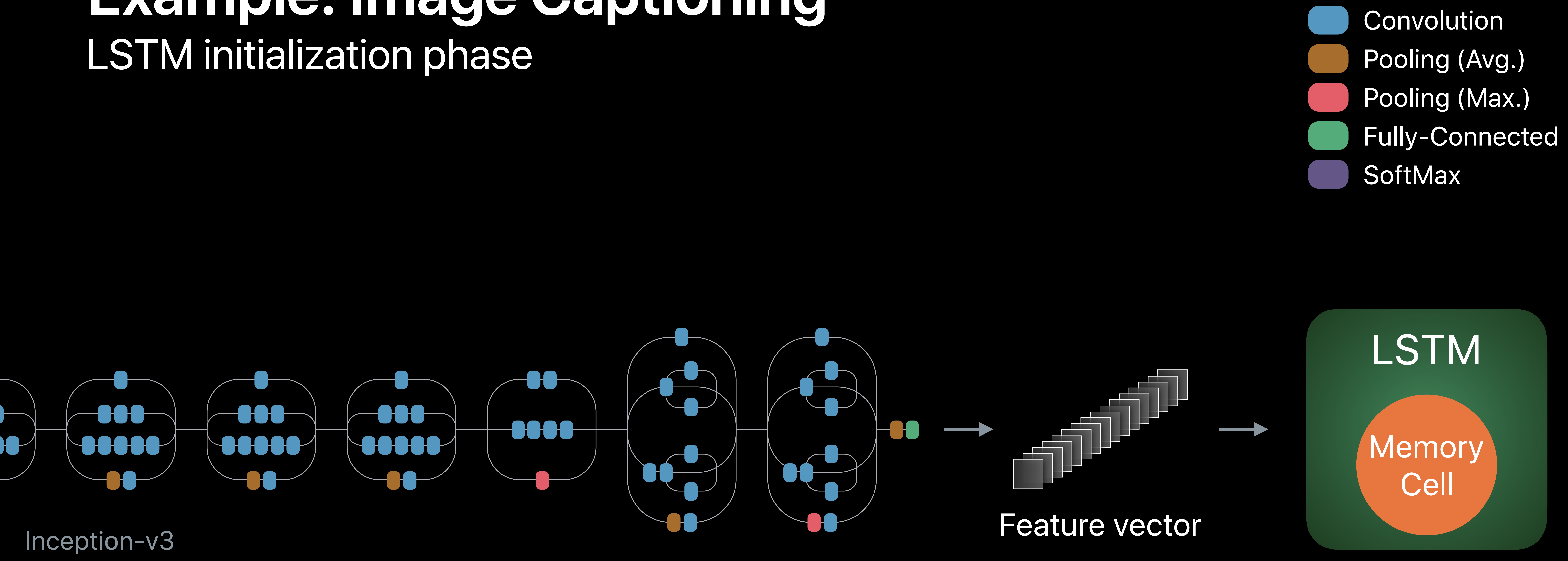
Example: Image Captioning

LSTM initialization phase



Example: Image Captioning

LSTM initialization phase



Example: Image Captioning

Caption generation phase

- Convolution
- Pooling (Avg.)
- Pooling (Max.)
- Fully-Connected
- SoftMax

Input

Sentence start token

Output

Example: Image Captioning

Caption generation phase

Example: Image Captioning

Caption generation phase

Example: Image Captioning

Caption generation phase

Example: Image Captioning
Caption generation phase

Caption Generation

Top three captions:

1.
2.
3.

Iteration 1

Caption	Probability
man	0.021986
a	0.862899
the	0.039906

Iteration 2

Caption	Probability

Caption Generation

Top three captions:
1.
2.
3.

Iteration 1

Caption	Probability
man	0.021986
a	0.862899
the	0.039906

Iteration 2

Caption	Probability

Caption Generation

Top three captions:

1.
2.
3.

Iteration 1

Caption	Probability
man	0.021986
a	0.862899
the	0.039906

Iteration 2

Caption	Probability
man on	0.005290
man in	0.004869
man surfing	0.003914

Caption Generation

Top three captions:

1.
2.
3.

Iteration 1

Caption	Probability
man	0.021986
a	0.862899
the	0.039906

Iteration 2

Caption	Probability
man on	0.005290
man in	0.004869
man surfing	0.003914
a man	0.385814
a person	0.136590
a surfer	0.116651

Caption Generation

Top three captions:

1.
2.
3.

Iteration 1

Caption	Probability
man	0.021986
a	0.862899
the	0.039906

Iteration 2

Caption	Probability
man on	0.005290
man in	0.004869
man surfing	0.003914
a man	0.385814
a person	0.136590
a surfer	0.116651
the man	0.014275
the surfer	0.012315
the young	0.003500

Caption Generation

Top three captions:

1.
2.
3.

Iteration 1

Caption	Probability
man	0.021986
a	0.862899
the	0.039906

Iteration 2

Caption	Probability
man on	0.005290
man in	0.004869
man surfing	0.003914
a man	0.385814
a person	0.136590
a surfer	0.116651
the man	0.014275
the surfer	0.012315
the young	0.003500

Caption Generation

Top three captions:
1.
2.
3.

Iteration 2

Caption	Probability
man on	0.005290
man in	0.004869
man surfing	0.003914
a man	0.385814
a person	0.136590
a surfer	0.116651
the man	0.014275
the surfer	0.012315
the young	0.003500

Iteration 3

Caption	Probability
a man riding	0.115423
a man on	0.060142
a man is	0.048678
a person riding	0.041114
a person on	0.031153
a person in	0.014218
a surfer is	0.027462
a surfer riding	0.016631
a surfer in	0.015737

Caption Generation

Top three captions:

1.
2.
3.

Iteration 2

Caption	Probability
man on	0.005290
man in	0.004869
man surfing	0.003914
a man	0.385814
a person	0.136590
a surfer	0.116651
the man	0.014275
the surfer	0.012315
the young	0.003500

Iteration 3

Caption	Probability
a man riding	0.115423
a man on	0.060142
a man is	0.048678
a person riding	0.041114
a person on	0.031153
a person in	0.014218
a surfer is	0.027462
a surfer riding	0.016631
a surfer in	0.015737

Caption Generation

Top three captions:

1.
2.
3.

Iteration 3

Caption	Probability
a man riding	0.115423
a man on	0.060142
a man is	0.048678
a person riding	0.041114
a person on	0.031153
a person in	0.014218
a surfer is	0.027462
a surfer riding	0.016631
a surfer in	0.015737

Iteration 4

Caption	Probability
a man riding a	0.100079
a man riding on	0.008264
a man riding the	0.002604
a man on a	0.055997
a man on his	0.001288
a man on the	0.001211
a man is surfing	0.021393
a man is riding	0.015222
a man is on	0.003434

Caption Generation

Top three captions:

1.
2.
3.

Iteration 3

Caption	Probability
a man riding	0.115423
a man on	0.060142
a man is	0.048678
a person riding	0.041114
a person on	0.031153
a person in	0.014218
a surfer is	0.027462
a surfer riding	0.016631
a surfer in	0.015737

Iteration 4

Caption	Probability
a man riding a	0.100079
a man riding on	0.008264
a man riding the	0.002604
a man on a	0.055997
a man on his	0.001288
a man on the	0.001211
a man is surfing	0.021393
a man is riding	0.015222
a man is on	0.003434

Caption Generation

Top three captions:

1. a man riding a wave on top of a surfboard
2. a man on a surfboard riding a wave
3. a man riding a wave on a surfboard

Caption Generation

Top three captions:

1. a man riding a wave on top of a surfboard
2. a man on a surfboard riding a wave
3. a man riding a wave on a surfboard

Demo

Image captioning — CNN + LSTM

Summary

GPU accelerated primitives

- Expanded support for Image Processing and Convolutional Neural Networks
- Added support for Linear Algebra and Recurrent Neural Networks

Optimized for iOS and macOS

New Neural Network Graph API

Related Sessions

Introducing Metal 2	Executive Ballroom	Tuesday 1:50PM
Introducing Core ML	Hall 3	Tuesday 3:10PM
VR with Metal 2	Hall 3	Wednesday 10:00AM
Vision Framework: Building on Core ML	Hall 2	Wednesday 3:10PM
Core ML in depth	Hall 3	Thursday 09:00AM
Accelerate and Sparse Solvers	Executive Ballroom	Thursday 10:00AM
Metal 2 Optimization and Debugging	Grand Ballroom B	Thursday 3:10PM

Labs

Metal 2 Lab	Technology Lab	Friday 09:00AM–12:00PM

More Information

https://developer.apple.com/wwdc17/608

WWDC17