

# Snack Classifier开发实验报告

学号	姓名
191220162	张乐简

## Snack Classifier开发实验报告

- 概述
- 实验内容
  - Snack Classifier
    - 训练模型
    - 使用模型
  - HealthyOrNotHealthySnackClassifier
    - 构造数据
    - 使用模型
- 反思

## 概述

该实验利用CreateML和提供的零食数据集训练出一个零食分类器，同时在代码中使用它来判断给定照片的零食种类。另外，也要求改造数据集，再训练一个健康/非健康食品分类器。

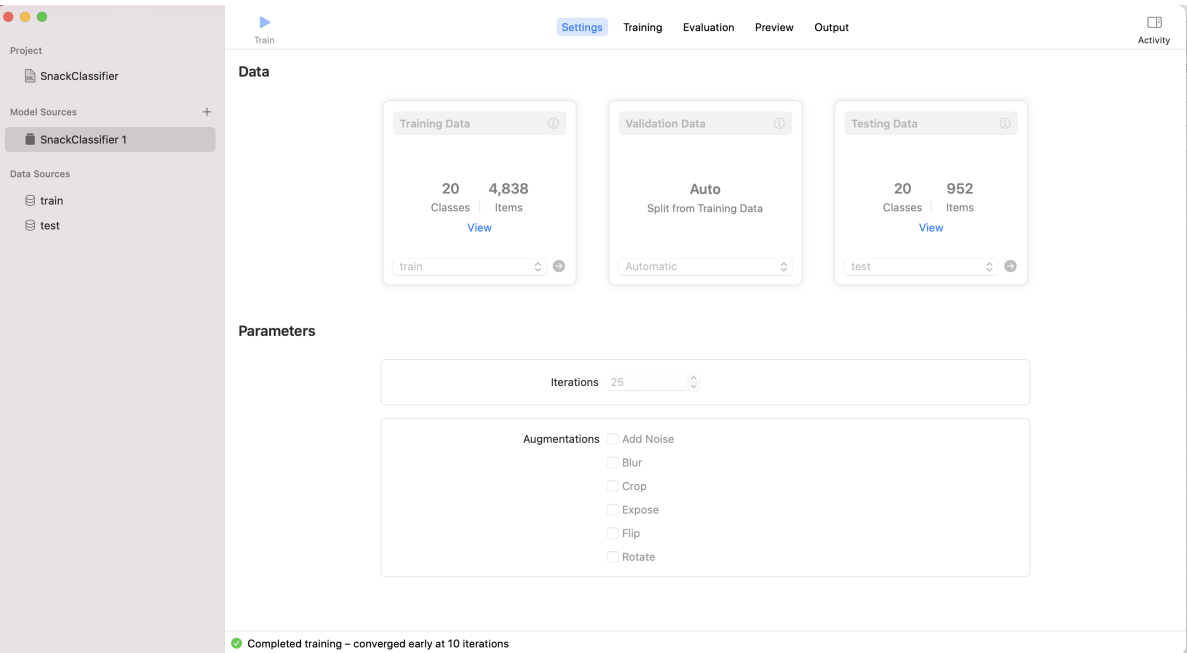
## 实验内容

### Snack Classifier

#### 训练模型

简单地使用CreateML即可。

训练过程：



训练结果：

The screenshot shows the SnackClassifier app interface. On the left, there's a sidebar with 'Project' (SnackClassifier), 'Model Sources' (SnackClassifier 1), and 'Data Sources' (train, test). The main area has tabs for 'Settings', 'Training', 'Evaluation' (selected), 'Preview', and 'Output'. Under 'Evaluation', there's a 'test' section showing results for Dec 10, 2021 at 7:20 PM. A table displays classification results for 20 classes. At the bottom, a green status bar indicates 'Completed training - converged early at 10 iterations'.

Class	Item Count	Precision	Recall
apple	50	51%	84%
banana	50	89%	94%
cake	50	81%	76%
candy	50	87%	90%
carrot	50	86%	88%
cookie	50	73%	88%
doughnut	50	96%	88%
grape	50	98%	90%
hot dog	50	96%	92%
ice cream	50	95%	78%
juice	50	98%	96%
muffin	48	91%	85%
orange	50	93%	76%
pineapple	40	95%	90%
popcorn	40	97%	90%
pretzel	25	96%	92%
salad	50	88%	90%
strawberry	49	93%	86%
waffle	50	100%	94%
watermelon	50	96%	94%

## 使用模型

首先，需要创建模型，并为模型提供输入。创建模型这一部分已经在框架代码中实现，不再赘述。在演示课件中，为模型提供的输入类型为CMSampleBuffer，而框架最后为我提供的参数类型为UIImage。开始时我在StackOverflow上寻得了将UIImage转换为CMSampleBuffer的方法并使用，但发现准确率很低，于是转而直接将输入参数的cglImage成员作为模型的输入。尽管后来发现准确率低似乎与CMSampleBuffer无关，但考虑到cglImage已经有不错的表现，便不再切换回去。

代码如下：

```
DispatchQueue.main.async {
    let handler = VNImageRequestHandler(cgImage:
sampleBuffer.cgImage!,
orientation: CGImagePropertyOrientation(sampleBuffer.imageOrientation))
    do {
        try handler.perform([self.classificationRequest])
        try handler.perform([self.huhclassificationRequest])
    } catch {
        print("Failed to perform classification: \(error)")
    }
    self.semaphore.signal()
}
```

得到模型输出的判断后，将该判断根据confidence进行筛选，决定显示它还是显示“我不确定”。代码如下。

```
func processObservations(for request: VNRequest, error: Error?) {
    if let results = request.results as? [VNClassificationObservation] {
        if results.isEmpty {
            self.resultsLabel.text = "Nothing found\n"
        } else {
            let result = results[0].identifier
            let confidence = results[0].confidence
            if confidence < 0.7 {
                self.resultsLabel.text = "我不确定" + result + "\n"
            }
        }
    }
}
```

```

self.resultsLabel.text=self.resultsLabel.text!+String(format: "%.1f%%",
confidence * 100)
}else{
self.resultsLabel.text = result+"\n"

self.resultsLabel.text=self.resultsLabel.text!+String(format: "%.1f%%",
confidence * 100)
}
}
} else if let error = error {
self.resultsLabel.text = "Error: \(error.localizedDescription)"
} else {
self.resultsLabel.text = "???"
}
showResultsView(delay: 0.5)
}

```

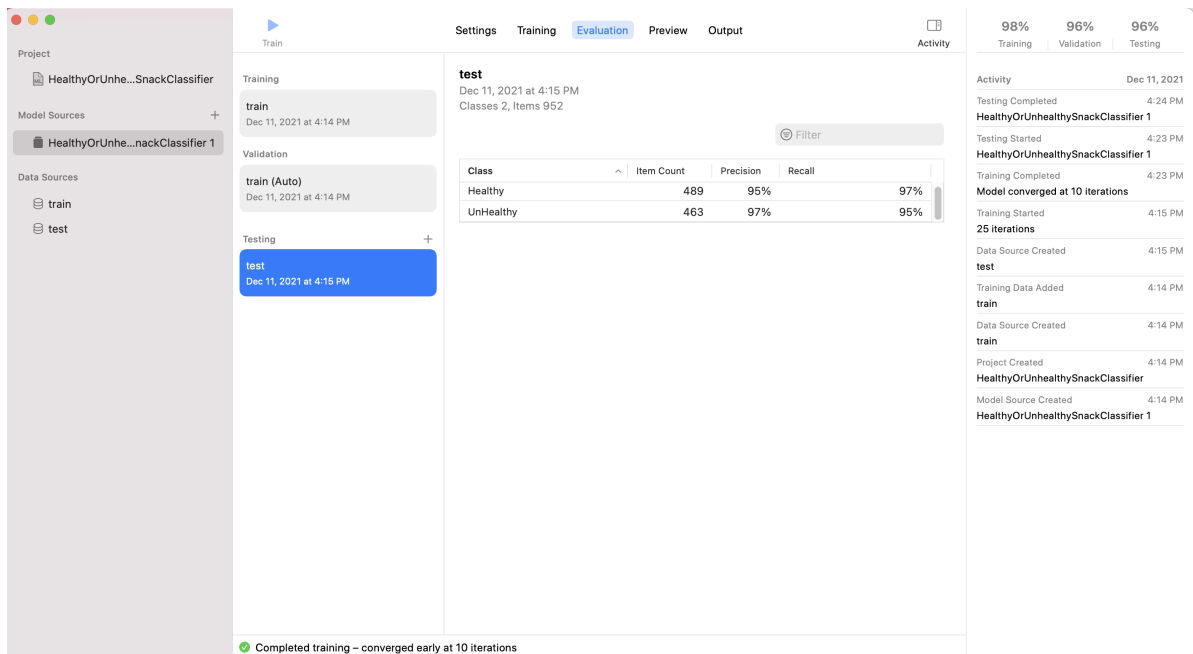
## HealthyOrNotHealthySnackClassifier

### 构造数据

将apple,banana,carrot,grape,juice,orange,pineapple,salad,strawberry,watermelon归为Heathy, 其余类别的食物归类为Unhealthy。将对应种类的照片从snack文件夹中对应类别下提取出来, 整合成 Healthy和Unhealthy两个大文件夹, 用于训练; 对测试集采用同样处理。这样得到训练数据与测试数据后, 再次使用CreateML进行训练, 得到模型。

训练过程与上一部分一致。

The screenshot displays the Apple CreateML application interface. On the left sidebar, the project is named 'HealthyOrUnhe...SnackClassifier' and the model source is 'HealthyOrUnhe...nackClassifier 1'. The data sources are 'train' and 'test'. The main area shows the 'Data' section with three data sets: Training Data (2 classes, 4,838 items), Validation Data (Auto, Split from Training Data), and Testing Data (2 classes, 952 items). The 'Parameters' section shows 'Iterations' set to 25 and 'Augmentations' with options like Add Noise, Blur, Crop, Expose, Flip, and Rotate. The 'Activity' log on the right shows the training process, including 'Testing Completed', 'HealthyOrUnhealthySnackClassifier 1', 'Testing Started', 'HealthyOrUnhealthySnackClassifier 1', 'Training Completed', 'Model converged at 10 iterations', 'Training Started', '25 iterations', 'Data Source Created', 'test', 'Training Data Added', 'train', 'Data Source Created', 'train', 'Project Created', 'HealthyOrUnhealthySnackClassifier', 'Model Source Created', and 'HealthyOrUnhealthySnackClassifier 1'. The status at the bottom indicates 'Completed training - converged early at 10 iterations'.



## 使用模型

使用模型的代码与SnackClassifier的使用方法基本一致，不再赘述。只是由于需要单独处理它的结果，另写一个processObservation函数来处理它的判断结果。

代码如下：

```
func processObservations2(for request: VNRequest, error: Error?) {
    if let results = request.results as? [VNClassificationObservation] {
        self.resultsLabel.text=self.resultsLabel.text!+"\n"+"健康判断: "
        if results.isEmpty {

        } else {
            let result = results[0].identifier
            let confidence = results[0].confidence
            if confidence<0.7{
                self.resultsLabel.text=self.resultsLabel.text!+"我不确定\n"

                self.resultsLabel.text=self.resultsLabel.text!+String(format: "%.1f%%",
confidence * 100)
            }else{
                self.resultsLabel.text = self.resultsLabel.text!+result+"\n"

                self.resultsLabel.text=self.resultsLabel.text!+String(format: "%.1f%%",
confidence * 100)
            }
        }
    } else if let error = error {
        self.resultsLabel.text = "Error: \(error.localizedDescription)"
    } else {
        self.resultsLabel.text = "???"
    }
}
```

## 反思

在模型训练结束后，我立刻试图将其放入代码中使用，但无论是将图片转为CVPixelBuffer输入模型，还是直接使用cglImage作为参数传入模型，判断的结果都出现明显的错误。开始时，我以为是模型训练问题，但发现模型在Preview中表现很好；而且，模型在Preview中对一张图片的判断和模拟器中对同一张图片的判断完全不同，在这方面四处搜寻解决方案无果，花费大量时间。后来发现此系模拟器问题，在真机上运行模型则可以正常运行。以后需要更重视运行环境对代码的影响。