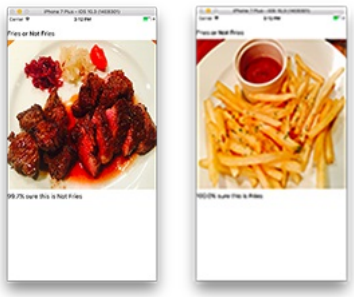
Create a Swift iOS app using Microsoft Custom Vision

With Microsoft's Custom Vision Service, it's easy and fast to build, deploy, and improve your own custom image classifier. A custom image classifier can help you identify images of "Bicycles" vs "Cars" or "Steak" vs "Fries".

After you create your custom image classifier, it's easy to build into any app by using a REST API provided by the Custom Vision service. In addition, you can improve your classifier overtime by using the web interface to upload images, tag images from end users and re-train your model.



This lab includes the following tasks:

- Creating a new image classifier project
- Creating your iOS app in Xcode and Swift
- Improving your classifier over time

Requirements

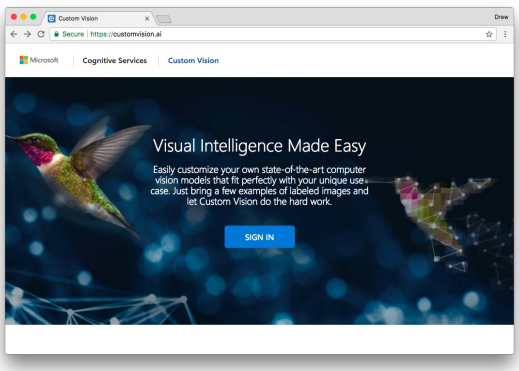
The following will be required for you to complete this lab:

- Microsoft Account (MSA) for signing into the Custom Vision service
- At least two sets of images ready to be tagged. Each set should have at least 30 images. (You can use the provided images in the source directory)
- Additional images for testing your model.
- Software: Xcode
- Optional: iPhone to test your app on a device

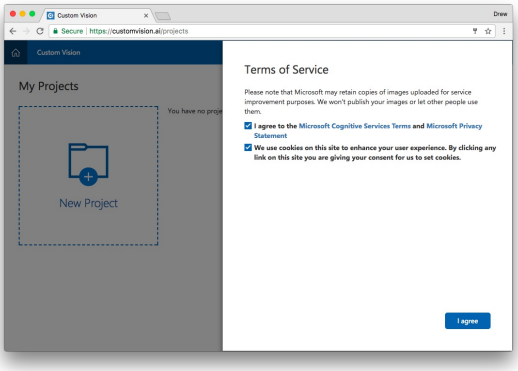
Creating a new image classifier project

New Project

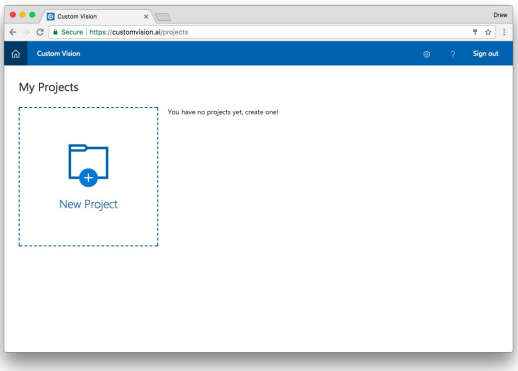
1. Open a web browser to http://customvision.ai and sign in using your Microsoft Account (MSA)



2. If this is your first time using the service, agree to the Terms of Service by checking the boxes and clicking I agree.

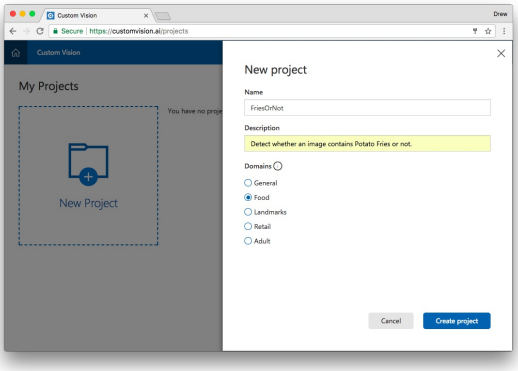


3. Click **New Project** to create your first project.



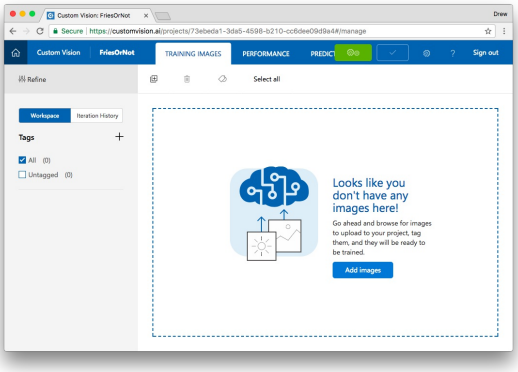
4. Enter a name and description for your project.

5. Select one of the provided Domains, or if you are not sure use General.

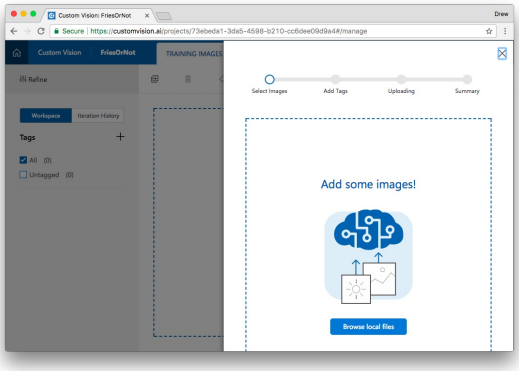


Add Images

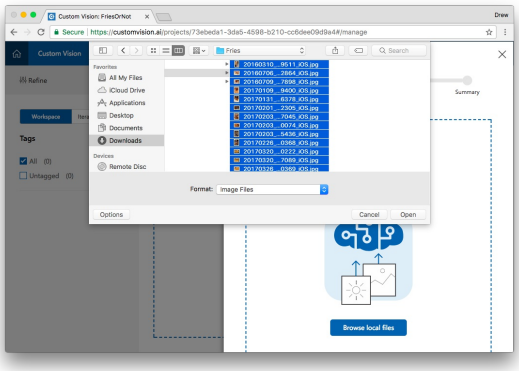
1. Click **Add images** to add images to your project.



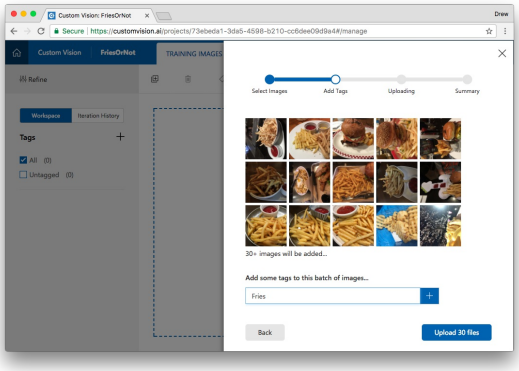
2. Click **Browse local files** to select images from your local disk.



3. Select at least 30 images for your first Tag set. Click **Open**.

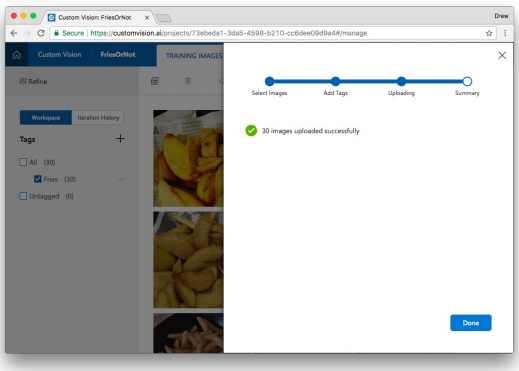


4. Type in a Tag and press the **+** button. You can add additional tags.



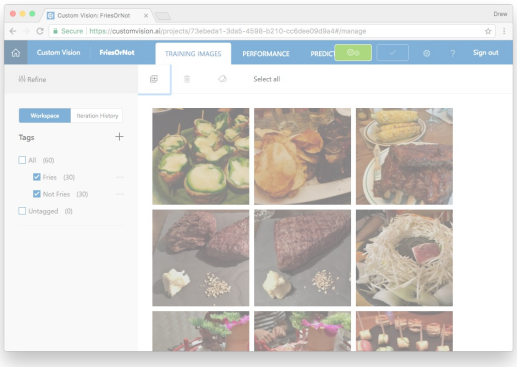
5. Now click **Upload ## files**

6. Repeat these steps for at least one more Tag set.

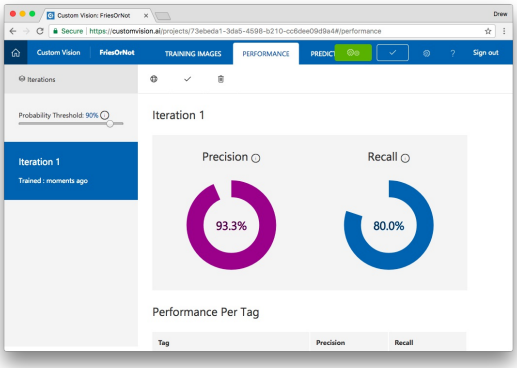


Train your classifier

1. At the top of the window, find the green button called **Train**. Click it!



2. This is all it takes to train your new custom image classifier.
3. Once trained, you will be shown precision and recall indicators which tell you about the performance of your classifier.



> **Precision** - How likely is a tagged image to be correct?
>
> **Recall** - How often did your classifier tag correctly?

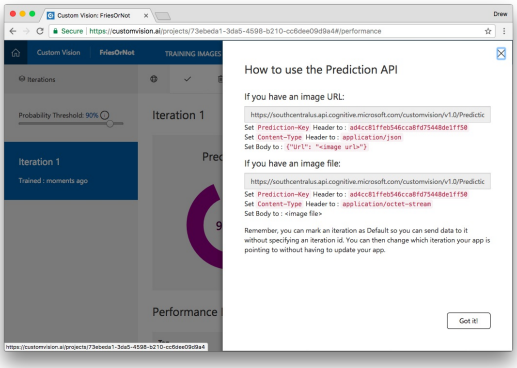
4. If you are happy with this iteration of your classifier, click the Check mark to make this the default.

Test your model

1. At the top of the window, find another button called **Quick Test**
2. Either enter a URL of an image, or **Browse local files** to select an image from your local disk.
3. The Custom Vision service will now use your image classifier model to evaluate this new image and give you a prediction.

Collect detail for your Prediction API

1. Click the **Performance** tab again.
2. At the top of this window, click "Prediction URL".
3. Copy the Prediction URL for uploading an Image.
4. Copy the Prediction Key for accessing the API.

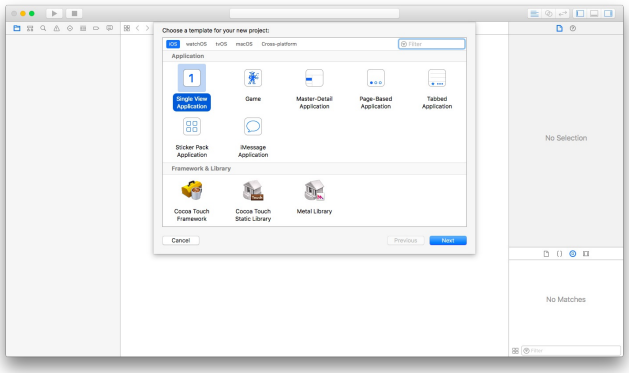


5. Paste these details to a Note for later use.

Creating your iOS app in Xcode and Swift

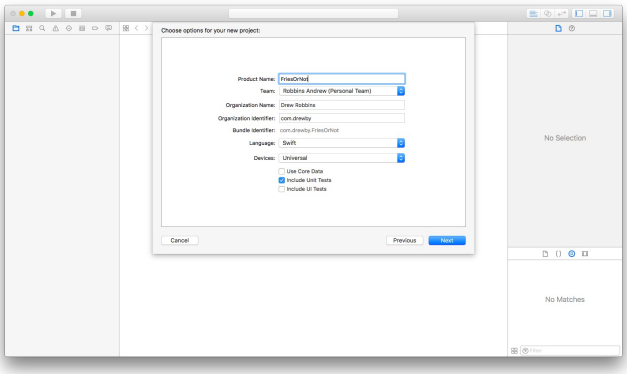
New iOS Project

1. Open Xcode from the /Applications directory.
2. Select **Create a new Xcode project** (or choose **File** > **New** > **Project**)
3. Select iOS at the top of the dialog.

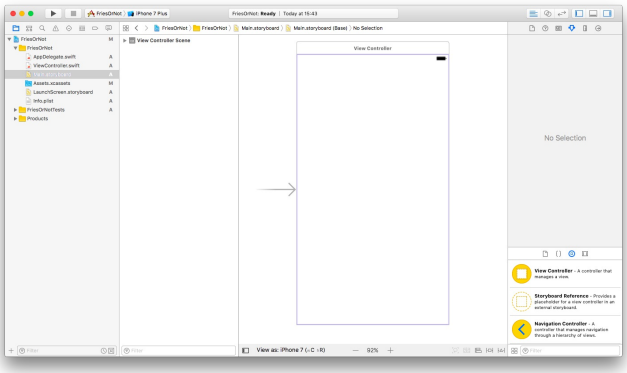


4. In the **Application** section, select **Single View Application** and then click **Next**.
5. Use the following values to name your app and choose additional options:

> **Product Name**: FriesOrNot
>
> **Team**: None
>
> **Organization Name**: Your name
>
> **Organization Identified**: Your org identifier, or com.example.
>
> **Language**: Swift
>
> **Devices**: Universal
>
> **Use Core Data**: Unselected
>
> **Include Unit Test**: Selected
>
> **Include UI Tests**: Unselected.

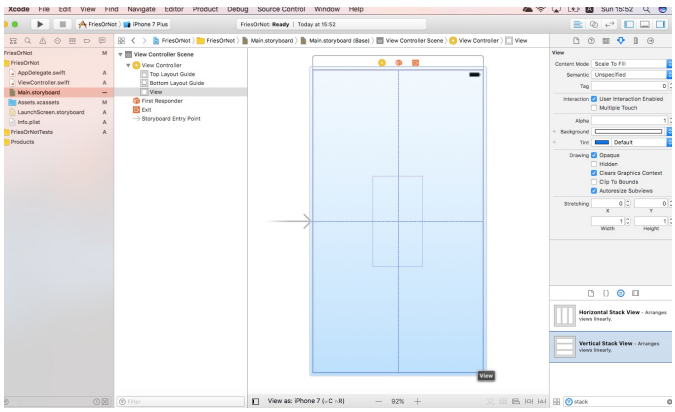


6. Click **Next**.
7. Select a location to save your project and click **Create**.
8. Click the **Run** button to make sure the empty project works.

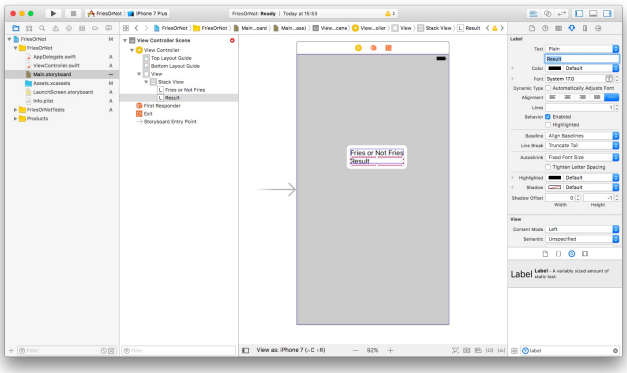


Add UI objects to Storyboard

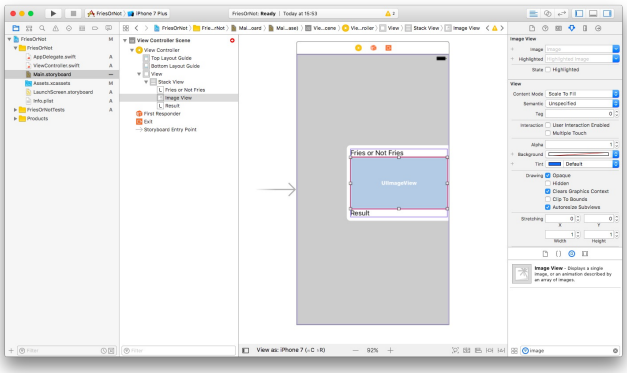
1. Open **Main.storyboard** by selecting it from Project Navigator
2. Open the Object library in the utility area (or choose **View** > **Utilities** > **Show Object Library**)
3. In the Object library, filter for "stack view".
4. Drag the **Vertical Stack View** to your scene.



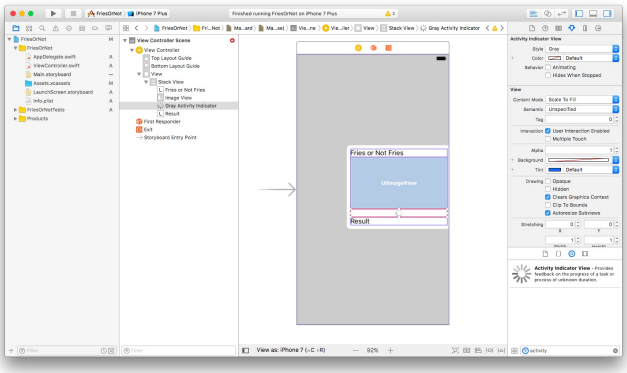
5. Now filter for "label" in Object library.
6. Drag a **Label** and drop it onto the Stack View.
7. Change the text for the label to "Fries or Not Fries".
8. Drag another **Label** to the Stack View and drop it below the previous one.
9. Change the text for this label to "Result".



10. Now filter for "image view" in Object library.
11. Drag an **Image View** to the Stack View and drop it in between the two labels.

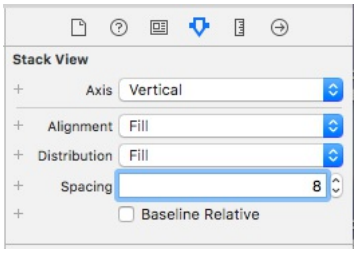


12. Now filter for "activity" in Object library.
13. Drag an **Activity Indicator View** to the Stack View and drop just below the image view.

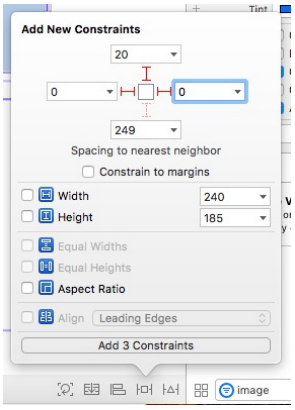


Set UI layout properties

1. Use the storyboard scene, or click in the Outline view to select the Stack View.
2. In the Attributes inspector, set the Spacing to 8.

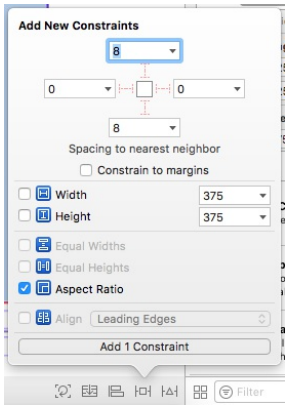


3. Click the Constraints button at the bottom of the canvas window.
4. Above **Spacing to nearest neighbor**, enter 20 for the Top and 0 for the Left and Right boundaries.
5. Click **Add 3 Constraints**

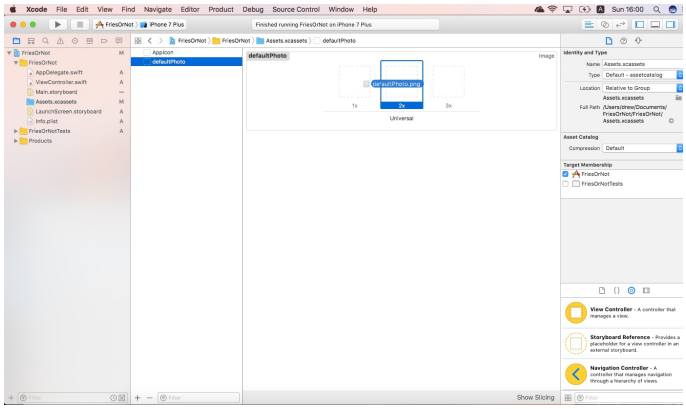


6. Now select the Image View.

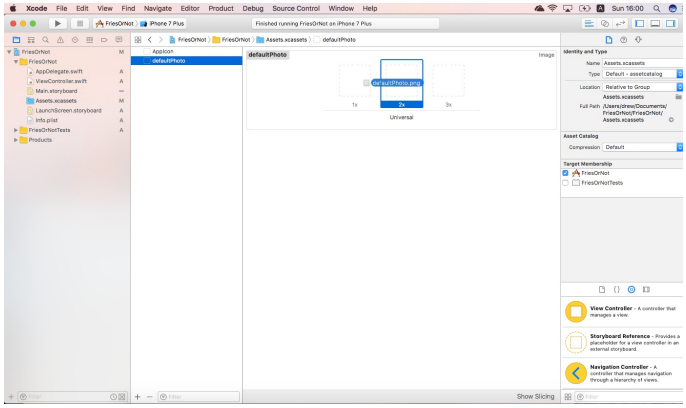
7. In the Attributes inspector, set the Height to the same as the Width (~350)



8. Click the Constraints button.
9. Check the box **Aspect Ratio** and click **Add 1 Constraint**
10. Open Assets.xcassets by clicking in the Project Navigator.
11. Click the **+** button to add a **New Image Set**
12. Double-click the new image set and rename to "defaultPhoto"
13. Drag the defaultPhoto.png from source files to the 2x slot in the image set.



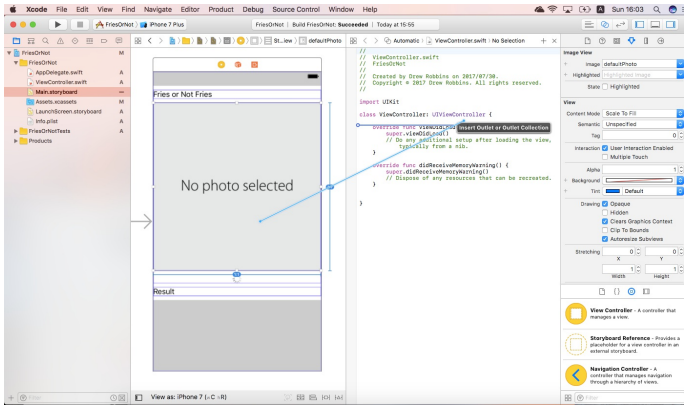
14. Select the Image View in the canvas again.
15. Set the **Image** attribute to defaultPhoto.



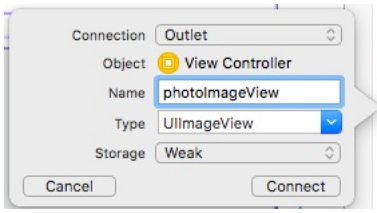
16. Check the **User Interaction Enabled** attribute.
17. Now in filter for "tap" in the Object library
18. Drag the tap gesture and drop on top of the Image View

Connect UI objects to code

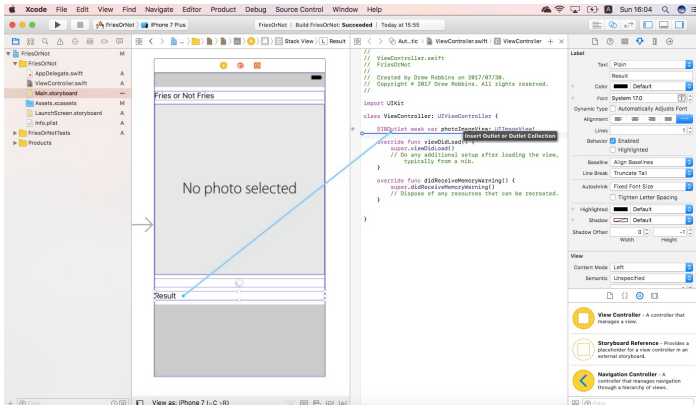
1. Click to show the Assistant editor. You should now see the Storyboard and Viewcontroller side-by-side.
2. Hold the Control-key and drag the result label to your ViewController just under the line starting with "class".



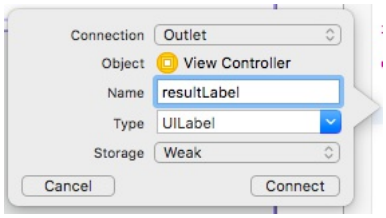
3. Type "photoImageView" for the **Name** and click **Connect**.



4. Hold the Control-key and drag the activity indicator to your ViewController just under the previous photoImageView definition.

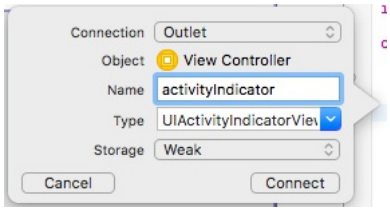


5. Type "resultLabel" for the **Name** and click **Connect**.



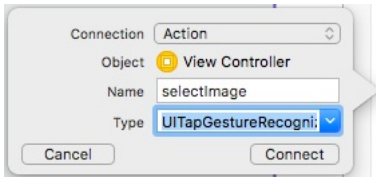
6. Hold the Control-key and drag the image view to your ViewController just under the previous resultLabel definition.

7. Type "activityIndicator" for the **Name** and click **Connect**.



8. Hold the Control-key and drag the Tap Gesture (from top of canvas) to your ViewController just above the closing curly brace.

9. Change the **Connection** to **Action** and the **Type** to **UITapGestureRecognizer**. **Name** the new action "selectImage"



Write code for Image Picker

1. Click to show Standard Editor
2. Make sure ViewController.swift is open in editor.
3. Change the ViewController class defintion to include two more implementations:

```
class ViewController: UIViewController,
                      UIImagePickerControllerDelegate,
                      UINavigationControllerDelegate {
```

4. Add the following code to the **selectImage** function:

```
    let imagePickerController = UIImagePickerController()
        imagePickerController.sourceType = .photoLibrary

        imagePickerController.delegate = self

        present(imagePickerController, animated: true, completion: nil)
```

5. Add the following function to ViewController:

```
func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
    dismiss(animated: true, completion: nil)
}
```

6. Add the following function to ViewController:

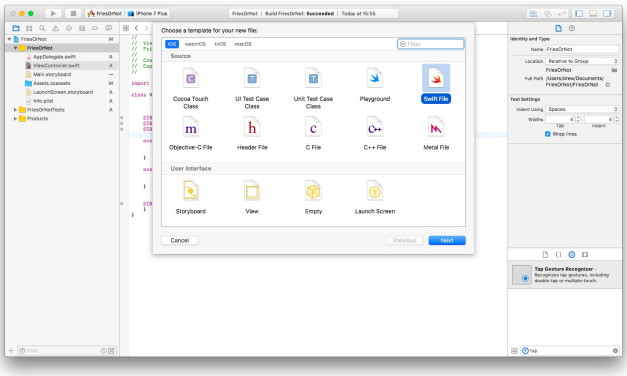
```
func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any]) {
    guard let selectedImage = info UIImagePickerControllerOriginalImage] as? UIImage else {
        fatalError("Expected an image, but was provided \(info)")
    }

    photoImageView.image = selectedImage
    dismiss(animated: true, completion: nil)
}
```

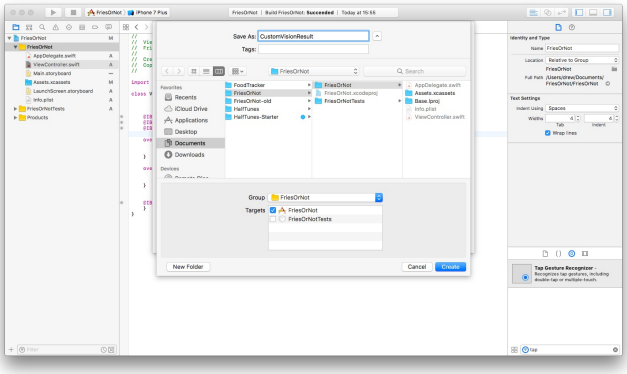
7. Open Info.plist
8. At the bottom of the list, click the **+** button.
9. Find "Privacy - Photo Library Usage"
10. Add a description in the Value column such as "See if photos in your library contain Fries or Not Fies."
11. Click the **Run** button to make sure the project works. Try tapping the Image View to select a photo from the photo library.

Create an object data model for Custom Vision

1. In Project Navigator, right-click the folder FriesOrNot and click **New File...**
2. Select **Swift File** and click **Next**.



3. Name the file "CustomVisionResult" and click **Create**..



4. In this file, create the following class:

```
struct CustomVisionPrediction {
    let TagId: String
    let Tag: String
    let Probability: Float

    public init(json: [String: Any]) {
        let tagId = json["TagId"] as? String
        let tag = json["Tag"] as? String
        let probability = json["Probability"] as? Float

        self.TagId = tagId!
        self.Tag = tag!
        self.Probability = probability!
    }
}
```

5. Now to the same file, add the following class:

```
struct CustomVisionResult {
    let Id: String
    let Project: String
    let Iteration: String
    let Created: String
    let Predictions: [CustomVisionPrediction]

    public init(json: [String: Any]) throws {
        let id = json["Id"] as? String
        let project = json["Project"] as? String
        let iteration = json["Iteration"] as? String
        let created = json["Created"] as? String

        let predictionsJson = json["Predictions"] as? [[String: Any]]

        var predictions = [CustomVisionPrediction]()
        for predictionJson in predictionsJson! {
            do
            {
```

```
                let cvp = CustomVisionPrediction(json: predictionJson)
                predictions.append(cvp)
            }
        }

        self.Id = id!
        self.Project = project!
        self.Iteration = iteration!
        self.Created = created!
        self.Predictions = predictions
    }
}
```

Create a service to call Custom Vision

1. In Project Navigator, right-click the folder FriesOrNot and click **New File...**
2. Select **Swift File** and click **Next**.
3. Name the file "CustomVisionService" and click **Create**..
4. In this file, create the following class:

```
class CustomVisionService {
    var preductionUrl = ""
    var predictionKey = ""
    var contentType = "application/octet-stream"

    var defaultSession = URLSession(configuration: .default)
    var dataTask: URLSessionDataTask?

    func predict(image: Data, completion: @escaping (CustomVisionResult?, Error?) -> Void) {

        // Create URL Request
        var urlRequest = URLRequest(url: URL(string: preductionUrl)!)
        urlRequest.addValue(predictionKey, forHTTPHeaderField: "Prediction-Key")
        urlRequest.addValue(contentType, forHTTPHeaderField: "Content-Type")
        urlRequest.httpMethod = "POST"

        // Cancel existing dataTask if active
        dataTask?.cancel()

        // Create new dataTask to upload image
        dataTask = defaultSession.uploadTask(with: urlRequest, from: image) { data, response, error in
            defer { self.dataTask = nil }

            if let error = error {
                completion(nil, error)
            } else if let data = data,
                let response = response as? HTTPURLResponse,
                response.statusCode == 200 {

                if let json = try? JSONSerialization.jsonObject(with: data) as? [String: Any],
                    let result = try? CustomVisionResult(json: json!) {
                    completion(result, nil)
                }
            }
        }

        // Start the new dataTask
        dataTask?.resume()
    }
}
```

5. To complete this class, you need to provide your Prediction URL and Prediction Key we copied earlier.
6. Copy the Prediction URL and paste in the quotes next to **preductionUrl**.
7. Copy the Prediction Key and paste in the quotes next to **predictionKey**.

Update ViewController to use CustomVisionService

1. Make sure ViewController.swift is open in the Standard Editor.
2. After the IBOutlet properties, add another property:

```
var service = CustomVisionService()
```

3. Add the following code to imagePickerController:

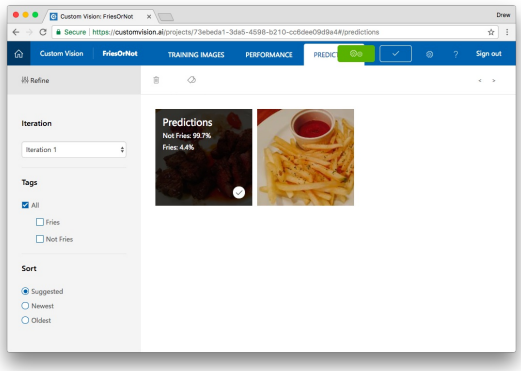
```
resultLabel.text = ""
self.activityIndicator.startAnimating()

let imageData = UIImageJPEGRepresentation(selectedImage, 0.8)!
service.predict(image: imageData, completion: { (result: CustomVisionResult?, error: Error?) in
    DispatchQueue.main.async {
        self.activityIndicator.stopAnimating()
        if let error = error {
            self.resultLabel.text = error.localizedDescription
        } else if let result = result {
            let prediction = result.Predictions[0]
            let probabilityLabel = String(format: "%.1f", prediction.Probability * 100)
            self.resultLabel.text = "\(probabilityLabel)% sure this is \(prediction.Tag)"
        }
    }
})
```

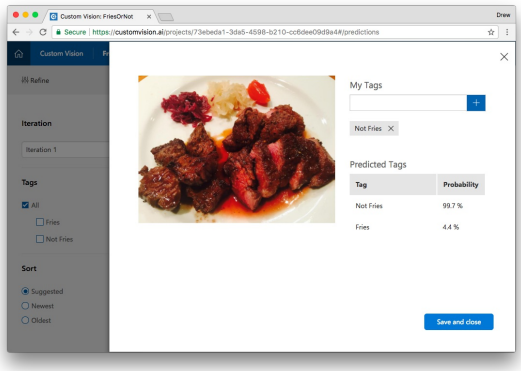
4. Click the **Run** button to test your project. Now when you pick an Image, the app should request a prediction from Custom Vision. Once returned, it will display the Tags and confidence for each Tag.

Improving your classifier over time

1. Open a web browser to http://customvision.ai and sign in using your Microsoft Account (MSA).
2. Click the **Predictions** tab
3. On this tab, you will see images that were submitted for Predictions.
4. If you hover your mouse cursor over an image, you'll see how the confidence for each Tag for that image.



5. If you would like to further train your algorithm, click the image.
6. Add a Tag and click **Save and close**



7. Click the green **Train** button to retrain your model using your new inputs.