Framework

Core ML

Integrate machine learning models into your app.

iOS 11.0+ | iPadOS 11.0+ | Mac Catalyst 13.0+ | macOS 10.13+ | tvOS 11.0+ | visionOS 1.0+ | watchOS 4.0+

Overview

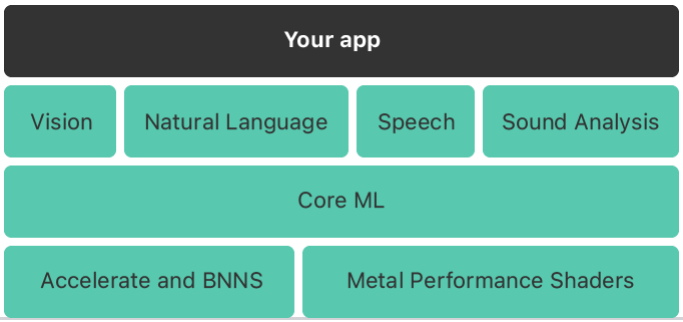
Use Core ML to integrate machine learning models into your app. Core ML provides a unified representation for all models. Your app uses Core ML APIs and user data to make predictions, and to train or fine-tune models, all on a person's device.

A model is the result of applying a machine learning algorithm to a set of training data. You use a model to make predictions based on new input data. Models can accomplish a wide variety of tasks that would be difficult or impractical to write in code. For example, you can train a model to categorize photos, or detect specific objects within a photo directly from its pixels.

You build and train a model with the Create ML app bundled with Xcode. Models trained using Create ML are in the Core ML model format and are ready to use in your app. Alternatively, you can use a wide variety of other machine learning libraries and then use Core ML Tools to convert the model into the Core ML format. Once a model is on a person's device, you can use Core ML to retrain or fine-tune it on-device, with that person's data.

Core ML optimizes on-device performance by leveraging the CPU, GPU, and Neural Engine while minimizing its memory footprint and power consumption. Running a model strictly on a person's device removes any need for a network connection, which helps keep a person's data private and your app responsive.

The framework is the foundation for domain-specific frameworks and functionality. It supports Vision for analyzing images, Natural Language for processing text, Speech for converting audio to text, and Sound Analysis for identifying sounds in audio. Core ML itself builds on top of low-level primitives like Accelerate and BNNS, as well as Metal Performance Shaders.



Topics

Core ML models

Getting a Core ML Model

Obtain a Core ML model to use in your app.

Updating a Model File to a Model Package

Convert a Core ML model file into a model package in Xcode.

{} Integrating a Core ML Model into Your App

Add a simple model to an app, pass input data to the model, and process the model's predictions.

`class MLModel`

An encapsulation of all the details of your machine learning model.

Model Customization

Expand and modify your model with new layers.

Model Personalization

Update your model to adapt to new data.

Model inputs and outputs

Making Predictions with a Sequence of Inputs

Integrate a recurrent neural network model to process sequences of inputs.

`class` `MLFeatureValue`

A generic wrapper around an underlying value and the value's type.

`struct` `MLSendableFeatureValue`

A sendable feature value.

`protocol` `MLFeatureProvider`

An interface that represents a collection of values for either a model's input or its output.

`class` `MLDictionaryFeatureProvider`

A convenience wrapper for the given dictionary of data.

`protocol` `MLBatchProvider`

An interface that represents a collection of feature providers.

`class` `MLArrayBatchProvider`

A convenience wrapper for batches of feature providers.

`class` `MLModelAsset`

An abstraction of a compiled Core ML model asset.

App integration

Downloading and Compiling a Model on the User's Device

Install Core ML models on the user's device dynamically at runtime.

Model Integration Samples

Integrate tabular, image, and text classifcation models into your app.

Model encryption

Generating a Model Encryption Key

Create a model encryption key to encrypt a compiled model or model archive.

Encrypting a Model in Your App

Encrypt your app's built-in model at compile time by adding a compiler flag.

Compute devices

`enum` `MLComputeDevice`

Compute devices for framework operations.

`class MLCPUComputeDevice`

An object that represents a CPU compute device.

`class MLGPUComputeDevice`

An object that represents a GPU compute device.

`class MLNeuralEngineComputeDevice`

An object that represents a Neural Engine compute device.

`protocol MLComputeDeviceProtocol`

An interface that represents a compute device type.

Compute plan

`class MLComputePlan`

A class representing the compute plan of a model.

`enum MLModelStructure`

An enum representing the structure of a model.

`struct MLComputePolicy`

The compute policy determining what compute device, or compute devices, to execute ML workloads on.

`func withMLTensorComputePolicy<R>(MLComputePolicy, () async throws -> R) async rethrows -> R`

Calls the given closure within a task-local context using the specified compute policy to influence what compute device tensor operations are executed on.

`func withMLTensorComputePolicy<Result>(MLComputePolicy, () throws -> Result) rethrows -> Result`

Calls the given closure within a task-local context using the specified compute policy to influence what compute device tensor operations are executed on.

Model state

`class MLState`

Handle to the state buffers.

`class MLStateConstraint`

Constraint of a state feature value.

Model tensor

`struct MLTensor`

A multi-dimensional array of numerical or Boolean scalars tailored to ML use cases, containing methods to perform transformations and mathematical operations efficiently using a ML compute device.

`protocol MLTensorScalar`

A type that represents the tensor scalar types supported by the framework. Don't use this type directly.

`protocol MLTensorRangeExpression`

A type that can be used to slice a dimension of a tensor. Don't use this type directly.

`func pointwiseMin(_:_:)`

Computes the element-wise minimum of two tensors.

`func pointwiseMax(_:_:)`

Computes the element-wise minimum between two tensors.

`func withMLTensorComputePolicy(_:_:)`

Calls the given closure within a task-local context using the specified compute policy to influence what compute device tensor operations are executed on.

Model structure

`enum MLModelStructure`

An enum representing the structure of a model.

Model errors

`struct MLModelError`

Information about a Core ML model error.

`enum Code`

Information about a Core ML model error.

```
let MLModelErrorDomain: String
```

The domain for Core ML errors.

Model deployments

~~`class MLModelCollection`~~

A set of Core ML models from a model deployment.

Deprecated

Reference

CoreML Enumerations