Machine learning models

Create, optimize, and deploy models for on-device execution.

Overview

When the available intelligent frameworks or generative technologies don't provide the features you need, Apple provides machine learning frameworks that help you:

- Create models from your own training data.
- Run generative models, stateful models, and transformer models efficiently.
- Convert models from other training libraries to run on-device.
- Preview your model's behavior from sample data or live inputs.
- Analayze the performance of your model in Xcode and Instruments.

Build models to analyze text, images, or other types of data your app needs. If you already have your own machine learning models, convert them to the Core ML model format and integrate them into your app. Apple also provides frameworks to help with highly demanding machine learning tasks that involve graphics and real-time signal processing.

As you design your models, it's important to keep the intended experience of your app in mind. The HIG offers machine learning guidance and best practices to help you create apps that use machine learning.

Collect and prepare your data for training

When you create a new model the starting point is always the same — your training and testing data. The quality of your data determines the quality of your results, so choose data that reflects a wide variety of possibility for your training use case. For example, when you create an image classification model to recognize animals, begin by gathering at least 10 images per animal that

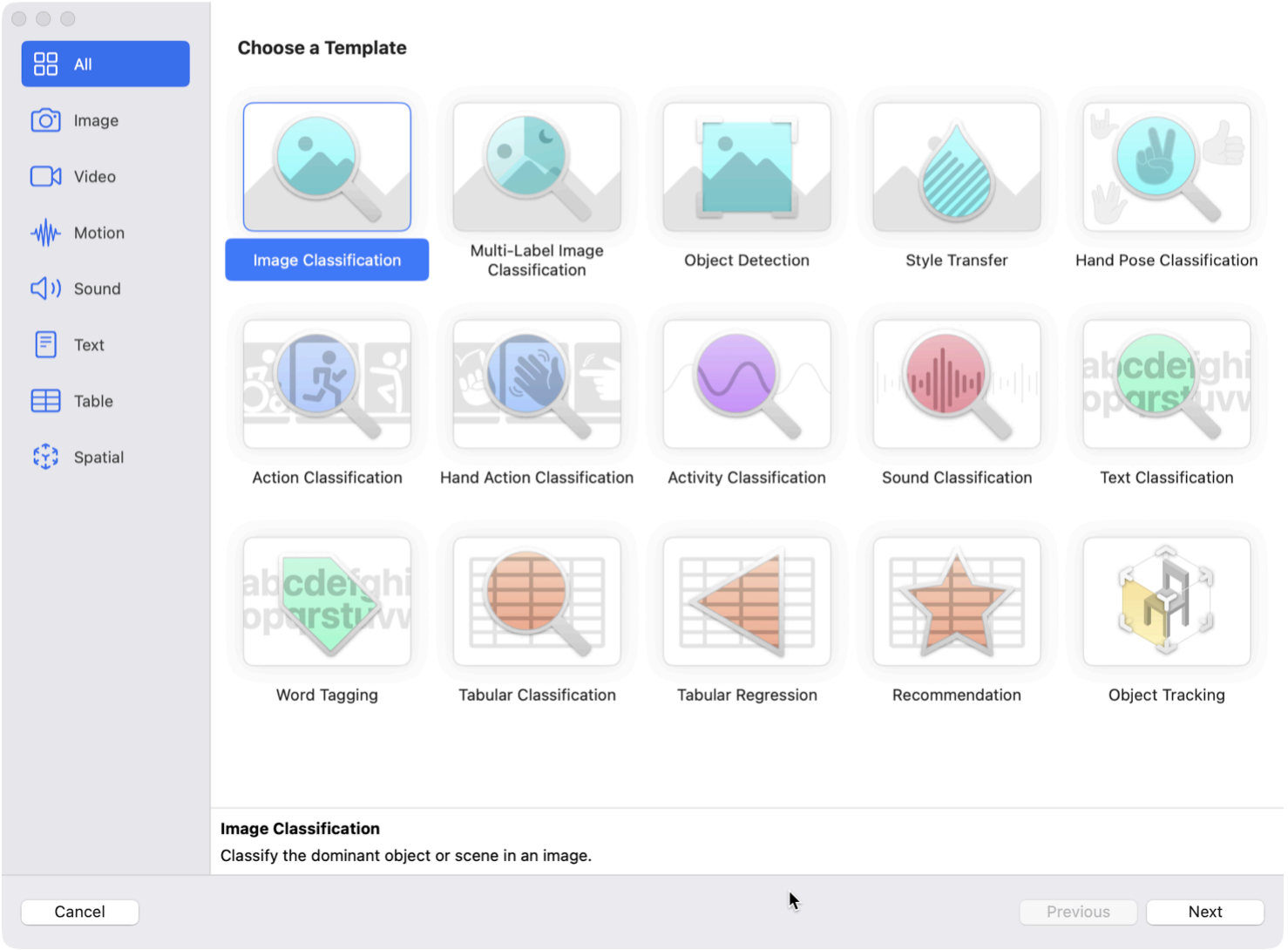
best represent what you expect the model to see. Create ML supports several types of data sources, each with its own arrangement of files within a parent folder. In you parent folder, organize data into subfolders and use the folder name as your training label.

If you use the Create ML framework to programmatically create and train a model — like a text classifier that identifies the sentiment expressed in a sentence — prepare your data for training by using TabularData.

Build and train on-device models with no code

Many system frameworks can be extended or customized to your specific use case. If you're working in a specialized domain that requires using your own data — or you want to extend the capabilities of an existing framework – the Create ML app makes it easier to adapt system models. For example, if you use the Sound Analysis framework and notice it can't classify your sound, use the Create ML app to train a sound classification model that's trained to identify your sound. After you train the model, load it with the Sound Analysis framework.

Built with the Create ML and Create ML Components frameworks, the Create ML app provides an approachable interface for creating models using your own data. In Xcode, choose Xcode > Open Developer Tool > Create ML. Choose a template that aligns with the task you want to customize, then provide your data, train, evaluate, and iterate on your model.



After training your model, use the Create ML app to visualize and debug your annotations by clicking on the data source. The default view shows a distribution of your data, and the Explore page lets you see into specific object or class labels to visualize your annotations. The app provides you with a model that's ready to integrate into your app with Core ML.

Model conversion and optimization

Bring any model to the device if you want to experiment with it or deploy it. All you need is the model to be in the Core ML format. Core ML is the go-to framework for deploying models on-device, and you can download and explore models that are already in the Core ML format to experiment with or use for your feature.

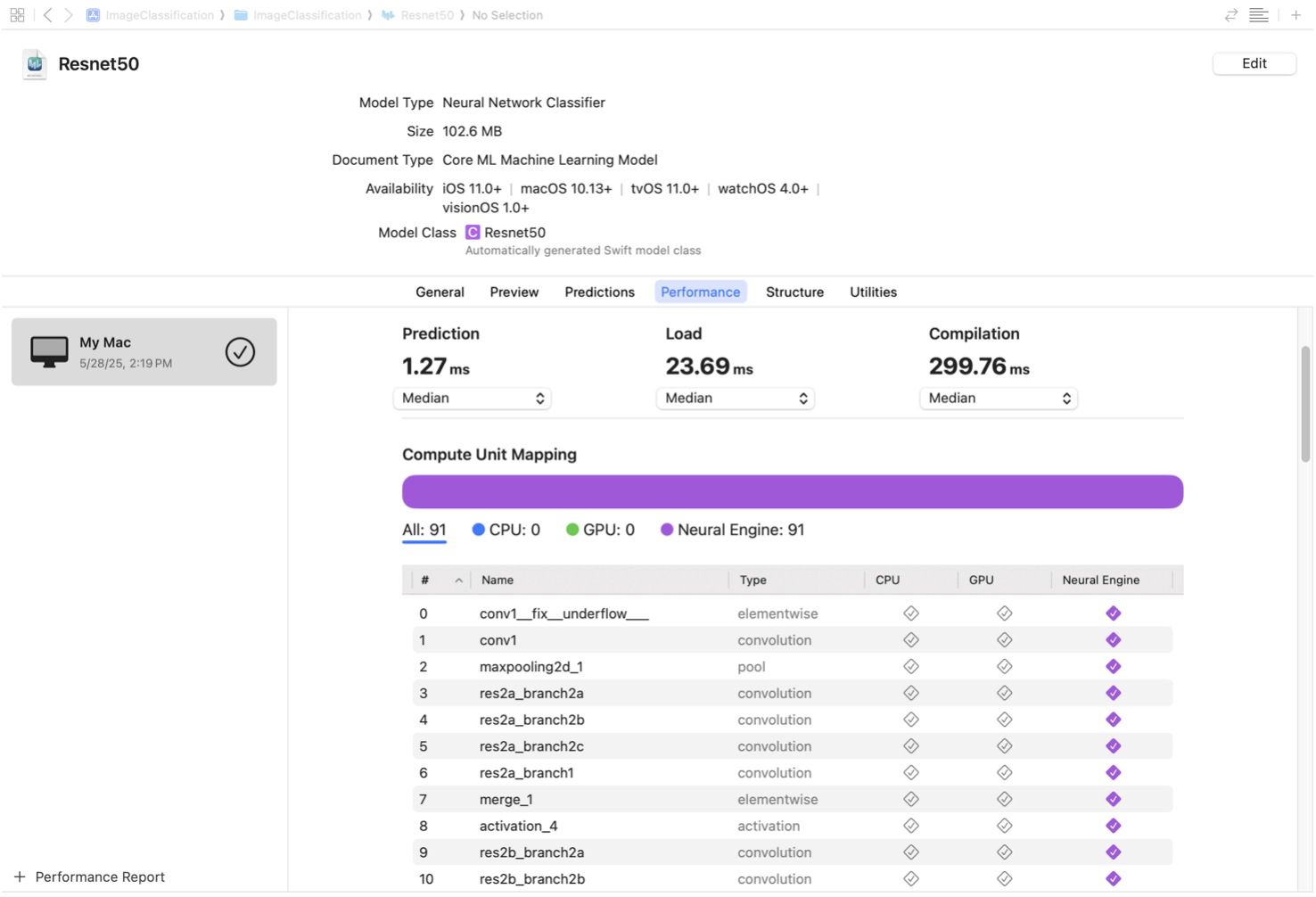
If you created a model using training libraries like MLX or PyTorch, use Core ML Tools to convert it to the Core ML format. Core ML Tools provides utilities and workflows for transforming trained models to the Core ML format. The workflows that Core ML Tools provide apply optimizations for on-device execution. Converting your model optimizes it for Apple devices, which requires less space on device, uses less power, and reduces latency when making predictions. Core ML Tools

provides a number of compression techniques to help you optimize the model representation and parameters, while maintaining good accuracy.

When you have an optimized and prepared model, you're ready to integrate it with system frameworks. For example, if your model performs image analysis, load the model with the Vision framework.

Analyze the performance of your model with Xcode

Evaluating the performance of models is an important task of machine learning. In Xcode, preview your model's behavior by using sample data files or using the device's camera and microphone. Review the performance of your model's predictions directly from Xcode, or profile your app in Instruments to get a thorough performance analysis. After you add a model to your project, select it to get insights about the expected prediction latency, load times, and introspect where a particular operation is supported and run.



To build a deeper understanding of the model you're working with, Xcode allows you to visualize the structure of the full model architecture and dive into the details of any operation. This visualization helps you debug issues and find performance enhancing opportunities.

Model deployment and execution on device

You use Core ML to integrate and run your model directly into your app. At runtime, Core ML makes use of all available compute and optimizes task execution across CPU, GPU, and Neural Engine. There are several technologies that underly Core ML that are available when you need fine-grained control over machine learning task execution.

To sequence and integrate machine learning with demanding graphics workloads, use Core ML models with both Metal Performance Shaders Graph and Metal. MPS Graph enables you to sequence tasks with other workloads, which optimizes GPU utilization. Use MPS Graph to load your Core ML model or programmatically build, compile, and execute computational graphs.

When running real-time signal processing on the CPU, use the BNNS Graph API in Accelerate. BNNS Graph works with Core ML models to enable real-time and latency-sensitive inference on the CPU, along with strict control over memory allocations. Use the Graph Builder to create graphs of operations that allow for writing routines or even small machine learning models to run in real-time on the CPU.