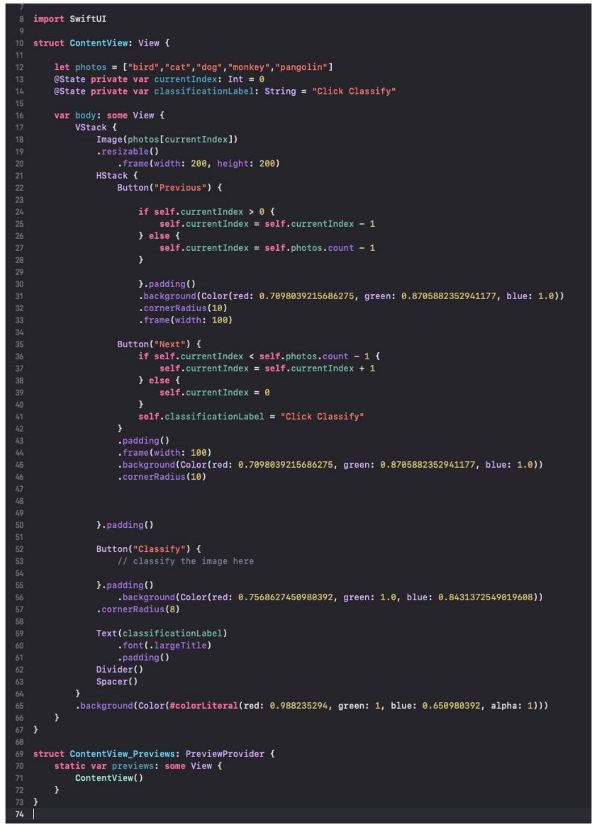
Core ML

教學

Image Classification – 現有模型

- 下載專案：
- 專案功能：
  - 可以翻到前/後一張圖片
  - “Classify”按鈕（無實裝功能）
  - “Click Classify”label（無實裝功能）



```swift

import SwiftUI

struct ContentView: View {

    let photos = ["bird","cat","dog","monkey","pangolin"]
    @State private var currentIndex: Int = 0
    @State private var classificationLabel: String = "Click Classify"

    var body: some View {
        VStack {
            Image(photos[currentIndex])
            .resizable()
                .frame(width: 200, height: 200)
            HStack {
                Button("Previous") {

                    if self.currentIndex > 0 {
                        self.currentIndex = self.currentIndex - 1
                    } else {
                        self.currentIndex = self.photos.count - 1
                    }

                    }.padding()
                    .background(Color(red: 0.7098039215686275, green: 0.8705882352941177, blue: 1.0))
                    .cornerRadius(10)
                    .frame(width: 100)

                Button("Next") {
                    if self.currentIndex < self.photos.count - 1 {
                        self.currentIndex = self.currentIndex + 1
                    } else {
                        self.currentIndex = 0
                    }
                    self.classificationLabel = "Click Classify"
                }
                .padding()
                .frame(width: 100)
                .background(Color(red: 0.7098039215686275, green: 0.8705882352941177, blue: 1.0))
                .cornerRadius(10)



            }.padding()

            Button("Classify") {
                // classify the image here

            }.padding()
                .background(Color(red: 0.7568627450980392, green: 1.0, blue: 0.8431372549019608))
            .cornerRadius(8)

            Text(classificationLabel)
                .font(.largeTitle)
                .padding()
            Divider()
            Spacer()
        }
        .background(Color(#colorLiteral(red: 0.988235294, green: 1, blue: 0.650980392, alpha: 1)))
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}

```

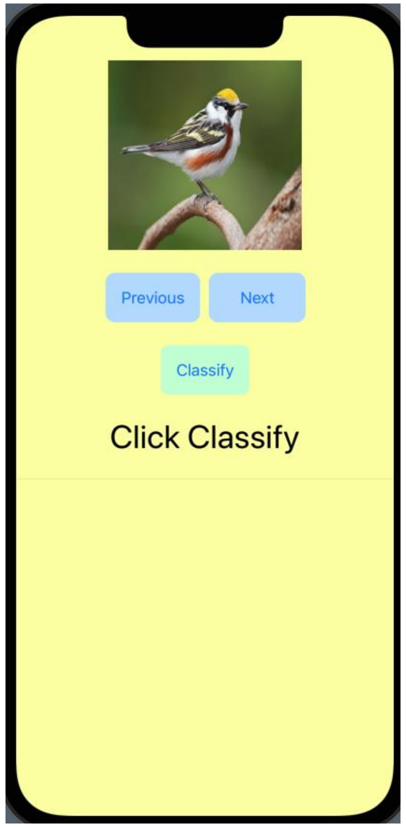
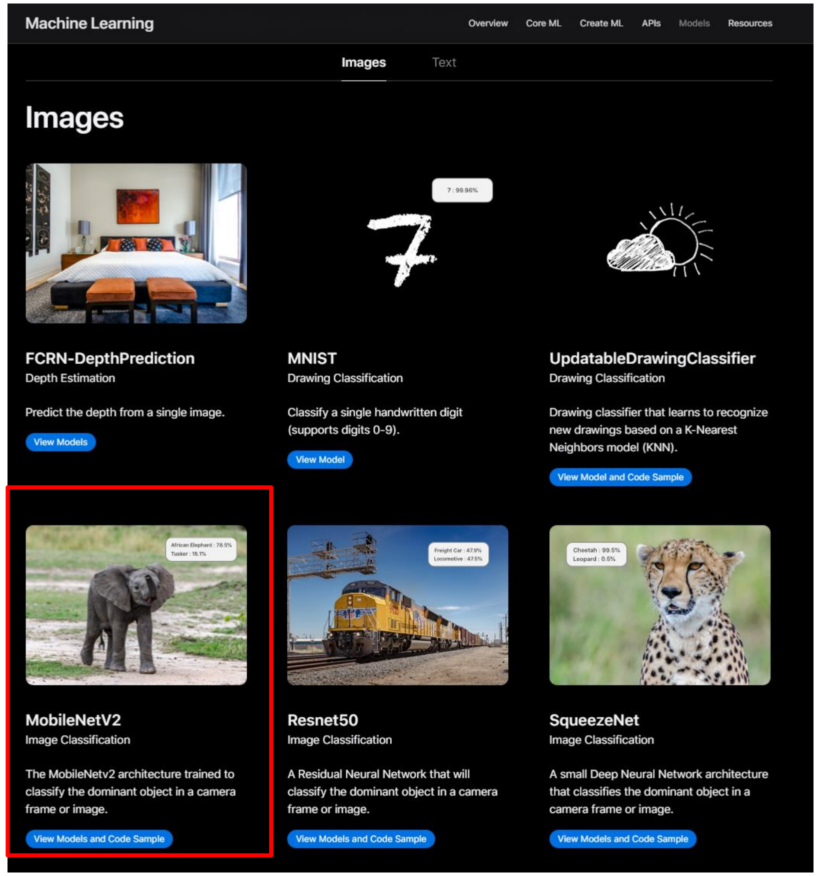


Image Classification – 現有模型

- 1. 前往 https://developer.apple.com/machine-learning/models/ 下載 MobileNetV2 模型



Machine Learning

Overview · Core ML · Create ML · APIs · Models · Resources

Images · Text

Images

FCRN-DepthPrediction
Depth Estimation

Predict the depth from a single image.

View Models

MNIST
Drawing Classification

Classify a single handwritten digit (supports digits 0-9).

View Model

UpdatableDrawingClassifier
Drawing Classification

Drawing classifier that learns to recognize new drawings based on a K-Nearest Neighbors model (KNN).

View Model and Code Sample

MobileNetV2
Image Classification

The MobileNetv2 architecture trained to classify the dominant object in a camera frame or image.

View Models and Code Sample

Resnet50
Image Classification

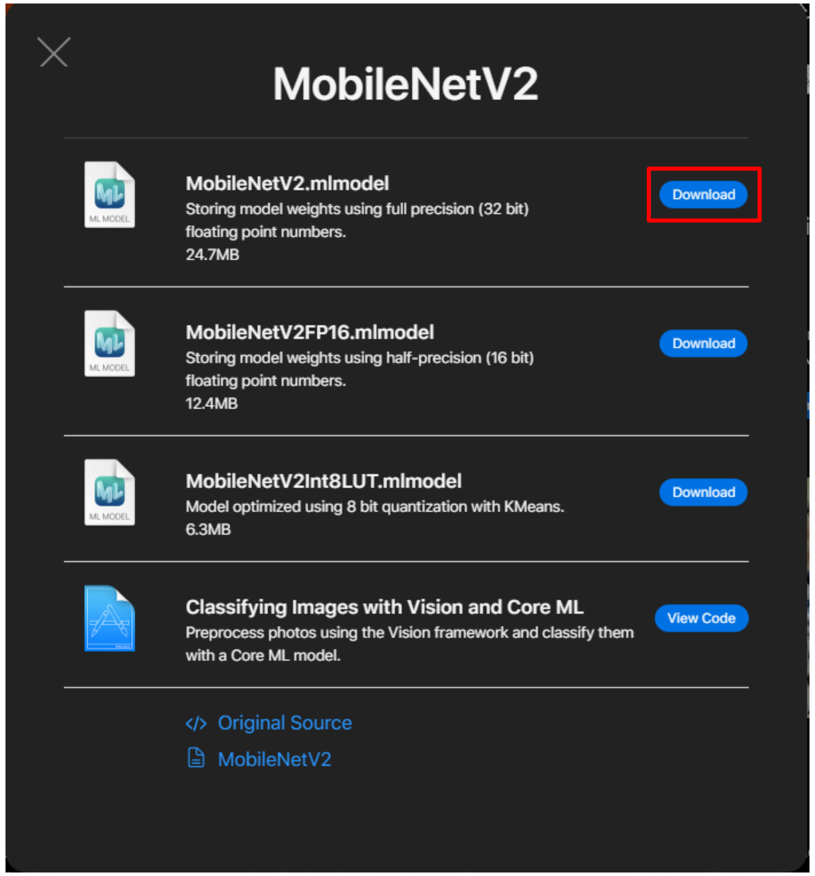
A Residual Neural Network that will classify the dominant object in a camera frame or image.

View Models and Code Sample

SqueezeNet
Image Classification

A small Deep Neural Network architecture that classifies the dominant object in a camera frame or image.

View Models and Code Sample



MobileNetV2

MobileNetV2.mlmodel
Storing model weights using full precision (32 bit) floating point numbers.
24.7MB
Download

MobileNetV2FP16.mlmodel
Storing model weights using half-precision (16 bit) floating point numbers.
12.4MB
Download

MobileNetV2Int8LUT.mlmodel
Model optimized using 8 bit quantization with KMeans.
6.3MB
Download

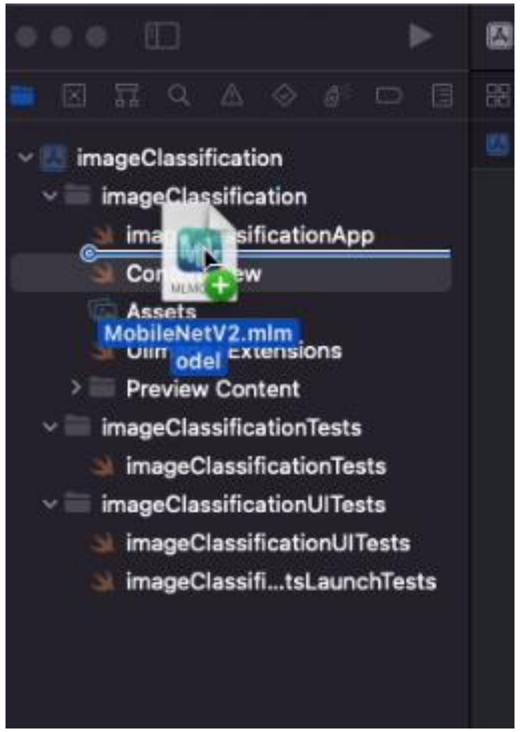
Classifying Images with Vision and Core ML
Preprocess photos using the Vision framework and classify them with a Core ML model.
View Code

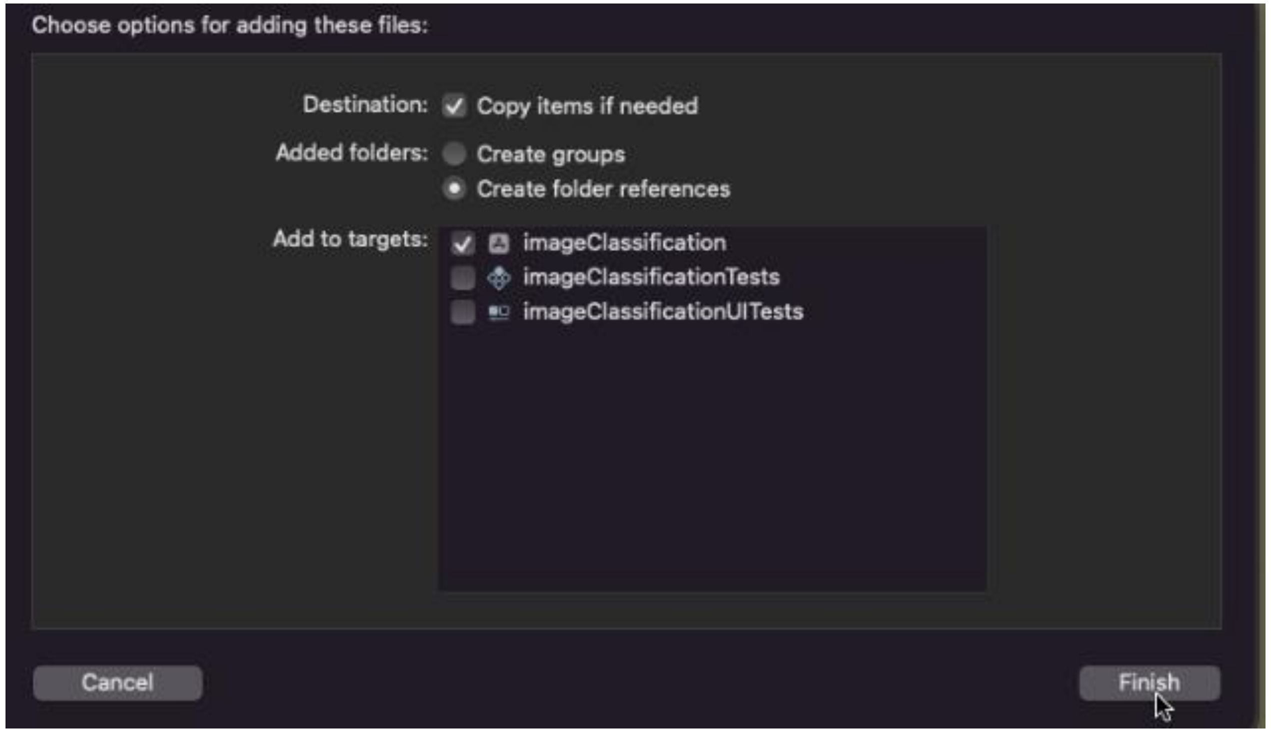
Original Source

MobileNetV2

Image Classification – 現有模型

- 2. 把下載好的.mlmodel拖進專案裡




Choose options for adding these files:

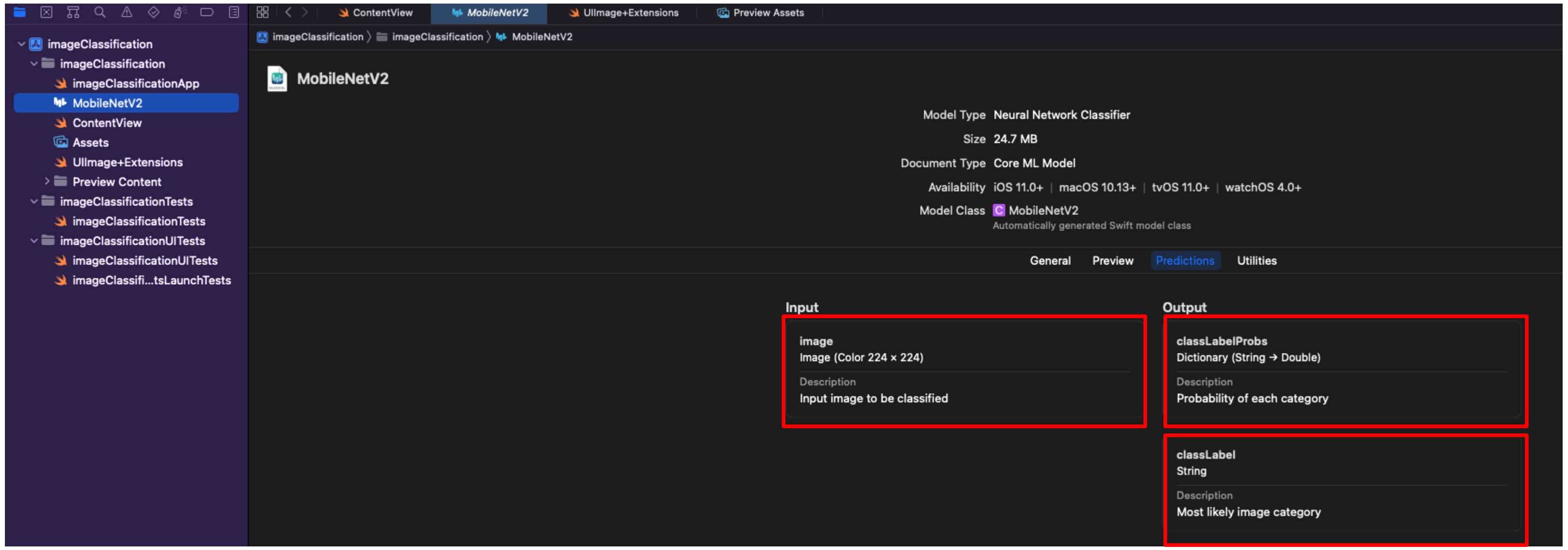
Destination: ✓ Copy items if needed

Added folders: ○ Create groups
● Create folder references

Add to targets: ✓ imageClassification
☐ imageClassificationTests
☐ imageClassificationUITests

Image Classification – 現有模型

- 3. 匯入MobileNetV2後我們可以在model的Predictions看到他的Input與Output格式



ContentView | MobileNetV2 | UIImage+Extensions | Preview Assets

imageClassification 〉 imageClassification 〉 MobileNetV2

- imageClassification
  - imageClassification
    - imageClassificationApp
    - MobileNetV2
    - ContentView
    - Assets
    - UIImage+Extensions
    - Preview Content
  - imageClassificationTests
    - imageClassificationTests
  - imageClassificationUITests
    - imageClassificationUITests
    - imageClassifi...tsLaunchTests

MobileNetV2

Model Type	Neural Network Classifier
Size	24.7 MB
Document Type	Core ML Model
Availability	iOS 11.0+ | macOS 10.13+ | tvOS 11.0+ | watchOS 4.0+
Model Class	MobileNetV2 Automatically generated Swift model class

General | Preview | Predictions | Utilities

Input

image
Image (Color 224 × 224)
Description
Input image to be classified

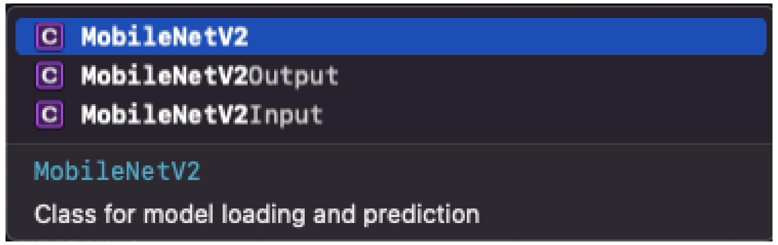
Output

classLabelProbs
Dictionary (String → Double)
Description
Probability of each category

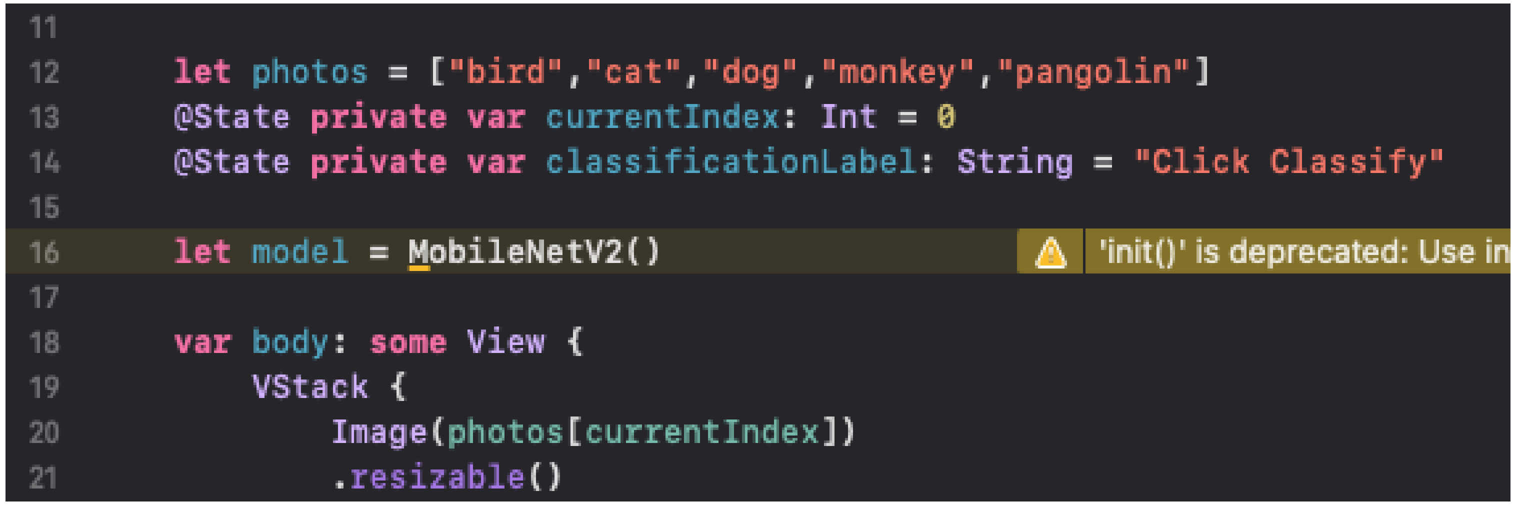
classLabel
String
Description
Most likely image category

Image Classification – 現有模型

- 4. 創建MobileNetV2() 實例 Instance



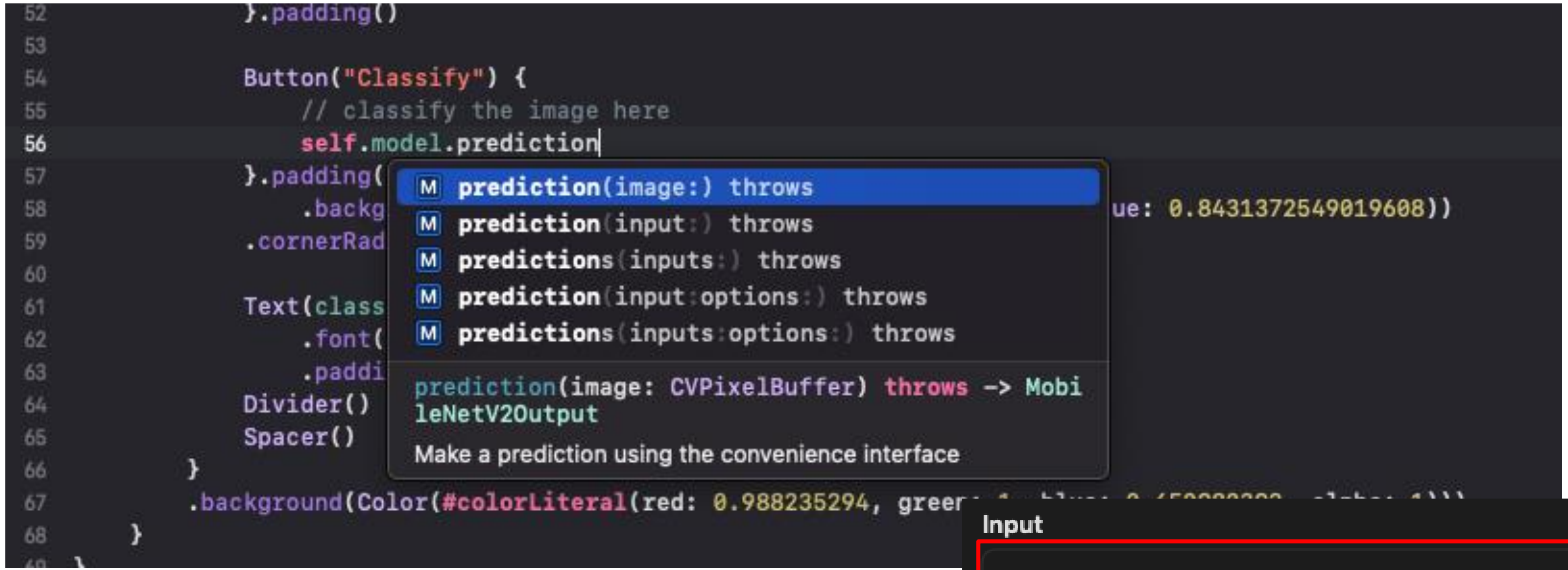
```
C MobileNetV2
C MobileNetV2Output
C MobileNetV2Input
MobileNetV2
Class for model loading and prediction
```



```

    let photos = ["bird","cat","dog","monkey","pangolin"]
    @State private var currentIndex: Int = 0
    @State private var classificationLabel: String = "Click Classify"

    let model = MobileNetV2()                    ⚠ 'init()' is deprecated: Use in

    var body: some View {
        VStack {
            Image(photos[currentIndex])
            .resizable()
```

Image Classification – 現有模型

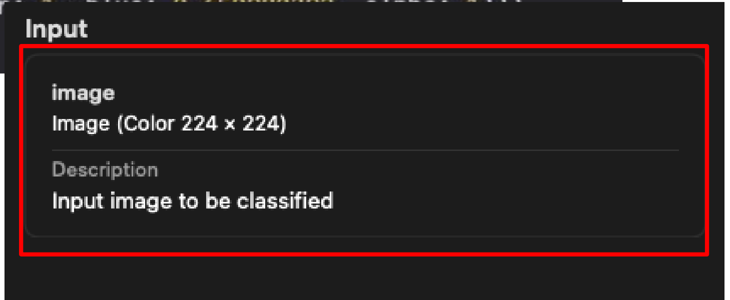
- 5a. 我們可以透過 self.model.查看提示，提示顯示有不同的prediction function，而我們要使用的是可以傳入image的prediction function。而依照提示我們必須要傳入`(image: CVPixelBuffer)`格式的照片。



```
            }.padding()

            Button("Classify") {
                // classify the image here
                self.model.prediction
            }.padding(
                .backg                                                  ue: 0.8431372549019608))
            .cornerRad

            Text(class
                .font(
                .paddi
            Divider()
            Spacer()
        }
        .background(Color(#colorLiteral(red: 0.988235294, green
    }
```

```
M  prediction(image:) throws
M  prediction(input:) throws
M  predictions(inputs:) throws
M  prediction(input:options:) throws
M  predictions(inputs:options:) throws

prediction(image: CVPixelBuffer) throws -> Mobi
leNetV2Output
Make a prediction using the convenience interface
```

- 5b. Model需要縮放Image （ 224*224 ）大小的需求



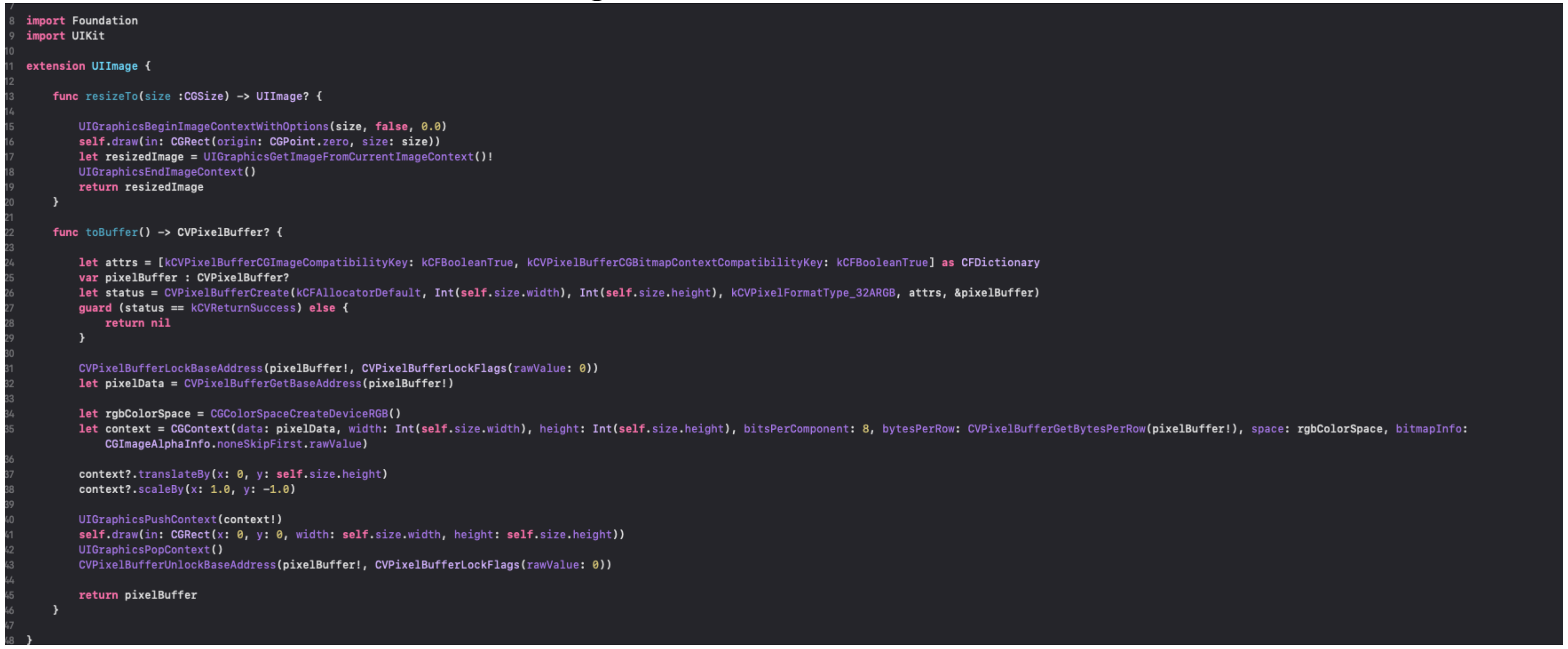
Input

image
Image (Color 224 × 224)

Description
Input image to be classified

Image Classification – 現有模型

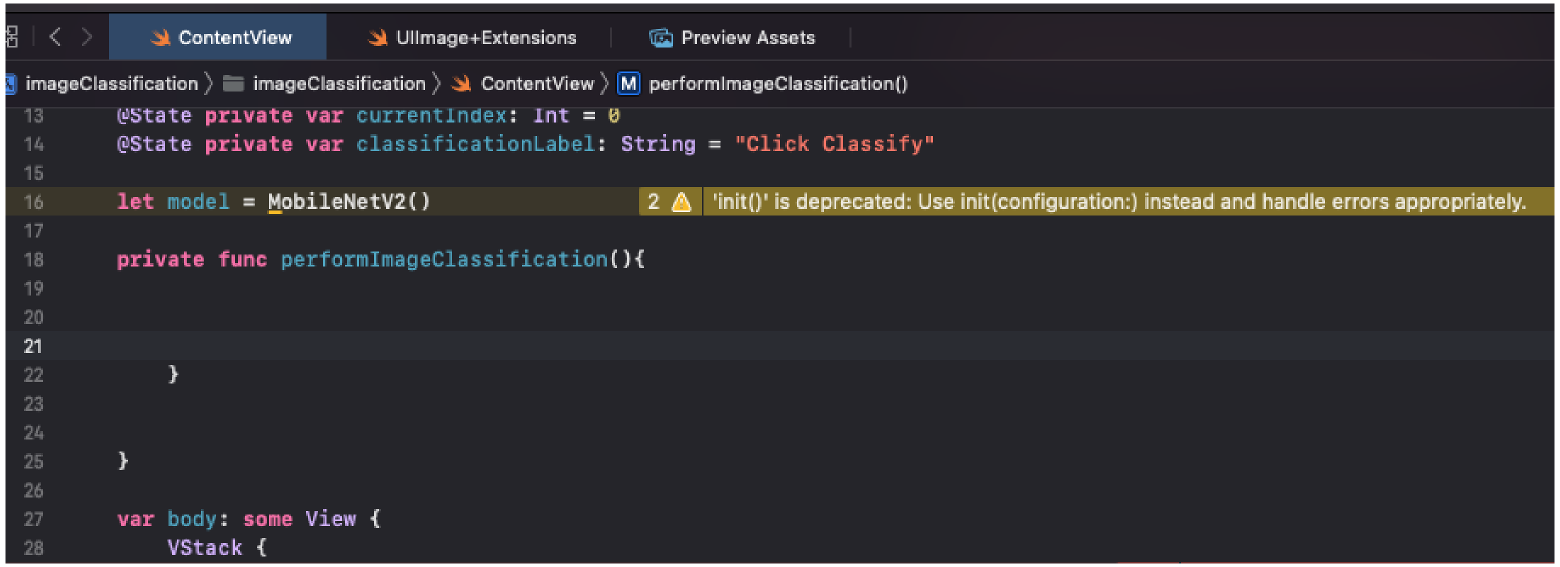
- 6. 為了滿足傳入Input的需求，專案裡已有準備UIImage+Extensions的擴充程式碼
  - 第一個function resizeTo : 傳入你需要的CGSize，function回傳特定大小的Image
  - 第二個function tobuffer: 將Image轉換成CVPixelBuffer格式



```swift
import Foundation
import UIKit

extension UIImage {

    func resizeTo(size :CGSize) -> UIImage? {

        UIGraphicsBeginImageContextWithOptions(size, false, 0.0)
        self.draw(in: CGRect(origin: CGPoint.zero, size: size))
        let resizedImage = UIGraphicsGetImageFromCurrentImageContext()!
        UIGraphicsEndImageContext()
        return resizedImage
    }

    func toBuffer() -> CVPixelBuffer? {

        let attrs = [kCVPixelBufferCGImageCompatibilityKey: kCFBooleanTrue, kCVPixelBufferCGBitmapContextCompatibilityKey: kCFBooleanTrue] as CFDictionary
        var pixelBuffer : CVPixelBuffer?
        let status = CVPixelBufferCreate(kCFAllocatorDefault, Int(self.size.width), Int(self.size.height), kCVPixelFormatType_32ARGB, attrs, &pixelBuffer)
        guard (status == kCVReturnSuccess) else {
            return nil
        }

        CVPixelBufferLockBaseAddress(pixelBuffer!, CVPixelBufferLockFlags(rawValue: 0))
        let pixelData = CVPixelBufferGetBaseAddress(pixelBuffer!)

        let rgbColorSpace = CGColorSpaceCreateDeviceRGB()
        let context = CGContext(data: pixelData, width: Int(self.size.width), height: Int(self.size.height), bitsPerComponent: 8, bytesPerRow: CVPixelBufferGetBytesPerRow(pixelBuffer!), space: rgbColorSpace, bitmapInfo:
            CGImageAlphaInfo.noneSkipFirst.rawValue)

        context?.translateBy(x: 0, y: self.size.height)
        context?.scaleBy(x: 1.0, y: -1.0)

        UIGraphicsPushContext(context!)
        self.draw(in: CGRect(x: 0, y: 0, width: self.size.width, height: self.size.height))
        UIGraphicsPopContext()
        CVPixelBufferUnlockBaseAddress(pixelBuffer!, CVPixelBufferLockFlags(rawValue: 0))

        return pixelBuffer
    }

}
```

Image Classification – 現有模型

- 7. 回到ContentView宣告一個private的執行圖像分類function : performImageClassification()



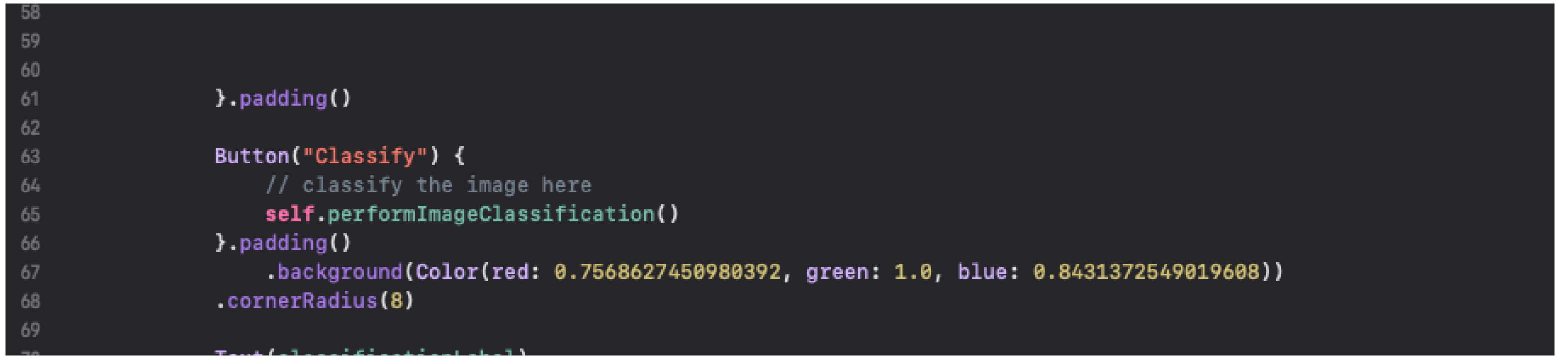
ContentView | UIImage+Extensions | Preview Assets

imageClassification 〉 imageClassification 〉 ContentView 〉 M performImageClassification()

```
    @State private var currentIndex: Int = 0
    @State private var classificationLabel: String = "Click Classify"

    let model = MobileNetV2()                    2 ⚠ 'init()' is deprecated: Use init(configuration:) instead and handle errors appropriately.

    private func performImageClassification(){



        }


    }

    var body: some View {
        VStack {
```

Image Classification – 現有模型

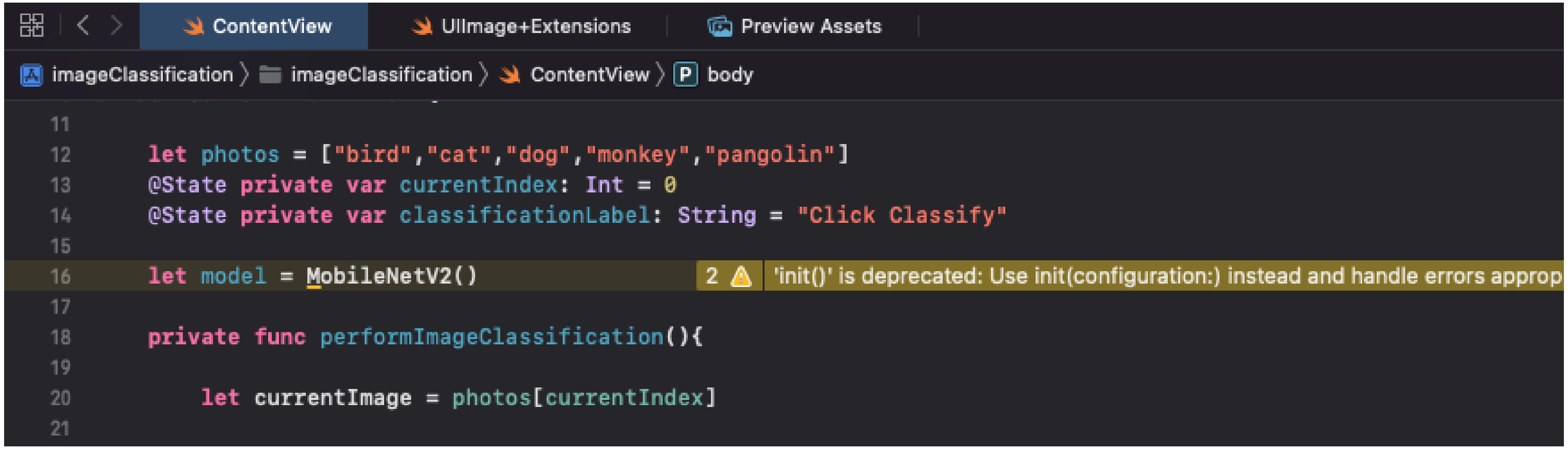
- 8. 在”Calassify”按鈕function中呼叫剛剛的function



```
            }.padding()

            Button("Classify") {
                // classify the image here
                self.performImageClassification()
            }.padding()
                .background(Color(red: 0.7568627450980392, green: 1.0, blue: 0.8431372549019608))
            .cornerRadius(8)

```

Image Classification – 現有模型

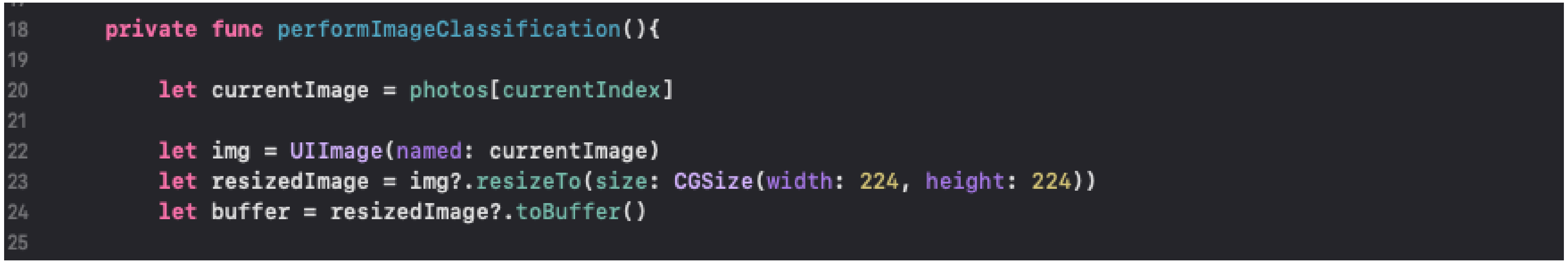
- 9. 宣告currentImage變數選擇目前的照片 photos[currentIndex]


ContentView
UIImage+Extensions
Preview Assets
imageClassification 〉 imageClassification 〉 ContentView 〉 P body

```
    let photos = ["bird","cat","dog","monkey","pangolin"]
    @State private var currentIndex: Int = 0
    @State private var classificationLabel: String = "Click Classify"

    let model = MobileNetV2()          2 ⚠ 'init()' is deprecated: Use init(configuration:) instead and handle errors approp

    private func performImageClassification(){

        let currentImage = photos[currentIndex]

```

Image Classification – 現有模型

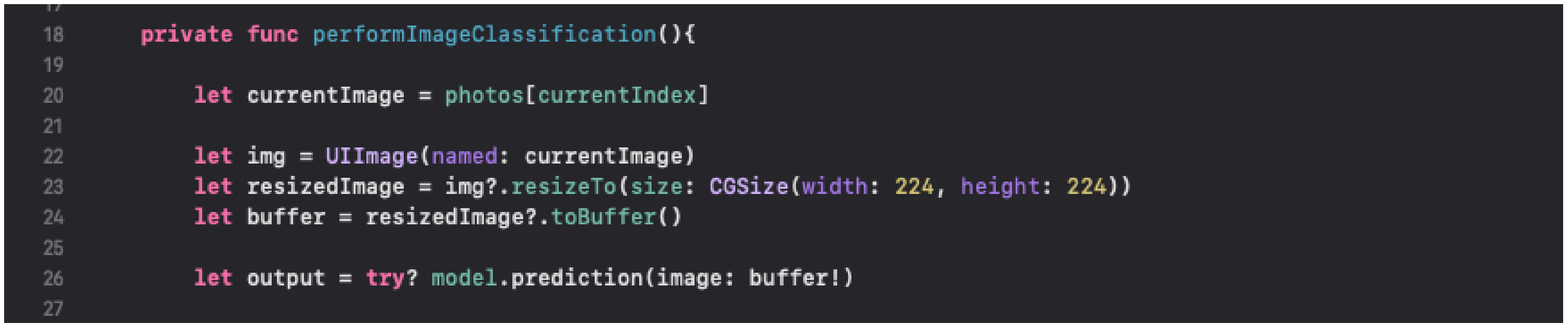
- 10a. img: 取得UIImage， 傳入currentImage
- 10b. resizedImage: 將img縮放成model要求的大小 (224*224)
- 10c. buffer : 使用toBuffer把resizedImage轉換成CVPixelBuffer格式



```
    private func performImageClassification(){

        let currentImage = photos[currentIndex]

        let img = UIImage(named: currentImage)
        let resizedImage = img?.resizeTo(size: CGSize(width: 224, height: 224))
        let buffer = resizedImage?.toBuffer()

```

Image Classification – 現有模型

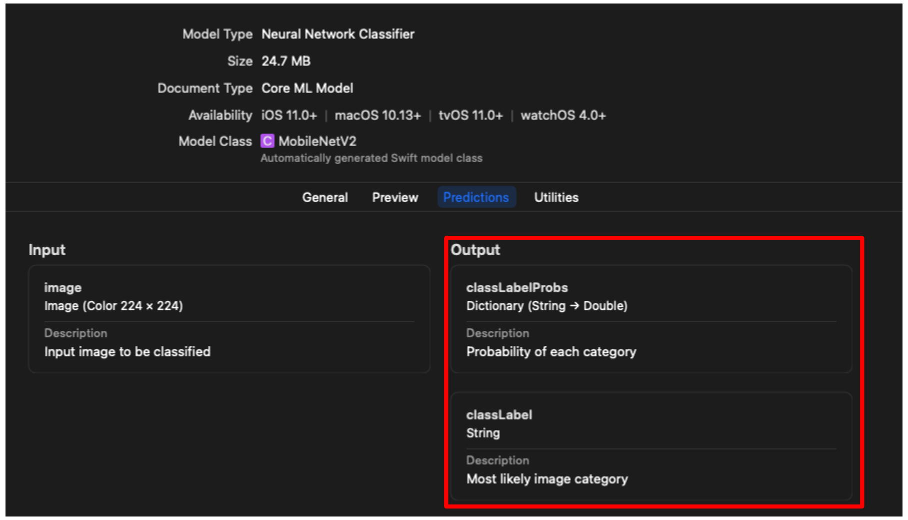
- 11. output: 將轉好的Cvpixelbuffer傳入model.prediction就能取得模型的輸出



```
    private func performImageClassification(){

        let currentImage = photos[currentIndex]

        let img = UIImage(named: currentImage)
        let resizedImage = img?.resizeTo(size: CGSize(width: 224, height: 224))
        let buffer = resizedImage?.toBuffer()

        let output = try? model.prediction(image: buffer!)

```

Image Classification – 現有模型

- 12. 在model的敘述中可以得知output會包含兩種不同的資料：classLabelProbs和classLabel



Model Type Neural Network Classifier
Size 24.7 MB
Document Type Core ML Model
Availability iOS 11.0+ | macOS 10.13+ | tvOS 11.0+ | watchOS 4.0+
Model Class MobileNetV2
Automatically generated Swift model class

Input

image
Image (Color 224 × 224)
Description
Input image to be classified

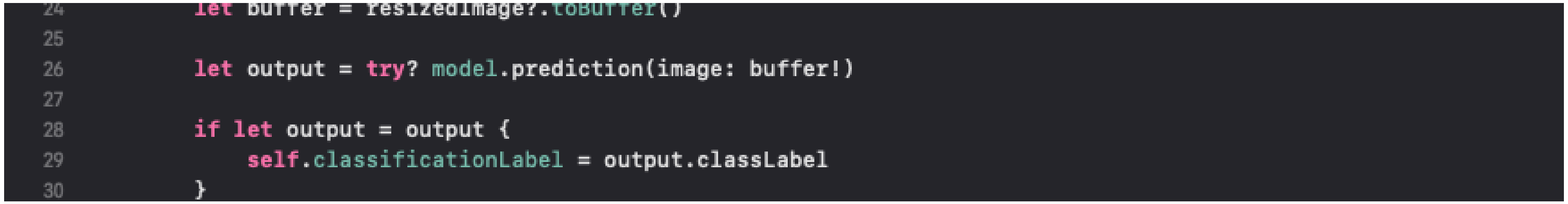
Output

classLabelProbs
Dictionary (String → Double)
Description
Probability of each category

classLabel
String
Description
Most likely image category

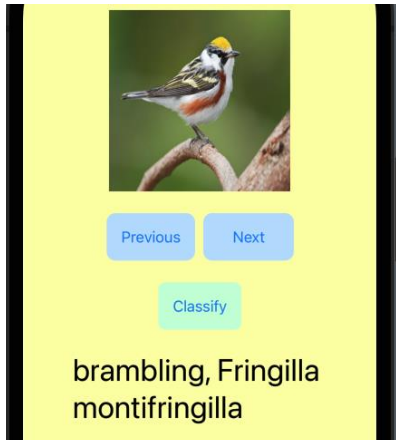
Image Classification – 現有模型

- 12a. 使用classLabel ：模型認為最像的label

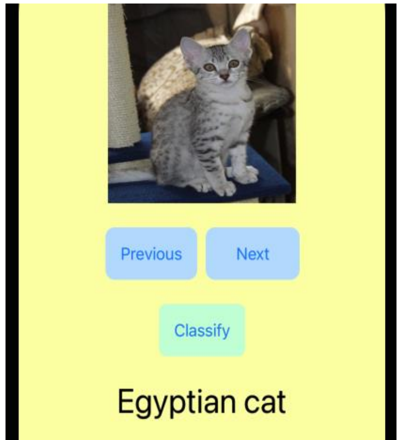


```
        let buffer = resizedImage?.toBuffer()

        let output = try? model.prediction(image: buffer!)

        if let output = output {
            self.classificationLabel = output.classLabel
        }
```

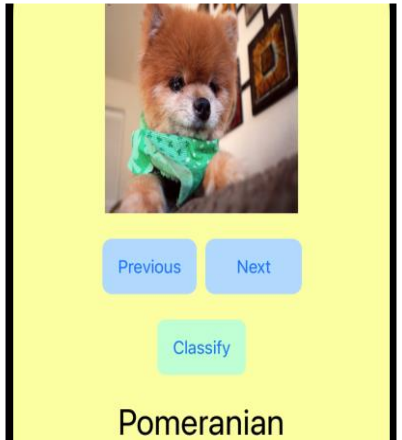
- 執行模擬器就可以點擊“Classify”按鈕對圖像做分類。



brambling, Fringilla montifringilla



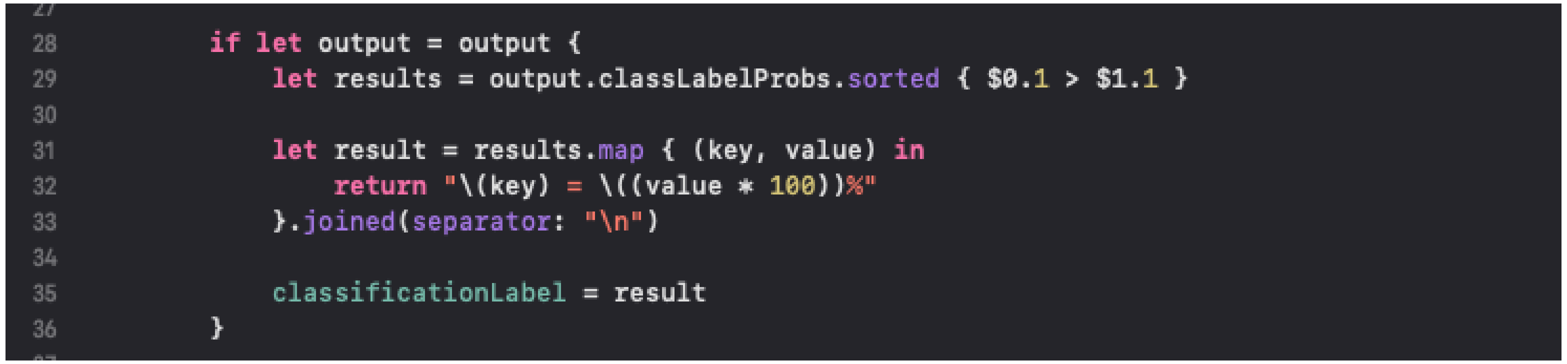
Egyptian cat



Pomeranian

Image Classification – 現有模型

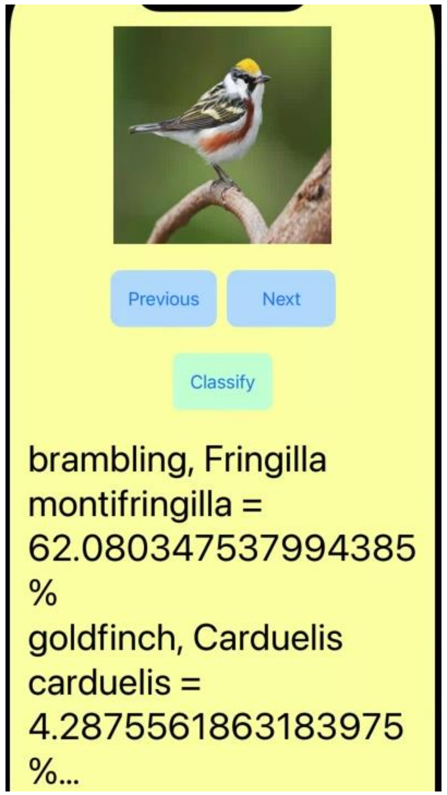
- 12b. 使用classLabelProbs ：每個類組的準確率，格式為Dictionary (“Key” : “value”)
  - 利用sorted 把classLabelProbs重新排列好Key的順序
  - 利用map 把classLabelProbs的印出想要的格式並且回傳給result



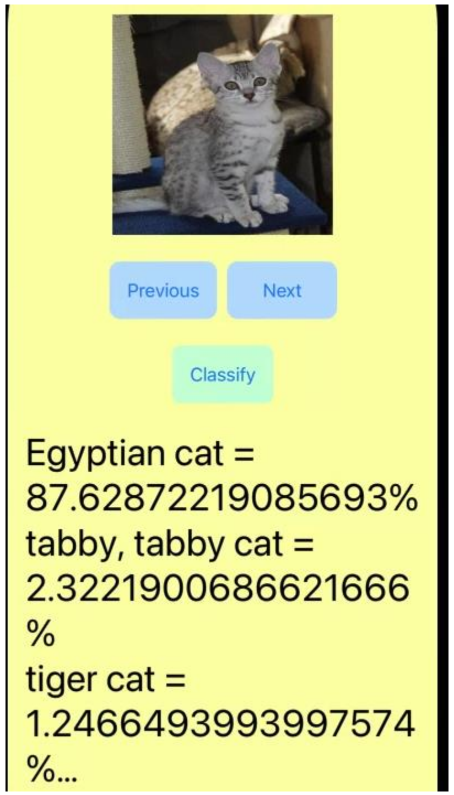
```
        if let output = output {
            let results = output.classLabelProbs.sorted { $0.1 > $1.1 }

            let result = results.map { (key, value) in
                return "\(key) = \((value * 100))%"
            }.joined(separator: "\n")

            classificationLabel = result
        }
```

Image Classification – 現有模型

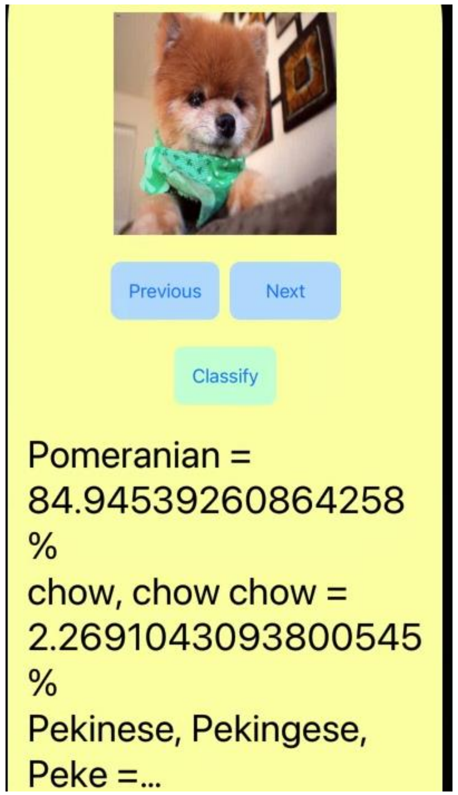
- 12b. 執行模擬器就可以看到模型對圖片的每個類組準確率



brambling, Fringilla montifringilla = 62.080347537994385 %
goldfinch, Carduelis carduelis = 4.2875561863183975 %...



Egyptian cat = 87.62872219085693%
tabby, tabby cat = 2.3221900686621666 %
tiger cat = 1.2466493993997574 %...



Pomeranian = 84.94539260864258 %
chow, chow chow = 2.2691043093800545 %
Pekinese, Pekingese, Peke =...

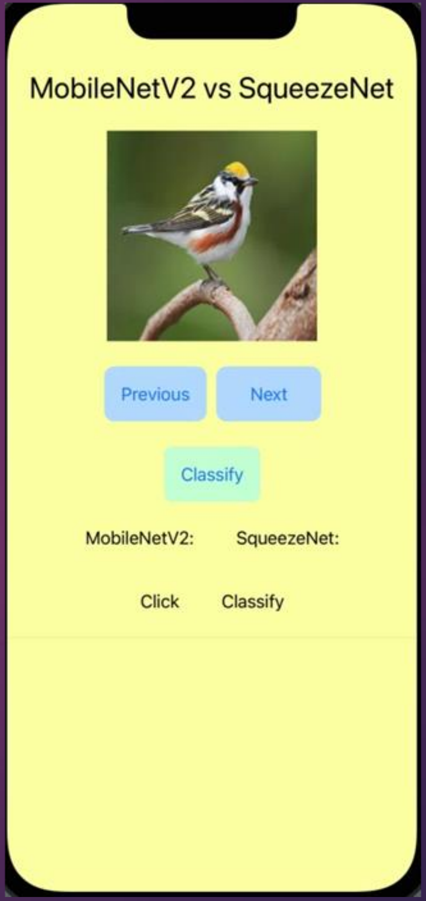
Lab 12
Core ML

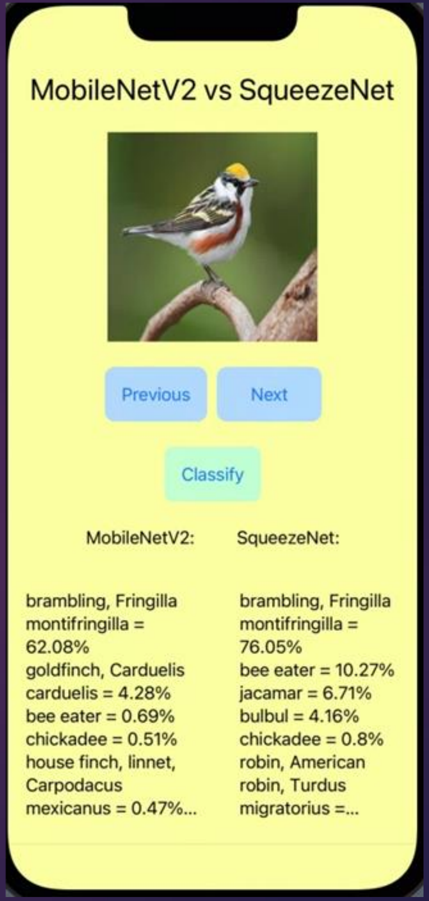
IOS APP DEVELOPMENT

6/15/2022

Task

- 結合Core ML框架實作出擁有圖像分類功能的App
  - 1. 使用兩種模型對圖片做辨識 40%
    - 可使用現有模型、用Create ML訓練的模型
  - 2.顯示模型結果：
    - 對圖片辨識的類別（Label） 30%
    - 對圖片辨識的機率（Probability）30%





MobileNetV2:	SqueezeNet:
brambling, Fringilla montifringilla = 62.08%	brambling, Fringilla montifringilla = 76.05%
goldfinch, Carduelis carduelis = 4.28%	bee eater = 10.27%
bee eater = 0.69%	jacamar = 6.71%
chickadee = 0.51%	bulbul = 4.16%
house finch, linnet, Carpodacus mexicanus = 0.47%...	chickadee = 0.8%
	robin, American robin, Turdus migratorius =...