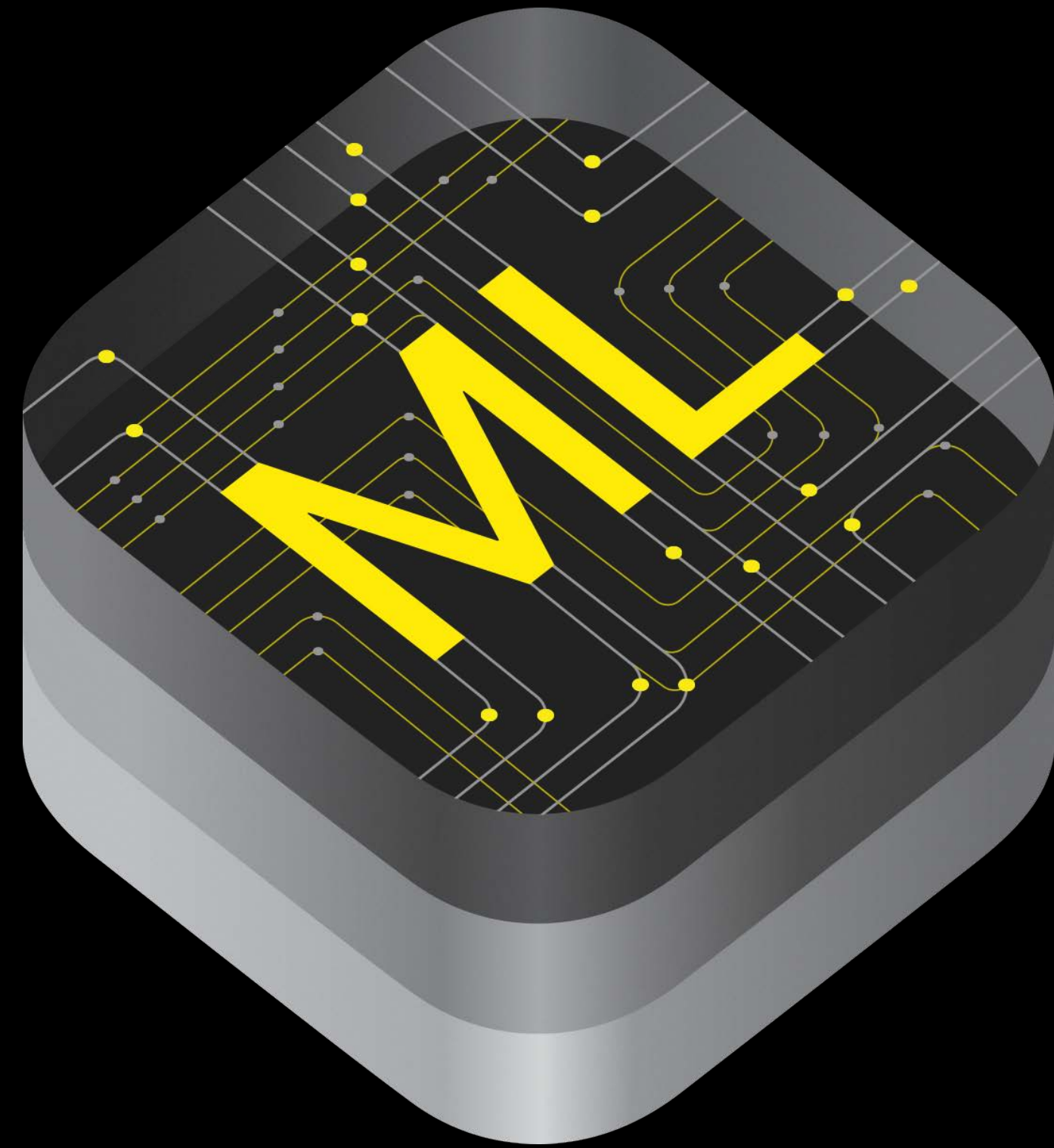


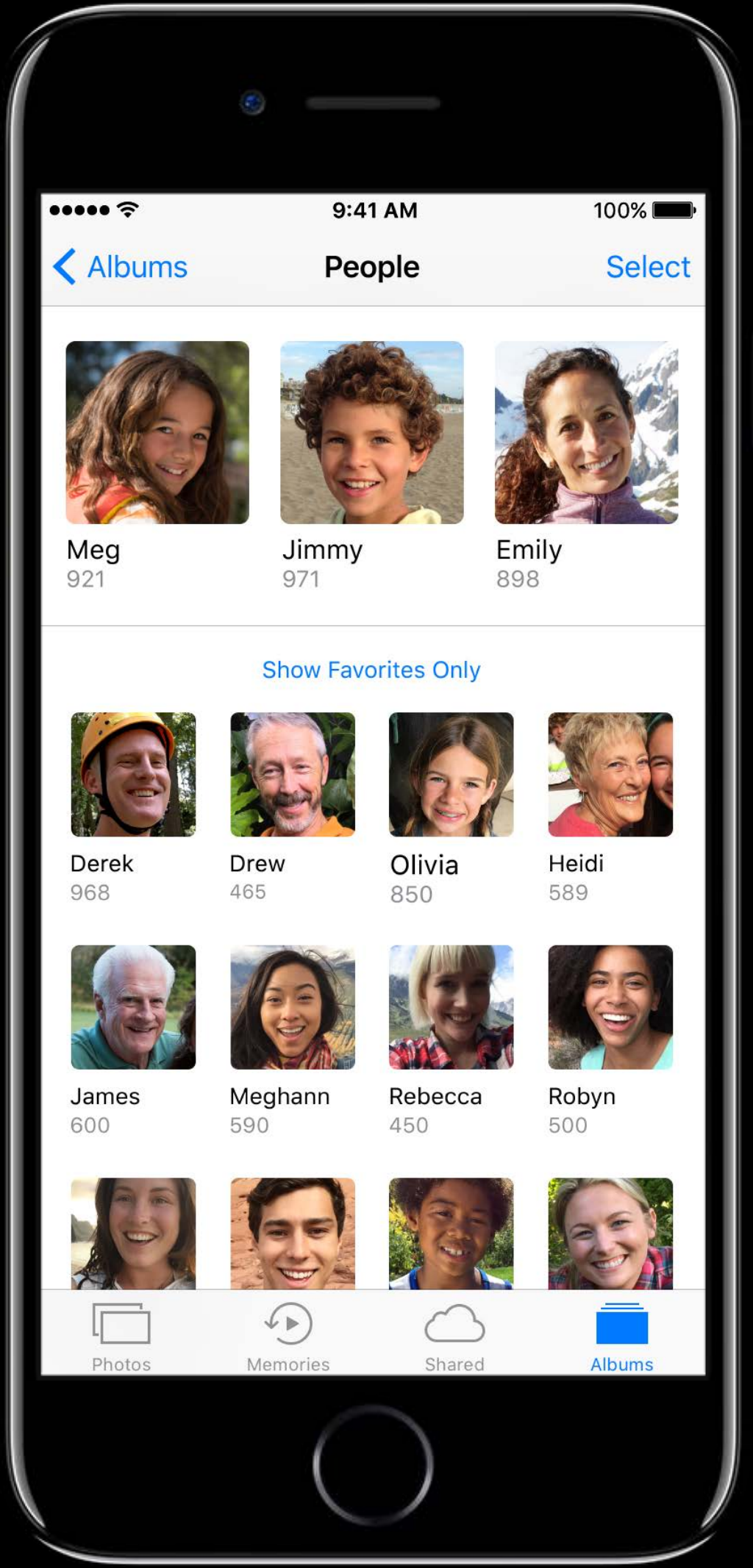
Introducing Core ML

Gaurav Kapoor, Core ML

Michael Siracusa, Core ML

Lizi Ottens, Core ML





9:41 AM 100%

Albums People Select



Meg
921



Jimmy
971



Emily
898

Show Favorites Only



Derek
968



Drew
465



Olivia
850



Heidi
589



James
600



Meghann
590



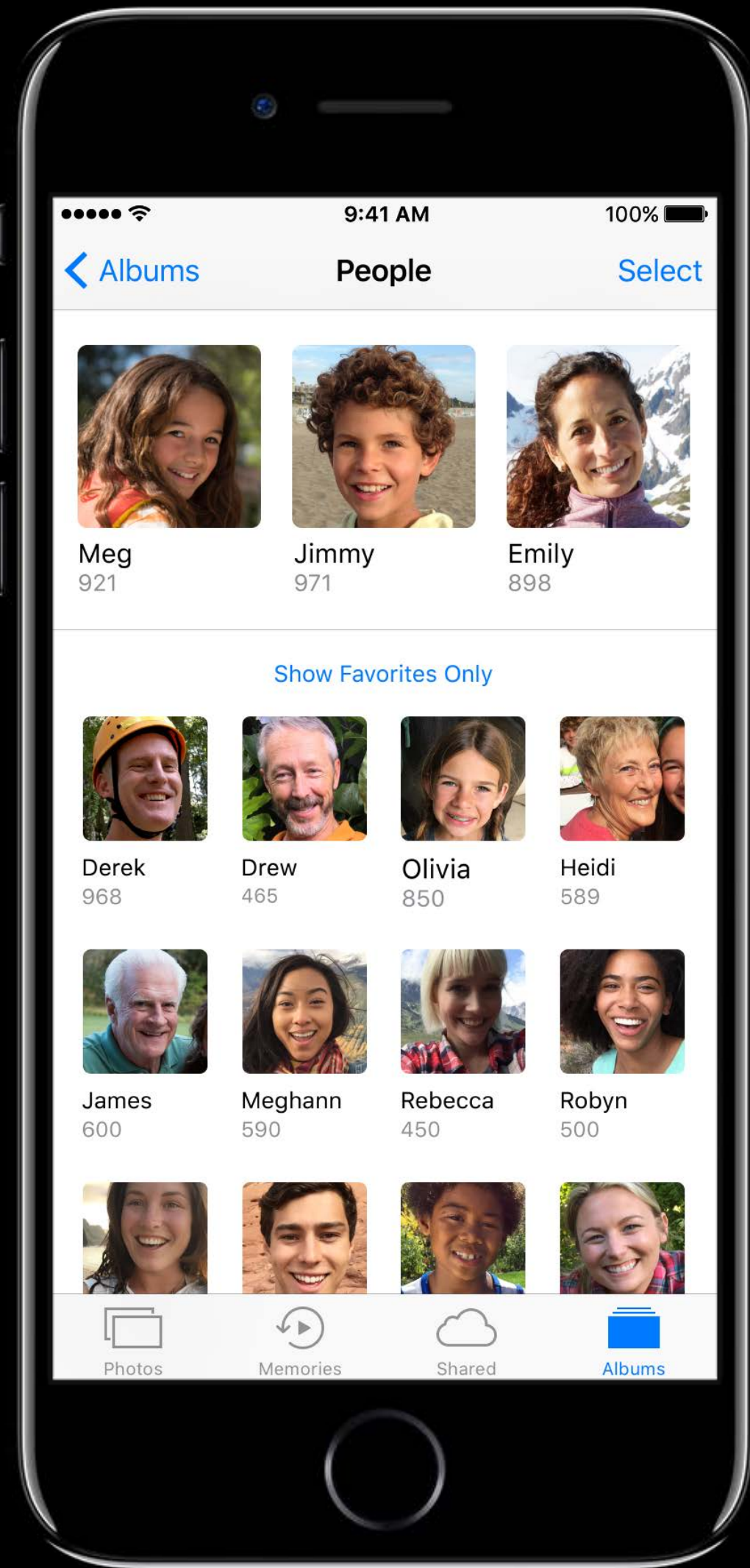
Rebecca
450

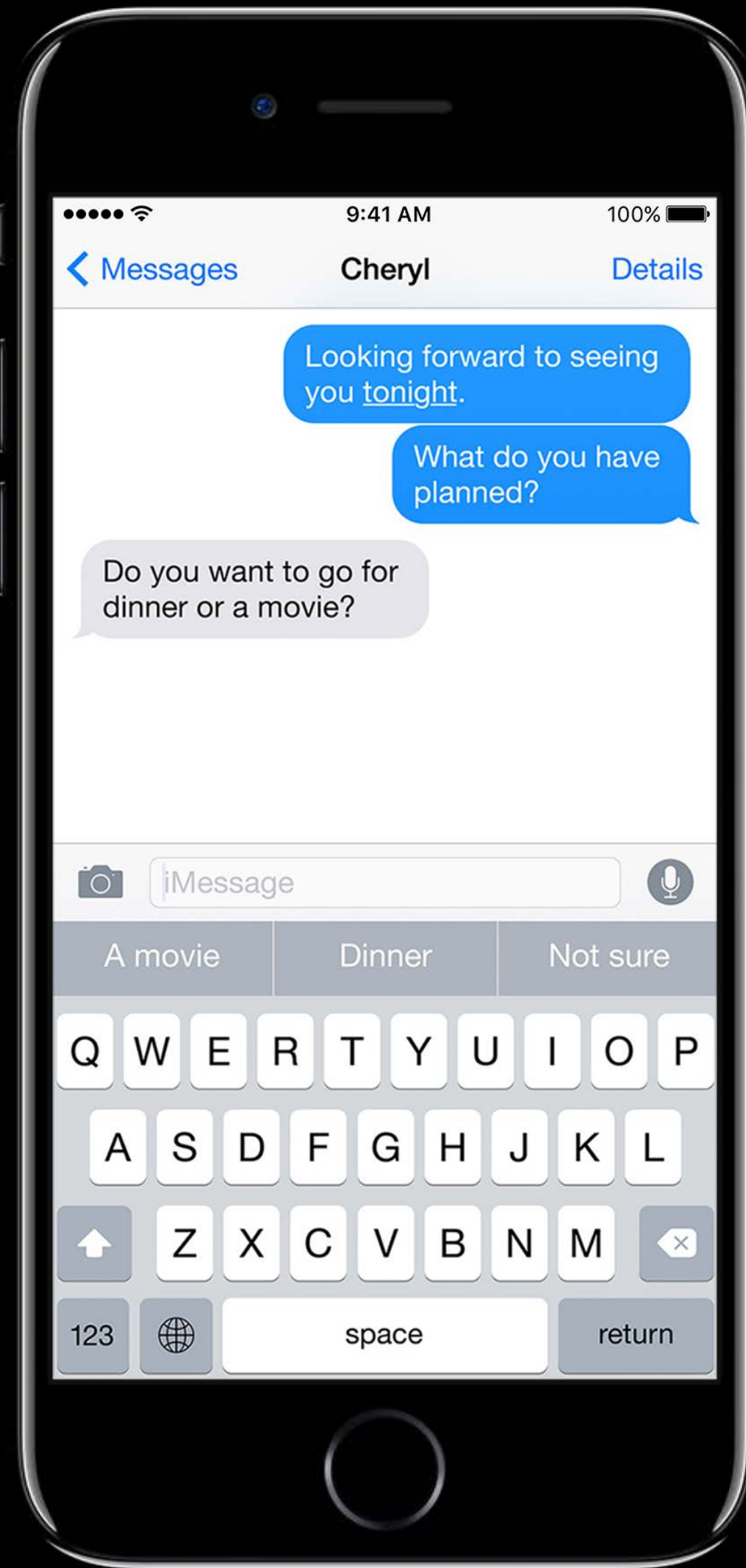
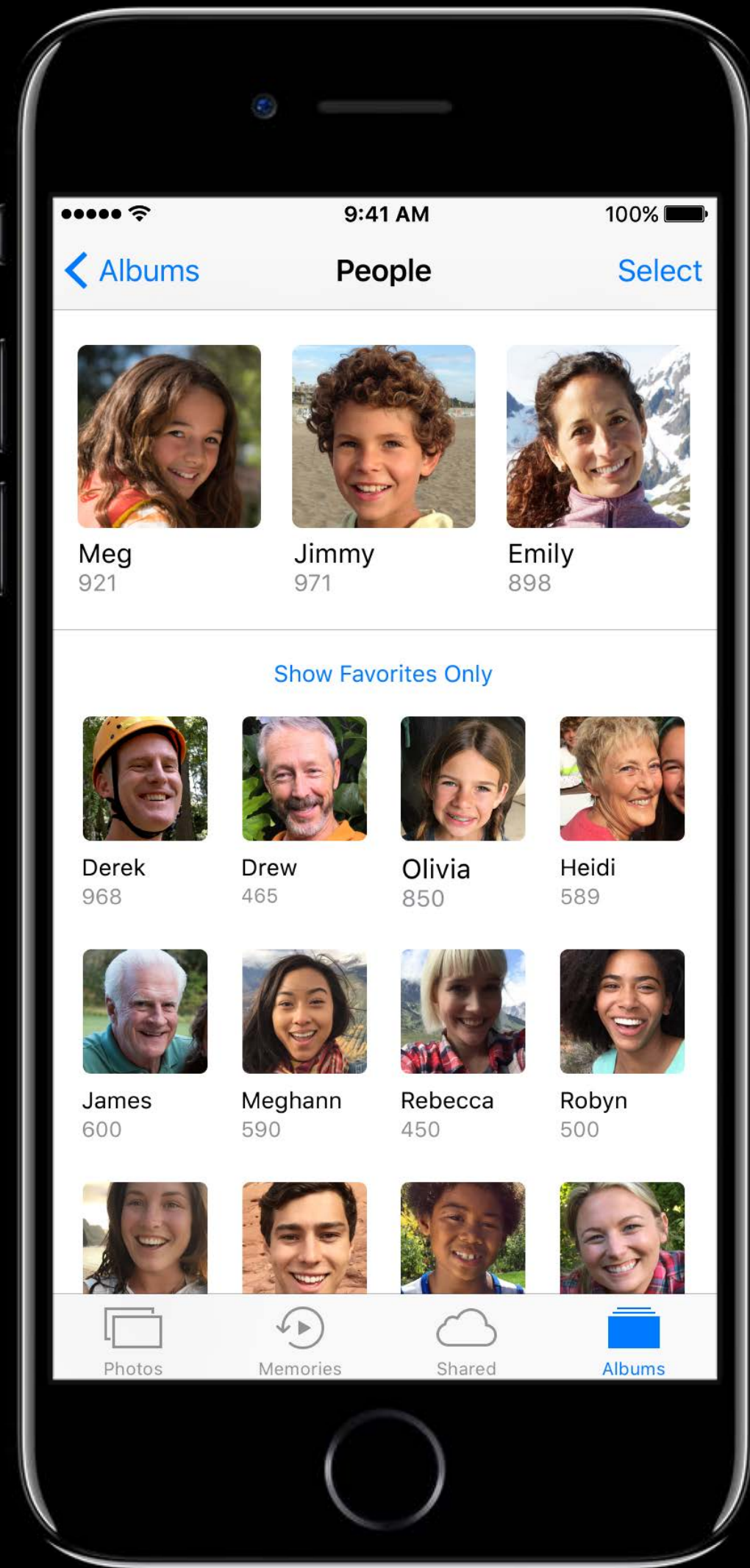


Robyn
500



Photos Memories Shared Albums





Real Time Image Recognition

Text Prediction

Entity Recognition

Sentiment Analysis

Handwriting Recognition

Style Transfer

Search Ranking

Machine Translation

Image Captioning

Personalization

Face Detection

Emotion Detection

Speaker Identification

Music Tagging

Text Summarization

Real Time Image Recognition

Text Prediction

Entity Recognition

Sentiment Analysis

Handwriting Recognition

Style Transfer

Search Ranking

Mac

Image Captioning

Personalization

Face Detection

Emotion Detection

Speaker Identification

Music Tagging

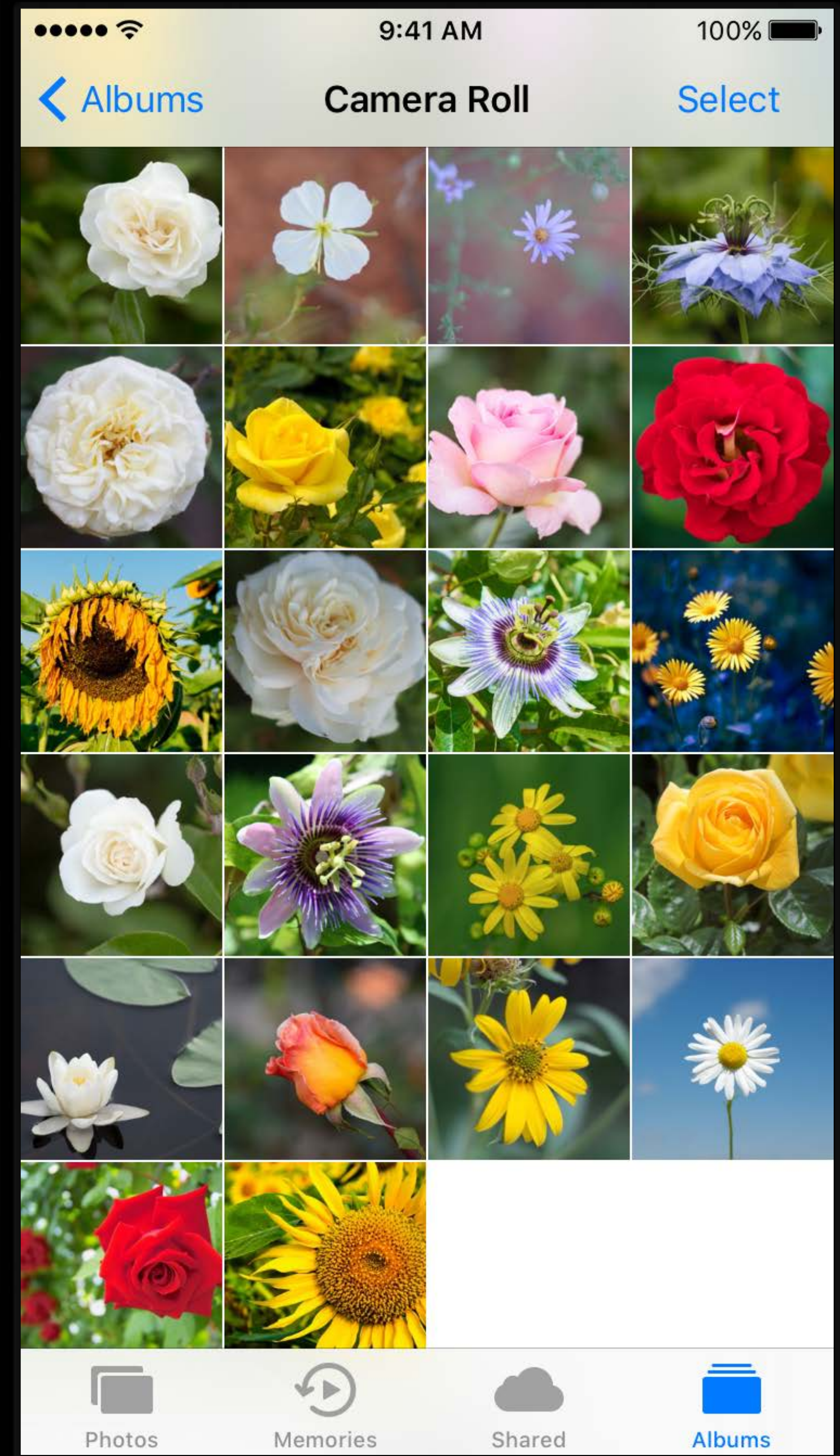
Text Summarization



Why?

Task

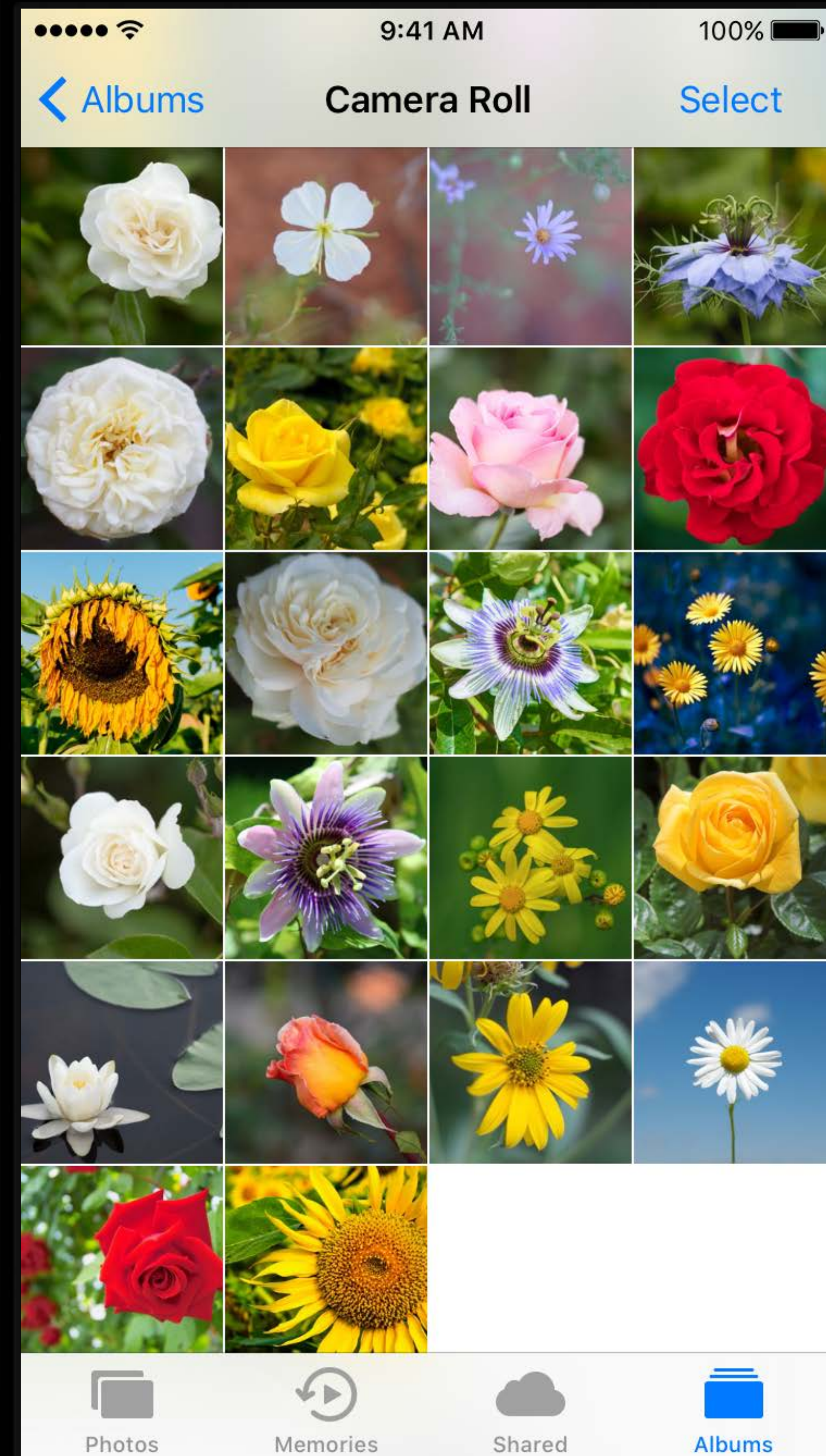
Show all images of roses



Task

Show all images of roses

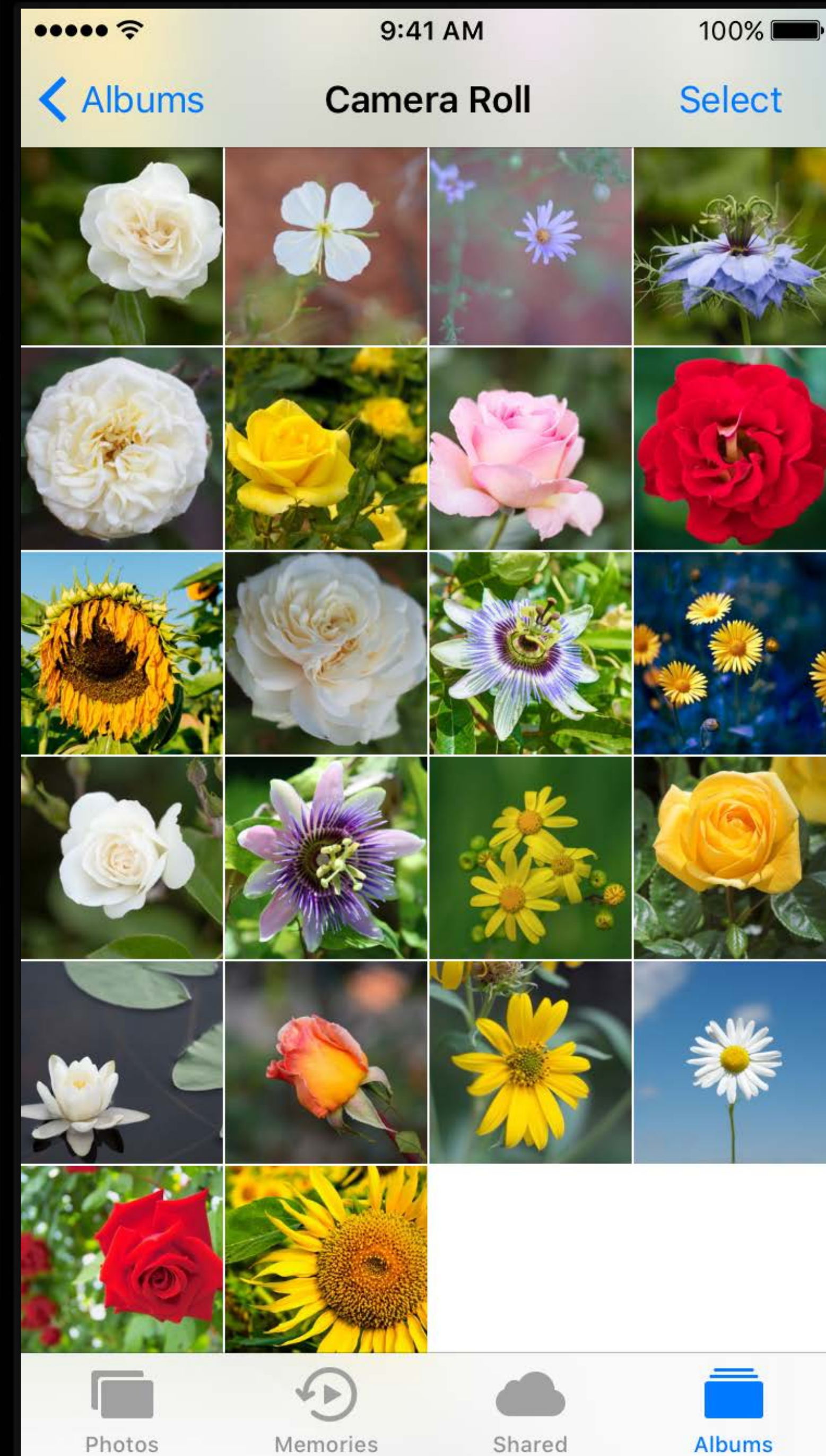
```
// Use color  
if color == "reddish"
```



Task

Show all images of roses

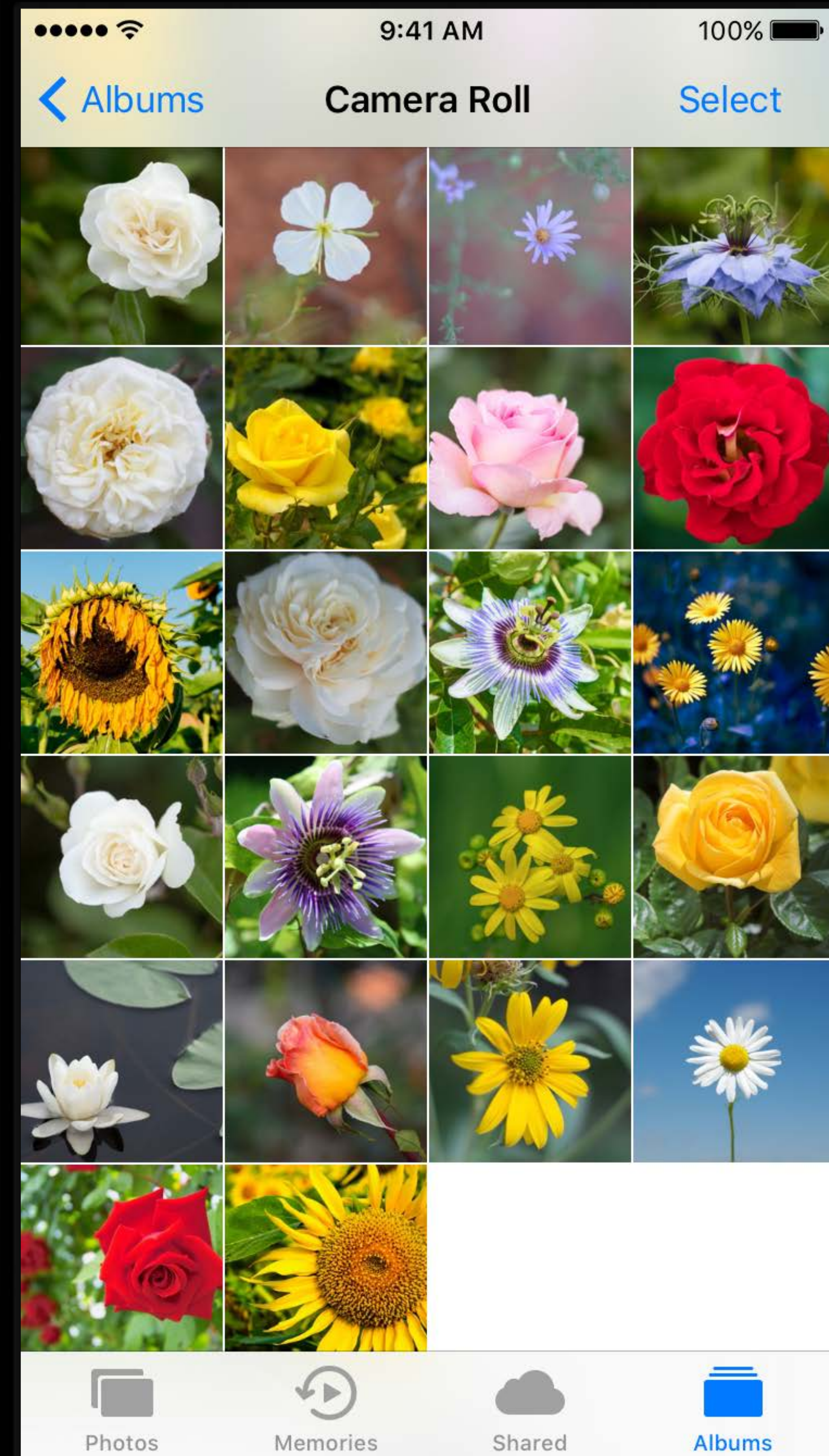
```
// Use color  
if color == "reddish"  
  
// Use shape  
if shape == ???
```



Task

Show all images of roses

Machine Learning



Training

Training

Offline



+ Labels

Training

Offline



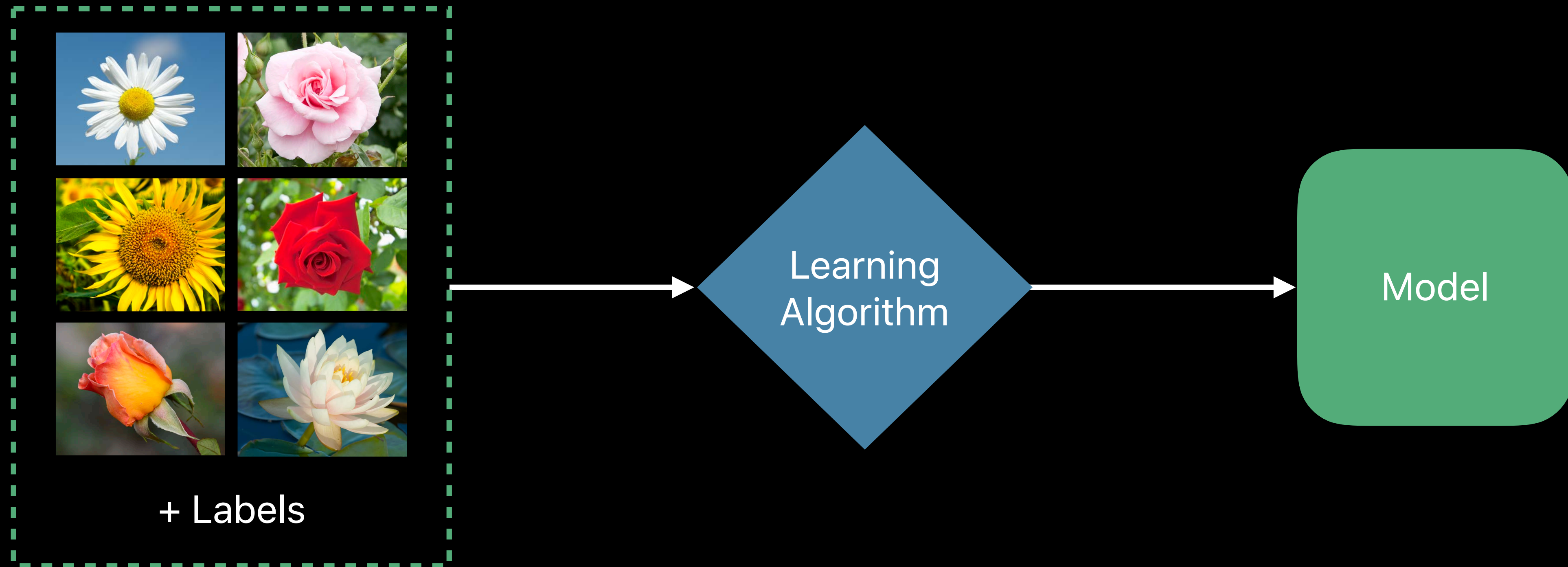
+ Labels



Learning
Algorithm

Training

Offline



Inference



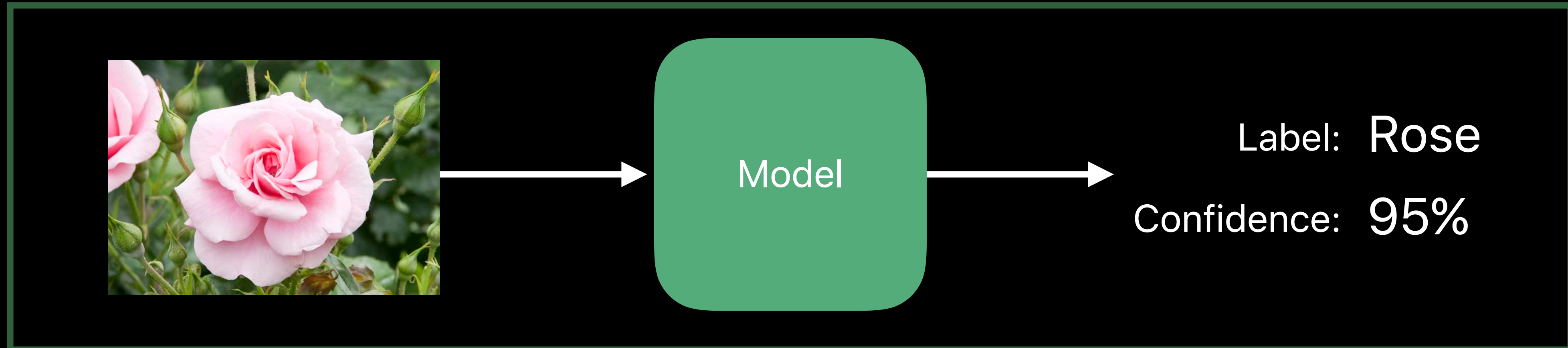
Model

Inference

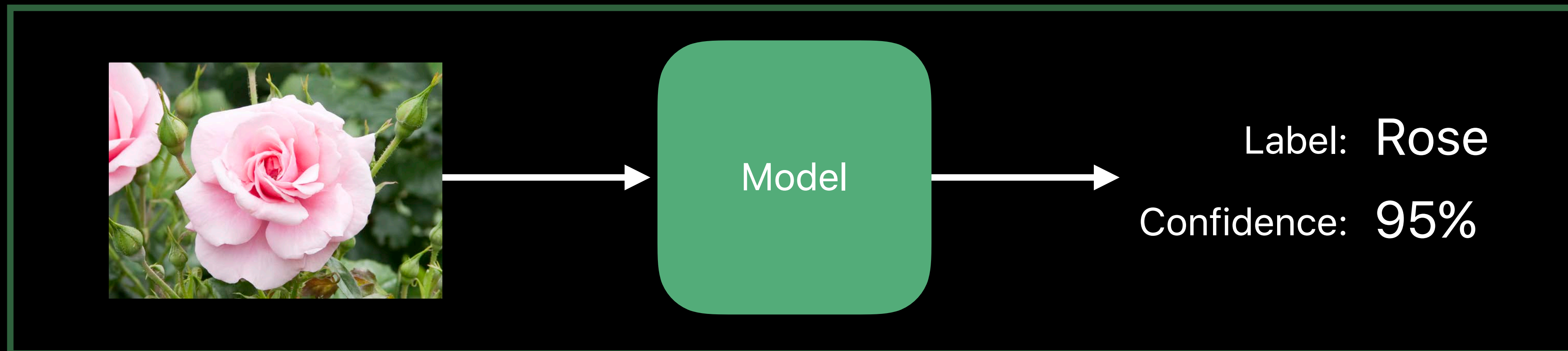
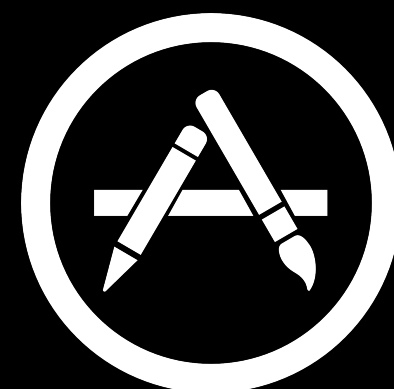


Model

Inference



Inference



Challenging!

```
void convolutionLayer(int kernelWidth,
                    int kernelHeight,
                    int inputFeatureChannels,
                    int outputFeatureChannels,
                    int strideX,
                    int strideY,
                    int numRows,
                    int numCols,
                    float* input,
                    float* output,
                    float* weights,
                    int widthPadding,
                    int heightPadding,
                    float alpha,
                    float beta) {
    memset(output, 0, ((numRows - kernelWidth + 2*widthPadding)/strideX + 2*widthPadding)/2 + 1)*outputFeatureChannels * sizeof(float));
    for (int depthInd = 0; depthInd < outputFeatureChannels; depthInd++) {
        // loop over input (color) channels
        for (int colorInd = 0; colorInd < inputFeatureChannels; colorInd++) {
            int numRowsOut = (numRows - kernelWidth + 2*widthPadding)/strideY + 1;
            int numColsOut = (numCols - kernelHeight + 2*widthPadding)/strideX + 1;
            // loop over the pixels of the image
            for (int i=0; i < numRowsOut; i++) {
                for(int j=0; j < numColsOut; j++) {
                    // loop over this kernel
                    for(int m=0; m < kernelWidth; m++) {
                        int mm = kernelWidth - 1 - m;
                        for(int n=0; n < kernelHeight; n++) {
                            int nn = kernelHeight - 1 - n;
                            int ii = i + (m - kernelWidth/2);
                            int jj = j + (n - kernelHeight/2);
                            if( ii >= 0 && ii < numRows && jj >= 0 && jj < numCols) {
                                float weight = weights[nn + mm * kernelHeight * kernelWidth];
                                float value = input[jj + ii*numCols + colorInd * outputFeatureChannels] * weight;
                                output[j + i*numColsOut + depthInd*numColsOut * outputFeatureChannels] += value;
                            }
                        }
                    }
                }
            }
        }
    }
    // loop and apply nonlinearity
    for (int i = 0; i < ((numRows - kernelWidth + 2*widthPadding)/strideX + 1) * ((numRows - kernelWidth + 2*widthPadding)/strideY + 1) * outputFeatureChannels;
    i++) {
        output[i] = alpha*tanh(beta*output[i]);
    }
}
```

Challenges

Correctness

Performance

Energy Efficiency

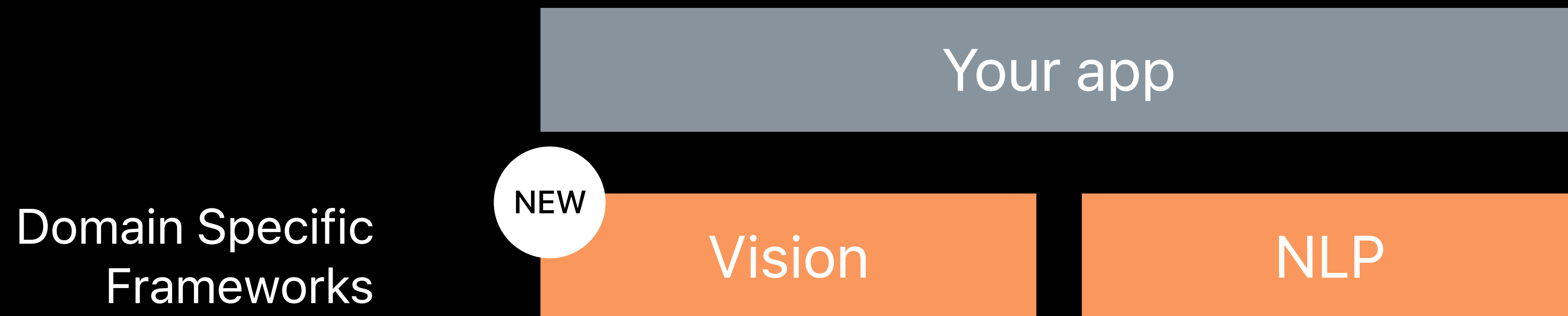
ML Frameworks

ML Frameworks

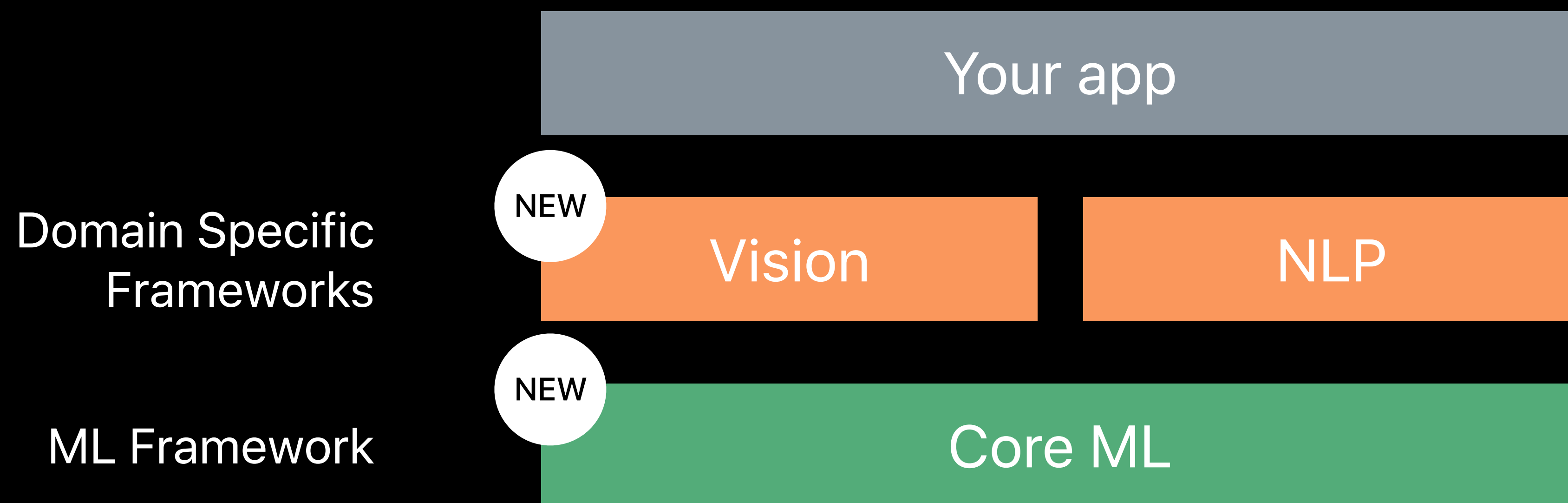


Your app

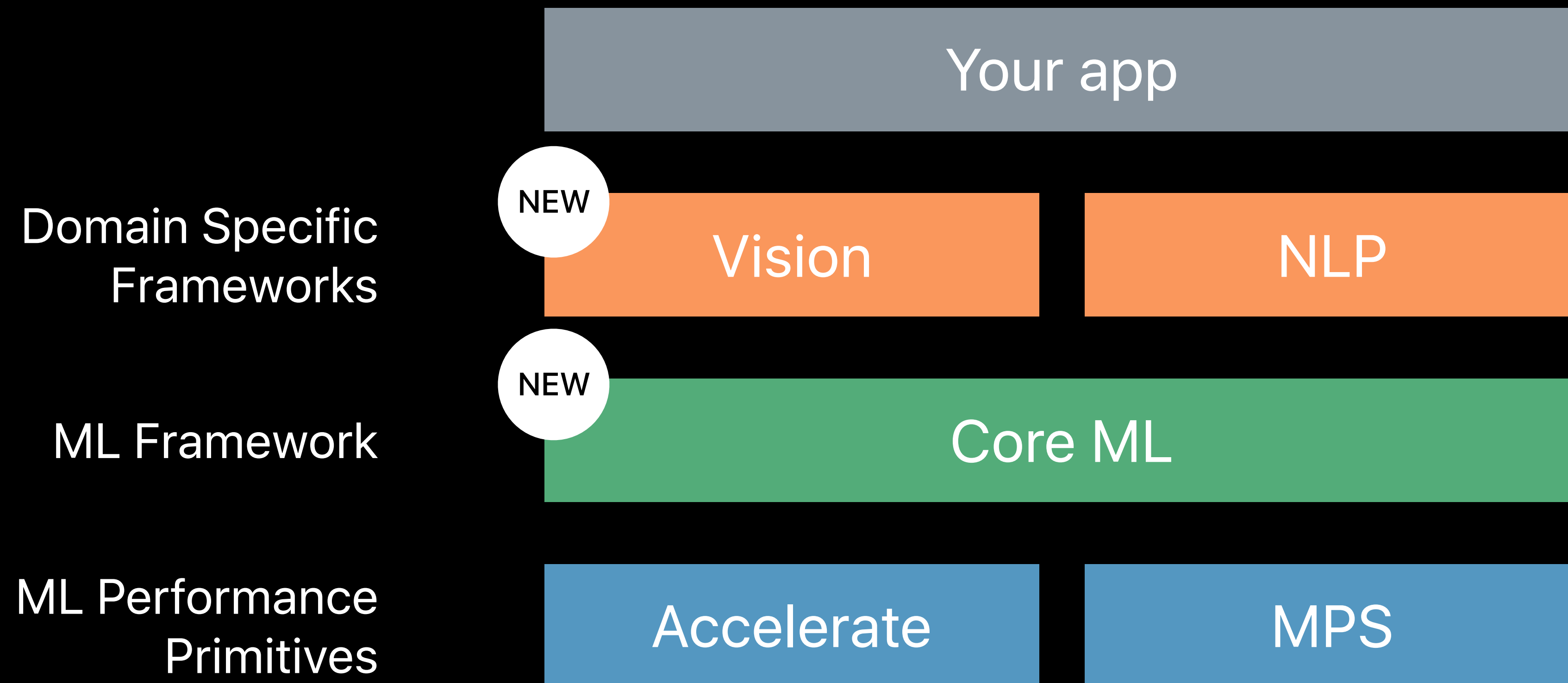
ML Frameworks



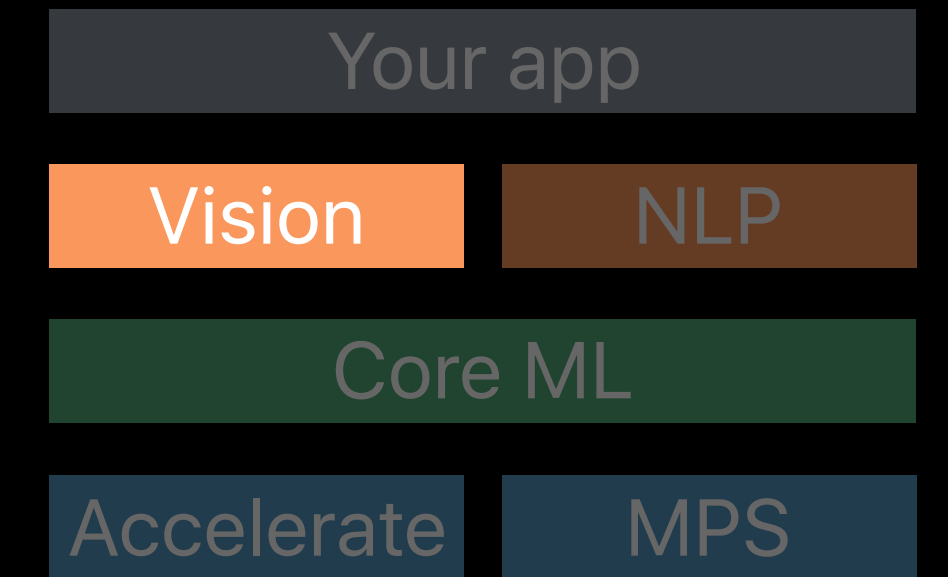
ML Frameworks



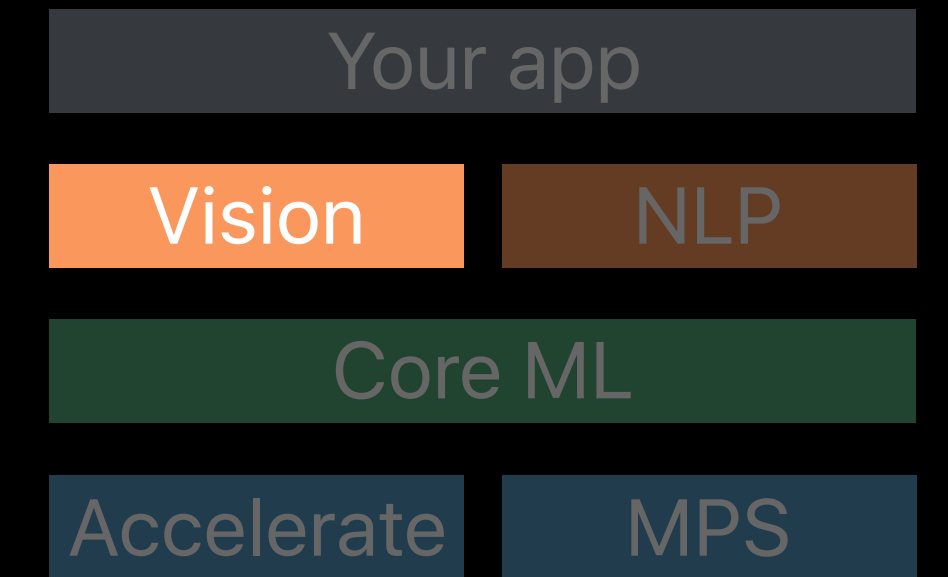
ML Frameworks



Vision Framework

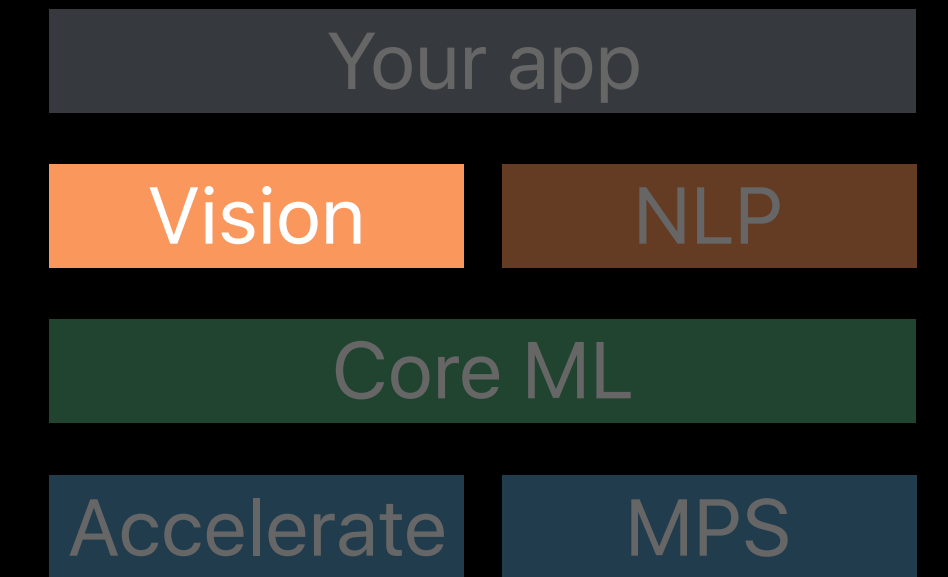


Vision Framework



Object Tracking

Vision Framework

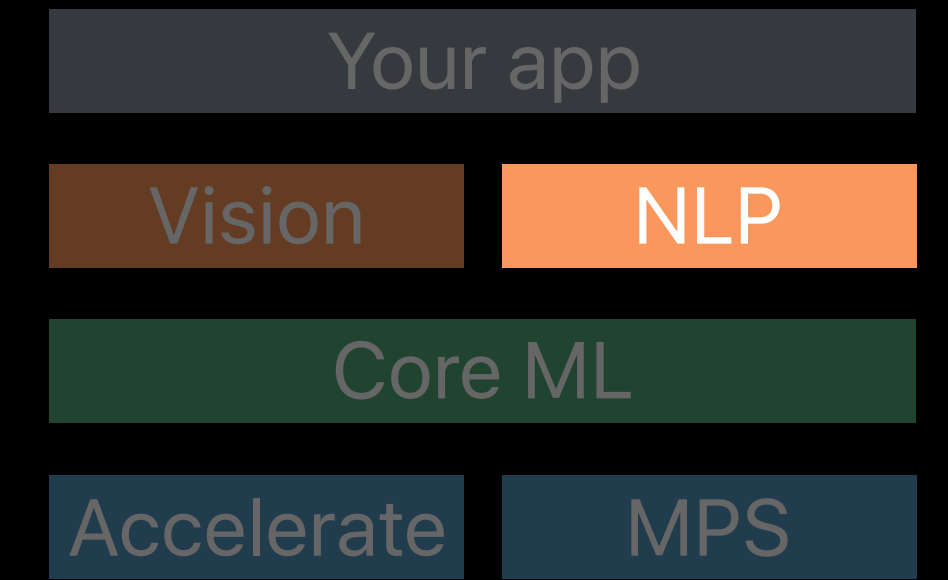


Object Tracking

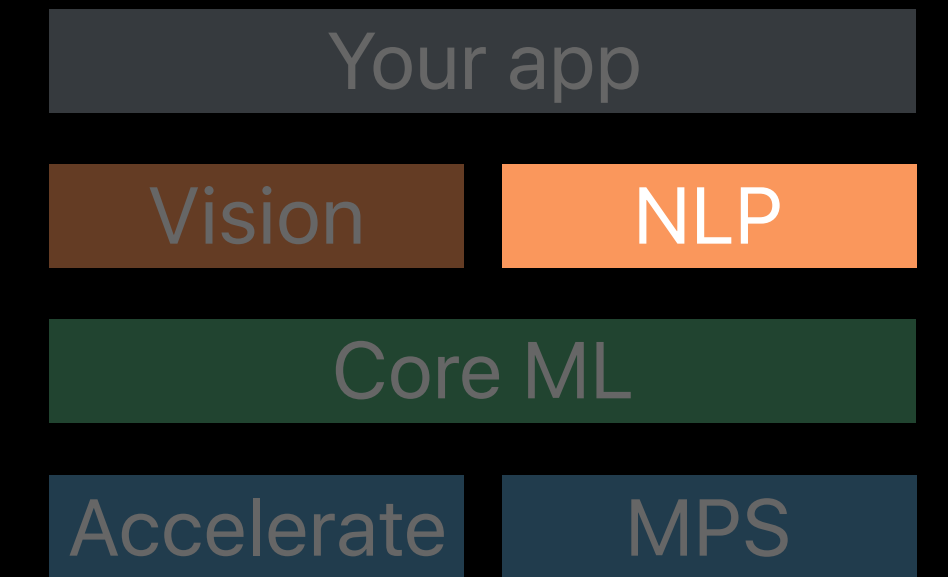


Face Detection

Natural Language Processing

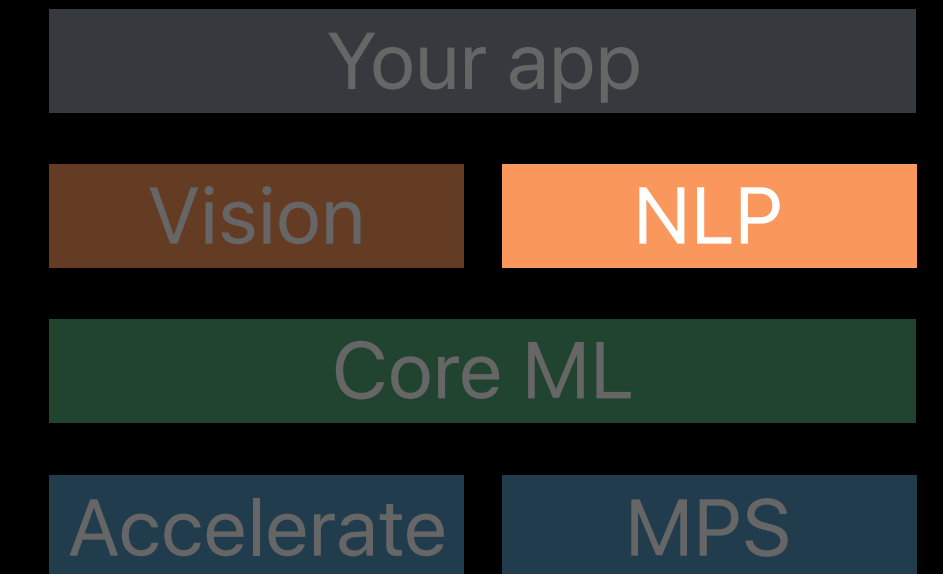


Natural Language Processing

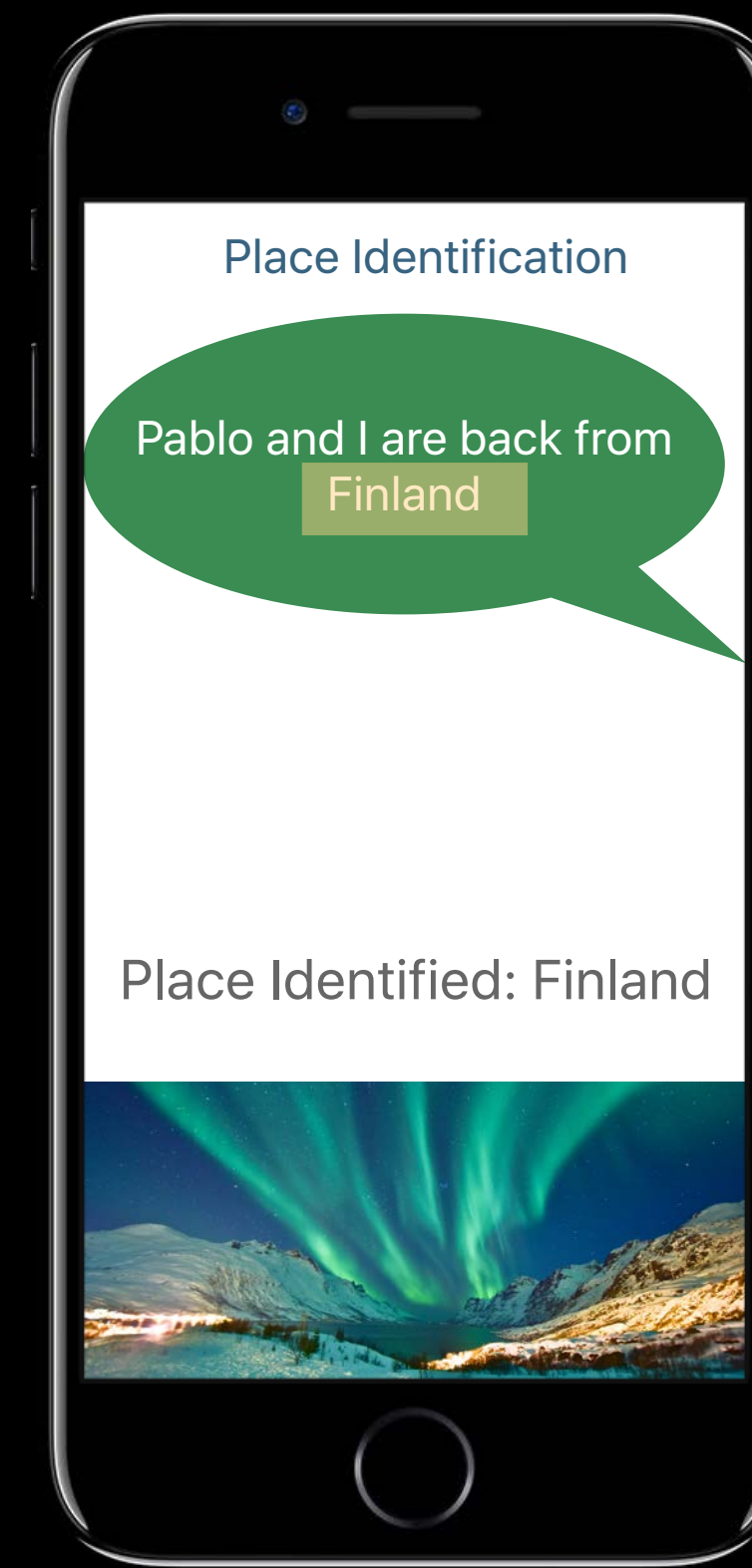


Language Identification

Natural Language Processing

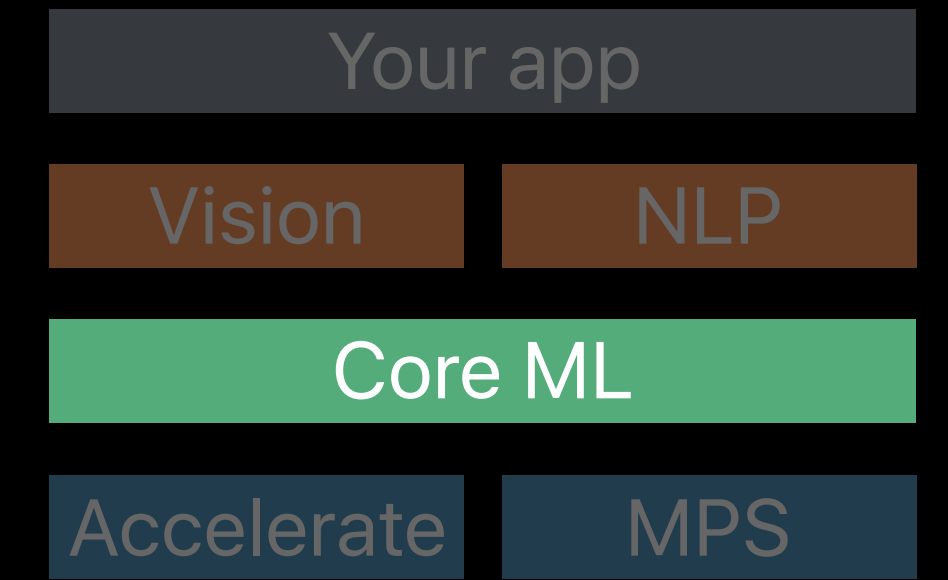


Language Identification

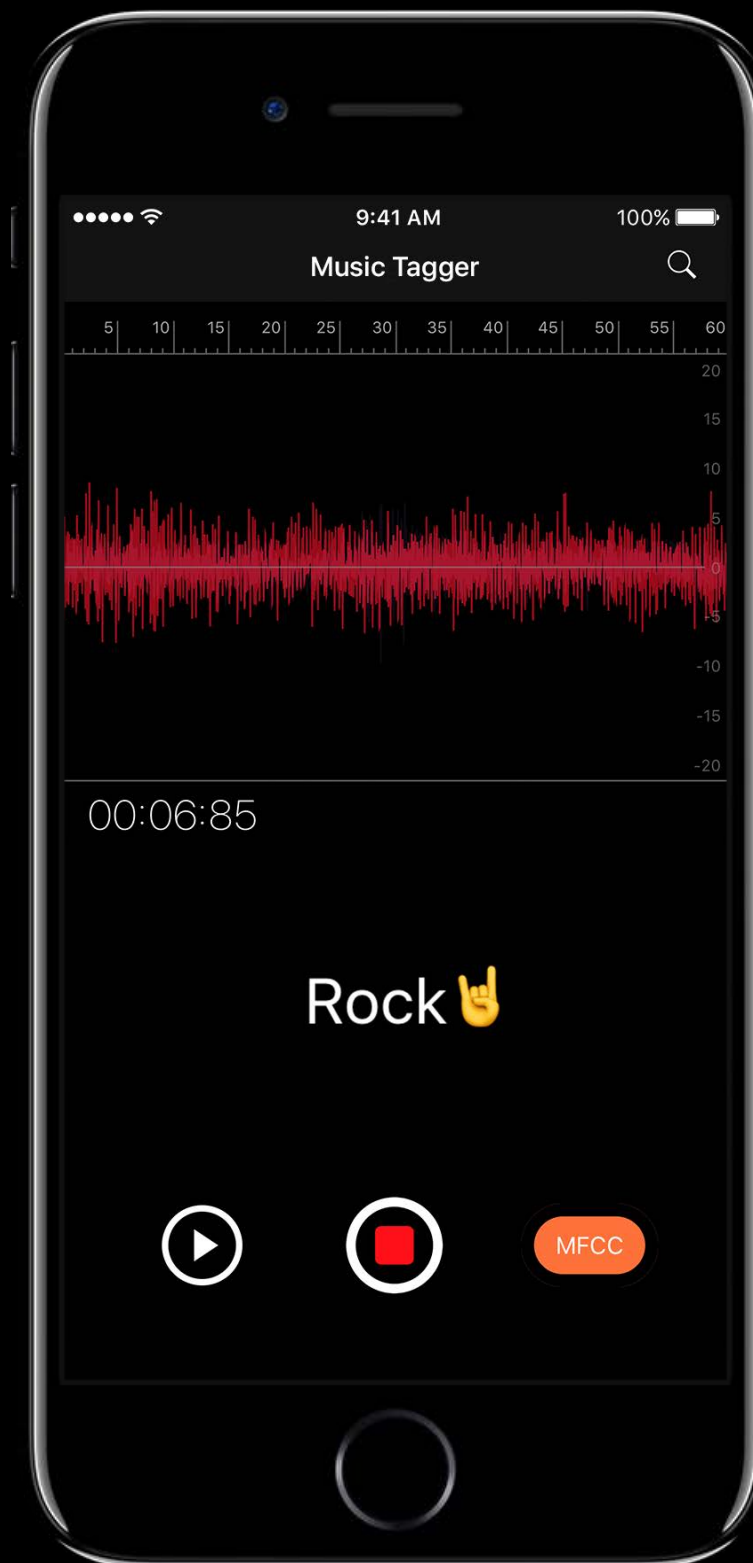
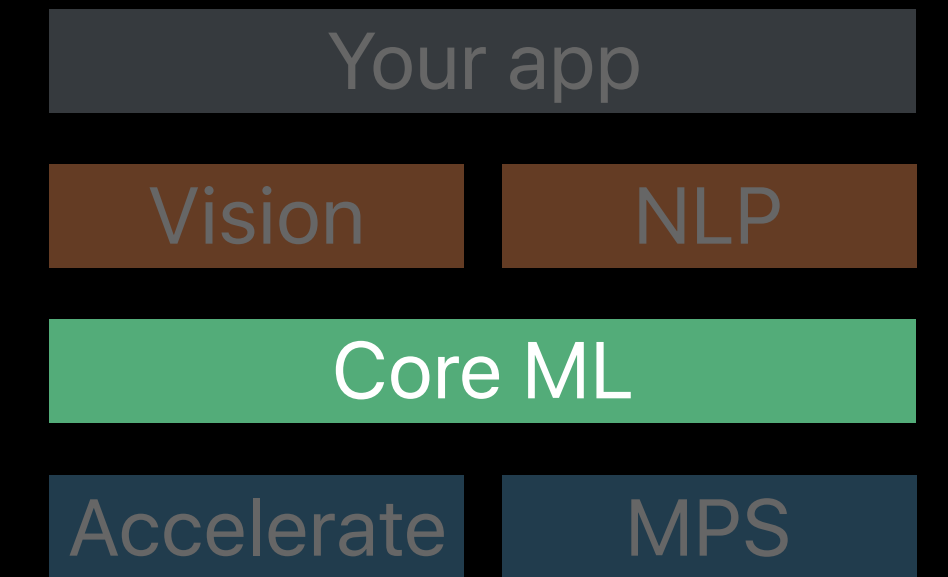


Named Entity Recognition

Core ML

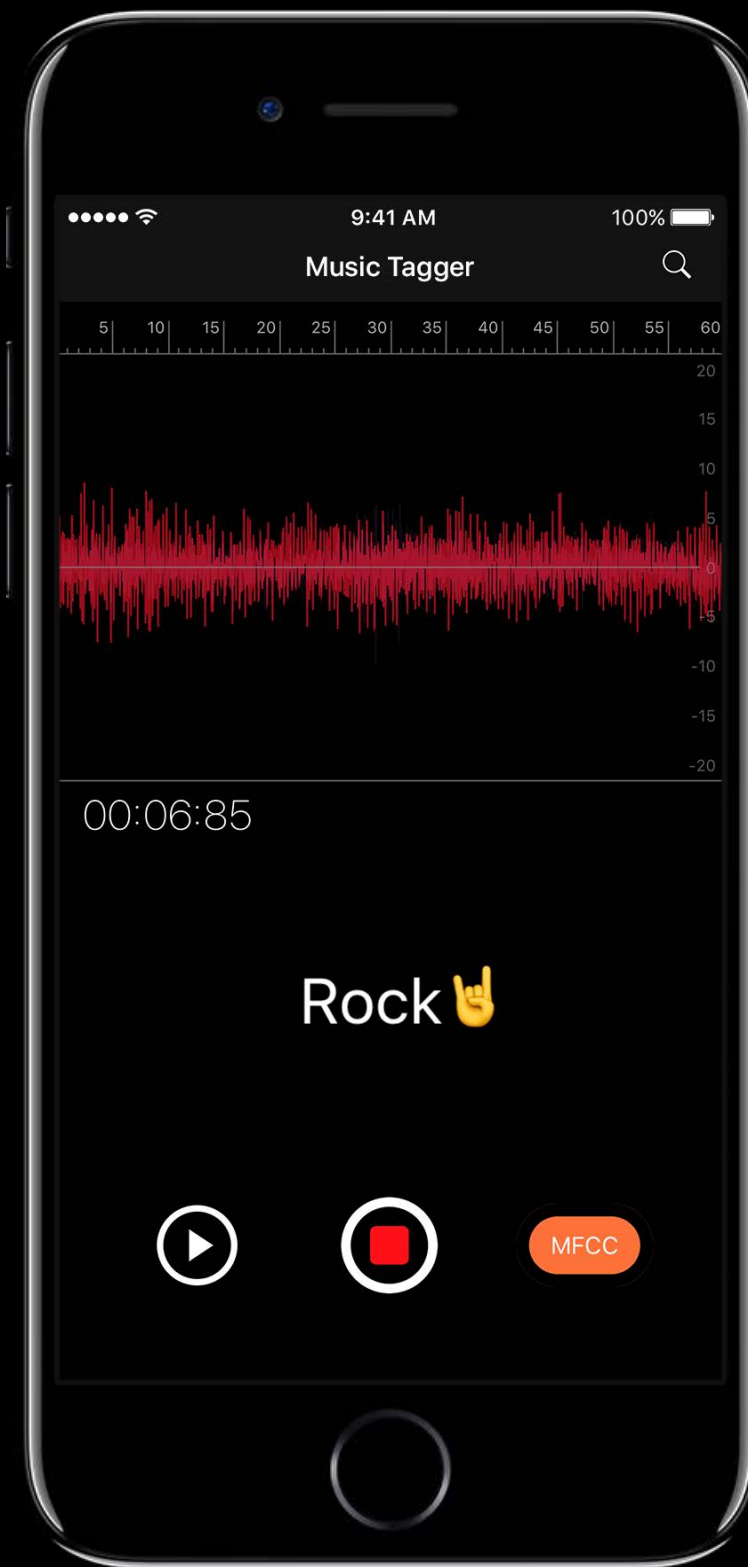
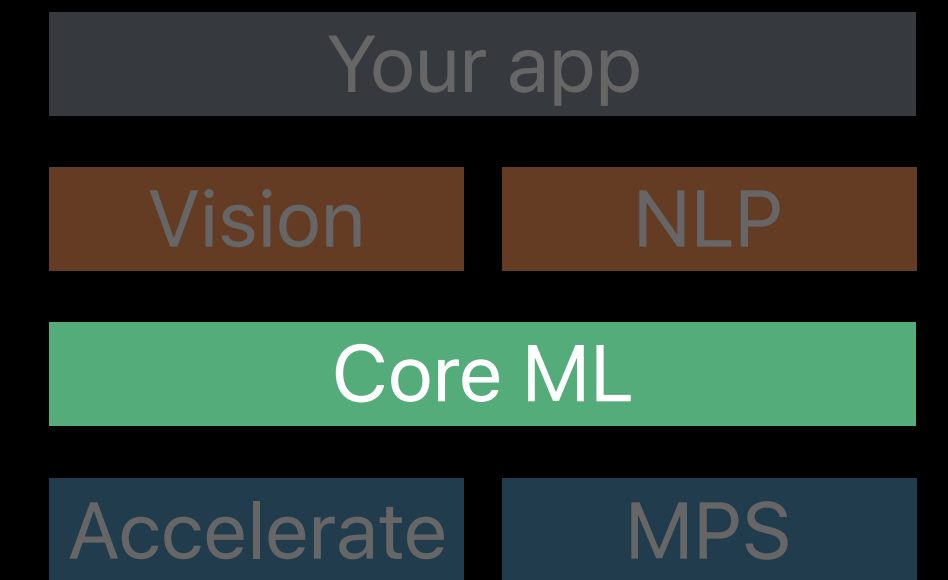


Core ML



Music Tagging

Core ML



Music Tagging

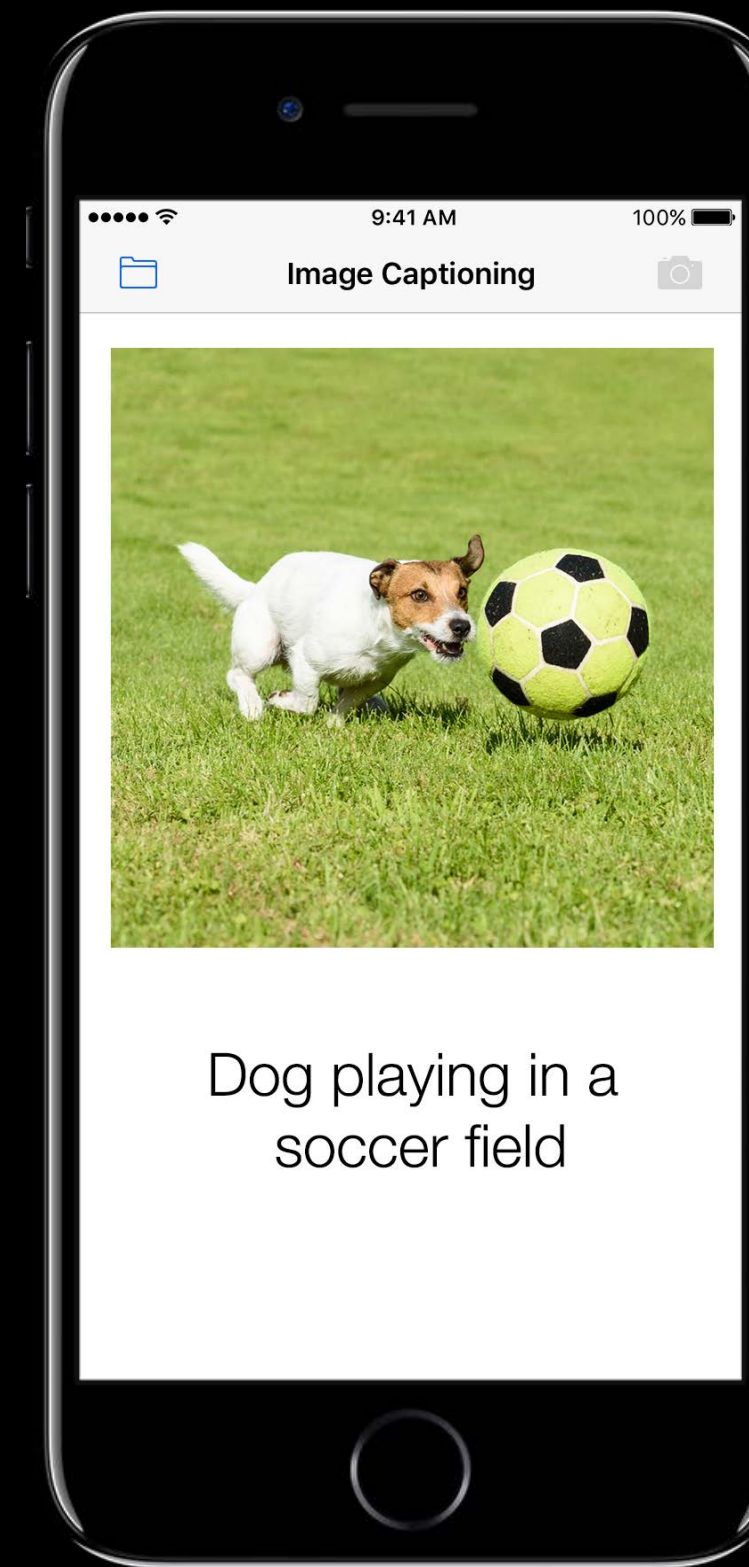
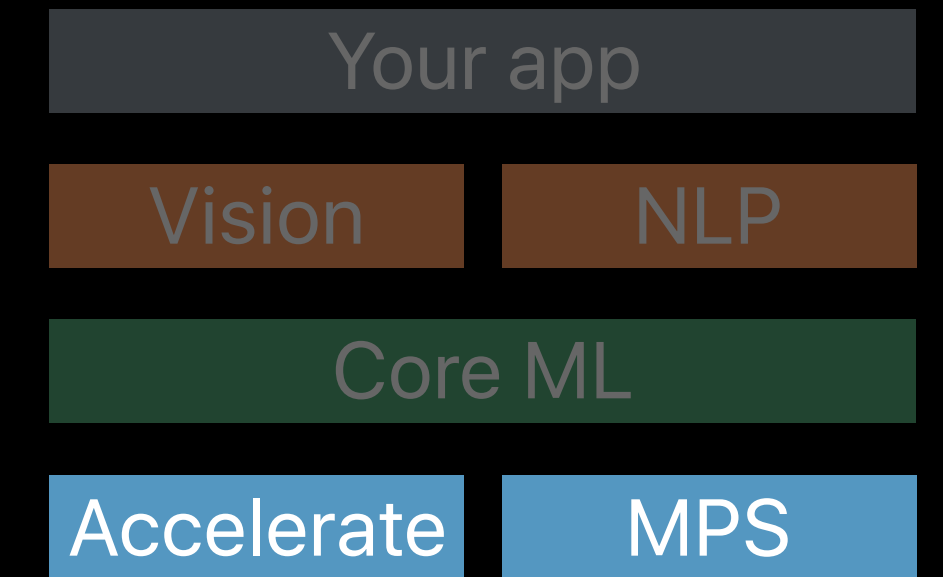


Image Captioning

Accelerate and MPS

High performance math

Inference for custom ML models



The Apple logo followed by the text 'A10X' in a large, white, sans-serif font, centered on a dark grey background.

Run on Device

Run on Device

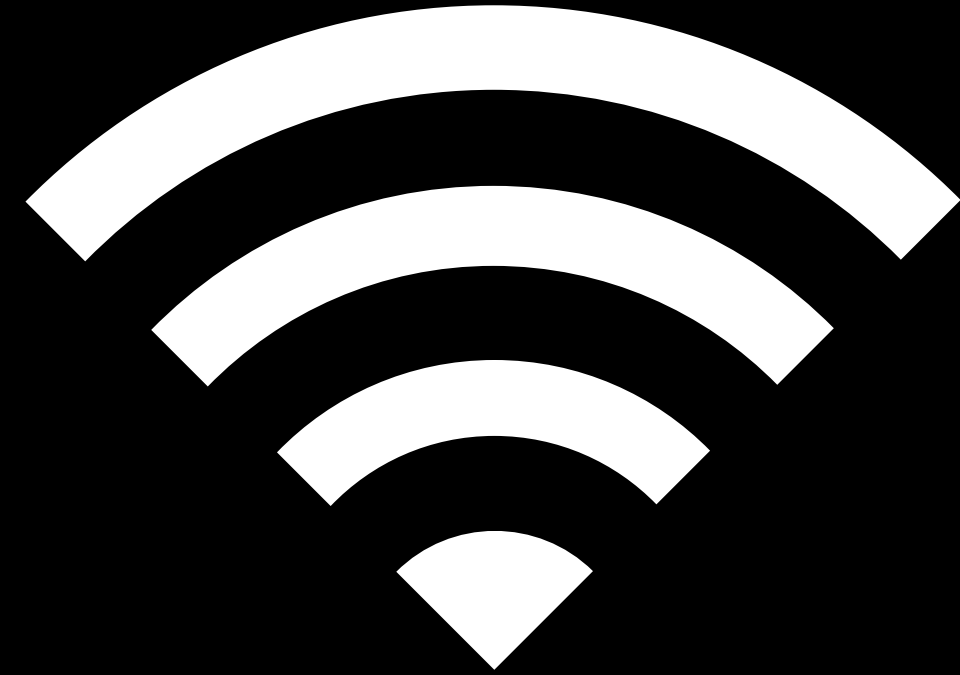


User Privacy

Run on Device



User Privacy

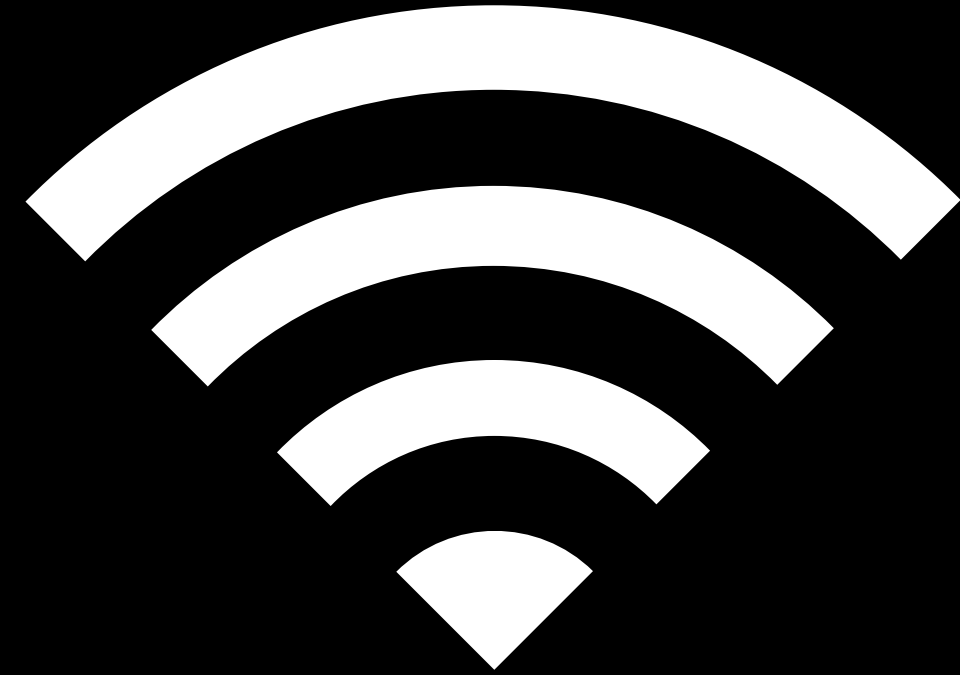


Data Cost

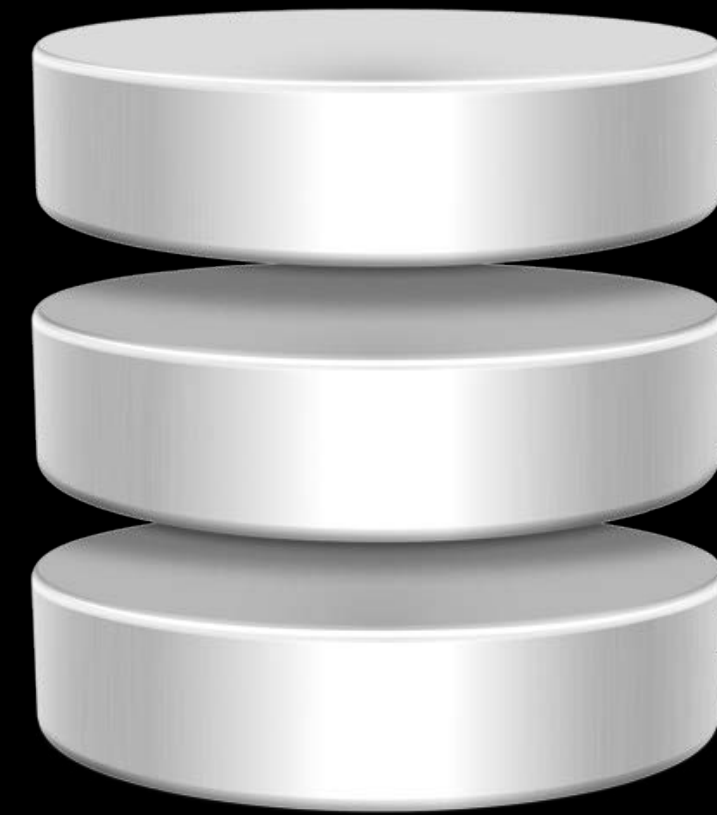
Run on Device



User Privacy



Data Cost



Server Cost

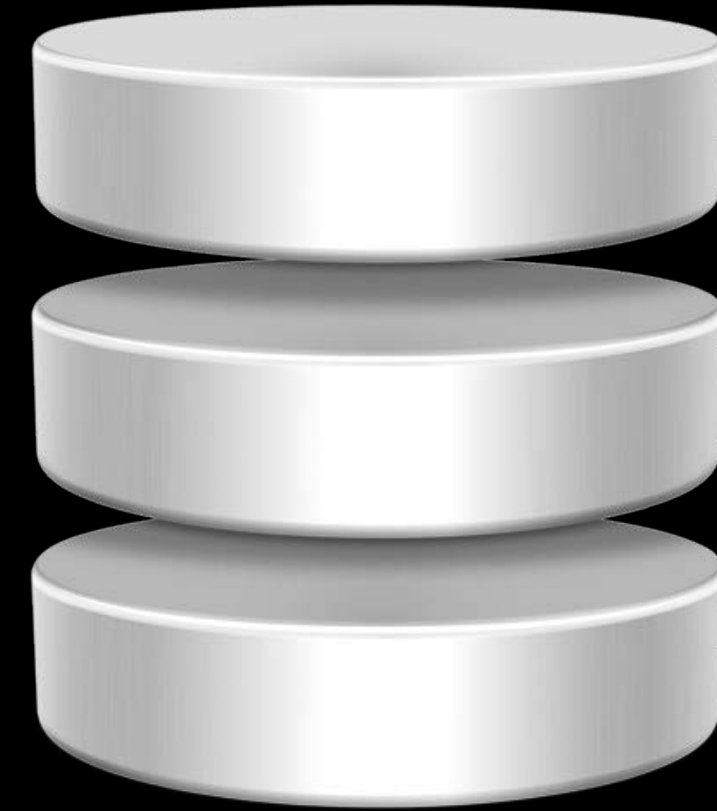
Run on Device



User Privacy



Data Cost




Server Cost



Always Available



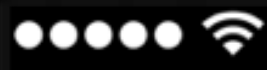
9:41 AM

100% 




dining table

35.8 %



9:41 AM

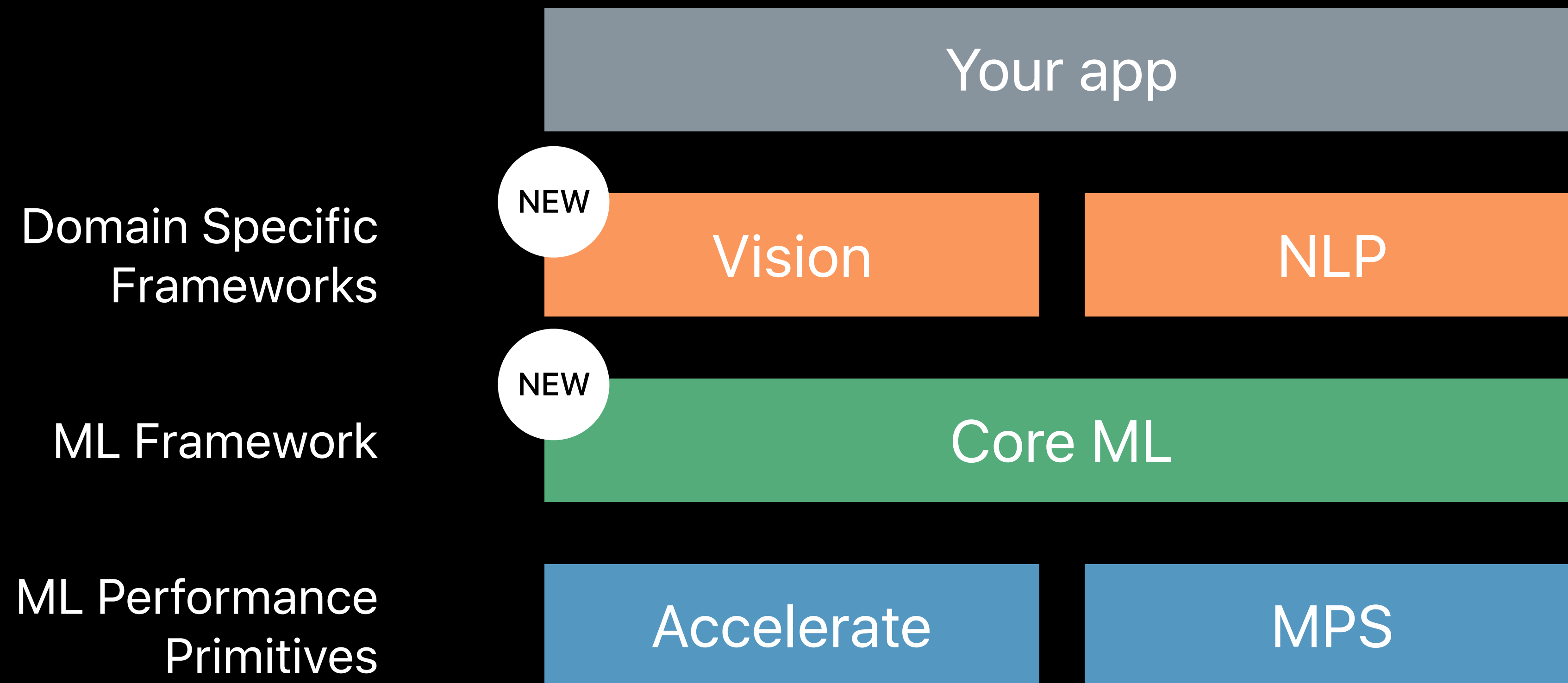
100% 



dining table

35.8 %

ML Frameworks



Core ML

Michael Siracusa, Core ML

Overview

Overview

Models

Overview

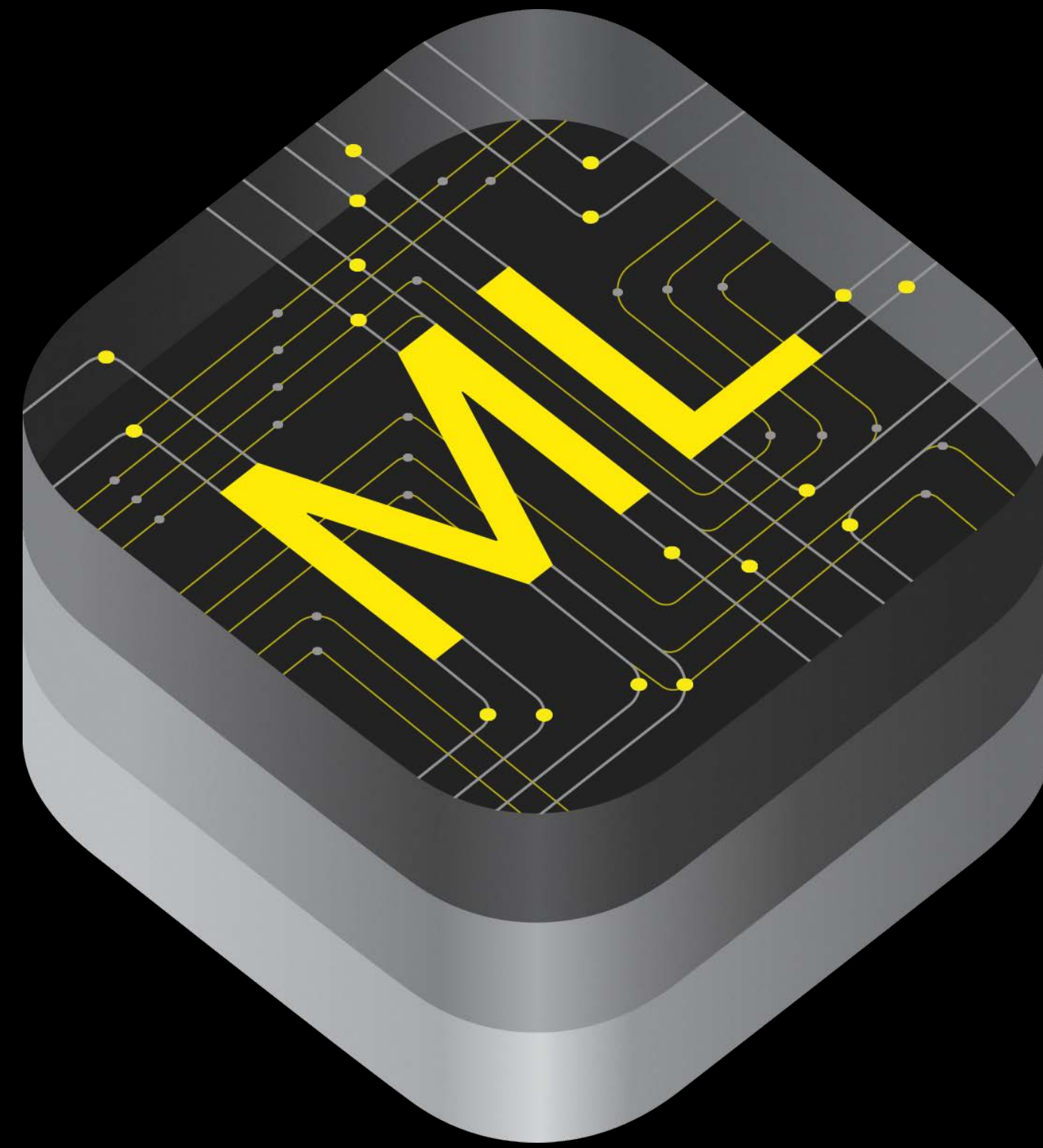
Models

Development Flow

Overview

Models

Development Flow



macOS

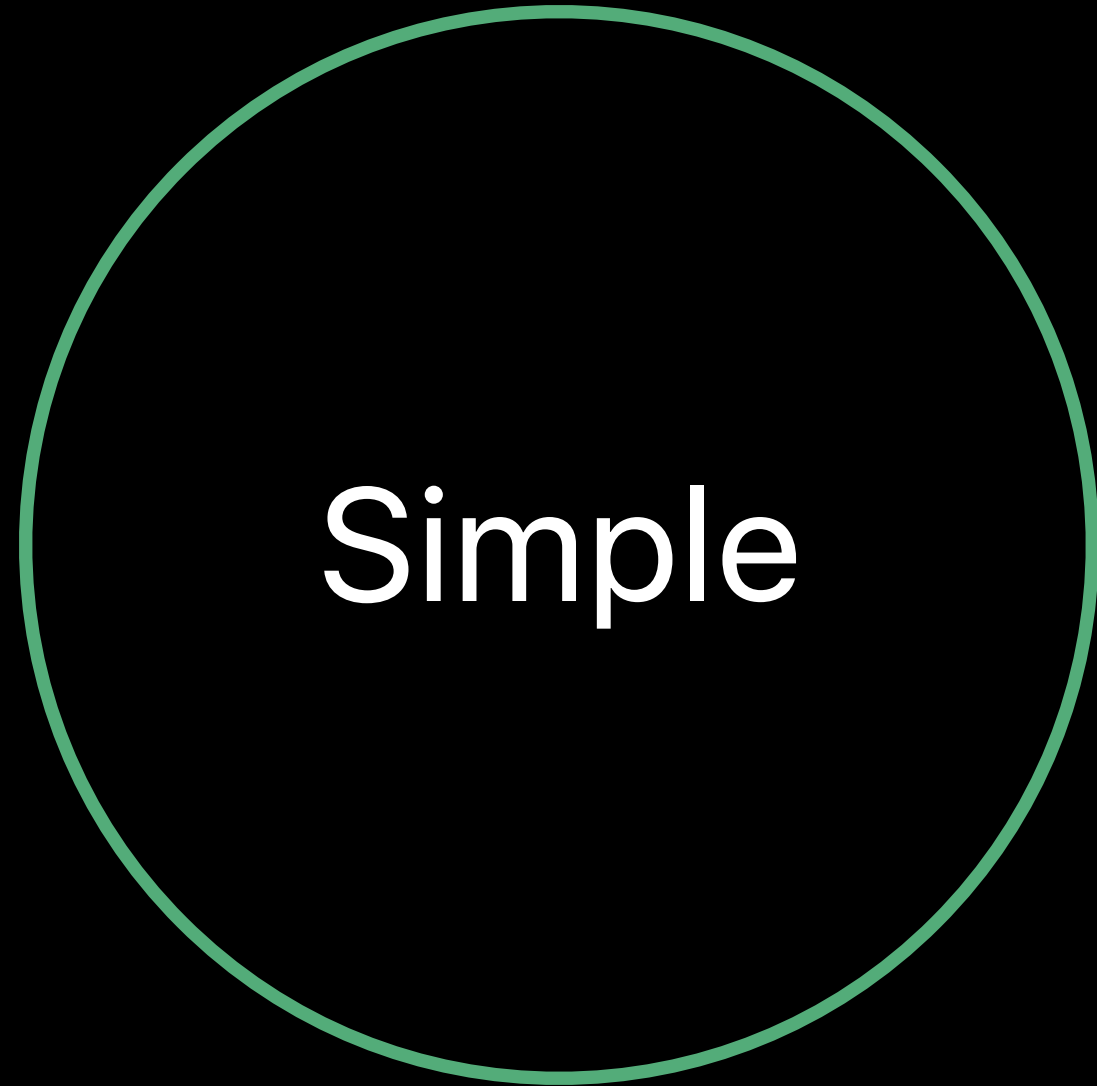
iOS

watchOS

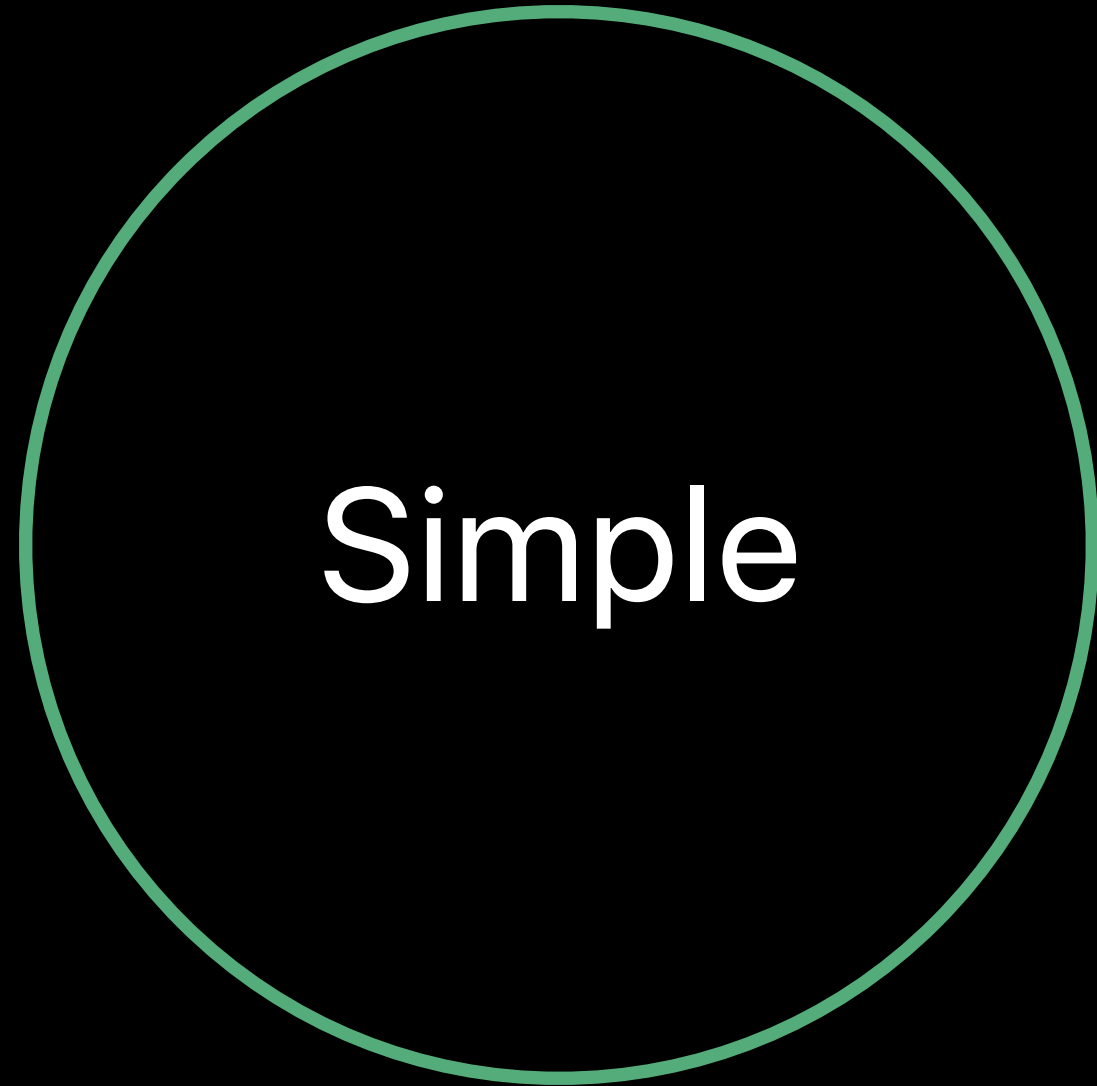
tvOS

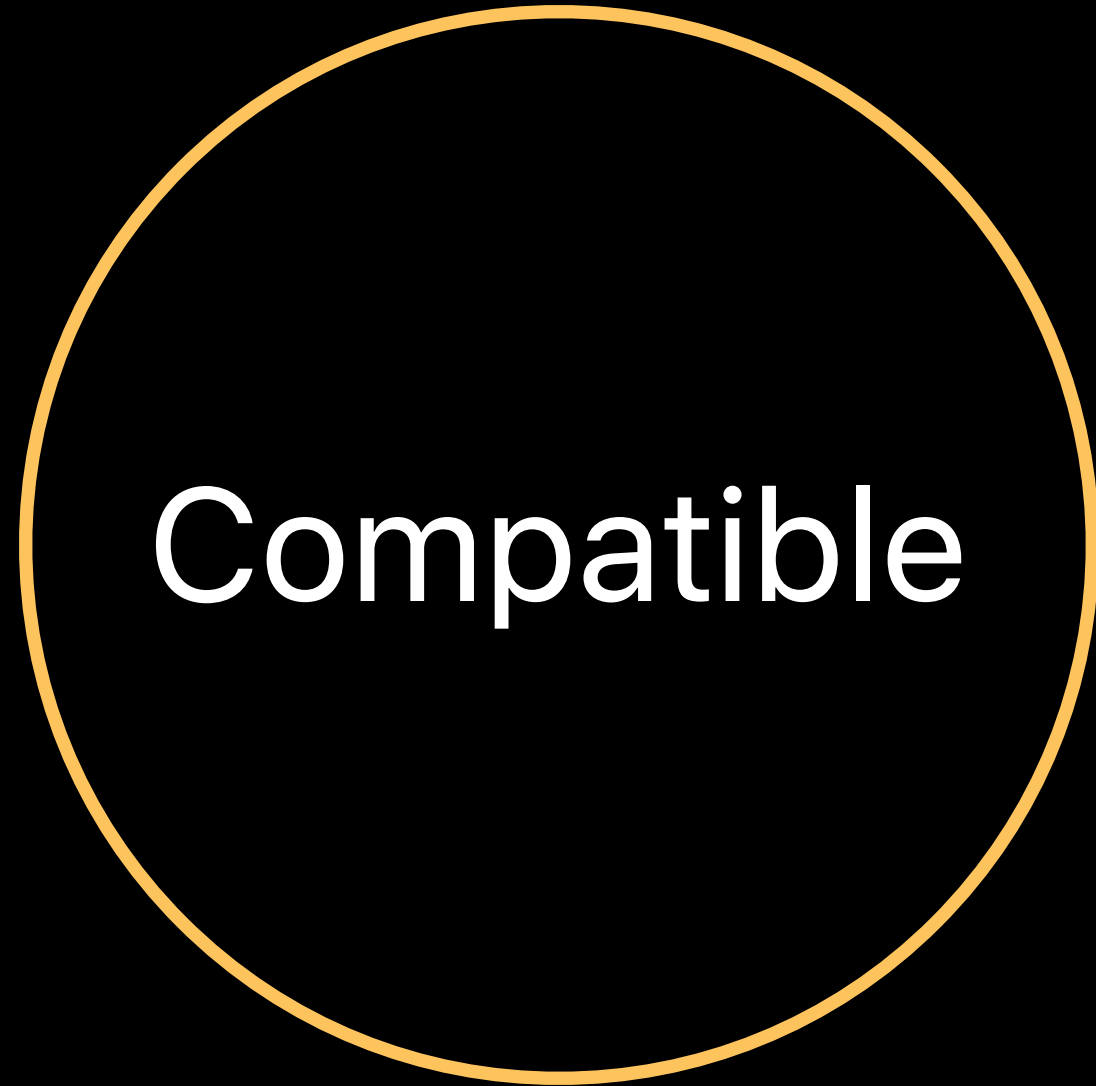
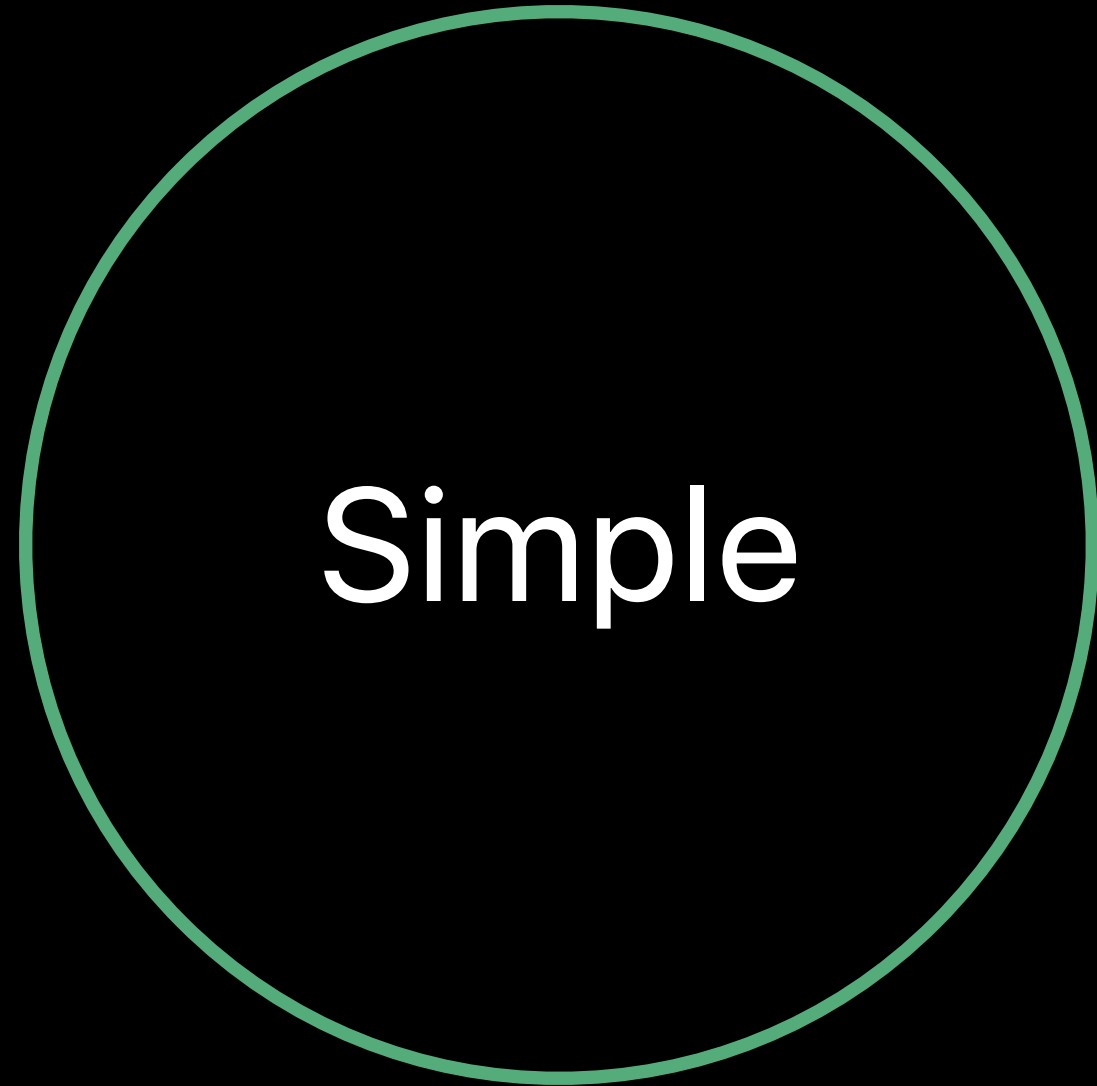


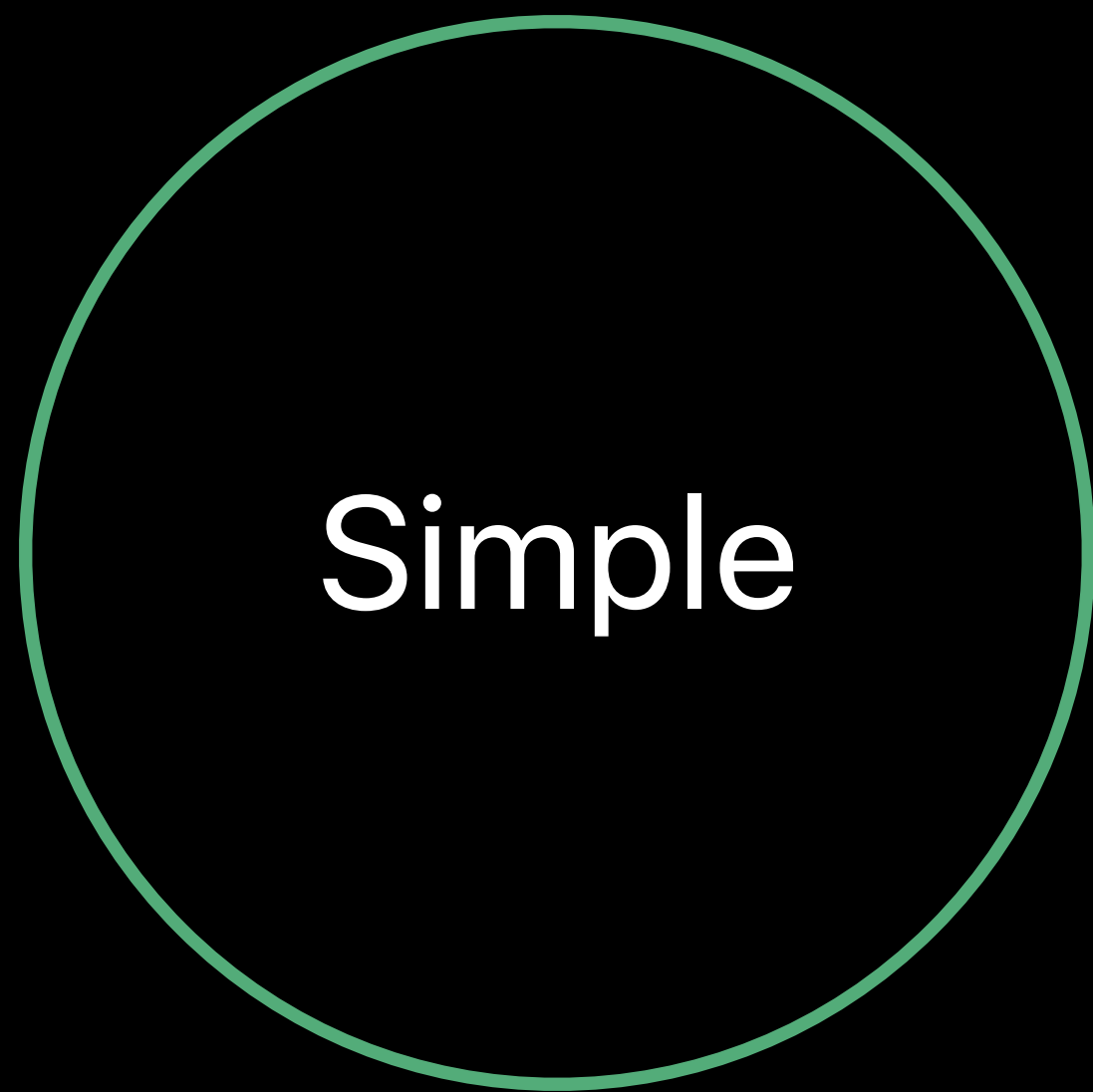
Focus on the **experience**
you are trying to enable



Simple







Unified inference API
Xcode integration



Performant

Fine tuned inference engines

Built on Accelerate and Metal



Compatible

Public model format

Support for popular training libraries

Overview

Models

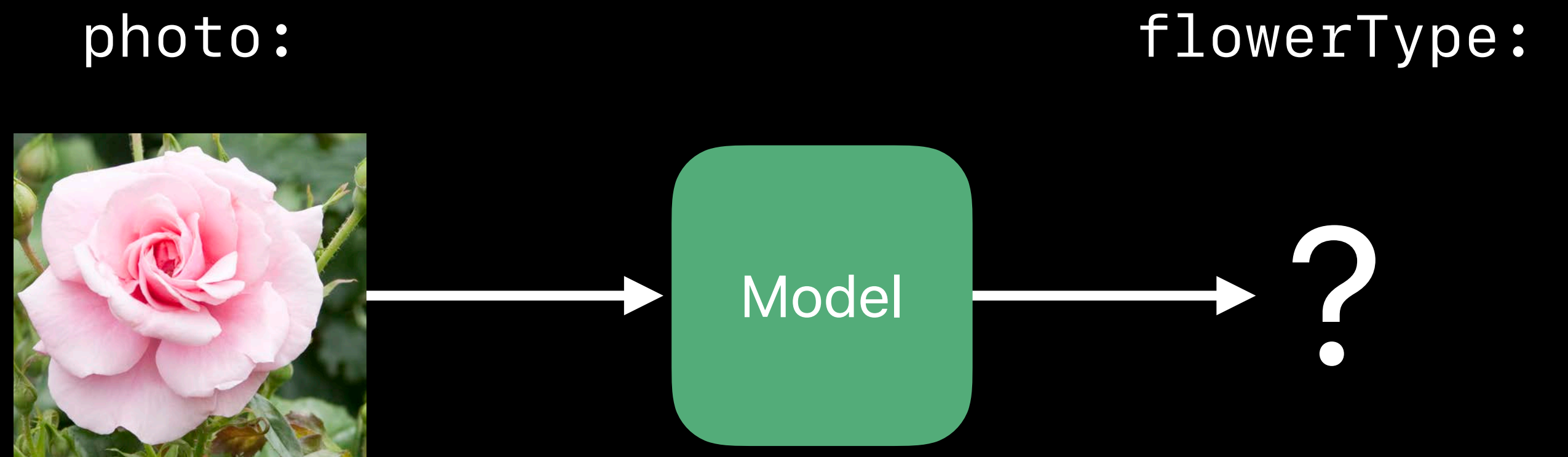
Development Flow

Model

Function learned from data

Observed inputs

Predicts outputs



Underlying Function

Sentiment Analysis

Handwriting Recognition

Translation

Scene Classification

Style Transfer

Music Tagging

Predicting Text

Underlying Function

Sentiment Analysis

That was totally awesome Leo! → 😊

Handwriting Recognition

7 → 7

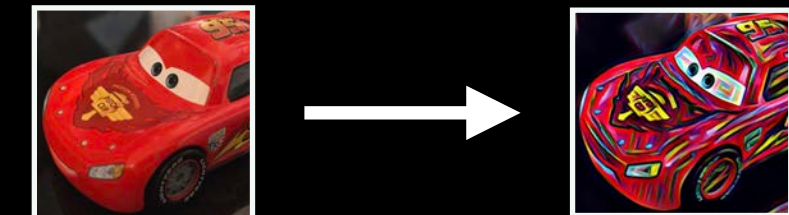
Translation

I love you mom → 사랑해 엄마

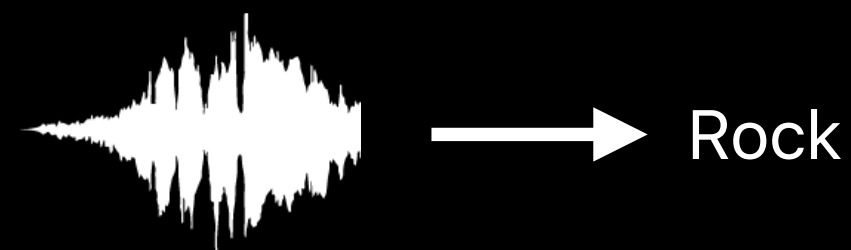
Scene Classification



Style Transfer



Music Tagging



Predicting Text

Do you know the way to → San Jose

Model Types

Sentiment Analysis

Handwriting Recognition

Translation

Scene Classification

Style Transfer

Music Tagging

Predicting Text

Model Types

Sentiment Analysis

Handwriting Recognition

Translation

Scene Classification

Style Transfer

Music Tagging

Predicting Text

Feed Forward
Neural Networks

Convolutional
Neural Networks

Recurrent
Neural Networks

Tree Ensembles

Support Vector Machines

Generalized Linear Models

Focus on Use Cases

Sentiment Analysis

Handwriting Recognition

Translation

Scene Classification

Style Transfer

Music Tagging

Predicting Text



Core ML Model

Single document

Public format



Where do models come from?

Sample Models

<https://developer.apple.com/machine-learning>

Core ML models

Ready to use

Task specific

Explore!

Places205-GoogLeNet

Detects the scene of an image from 205 categories such as an airport terminal, bedroom, forest, coast, and more.

[View original model details >](#)

 [Download Core ML Model](#)

File size: 24.8 MB

ResNet50

Detects the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, vehicles, people, and more.

[View original model details >](#)

 [Download Core ML Model](#)

File size: 102.6 MB

Tap Into ML Community

Thriving communities

Popular ML libraries

Many models

Caffe

dmlc
XGBoost

turi 

 Keras





Convert to Core ML

Core ML Tools



Convert to Core ML

Caffe

K Keras

dmlc
XGBoost

 *scikit*
learn →

turi 

LIBSVM



Convert to Core ML

Caffe

K Keras

dmlc
XGBoost

 scikit
learn

turi 

LIBSVM



Convert to Core ML

Caffe

K Keras

dmlc
XGBoost

 scikit
learn

turi 

LIBSVM



Open Source

Convert to Core ML

Caffe

K Keras

dmlc
XGBoost

scikit
learn

turi 

LIBSVM



Overview

Models

Development Flow

Model as Code

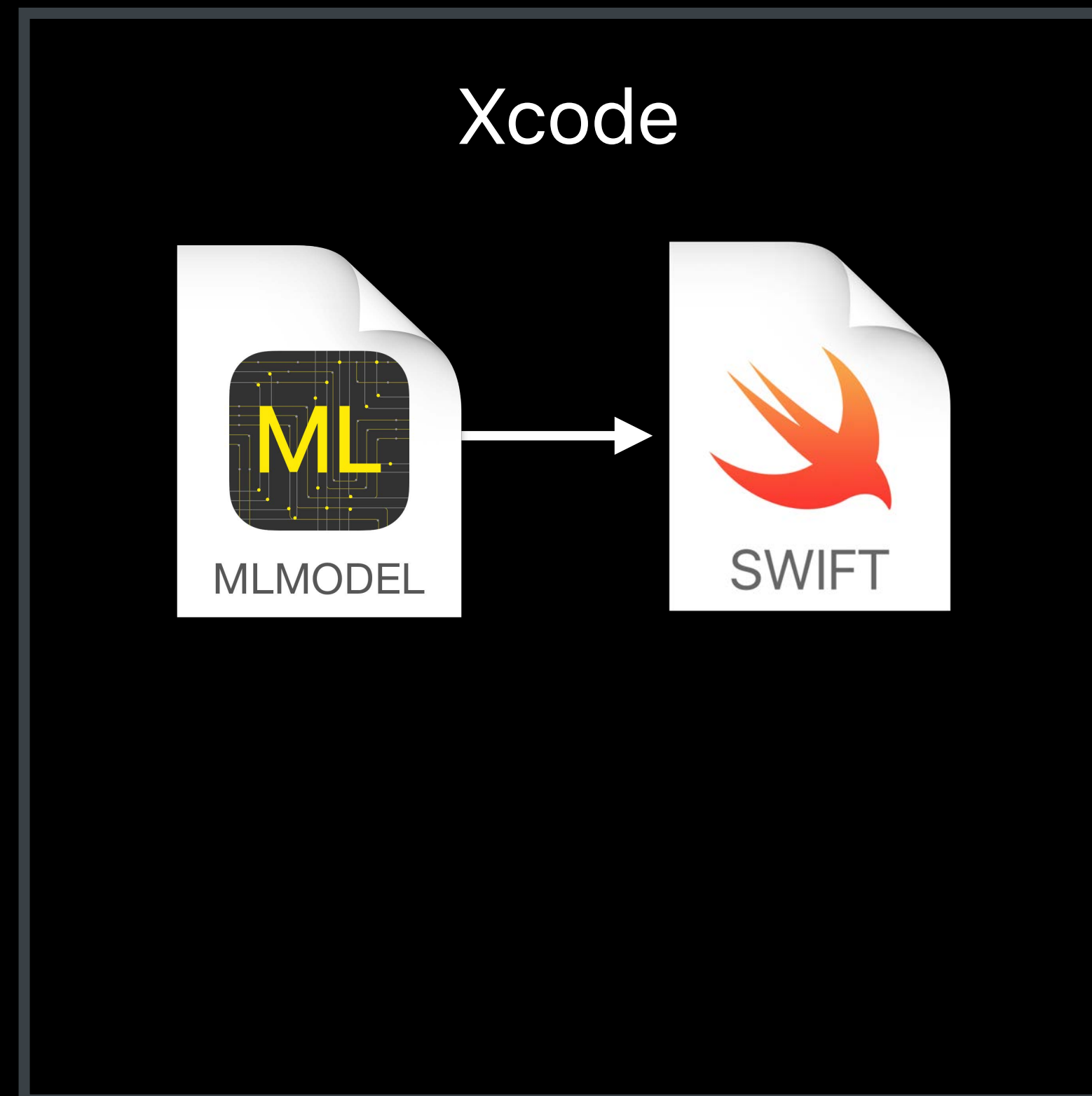


Model as Code

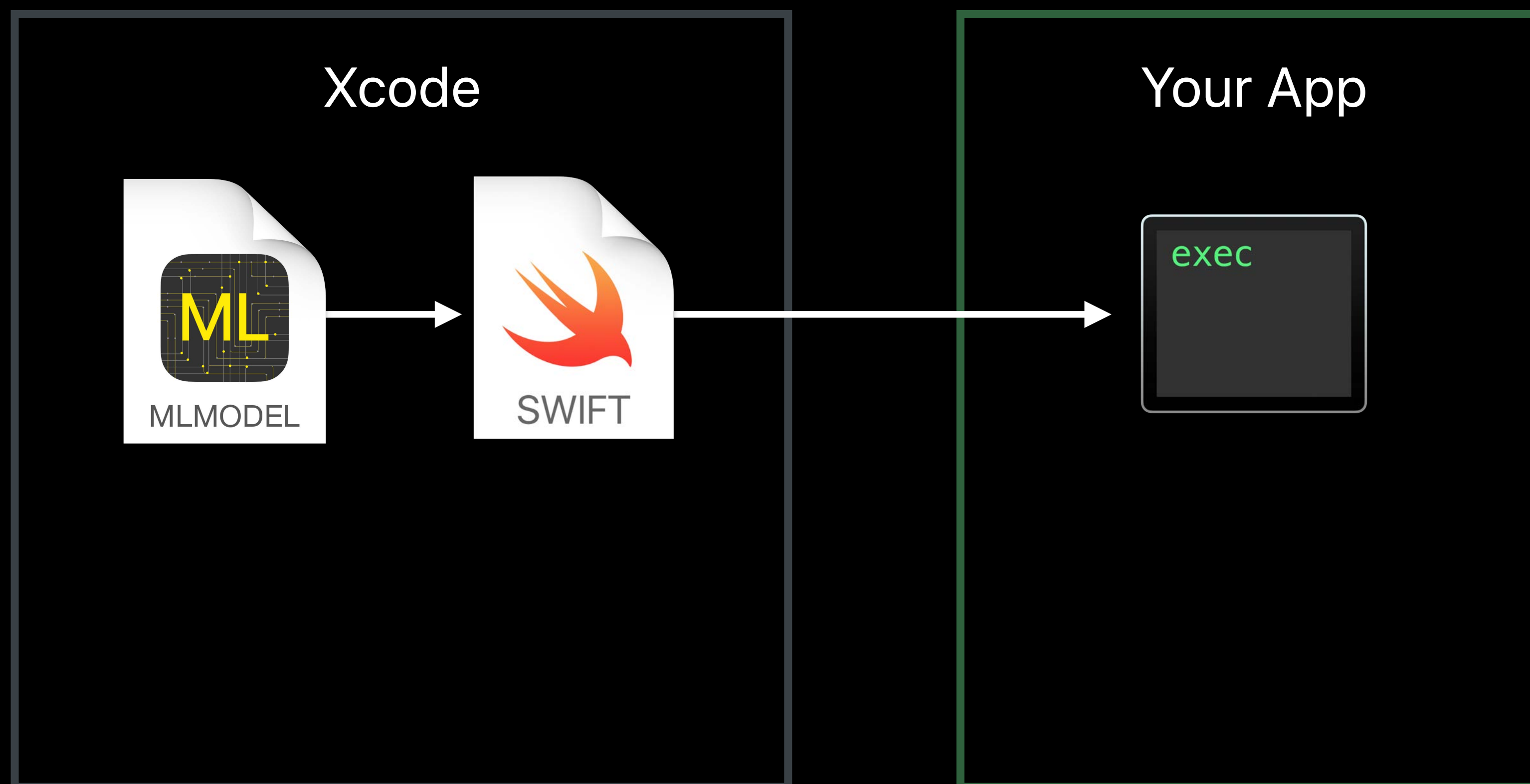
Xcode



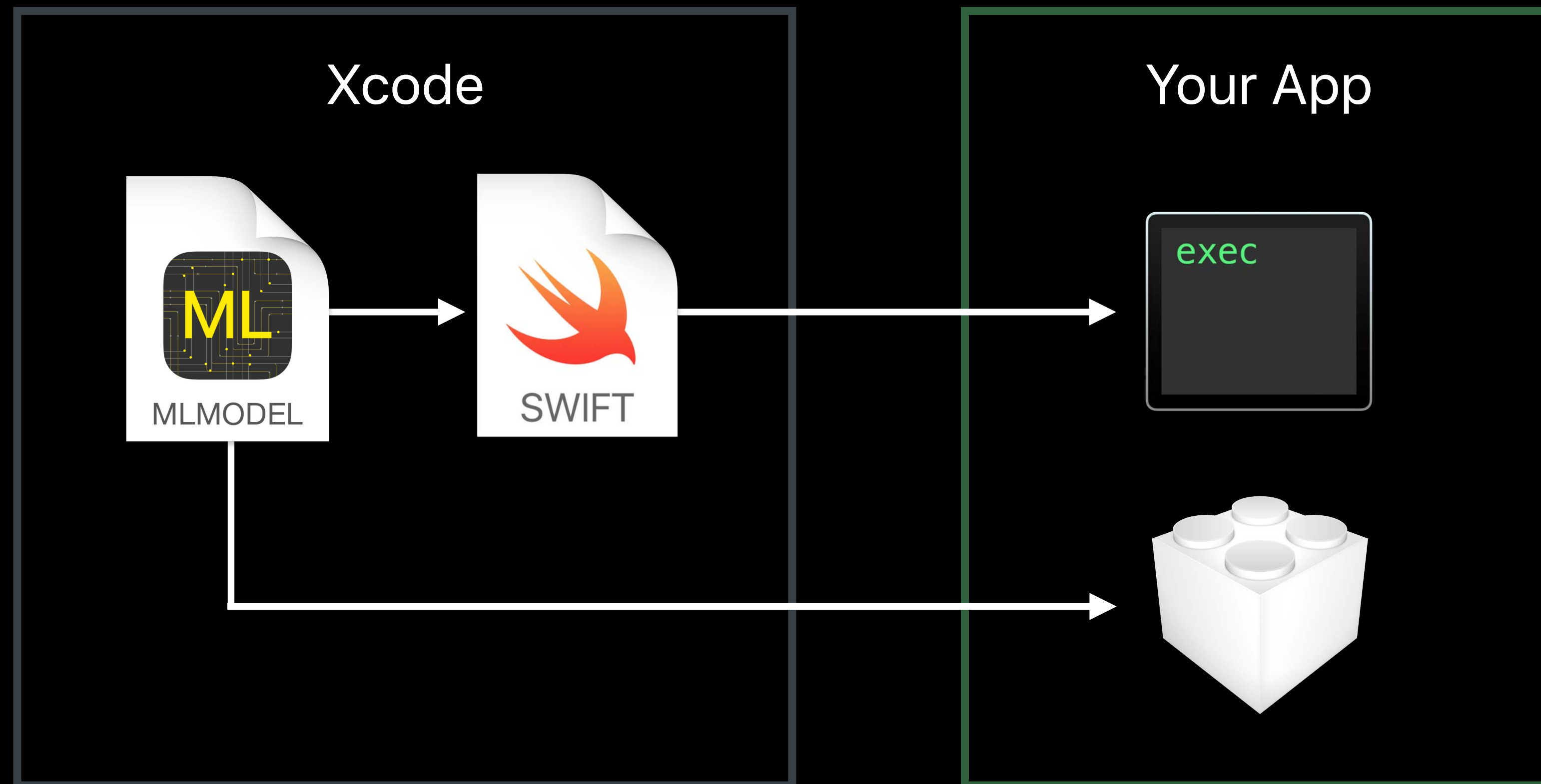
Model as Code



Model as Code

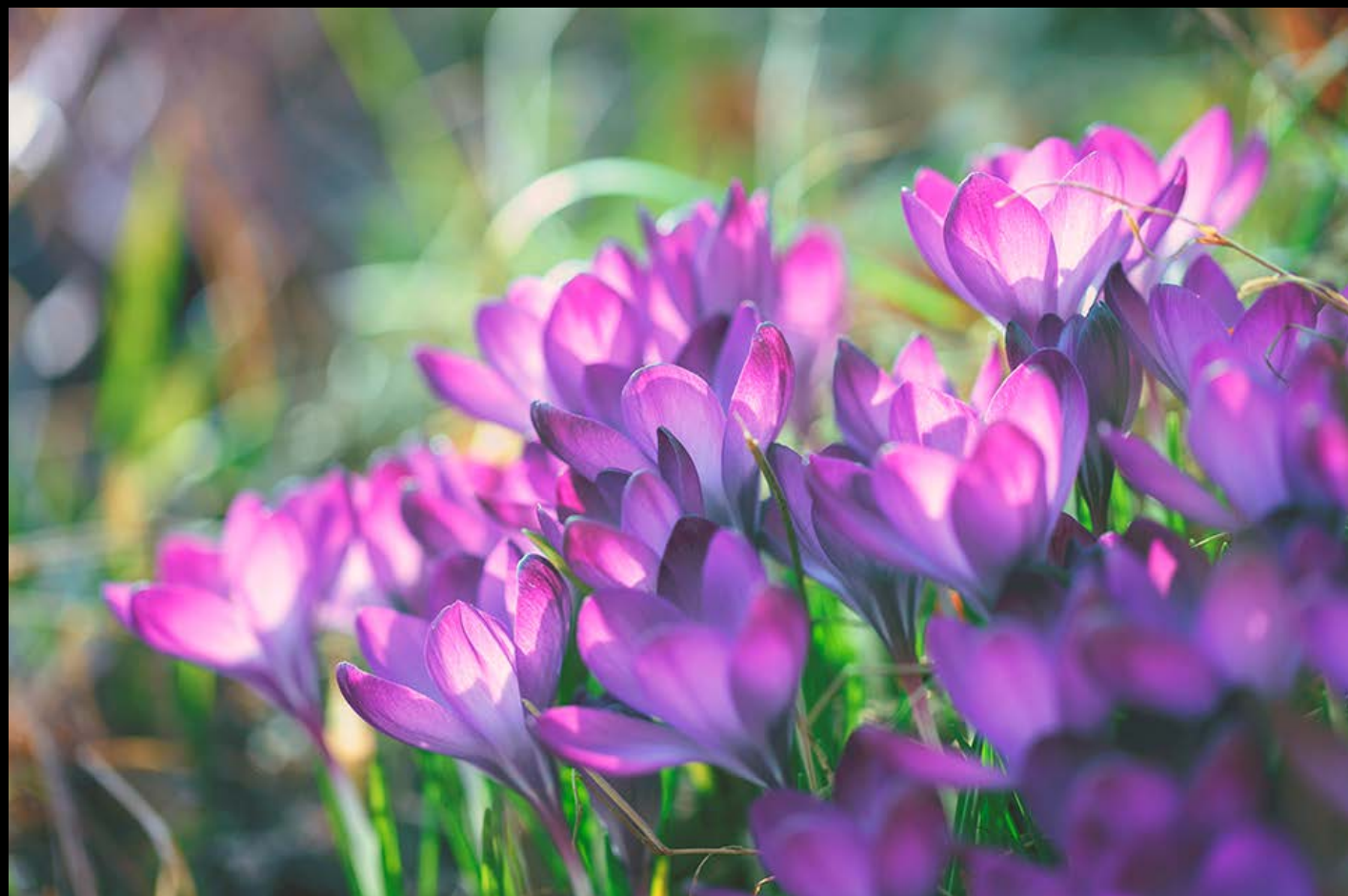


Model as Code



Development Flow

Lizi Ottens, Core ML





••••• 9:41 AM 100%

[← Back](#) Hello Flowers



rose

Getting the Model

Getting the Model



Getting the Model



+

Caffe

dmlc
XGBoost

turi 

K Keras

 scikit
learn

LIBSVM

Getting the Model



+

Caffe

dmlc
XGBoost

turi 

K Keras

 scikit
learn

LIBSVM

Convert





[Back](#)

Hello Flowers



rose

Demo

Image based flower identifier

Demo Recap

Image based flower identifier

Demo Recap

Xcode integration

▼ Machine Learning Model

Name	FlowerClassifier
Type	Neural Network Classifier
Size	41.6 MB
Author	Lizi Ottens
License	MIT
Description	Identify the type of flower present in an image.

▼ Model Class

 FlowerClassifier (Swift generated source) 

▼ Model Evaluation Parameters

Name	Type	Description
▼ inputs		
flowerImage	Image<RGB,227,227>	Input image of a flower
▼ outputs		
flowerType	String	Most likely flower type in image
flowerTypeProbs	Dictionary<String,Double>	Probability of each flower type


Demo Recap

Xcode integration

▼ Machine Learning Model

Name	FlowerClassifier
Type	Neural Network Classifier
Size	41.6 MB
Author	Lizi Ottens
License	MIT
Description	Identify the type of flower present in an image.

▼ Model Class

 FlowerClassifier (Swift generated source) 

▼ Model Evaluation Parameters

Name	Type	Description
▼ inputs		
flowerImage	Image<RGB,227,227>	Input image of a flower
▼ outputs		
flowerType	String	Most likely flower type in image
flowerTypeProbs	Dictionary<String,Double>	Probability of each flower type



Demo Recap

Xcode integration

▼ Machine Learning Model

Name	FlowerClassifier
Type	Neural Network Classifier
Size	41.6 MB
Author	Lizi Ottens
License	MIT
Description	Identify the type of flower present in an image.

▼ Model Class

 FlowerClassifier (Swift generated source) 

▼ Model Evaluation Parameters

Name	Type	Description
▼ inputs		
flowerImage	Image<RGB,227,227>	Input image of a flower
▼ outputs		
flowerType	String	Most likely flower type in image
flowerTypeProbs	Dictionary<String,Double>	Probability of each flower type

Demo Recap

Xcode integration

▼ Machine Learning Model

Name	FlowerClassifier
Type	Neural Network Classifier
Size	41.6 MB
Author	Lizi Ottens
License	MIT
Description	Identify the type of flower present in an image.

▼ Model Class

 FlowerClassifier (Swift generated source) 

▼ Model Evaluation Parameters

Name	Type	Description
▼ inputs		
flowerImage	Image<RGB,227,227>	Input image of a flower
▼ outputs		
flowerType	String	Most likely flower type in image
flowerTypeProbs	Dictionary<String,Double>	Probability of each flower type

Demo Recap

Simple usage

```
let flowerModel = FlowerClassifier()
if let prediction = try? flowerModel.prediction(flowerImage: image) {
    return prediction.flowerType
}
```

Demo Recap

Simple usage

```
let flowerModel = FlowerClassifier()
if let prediction = try? flowerModel.prediction(flowerImage: image) {
    return prediction.flowerType
}
```

Type of model abstracted

Demo Recap

Simple usage

```
let flowerModel = FlowerClassifier()
if let prediction = try? flowerModel.prediction(flowerImage: image) {
    return prediction.flowerType
}
```

Type of model abstracted

Input/output strongly typed

Generated Source

```
class FlowerClassifierInput {
    var flowerImage: CVPixelBuffer
}

class FlowerClassifierOutput {
    let flowerType: String
    let flowerTypeProbs: [String: Double]
}

class FlowerClassifier {
    convenience init()
    func prediction(flowerImage: CVPixelBuffer) throws -> FlowerClassifierOutput
}
```

Generated Source

```
class FlowerClassifierInput {  
    var flowerImage: CVPixelBuffer  
}
```

```
class FlowerClassifierOutput {  
    let flowerType: String  
    let flowerTypeProbs: [String: Double]  
}
```

```
class FlowerClassifier {  
    convenience init()  
    func prediction(flowerImage: CVPixelBuffer) throws -> FlowerClassifierOutput  
}
```

Generated Source

```
class FlowerClassifierInput {  
    var flowerImage: CVPixelBuffer  
}
```

```
class FlowerClassifierOutput {  
    let flowerType: String  
    let flowerTypeProbs: [String: Double]  
}
```

```
class FlowerClassifier {  
    convenience init()  
    func prediction(flowerImage: CVPixelBuffer) throws -> FlowerClassifierOutput  
}
```

Generated Source

```
class FlowerClassifierInput {  
    var flowerImage: CVPixelBuffer  
}
```

```
class FlowerClassifierOutput {  
    let flowerType: String  
    let flowerTypeProbs: [String: Double]  
}
```

```
class FlowerClassifier {  
    convenience init()  
    func prediction(flowerImage: CVPixelBuffer) throws -> FlowerClassifierOutput  
}
```

More Advanced

Underlying API

```
class FlowerClassifier {  
    convenience init()  
    func prediction(flowerImage: CVPixelBuffer) throws -> FlowerClassifierOutput  
}
```


More Advanced

Underlying API

```
class FlowerClassifier {  
    convenience init()  
  
    func prediction(flowerImage: CVPixelBuffer) throws -> FlowerClassifierOutput  
}
```

More Advanced

Underlying API

```
class FlowerClassifier {  
    convenience init()  
    let model: MLModel  
    func prediction(flowerImage: CVPixelBuffer) throws -> FlowerClassifierOutput  
}
```

Programmatic access to model for power users

MLModel

```
class MLModel {  
    var modelDescription: MLModelDescription  
    func prediction(from input: MLFeatureProvider) throws -> MLFeatureProvider  
}
```

MLModel

```
class MLModel {  
    var modelDescription: MLModelDescription  
    func prediction(from input: MLFeatureProvider) throws -> MLFeatureProvider  
}
```

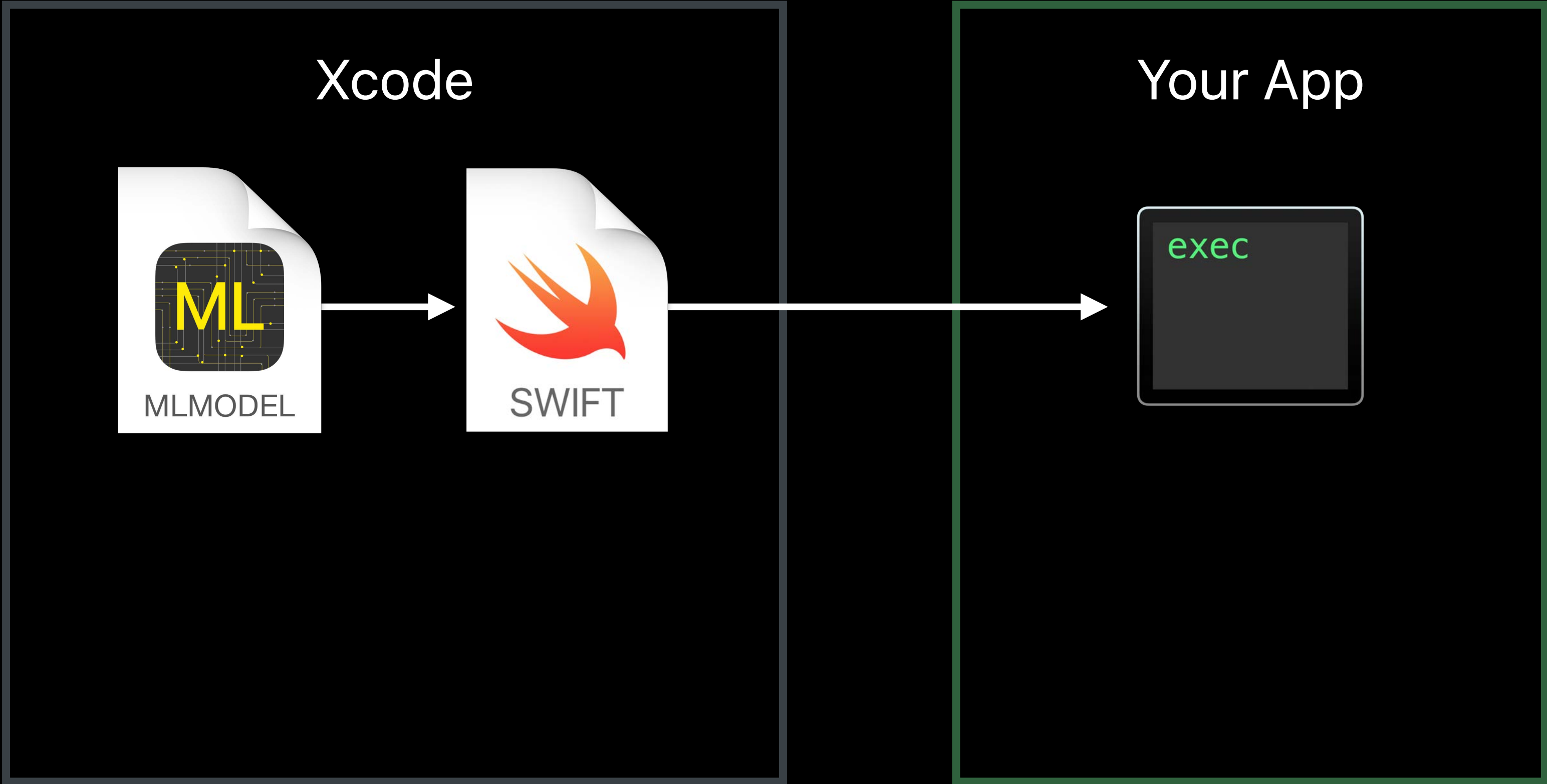
Access to model description

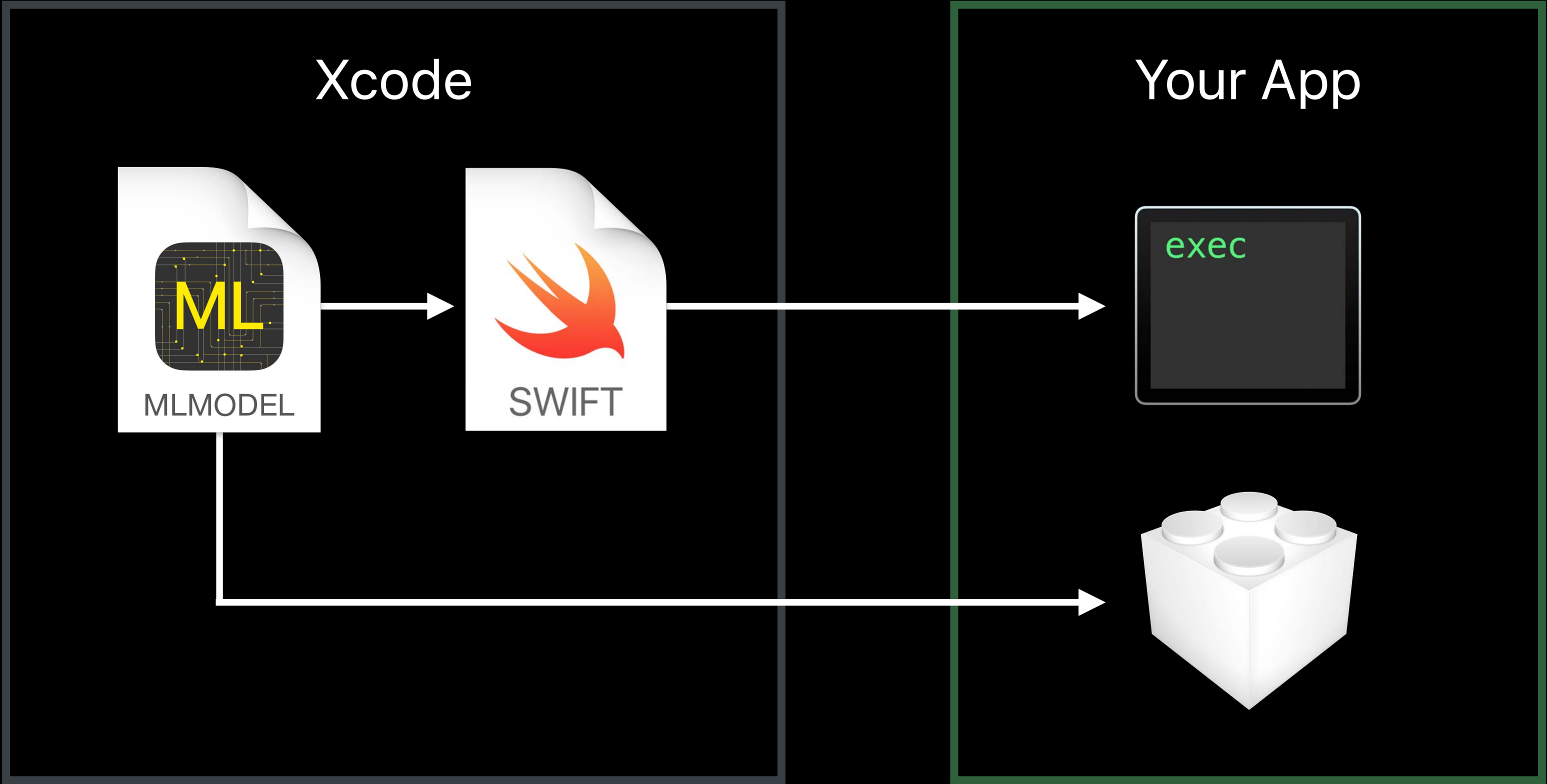
MLModel

```
class MLModel {  
    var modelDescription: MLModelDescription  
    func prediction(from input: MLFeatureProvider) throws -> MLFeatureProvider  
}
```

Access to model description

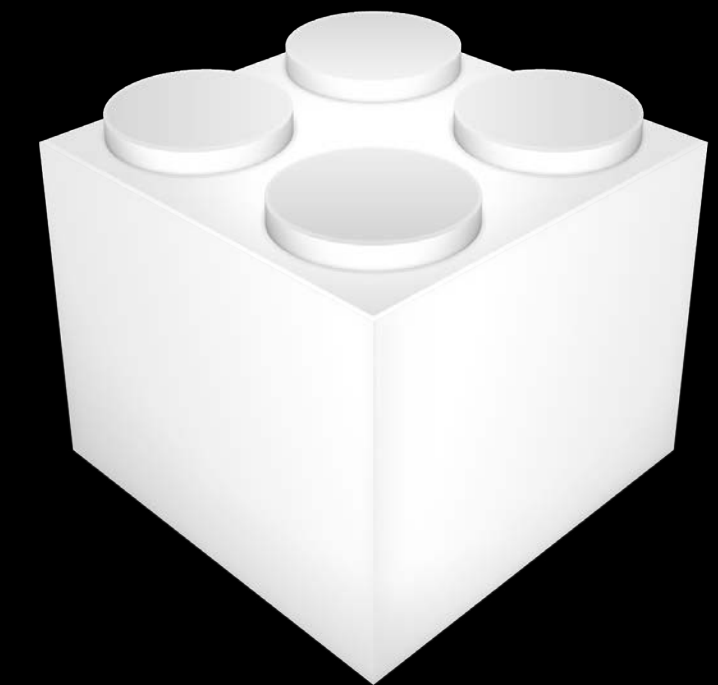
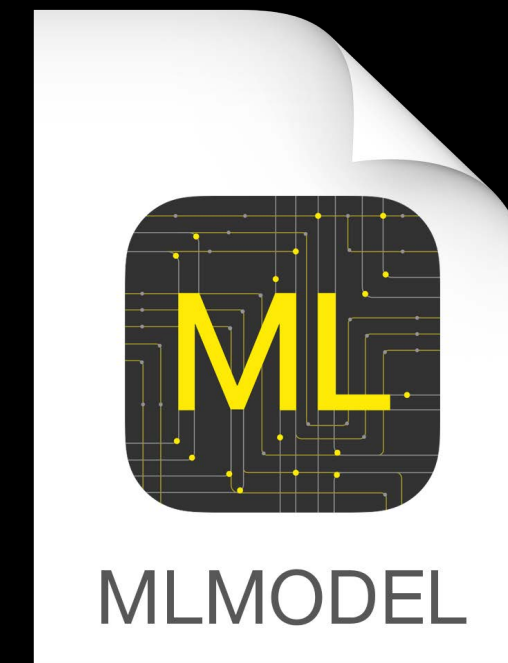
Flexibility in how input is provided





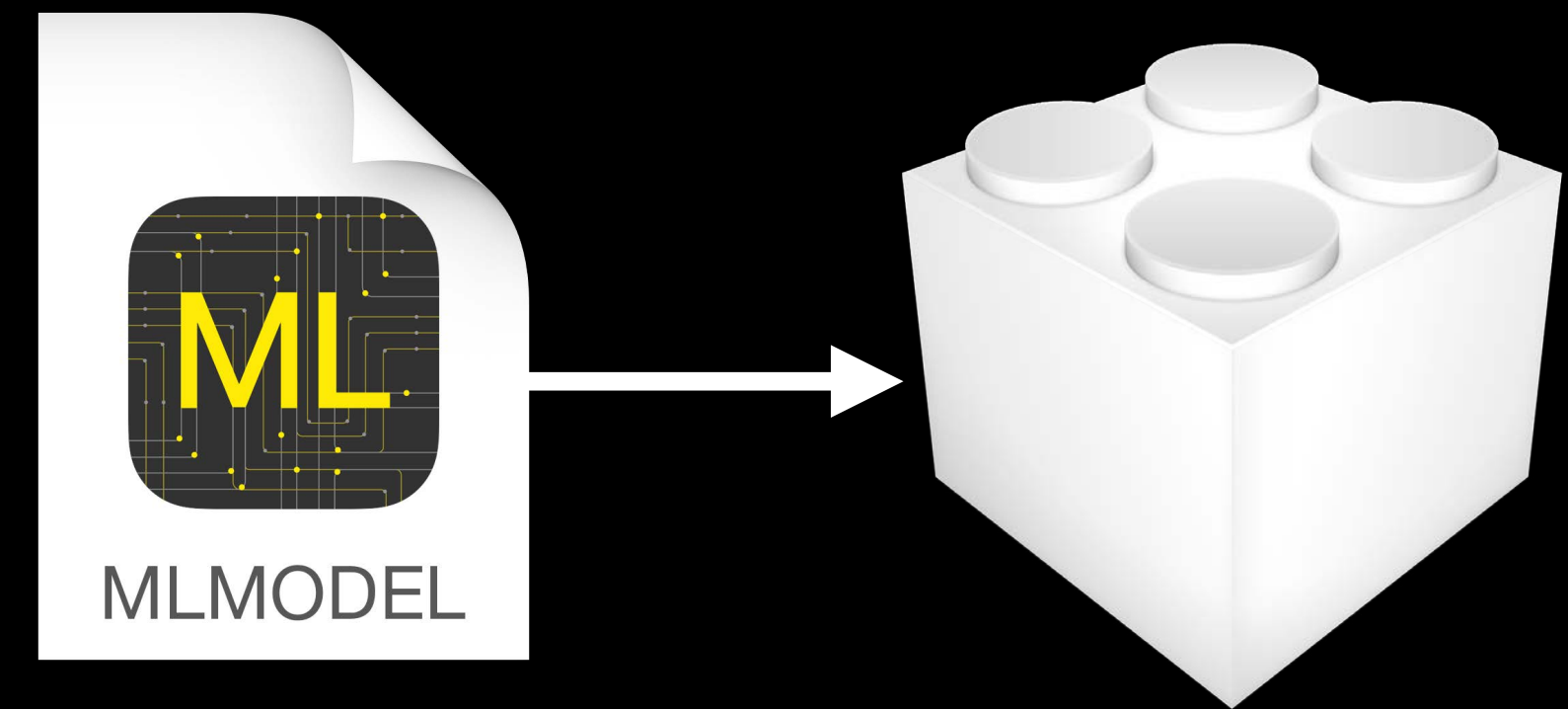
Behind the Scenes

Model compilation



Behind the Scenes

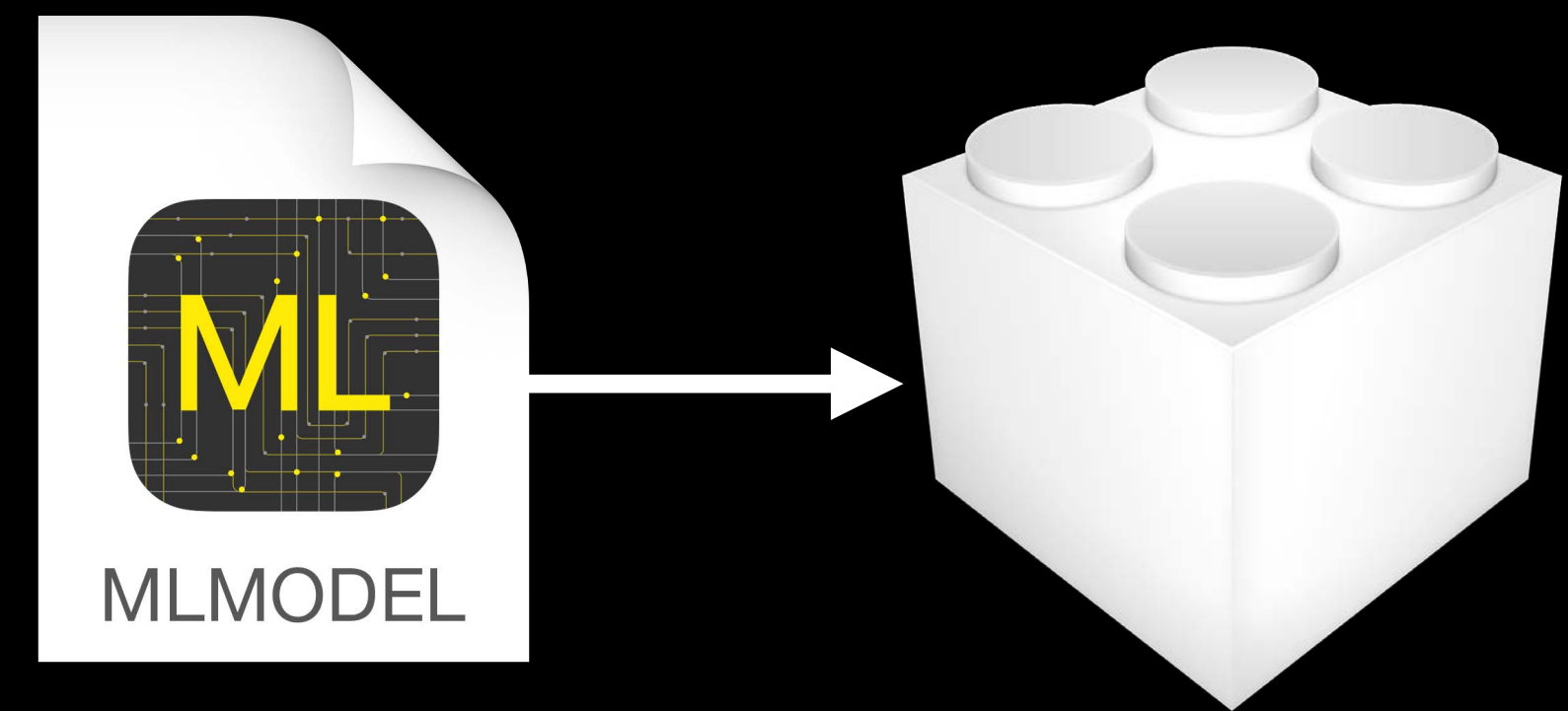
Model compilation



Behind the Scenes

Model compilation

Quick initialization

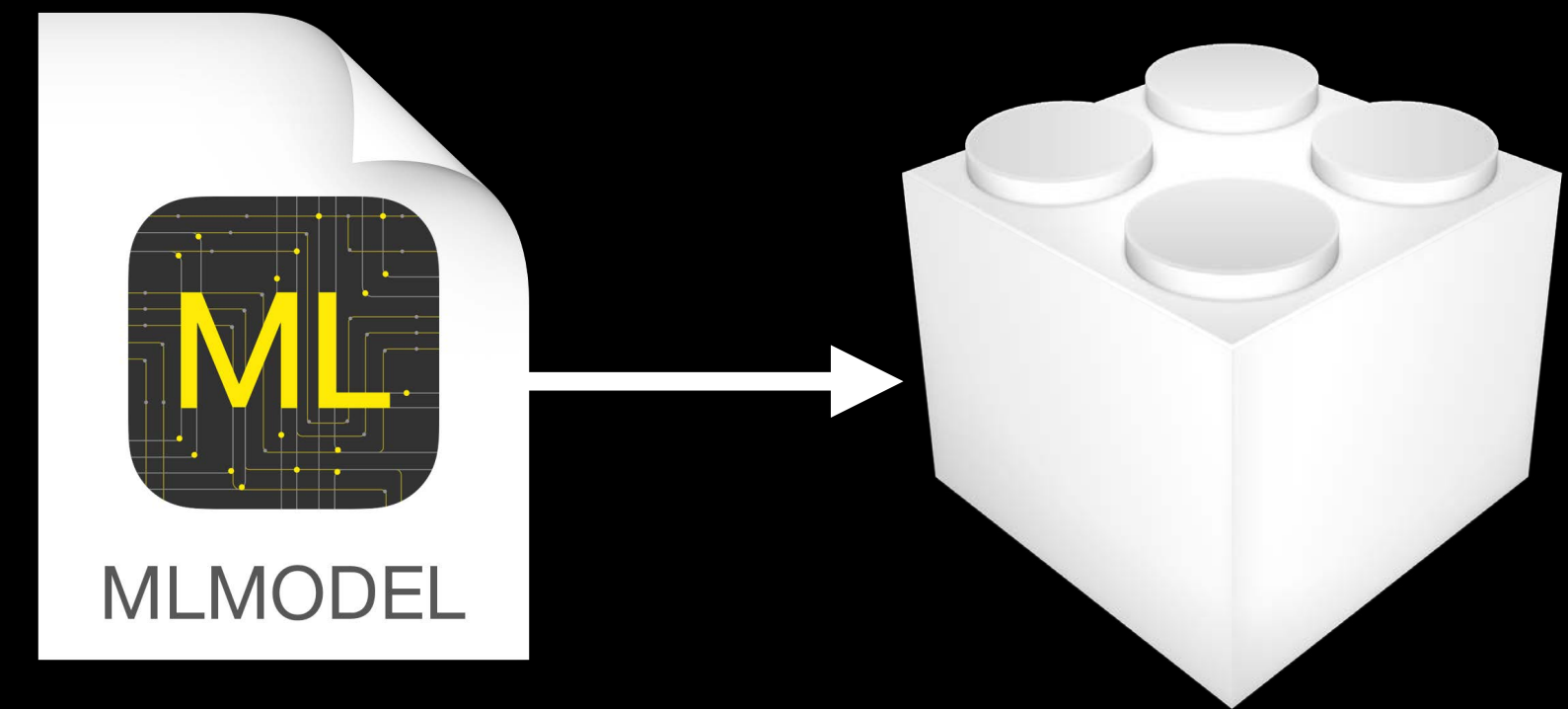


Behind the Scenes

Model compilation

Quick initialization

Optimized prediction



Model Goals



Model Goals

Reduce size



Model Goals

Reduce size

Improve accuracy



Model Goals

Reduce size

Improve accuracy

Decrease prediction times



Summary

Summary

Machine learning frameworks

Summary

Machine learning frameworks

Core ML

Summary

Machine learning frameworks

Core ML

Development flow in action

More Information

<https://developer.apple.com/wwdc17/703>

Related Sessions

[Natural Language Processing and your Apps](#)

Hall 3

Wednesday 9:00AM

[Vision Framework: Building on Core ML](#)

Hall 2

Wednesday 3:10PM

[Core ML in depth](#)

Hall 3

Thursday 9:00AM

[Accelerate and Sparse Solvers](#)

Executive Ballroom

Thursday 10:00AM

[Using Metal 2 for Compute](#)

Grand Ballroom A

Thursday 4:10PM

Labs

Core ML and Natural Language Processing Lab

Technology Lab D

Thu 11:00AM-3:30PM

Core ML & Natural Language Processing Lab

Technology Lab D

Fri 1:50AM-4:00PM

