// import ML Models developer.apple.com/machine-learning // youtube: CoreML with ARKit

places205-GoogLeNet

// ViewController.swift

```
import UIKit
import CoreML
import Vision
class ViewController: UIViewController, UINavigationControllerDelegate, UIImagePickerControllerDelegate{
    private var imagePicker = UIImagePickerController()
    private var model = GoogLeNetPlaces()
    @IBOutlet weak var photoImageView: UIImageView!

    override func viewDidLoad(){
        super.viewDidLoad()
        self.imagePicker.sourceType = .photoLibrary
        self.imagePicker.delegate = self


    }

func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info:[String:Any]){
    dismiss(animated: true, completion: nil)
    guard let pickedImage = info[UIImagePickerControllerOriginalImage] as? UIImage else{
        return
    }
    self.photoImageView.image = pickedImage
    processImage(image: pickedImage)
}

private func processImage(image: UIImage){
    guard let ciImage = CIImage(image: image) else{
        fatalError("Unable to create the ciImage object")
    }
    guard let visionModel = try? VNCoreMLModel(for: self.model.model) else{
        fatalError("Unable to create vision model")
    }
    let visionRequest = VNCoreMLRequest(model: visionModel){ request, error in   // completion block
        if error != error{
            return
        }
        guard let results = request.result as?[VNClassificationObservation] else{
            return
        }
        let classifications = results.map{ observation in   // array of strings
            "\(observation.identifier) \(observation.confidence * 100)"
        }
        DispatchQueue.main.async{
            self.descriptionTextView.text = classifications.joined(separator: "\n")
        }
    }
```

```
let visionRequestHandler = VNImageRequestHandler(ciImage: ciImage, orientation: .up, options: [:])
DispatchQueue.global(qos: .userInteractive).async{
    try! visionRequestHandler.perform([visionRequest])
}
}

@IBAction func openPhotoLibraryButtonPressed(){
    self.present(self.imagePicker, animated: true, completion: nil)

}

}
```

quality of service

on the main thread