## Setting Up

You will need a Mac and an iPhone for this tutorial.

### XCode

1. Download and install XCode on your Mac. Ensure the version is XCode 15 or newer.
2. Launch XCode and click 'Create New Project'.
3. Under the iOS tab, select 'App'.
4. Name your project and save.

### Connecting Your iPhone

1. Open your project in XCode. At the top bar select 'Window' > 'Devices and Simulators'.
2. Connect your iPhone to your Mac via cable.
3. In the list of devices, select your iPhone and check 'Connect via Network'.
4. You can now disconnect your iPhone.

## Create ML

Now let's build your image classification model.

### Find or Create a Dataset

1. Find a dataset that you are interested in or create one yourself. There are many datasets already available on the internet. For example, if you wanted to your model to recognize plants, you may want to use this dataset: https://www.kaggle.com/datasets/yudhaislamisulistya/plants-type-datasets
2. If you'd like to create your own dataset, name the main folder training_data, and create sub-folders of the types of images. See above link for example.
3. Similar to the training data, make another folder for your testing data. The testing data should have the same sub-folders but different images.

### Creating an Image Classification Model

1. Navigate back to your XCode project. At the top bar, 'XCode' > 'Open Developer Tool' > 'Create ML'.
2. Click 'New Document' in the bottom left corner. Select 'Image' > 'Image Classification'.
3. Name your project and save.
4. Press the + sign under training data and select your training data folder.

5. Optionally, you can select different augmentations under ‘Parameters’ that will add modified copies of your images to the training data.

**Training and Testing Your Model**

1. Now press the ‘Train’ button at the top left corner and let the training run. If you’re unhappy with the accuracy of your training, you may train more and adjust the number of iterations.
2. To test your model, select ‘Evaluation’ at the top. Then, press the + sign, select your testing data folder, and press ‘Test’.
3. Once you are satisfied with the accuracy of your image classification model, select ‘Output’ at the top and drag and drop the icon next to your model’s name into your XCode project.

## SwiftUI - ContentView

We will now code the file ContentView.swift to connect your image classification model to an app that integrates a live camera feed with real-time predictions.

**Coding ContentView**

1. Define a state object for the view model under line 10:

```
10  struct ContentView: View {
11      @StateObject private var viewModel = ContentViewModel()
```

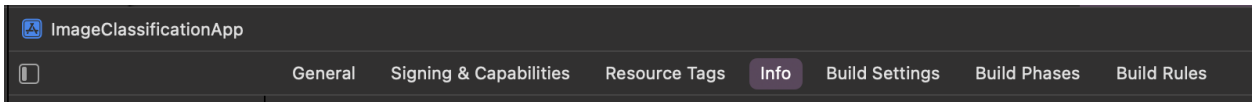
   We will create this ContentViewModel later so do not worry if XCode is throwing an error.
2. The UI will go inside the body. You can design the UI however you like, here is Apple’s tutorial: https://developer.apple.com/tutorials/swiftui/. For this tutorial, we’ll work with

the VStack provided to us.

```
13      var body: some View {
14          VStack {
15              // Prediction Type
16              HStack {
17                  Text("Prediction: ")
18                  Text (viewModel.prediction)
19              } // HStack
20
21              // Confidence Level
22              HStack {
23                  Text("Confidence: ")
24                  Text(viewModel.confidence)
25              } // HStack
26
27              // Camera Preview that the app uses to display previews
28              CameraPreview(session: viewModel.session).onAppear {
29
30                  // Ensure code is ran in background
31                  DispatchQueue.global().async {
32                      self.viewModel.setupSession()
33                  }
34              }
35          } // VStack
36      }
```

**Set Camera Permissions**

1. In the list of files on the left, select your app at the very top.
2. Select 'Info':




3. Under 'Custom iOS Target Properties', hover over any key and press + to add a new key.
4. Enter 'NSCameraUsageDescription' and press enter. It will autocorrect to 'Privacy - Camera Usage Description' if done correctly.
5. Optionally, you can enter a description for why your app requires the camera in the 'Value' section.

## SwiftUI - Camera Preview

Now we will code the camera preview which will use your phone camera to allow the image classification model to detect objects.

### Creating a New File

1. At the top left corner of your Mac, select 'File' > 'New' > 'File'.
2. Make sure you are in the iOS tab and select 'Swift File'.
3. Click 'Next', name the new file 'CameraPreview.swift', and click 'Create'.

### Coding

1. Import the needed libraries:

```
8  import SwiftUI
9  import AVKit
```

2. Define required properties:

```
11 struct CameraPreview: UIViewRepresentable {
12     var session: AVCaptureSession
```

3. Implement makeUIView function and setup the UIView and preview layer:

```
14     func makeUIView(context: Context) -> UIView {
15         // Create a UIView with the frame of the device's screen
16         let view = UIView(frame: UIScreen.main.bounds)
17
18         // Initialize the AVCaptureVideoPreviewLayer with the session
19         let previewLayer = AVCaptureVideoPreviewLayer(session: session)
20         previewLayer.frame = view.frame
21         previewLayer.videoGravity = .resizeAspectFill
22
23         // Add the previewLayer as a sublayer of the view's layer
24         view.layer.addSublayer(previewLayer)
25         return view
26     }
```

4. Implement the updateUIView function:

```
28     func updateUIView(_ uiView: UIView, context: Context) {}
```

## SwiftUI - ContentViewModel

Now we will create a class that incorporates video capturing, image classification model predictions, and updating the UI based on the predictions.

1. Create a new Swift File like how we did for the CameraPreview.swift, name it 'ContentViewModel.swift'.
2. Import needed libraries:

```swift
import AVKit
import Vision
import Combine
```

3. Setup the basic class structure:

```swift
class ContentViewModel: NSObject, ObservableObject, AVCaptureVideoDataOutputSampleBufferDelegate {
```

4. Define observable properties for UI updates:

```swift
    @Published var prediction: String = "--"
    @Published var confidence: String = "--"
```

5. Setup the AVCaptureSession and implement the setupSession function:

```swift
    let session = AVCaptureSession()

    func setupSession() {
        guard let device = AVCaptureDevice.default(for: .video) else { return }
        guard let input = try? AVCaptureDeviceInput(device: device) else { return }
        session.sessionPreset = .hd1280x720 // Set high-definition video capture

        let output = AVCaptureVideoDataOutput()
        output.setSampleBufferDelegate(self, queue: DispatchQueue(label: "videoQueue"))

        session.addInput(input)
        session.addOutput(output)
        session.startRunning() // Begin capturing video
    }
```

6. Implement the captureOutput function to handle video frames:

```
32      func captureOutput(_ output: AVCaptureOutput, didOutput sampleBuffer: CMSampleBuffer, from
            connection: AVCaptureConnection) {
33
34          guard let pixelBuffer: CVPixelBuffer = CMSampleBufferGetImageBuffer(sampleBuffer) else {
                return }
35
36          guard let model = try? VNCoreMLModel(for: ModelName.model) else { return }  ⊗ Cannot find '...
37
38          let request = VNCoreMLRequest(model: model) { (finishedReq, err) in
39              DispatchQueue.main.async {
40                  self.processResults(for: finishedReq)
41              }
42          }
43
44          try? VNImageRequestHandler(cvPixelBuffer: pixelBuffer, options: [:]).perform([request])
45      }
```

On line 36, replace 'ModelName' with the name of your image classification model.

7. Implement the processResults function:

```
47      private func processResults(for request: VNRequest) {
48          guard let results = request.results as? [VNClassificationObservation], let firstResult =
                results.first else {
49              prediction = "--"
50              confidence = "--"
51              return
52          }
53
54          if firstResult.confidence * 100 >= 20 {
55              prediction = firstResult.identifier.capitalized
56              confidence = String(format: "%.2f%%", firstResult.confidence * 100)
57          } else {
58              prediction = "--"
59              confidence = "--"
60          }
61      }
```

Now make sure you save all your files and we can press the run button on the left panel. Your phone should launch this app and start scanning objects around you!