ML and On-device Processing with Apple Frameworks

Introduction to Mobile Development and Machine Learning Basics Using CoreML

Sebastián A. Cruz Romero
University of Puerto Rico at Mayagüez
Department of Computer Science and Engineering
Computer Science and Engineering
sebastian.cruz6@upr.edu

PI: Dr. Wilfredo Lugo Beauchamp, PhD
University of Puerto Rico at Mayagüez
Department of Computer Science and Engineering
Computer Science and Engineering
wilfredo.lugo1@upr.edu

Outline

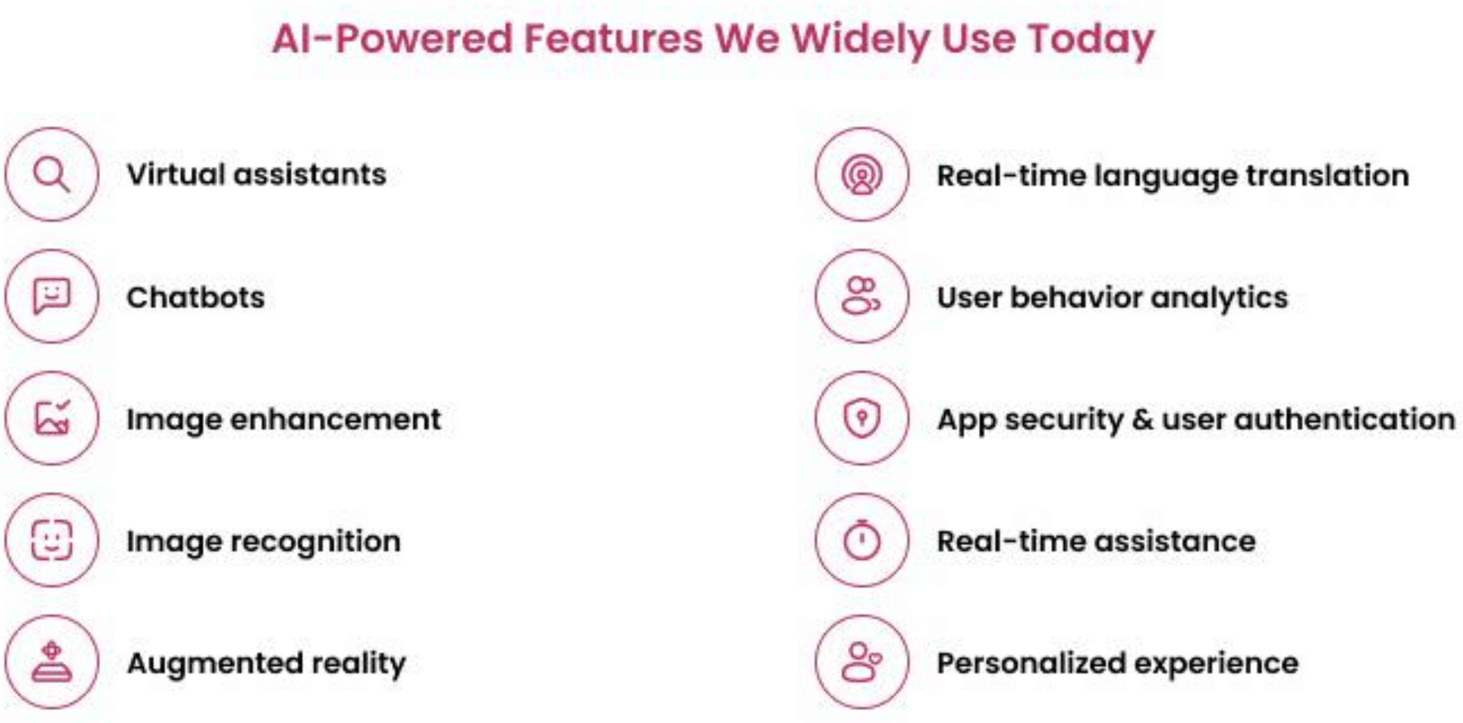
- Introduction
  - What and Why Mobile Development?
  - Understanding Machine Learning
- Apple’s Machine Learning Ecosystem
  - What is CoreML?
  - Key Steps to Using CoreML
  - CoreML for Image Recognition
- Benefits of On-Device Processing
- Challenges in Mobile AI Development
- Future of AI in Mobile Development
- WWDC23: *Use Core ML Tools for machine learning model compression*

Introduction

What is Mobile Development?

- Creating applications for mobile devices
- Key platforms: iOS and Android
- Languages: Swift (iOS) and Kotlin (Android)

Why Mobile Development and AI?

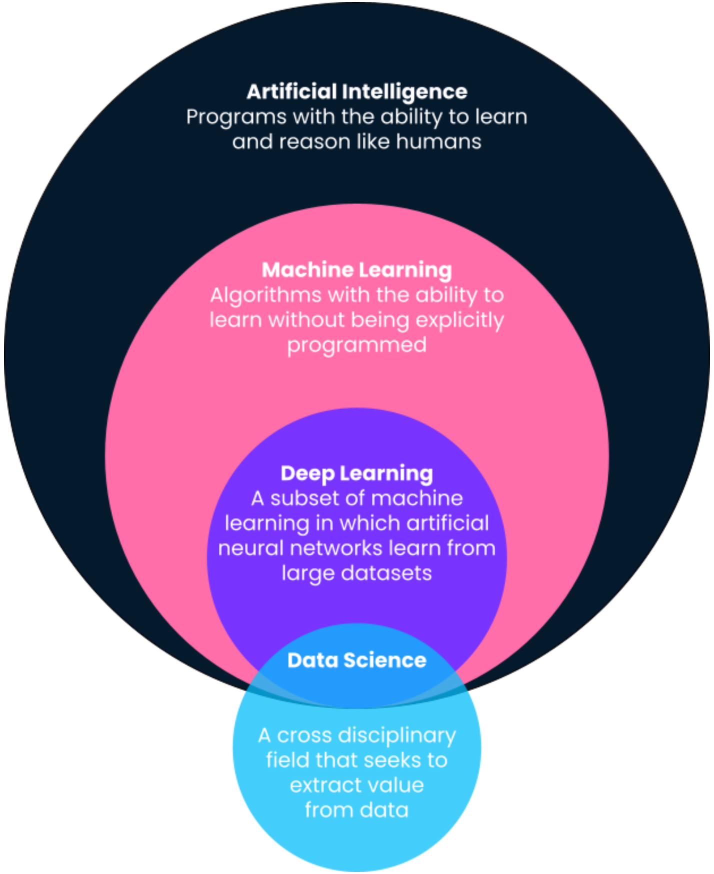


AI-Powered Features We Widely Use Today

- Virtual assistants
- Chatbots
- Image enhancement
- Image recognition
- Augmented reality
- Real-time language translation
- User behavior analytics
- App security & user authentication
- Real-time assistance
- Personalized experience

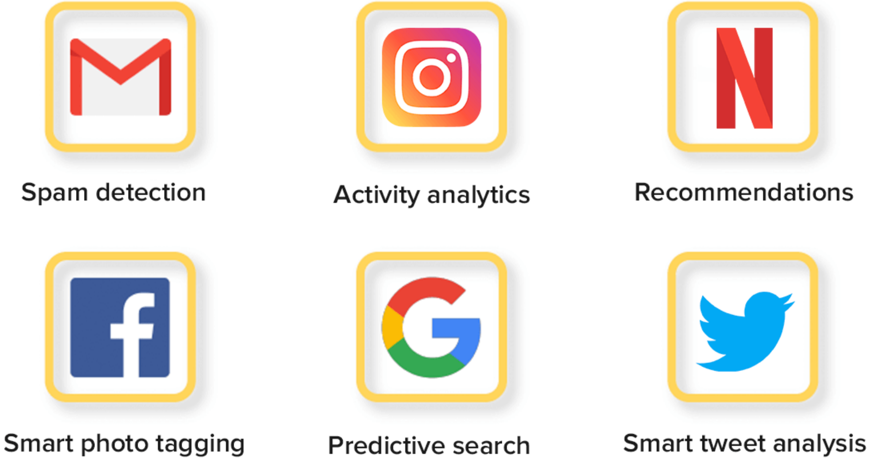
Understanding Machine Learning

- **Machine Learning** is a subset of AI that enables systems to learn from data and improve over time without being explicitly programmed.
- Key concepts:
  - Training data and models.
  - Types of ML: Supervised, unsupervised, and reinforcement learning.


Artificial Intelligence
Programs with the ability to learn and reason like humans
Machine Learning
Algorithms with the ability to learn without being explicitly programmed
Deep Learning
A subset of machine learning in which artificial neural networks learn from large datasets
Data Science
A cross disciplinary field that seeks to extract value from data

Understanding Machine Learning

- Example applications: Image Recognition, Natural Language Processing, and Recommendation Systems.



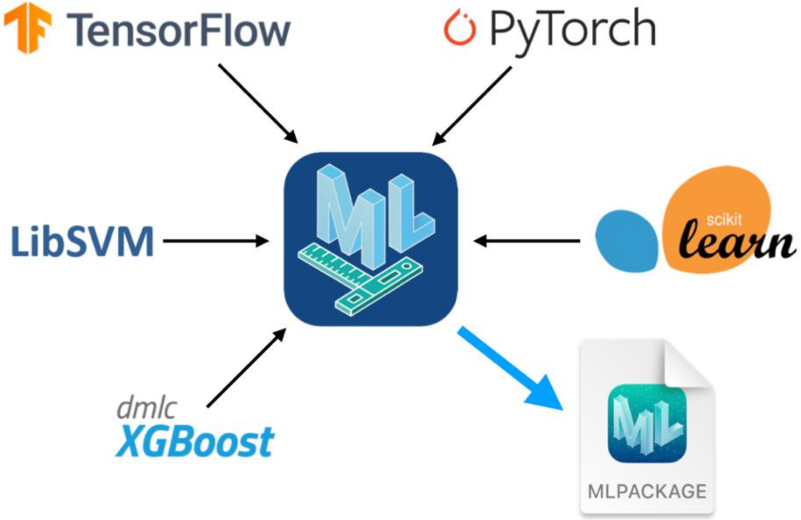
Apple’s Machine Learning Ecosystem

Apple’s Machine Learning Ecosystem

- **CoreML:** Apple’s framework for integrating machine learning models into apps.
- **CreateML:** Tool for training and testing models without needing deep coding expertise.
- **Vision Framework:** For image analysis, including object detection and facial recognition.
- **SiriKit:** For adding voice interaction capabilities to mobile apps like translation, transcription, etc.

What is CoreML?

- **CoreML** allows developers to use trained ML models directly on iOS, iPadOS, and macOS.
- Benefits:
  - Runs on-device (faster, more private).
  - Optimized for Apple’s hardware (A-series, and M-series chips)
- Supports popular format (eg. ONNX, TensorFlow, PyTorch).

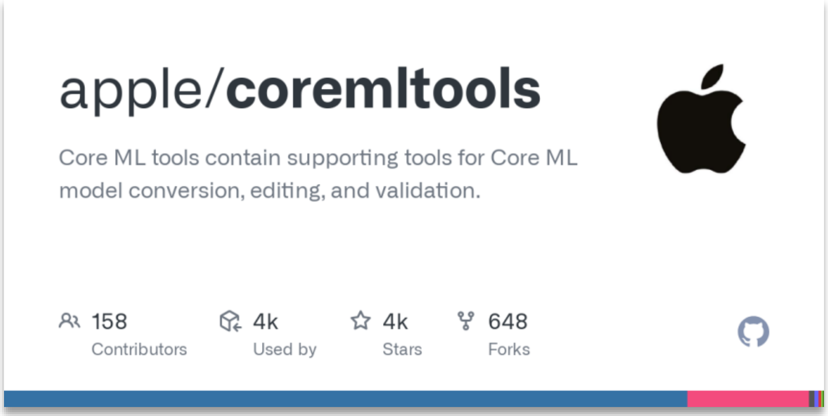


Key Steps for Using CoreML

- **Train a model:** Use frameworks like TensorFlow, PyTorch, or the Create ML platform.
- **Convert to CoreML:** Tools like CoreML Converter or ONNX to export model weights into `coremltools`.
- **Integrate into App:** import `.mlmodel` into Xcode and use CoreML APIs.
- **Optimize for On-Device Performance:** Use quantization and pruning to reduce model size.

Getting Started with CoreML

- **Tools to Explore:**
  - Xcode: Apple’s IDE for app development.
  - Swift Playgrounds: Learn Swift with hands-on examples.
  - CreateML: Simplified model training
- Apple’s Developer Documentation and Tutorials.
- Online Resources: GitHub repositories and open-source models.



apple/**coremltools**

Core ML tools contain supporting tools for Core ML model conversion, editing, and validation.

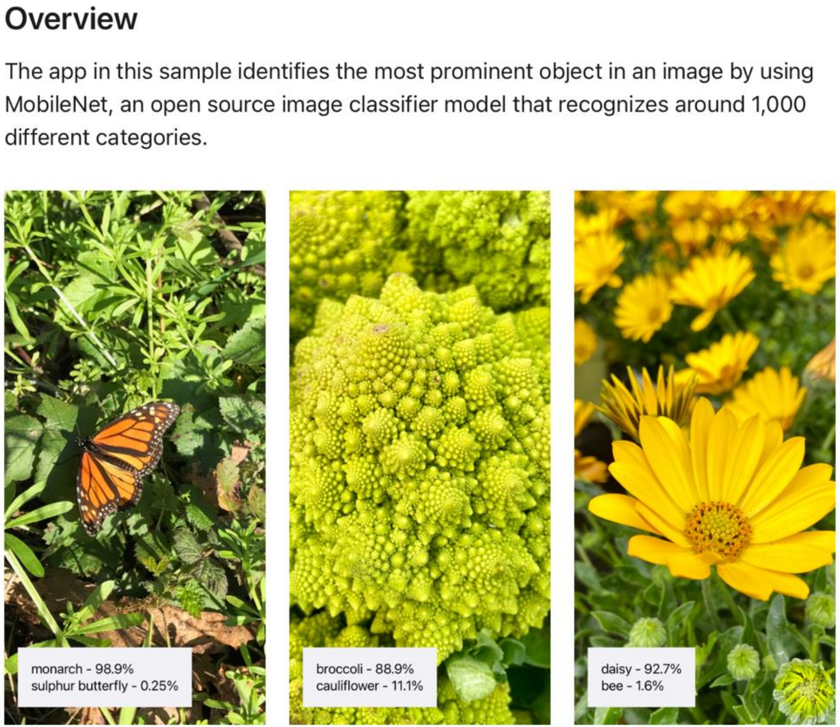
158 Contributors
4k Used by
4k Stars
648 Forks

CoreML for Image Recognition

Crop and scale photos using the Vision framework and classify them with a Core ML model.

Tools used:

- Vision, apply computer vision algorithms to perform a variety of tasks on input images and videos.
- CoreML, integrate machine learning models into your app.



Overview

The app in this sample identifies the most prominent object in an image by using MobileNet, an open source image classifier model that recognizes around 1,000 different categories.

monarch - 98.9%
sulphur butterfly - 0.25%

broccoli - 88.9%
cauliflower - 11.1%

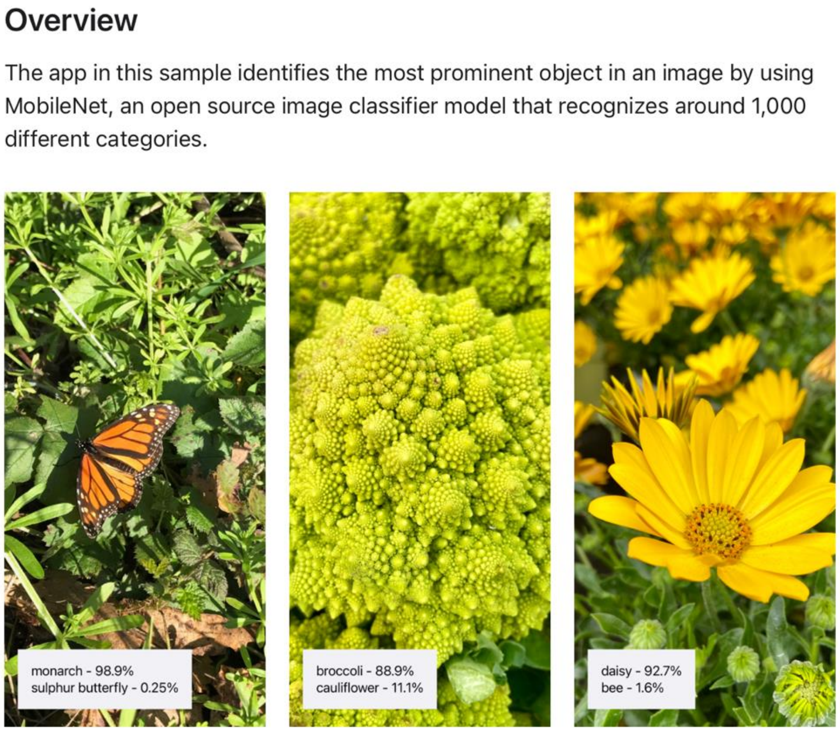
daisy - 92.7%
bee - 1.6%

Tutorial: Classifying Images with Vision and CoreML

CoreML for Image Recognition

Crop and scale photos using the Vision framework and classify them with a Core ML model.

- Each time a user takes a picture, the app passes it to a Vision image classification request.
- Image is pre-processed for adequate input to MobileNet architecture.
- CoreML behind the scenes outputs class during runtime.



Overview

The app in this sample identifies the most prominent object in an image by using MobileNet, an open source image classifier model that recognizes around 1,000 different categories.

monarch - 98.9%
sulphur butterfly - 0.25%

broccoli - 88.9%
cauliflower - 11.1%

daisy - 92.7%
bee - 1.6%

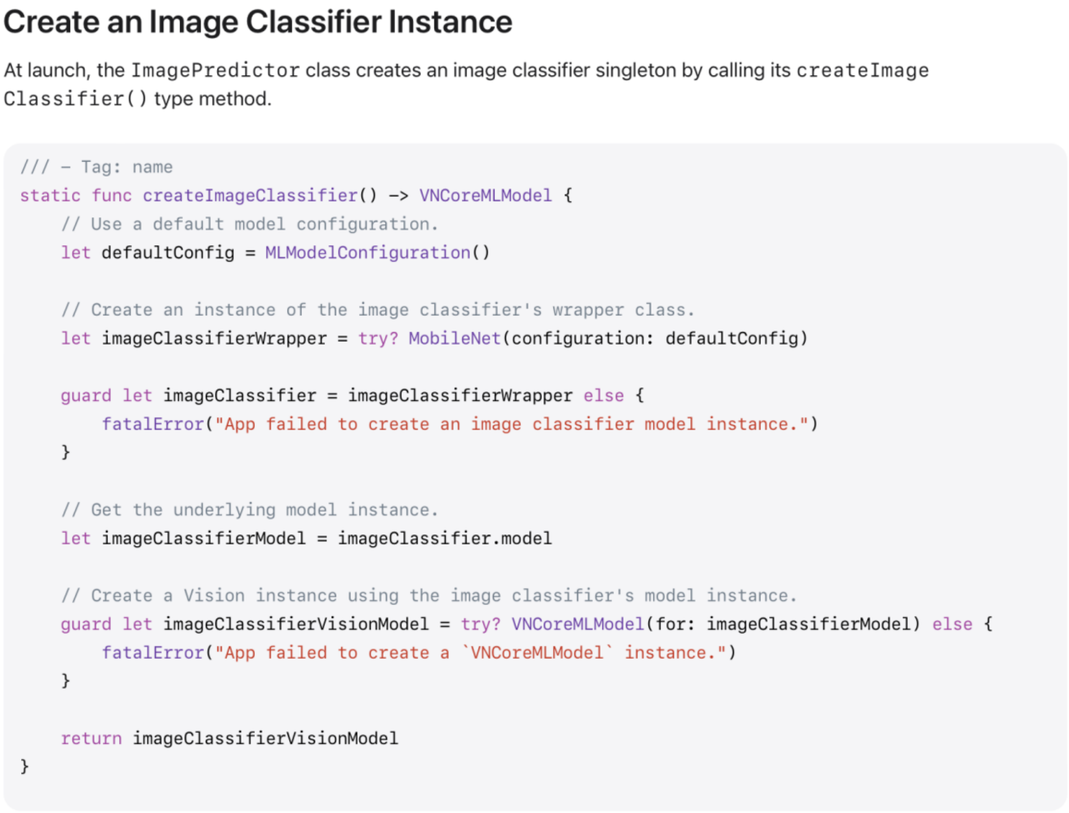
Tutorial: Classifying Images with Vision and CoreML

CoreML for Image Recognition

The method creates a Core ML model instance for Vision by:

1. Creating an instance of the model's wrapper class that Xcode auto-generates at compile time.
2. Retrieving the wrapper class instance's underlying MLModel property
3. Passing the model instance to a VNCoreMLModel initializer

The Image Predictor class minimizes runtime by only creating a single instance it shares across the app.



Create an Image Classifier Instance

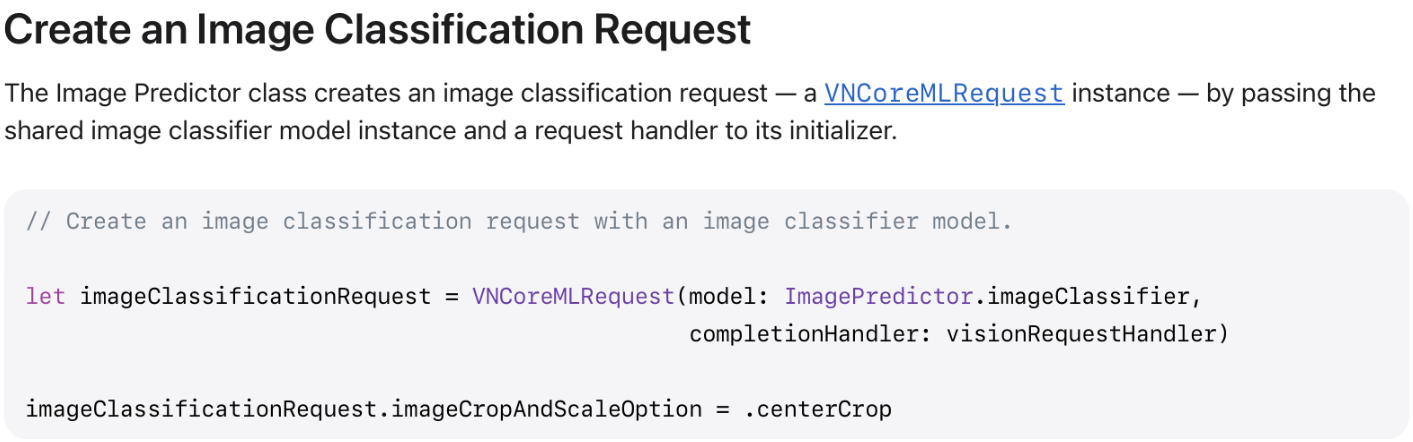
At launch, the `ImagePredictor` class creates an image classifier singleton by calling its `createImageClassifier()` type method.

```
/// - Tag: name
static func createImageClassifier() -> VNCoreMLModel {
    // Use a default model configuration.
    let defaultConfig = MLModelConfiguration()

    // Create an instance of the image classifier's wrapper class.
    let imageClassifierWrapper = try? MobileNet(configuration: defaultConfig)

    guard let imageClassifier = imageClassifierWrapper else {
        fatalError("App failed to create an image classifier model instance.")
    }

    // Get the underlying model instance.
    let imageClassifierModel = imageClassifier.model

    // Create a Vision instance using the image classifier's model instance.
    guard let imageClassifierVisionModel = try? VNCoreMLModel(for: imageClassifierModel) else {
        fatalError("App failed to create a `VNCoreMLModel` instance.")
    }

    return imageClassifierVisionModel
}
```

Tutorial: Classifying Images with Vision and CoreML

CoreML for Image Recognition

The Image classification request pre-processes the image and handles I/O for the model.



Create an Image Classification Request

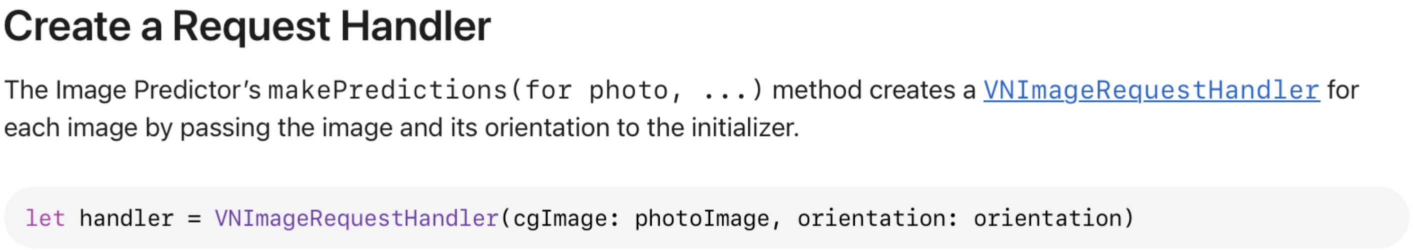
The Image Predictor class creates an image classification request — a VNCoreMLRequest instance — by passing the shared image classifier model instance and a request handler to its initializer.

```
// Create an image classification request with an image classifier model.

let imageClassificationRequest = VNCoreMLRequest(model: ImagePredictor.imageClassifier,
                                                 completionHandler: visionRequestHandler)

imageClassificationRequest.imageCropAndScaleOption = .centerCrop
```

Tutorial: Classifying Images with Vision and CoreML

CoreML for Image Recognition

Request handler can work with both images stored/taken by user or by initializing the URL of the image.



Create a Request Handler

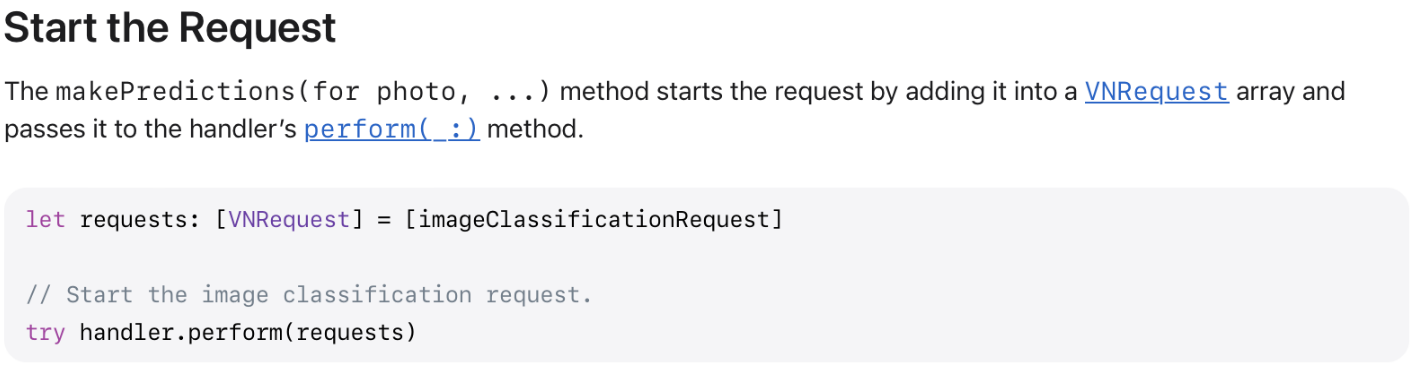
The Image Predictor's `makePredictions(for photo, ...)` method creates a [`VNImageRequestHandler`]() for each image by passing the image and its orientation to the initializer.

```
let handler = VNImageRequestHandler(cgImage: photoImage, orientation: orientation)
```

Tutorial: [Classifying Images with Vision and CoreML]()

CoreML for Image Recognition

To perform multiple Vision requests on the same image we can add multiple requests to the array you pass to the perform(_:) method’s requests parameter.



Start the Request

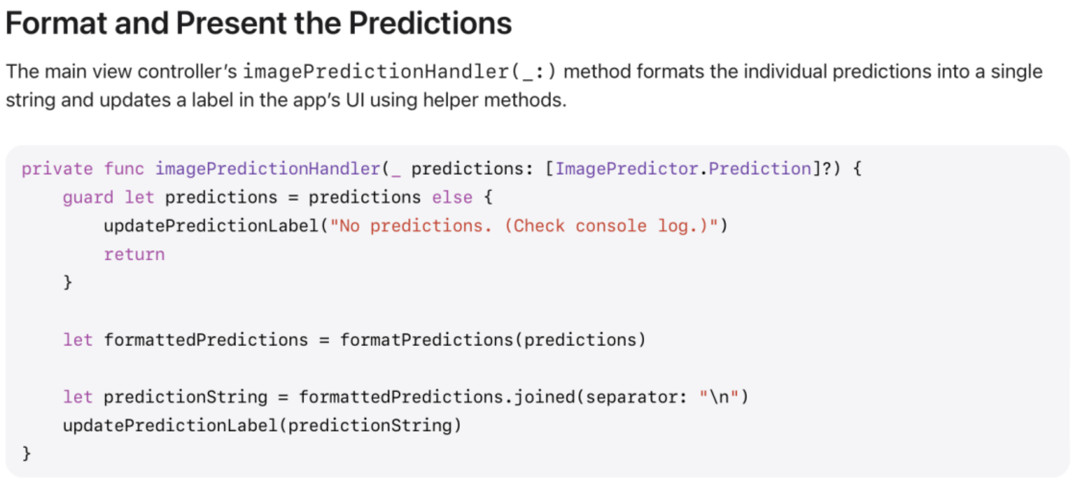
The `makePredictions(for photo, ...)` method starts the request by adding it into a `VNRequest` array and passes it to the handler’s `perform(_:)` method.

```
let requests: [VNRequest] = [imageClassificationRequest]

// Start the image classification request.
try handler.perform(requests)
```

Tutorial: *Classifying Images with Vision and CoreML*

CoreML for Image Recognition

- We constantly update the label of our predictions based on how we format them in our View file.



Format and Present the Predictions

The main view controller's `imagePredictionHandler(_:)` method formats the individual predictions into a single string and updates a label in the app's UI using helper methods.

```
private func imagePredictionHandler(_ predictions: [ImagePredictor.Prediction]?) {
    guard let predictions = predictions else {
        updatePredictionLabel("No predictions. (Check console log.)")
        return
    }

    let formattedPredictions = formatPredictions(predictions)

    let predictionString = formattedPredictions.joined(separator: "\n")
    updatePredictionLabel(predictionString)
}
```

Tutorial: Classifying Images with Vision and CoreML

Edge Computing Group
Unleashing Intelligence Everywhere

Benefits, Challenges, and Future of AI-Mobile Development

Benefits of On-Device Processing

- **Privacy:** Data stays on the device.
- **Performance:** No need for server round-trips.
- **Availability:** Works without internet connectivity.
- **Energy efficiency** with Apple Neural Engine (ANE).

Challenges in Mobile AI Development

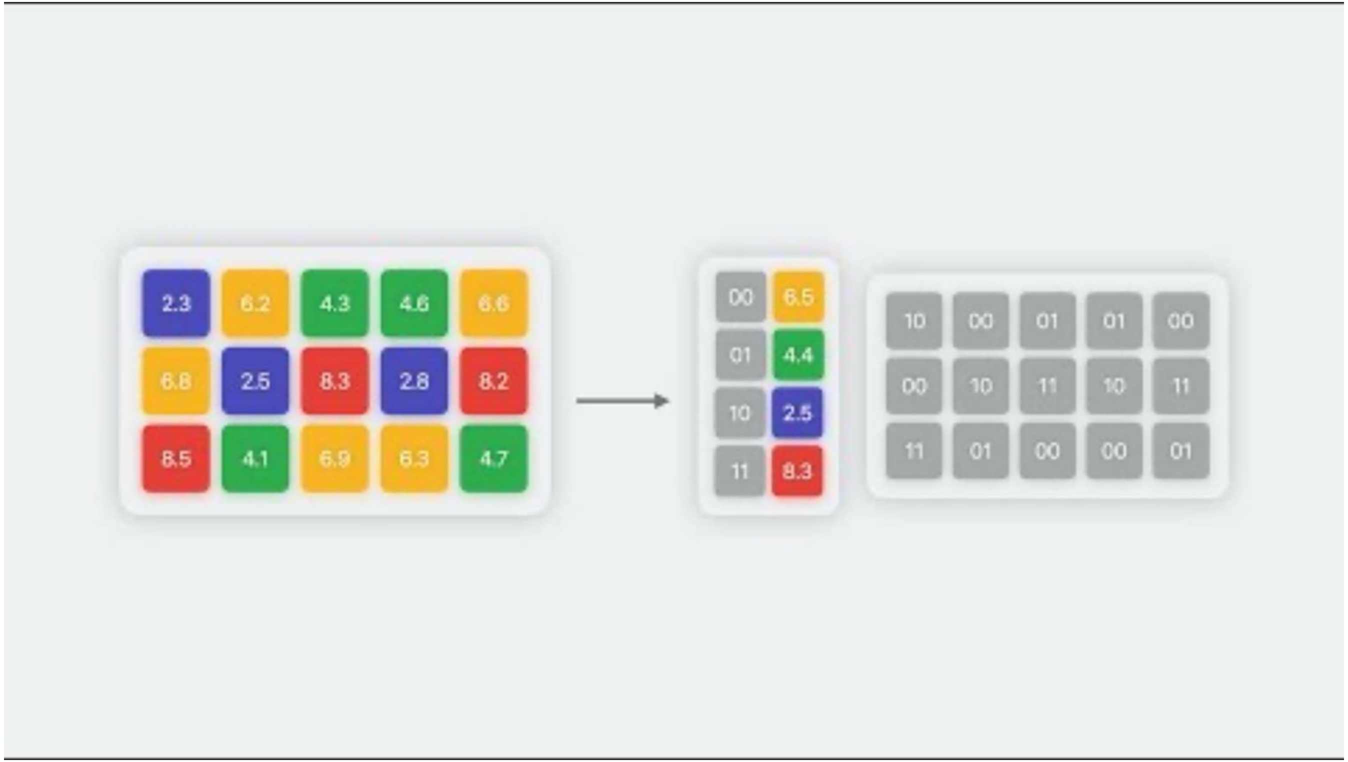
- **Limited device resources** (memory, battery, and processing power).
- **Model optimization** (size vs. accuracy trade-off).
- **Compatibility** across devices and iOS versions.

Future of AI in Mobile Development

- More sophisticated models running on mobile devices.
- Seamless integration of AR and AI for enhanced user experiences.
- Real-time on-device AI for personalized and dynamic applications.
  - Education
  - Healthcare
  - and more.

WWDC23: *Use CoreML Tools for machine learning model compression*

WWDC23: *Use CoreML Tools for machine learning model compression*



2.3	6.2	4.3	4.6	6.6
6.8	2.5	8.3	2.8	8.2
8.5	4.1	6.9	6.3	4.7

00	6.5
01	4.4
10	2.5
11	8.3

10	00	01	01	00
00	10	11	10	11
11	01	00	00	01

Q&A

Contact Information

Website

LinkedIn

GitHub

sebastian.cruz6@upr.edu

Acknowledgements

- Dr. Wilfredo Lugo Beauchamp, PhD
- Edge Computing Group
- NSF-EPSCoR Center for the Advancement of Wearable Technologies (CAWT)
- Center for Research & Development at the University of Puerto Rico at Mayagüez

This project is supported by **NSF Award OIA-1849243**

Edge Computing Group
Unleashing Intelligence Everywhere

EPSCoR
RESEARCH IN PUERTO RICO

CENTER FOR THE ADVANCEMENT OF WEARABLE TECHNOLOGIES