8장 Camera & CoreML

학습목표

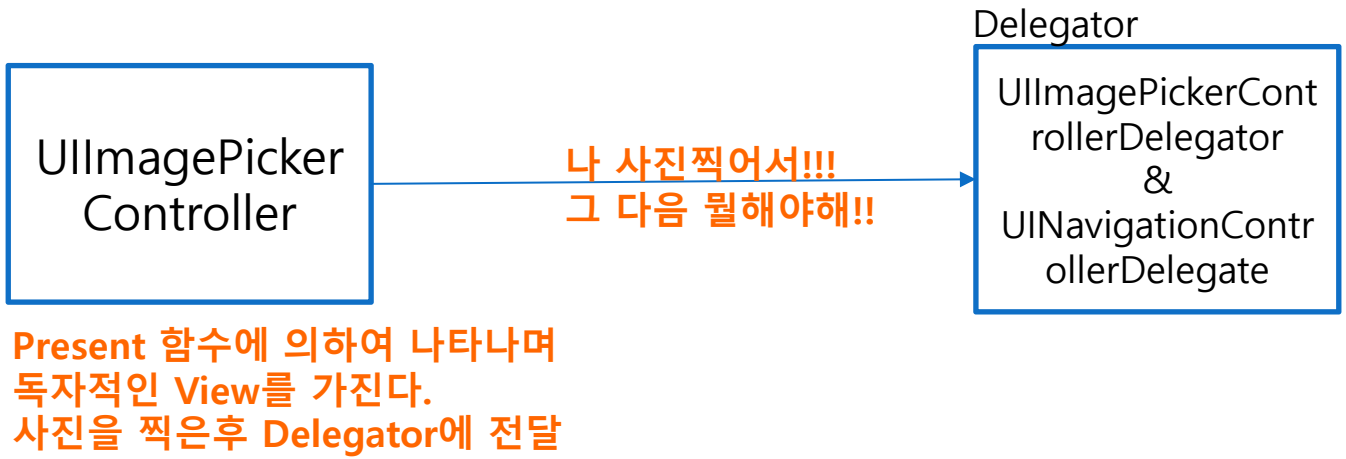
- 카메라 관련 API를 이해한다
- 카메라의 Still Image 캡쳐를 위한 UIImagePickerController를 사용할 수 있다.
- 카메라의 Live Image 캡쳐를 위한 API 사용할 수 있다
- CoreML의 개념을 이해한다
- 이미지 분류 CoreML 모델을 활용할 수 있다.
- 오브젝트 탐지 CoreML 모델을 활용할 수 있다.

Camera

- **iOS에서 카메라 사용 모드**
  - Still Image 캡쳐
    - iOS에서 제공하는 UIImagePickerController을 사용하여 이미지를 캡쳐하는 방법
    - 단순히 UIImagePickerController을 활성화 하면 된다. 프로그램이 매우 쉽다.
    - 최종적으로 촬영된 이미지에만 추가적인 연산을 할 수 있다
  - Live Image 캡쳐
    - 카메라를 통하여 영상에 나타나는 이미지에 추가적인 연산을 할 수 있다.
    - iOS에서 제공하는 카메라 API를 직접 사용하여야 한다. 프로그램이 복잡하다
      - AVCapture Device Input: 비디오/오디오 디바이스와 연결하는 객체
      - AVCapture Output: input으로 부터 받은 데이터를 처리하는 객체
      - AVCapture Session: Device Input와 Output를 연결해주는 객체

Still Image 캡쳐

- **UIImagePickerController**
  - Image를 관리하는 ViewController(별도의 UIView를 가지고 있음)이다.
  - iOS 라이브러리에서 제공하는 것으로 카메라, 앨범 등을 액세스한다.
  - 이것은 UIImagePickerControllerDelegate와 UINavigationControllerDelegate를 요구한다.
    - 즉 UIImagePickerController의 상태 변화가 생기면 이 Delegator에 알려준다.



Still Image 캡쳐

- **XXXController의 Launching**
  - 이것은 하나의 ViewController이다. 별도의 UIView를 가지고 있다
  - ViewController를 Launching하는 여러 방법이 있다(나중에 설명함)
  - 여러 방법중 하나가
    - present(xxxController, animated: true, completion: nil)
- **UIImagePickerController 설정 & Launching**

```
let imagePickerController = UIImagePickerController()
imagePickerController.delegate = ...

if UIImagePickerController.isSourceTypeAvailable(.camera) {
    imagePickerController.sourceType = .camera
}else{
    imagePickerController.sourceType = .photoLibrary
}

// UIImagePickerController이 활성화 된다
present(imagePickerController, animated: true, completion: nil)
```

Still Image 캡쳐

- **UIImagePickerController의 Delegator**

```
extension ViewController: UINavigationControllerDelegate, UIImagePickerControllerDelegate{

    // 사진을 찍은 경우 호출되는 함수
    func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) {
        let image = info[UIImagePickerController.InfoKey.originalImage] as! UIImage

        // 여기서 이미지에 대한 추가적인 작업을 한다

        picker.dismiss(animated: true, completion: nil)
    }

    // 사진 캡쳐를 취소하는 경우 호출 함수
    func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
            // imagePickerController을 죽인다
        picker.dismiss(animated: true, completion: nil)
    }
}
```

Still Image 캡쳐

- **프로젝트**
  - 카메라로부터 사진을 찍어서 화면에 나타나도록 한다.
  - 프로젝트 이름: ch08-123456-cameraCoreMl
  - ViewController.swift → StillImageViewController.swift
    - ViewController 클래스도 StillImageViewController 클래스로 변경
    - Main.storyboard에서 ViewController의 대응 클래스도 변경
  - Main.storyboard
    - 오브젝트 라이브러리로부터 UIImageView를 ViewController위에 끌어다 놓아라
      - UIImageView의 여백을 safeArea로부터 모두 0으로 하라
      - UIImageView의 Attribute Inspector을 열어서 View→Content Mode를 "Scale to Fill"로 선택하라.
    - 오브젝트 라이브러리에서 UILabel을 선택하여 UIImageView위에 놓아라
      - 텍스트를 "아무곳이나 클릭하세요"라고 하라
      - 오토레이아웃으로 상단여백을 10으로 하고 수평적으로 화면의 중앙에 오도록 하라
  - StillImageViewController
    - Main.storyboard의 UIImageView를 IBOutlet로 imageView변수로 선언하라
    - Main.storyboard의 UILabel를 IBOutlet로 messageLabel변수로 선언하라
    -

Still Image 캡쳐

- 카메라 사용 설정

Key	Type	Value
▼ Information Property List	Dictionary	(15 items)
Application Category	String	
Principal class	String	$(DEVELOPMENT_LANGUAGE)
Privacy - AppleEvents Sending...	String	$(EXECUTABLE_NAME)
Privacy - Bluetooth Peripheral...	String	$(PRODUCT_BUNDLE_IDENTIFIER)
Privacy - Calendars Usage Des...	String	6.0
Privacy - Camera Usage Descri...	String	$(PRODUCT_NAME)
Privacy - Contacts Usage Descr...	String	APPL
Privacy - Face ID Usage Descri...	String	1.0
Privacy - Health Records Usage...	String	1
Privacy - Health Share Usage D...	Boolean	YES
Privacy - Health Update Usage...	String	LaunchScreen
Main storyboard file base name		
▶ Required device capabilities		

반드시 이 문자열을 입력하기를...
실제 AppStore에 올릴 때 아무 이유없이 안 올라가는 원인 중에 하나임

Key	Type	Value
▼ Information Property List	Dictionary	(16 items)
Privacy - Camera Usage Description	String	이 앱은 카메라를 사용합니다.
Application Category	String	
Localization native development re...	String	$(DEVELOPMENT_LANGUAGE)
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle version	String	1
Application requires iPhone enviro	Boolean	YES

Still Image 캡쳐

- **UIImageView에 tapping Gesture 달기**
  - UIImageView를 클릭하면 사진찍기 모드로 전환한다.
  - UIImageView는 기본적으로 사용자와 Interaction을 하지 않는다. 따라서 아래와 같은 코딩이 필요하다

```
extension StillImageViewController{
    override func viewDidLoad() {
        super.viewDidLoad()

        let tapGesture = UITapGestureRecognizer(target: self, action: #selector(takePicture))
        imageView.addGestureRecognizer(tapGesture)

        messageLabel.layer.borderWidth = 2
        messageLabel.layer.borderColor = UIColor.red.cgColor
        // 반드시 설정하여야 한다
        imageView.isUserInteractionEnabled = true
    }
}
```

Still Image 캡쳐

- **UIImagePickerController 호출 구현**
  - takePicture 함수가 호출되면 UIImagePickerController로 전환한다
  - present 함수가 사용된다.

```
extension StillImageViewController{
    @objc func takePicture(sender: UITapGestureRecognizer){

        let imagePickerController = UIImagePickerController()
        imagePickerController.delegate = self

        if UIImagePickerController.isSourceTypeAvailable(.camera) {
            imagePickerController.sourceType = .camera
        }else{
            imagePickerController.sourceType = .photoLibrary
        }

        // UIImagePickerController이 활성화 된다
        present(imagePickerController, animated: true, completion: nil)
    }
}
```

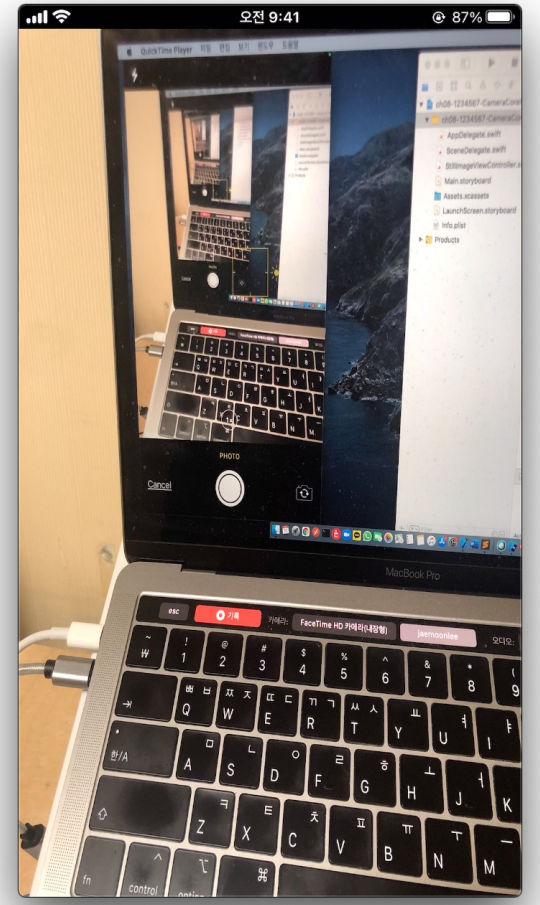
Still Image 캡쳐

- **UIImagePickerController의 Delegator 구현**
  - 선택된 사진을 UIImageView에 나타나게 한다.

```
extension StillImageViewController: UINavigationControllerDelegate, UIImagePickerControllerDelegate{

    // 사진을 캡쳐하는 경우 호출 함수
    func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) {

        // 사진을 가져온다
        let image = info[UIImagePickerController.InfoKey.originalImage] as! UIImage

        // 가져온 사진을 UIImageView에 나타나도록 한다
        imageView.image = image
        picker.dismiss(animated: true, completion: nil)
    }

    // 사진 캡쳐를 취소하는 경우 호출 함수
    func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
        picker.dismiss(animated: true, completion: nil)
    }
}
```

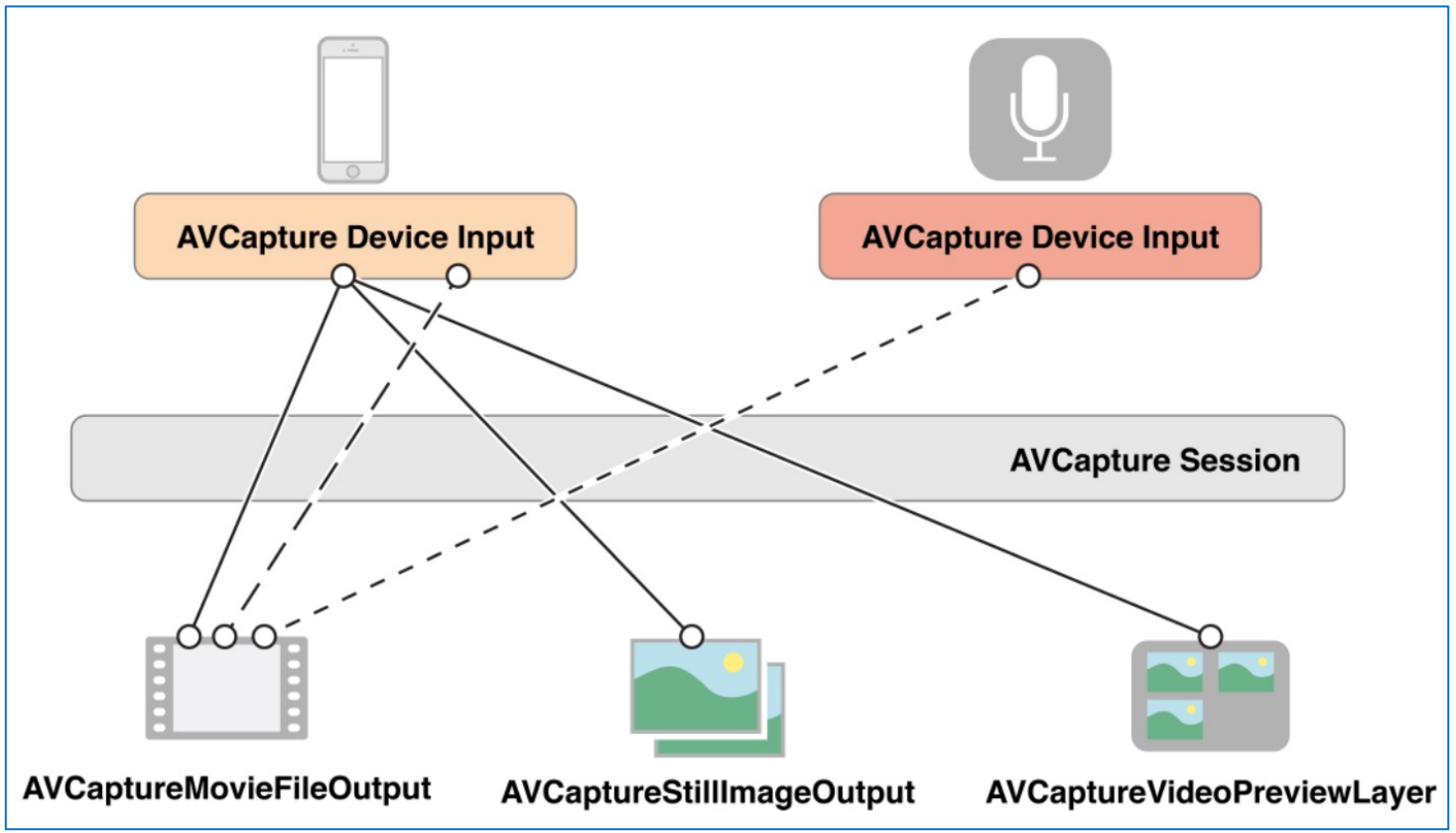
Still Image 캡쳐

- **실행결과**
  - 아이폰 화면 맥 컴퓨터로 보기
    - 아이폰을 맥 컴퓨터와 연결한다
    - QuickTime Player을 실행한다
    - QuickTime Player의 메뉴에서 "새로운 동영상 녹화"를 선택한다
    - 녹화버턴 우측에서 "아이폰"을 선택한다



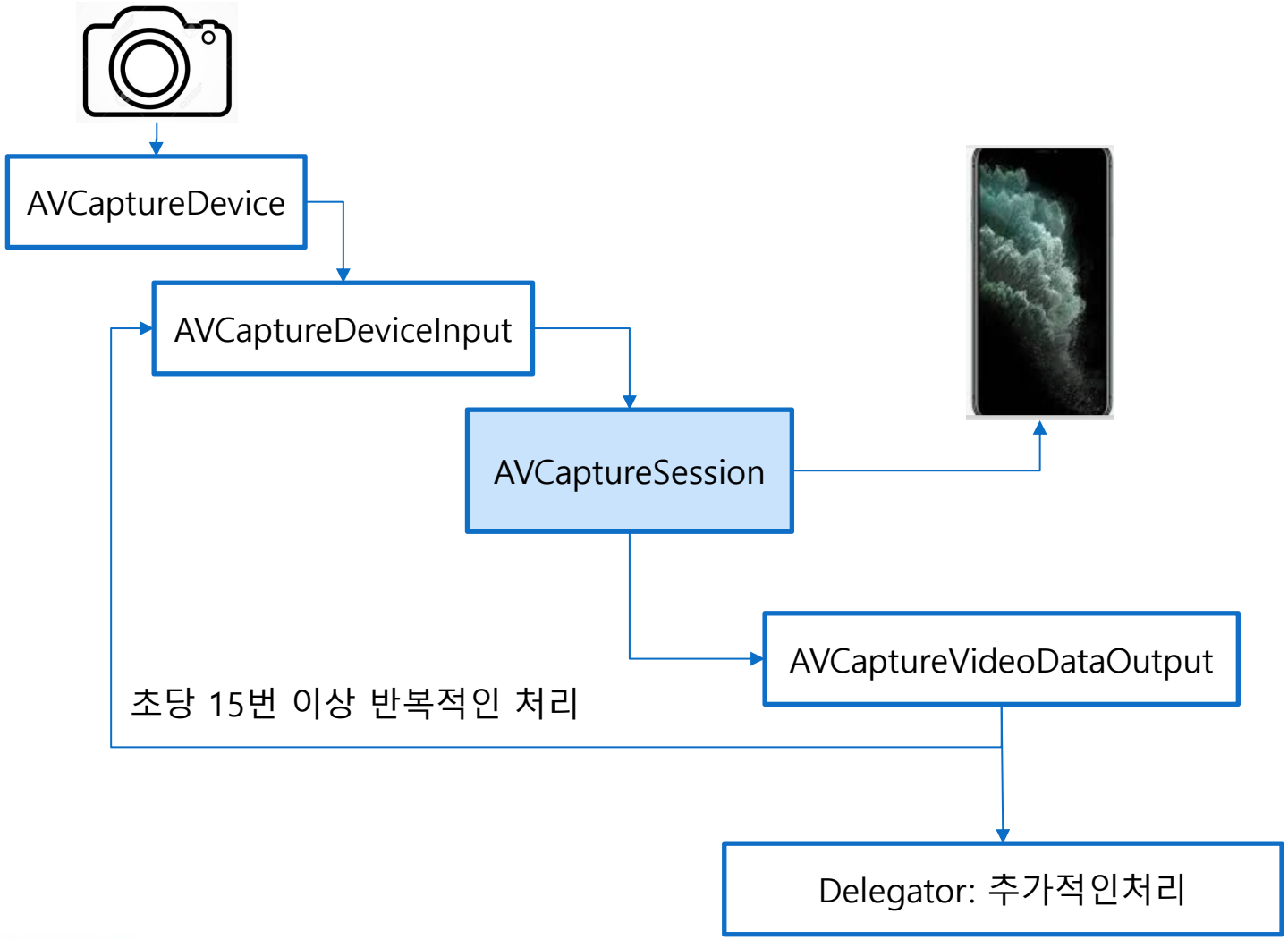
Live Image 캡쳐

- **카메라 API**
  - AVCapture Device Input: 비디오/오디오 디바이스와 연결하는 객체
  - AVCapture Output: input으로 부터 받은 데이터를 처리하는 객체
  - AVCapture Session: Device Input와 Output를 연결해주는 객체



Live Image 캡쳐

- 카메라 관련 객체 및 연결



Live Image 캡쳐

- **AVSeccession 설정**
  - Live Image 캡쳐의 모든 내용을 관장한다
    - sessionPreset를 통하여 캡쳐링의 퀄리티를 제어할 수 있다.

```
captureSession = AVCaptureSession()

captureSession.beginConfiguration()
captureSession.sessionPreset = .medium

// VideoInput 디바이스 부착
// VideoOutput 디바이스 부착
// Previewer 설정

captureSession.commitConfiguration()

// 캡쳐링을 시작한다
captureSession.startRunning()
```

Live Image 캡쳐

- **Video Input Device의 생성**
  - Device를 선택한다.
  - 선택된 Device에 대한 AVCaptureDeviceInput 객체를 생성한다.

```
func createVideoInput() -> AVCaptureDeviceInput? {

        if let device = AVCaptureDevice.default(.builtInWideAngleCamera, for: .video, position: .back){

            return try? AVCaptureDeviceInput(device: device)
        }
        return nil
}
```

Live Image 캡쳐

- **Video Output Device의 생성**
  - 비디오 output에 대한 포맷 설정
  - 비디오 output 처리를 위한 Delegator 설정
  - 비디오 output 처리를 위한 DispatchQueue 설정

```
func createVideoOutput() -> AVCaptureVideoDataOutput? {

        let videoOutput = AVCaptureVideoDataOutput()
        let settings: [String: Any] = [kCVPixelBufferPixelFormatTypeKey as String: NSNumber(value: kCVPix
elFormatType_32BGRA)]

        videoOutput.videoSettings = settings
        videoOutput.alwaysDiscardsLateVideoFrames = true
        videoOutput.setSampleBufferDelegate(self, queue: DispatchQueue.global())
        videoOutput.connection(with: .video)?.videoOrientation = .portrait
        return videoOutput
}
```

Live Image 캡쳐

- **Video Output Device의 생성**
  - Delegator
    - AVCaptureVideoDataOutputSampleBufferDelegate를 상속받고 captureOutput를 구현하여야 한다

```
extension XXXViewController: AVCaptureVideoDataOutputSampleBufferDelegate{

    func captureOutput(_ output: AVCaptureOutput, didOutput sampleBuffer: CMSampleBuffer, from connection: AVCaptureConnection) {

            // 여기서 이미지가 담겨져 온 sampleBuffer에 대한 처리를 하면된다.

    }
}
```

Live Image 캡쳐

- **Previewer의 설정**
  - AVCaptureSession으로부터 PreviewLayer을 가져온다
  - 가져온 PreviewLayer을 적절한 UIView에 subLayer로 부착한다

```
func attachPreviewer(captureSession: AVCaptureSession){

    let avCaptureVideoPreviewLayer = AVCaptureVideoPreviewLayer(session: captureSession)
    avCaptureVideoPreviewLayer.frame = imageView.layer.bounds
    avCaptureVideoPreviewLayer.videoGravity = .resize
    imageView.layer.addSublayer(avCaptureVideoPreviewLayer)
}
```

Live Image 캡쳐

- **프로젝트**
  - 기존의 프로젝트에서 LiveImageViewController.swift를 생성하라.
  - 스토리보드의 ViewController의 연결 클래스를 LiveImageViewController로 설정하라
  - 스토리보드의 ViewController에 있는 UIImageView에 대하여 LiveImageViewController에 IBOutlet로 imageView로 선언하라
  - 스토리보드의 ViewController에 있는 UILabel에 대하여 LiveImageViewController에 IBOutlet로 messageLabel로 선언하라
  - LiveImageViewController.swift에서
    - Import AVKit를 하라

Live Image 캡쳐

- **createVideoInput 함수 구현**

```
extension LiveImageViewController{
    func createVideoInput() -> AVCaptureDeviceInput? {

        if let device = AVCaptureDevice.default(.builtInWideAngleCamera, for: .video, position: .back){

            return try? AVCaptureDeviceInput(device: device)
        }
        return nil
    }
}
```

Live Image 캡쳐

- **createVideoOutput 함수 구현**

```
extension LiveImageViewController{
    func createVideoOutput() -> AVCaptureVideoDataOutput? {

        let videoOutput = AVCaptureVideoDataOutput()
        let settings: [String: Any] = [kCVPixelBufferPixelFormatTypeKey as String: NSNumber(value: kCVPi
xelFormatType_32BGRA)]

        videoOutput.videoSettings = settings
        videoOutput.alwaysDiscardsLateVideoFrames = true
        videoOutput.setSampleBufferDelegate(self, queue: DispatchQueue.global())
        videoOutput.connection(with: .video)?.videoOrientation = .portrait
        return videoOutput
    }
}
```

```
extension LiveImageViewController: AVCaptureVideoDataOutputSampleBufferDelegate{

    func captureOutput(_ output: AVCaptureOutput, didOutput sampleBuffer: CMSampleBuffer, from co
nnection: AVCaptureConnection) {

    // 여기서 이미지가 담겨져 온 sampleBuffer에 대한 처리를 하면된다.

    }
}
```

Live Image 캡쳐

- **attachPreviewer 함수 구현**

```
extension LiveImageViewController{
    func attachPreviewer(captureSession: AVCaptureSession){

        let avCaptureVideoPreviewLayer = AVCaptureVideoPreviewLayer(session: captureSession)
        avCaptureVideoPreviewLayer.frame = imageView.layer.bounds
        avCaptureVideoPreviewLayer.videoGravity = .resize
        imageView.layer.addSublayer(avCaptureVideoPreviewLayer)
    }
}
```

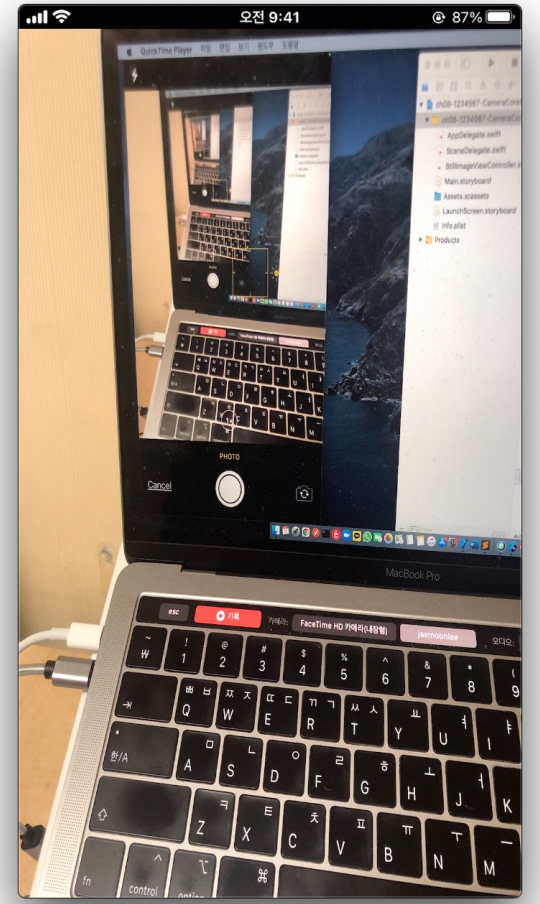
Live Image 캡쳐

- **전역변수 및 viewDidLoad 함수 구현**

```
class LiveImageViewController: UIViewController {

    @IBOutlet weak var imageView: UIImageView!
    var captureSession: AVCaptureSession!

    override func viewDidLoad() {
        super.viewDidLoad()

        captureSession = AVCaptureSession()

        captureSession.beginConfiguration()
        captureSession.sessionPreset = .medium

        guard let videoInput = createVideoInput() else{ return }
        captureSession.addInput(videoInput)
        guard let videoOutput = createVideoOutput() else{ return }
        captureSession.addOutput(videoOutput)

        attachPreviewer(captureSession: captureSession)
        captureSession.commitConfiguration()

        captureSession.startRunning()
    }
}
```

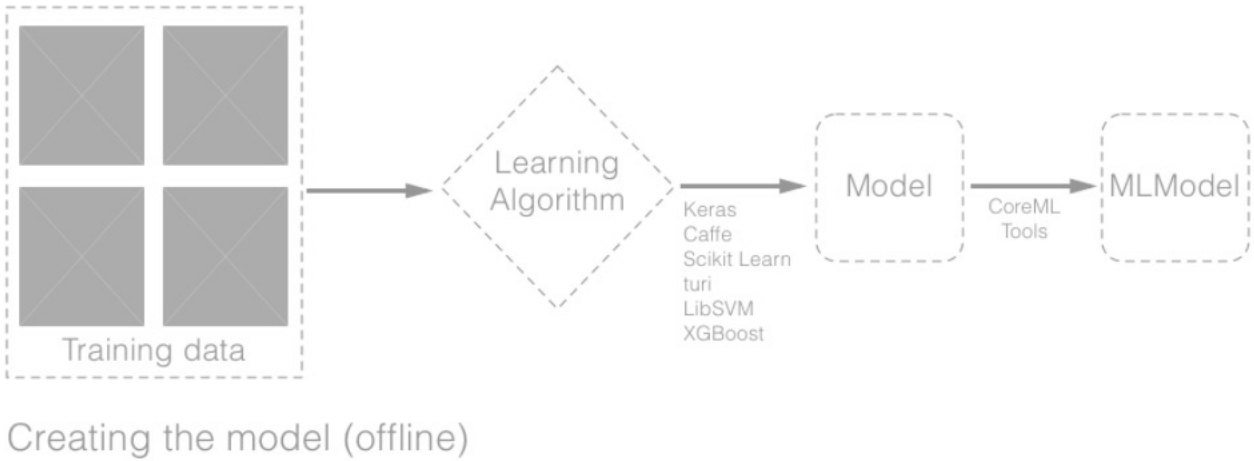
Live Image 캡쳐

- 실행



CoreML

- **What is CoreML?**
  - CoreML은 다양한 Machine Learning Model을 iOS로 가져와서 표준 API로 Wrapping하여 쉽게 인공지능을 개발할 수 있도록 하는 framework임



CoreML

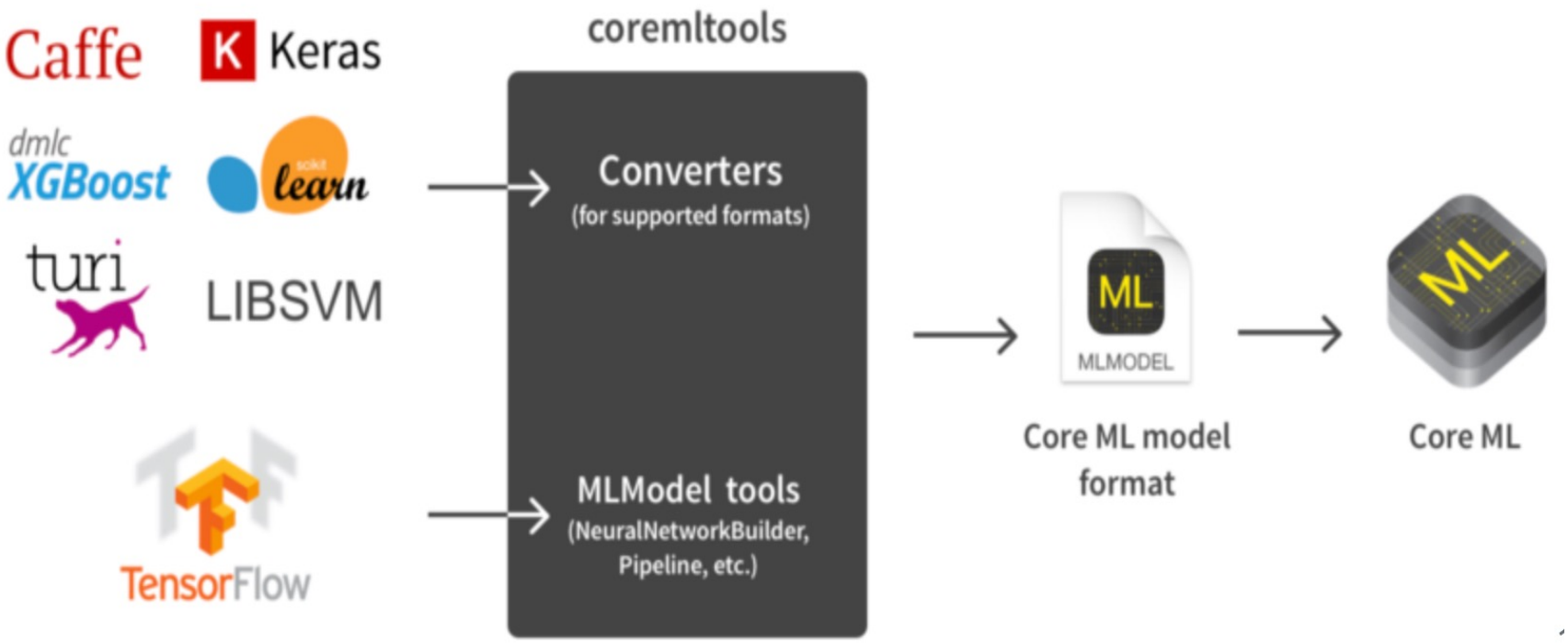
- **CoreML의 사용**

Core ML is huge, and it can be used to implement a multitude of functionalities. However, there are two direct applications of Core ML:

- First, the developers can use the pretrained models that already exist in the Core ML.
- Second, they can they can build their own custom ML model using frameworks like Caffe, Turi, and Keras, and then convert it into a Core ML model to use it in their iOS application.

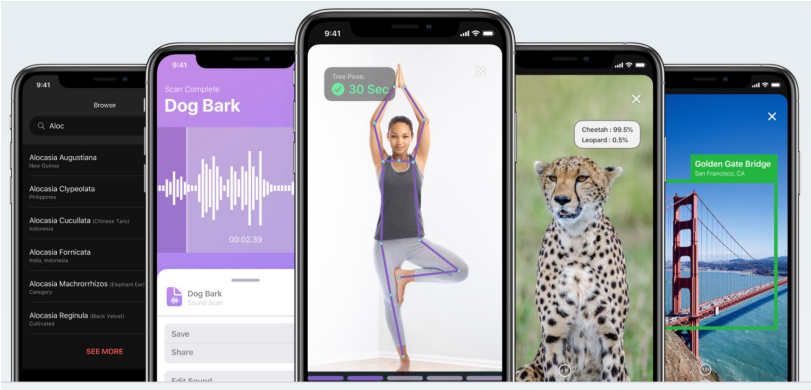
CoreML

- 외부 ML로 부터 CoreML 생성



Machine Learn in Apple

- https://developer.apple.com/machine-learning/core-ml/



Create ML

Build and train Core ML models right on your Mac with no code.

Learn more ›

Core ML Converters

Convert models from third-party training libraries into Core ML using the coremltools Python package.

Learn more ›

Models

Get started with models from the research community that have been converted to Core ML.

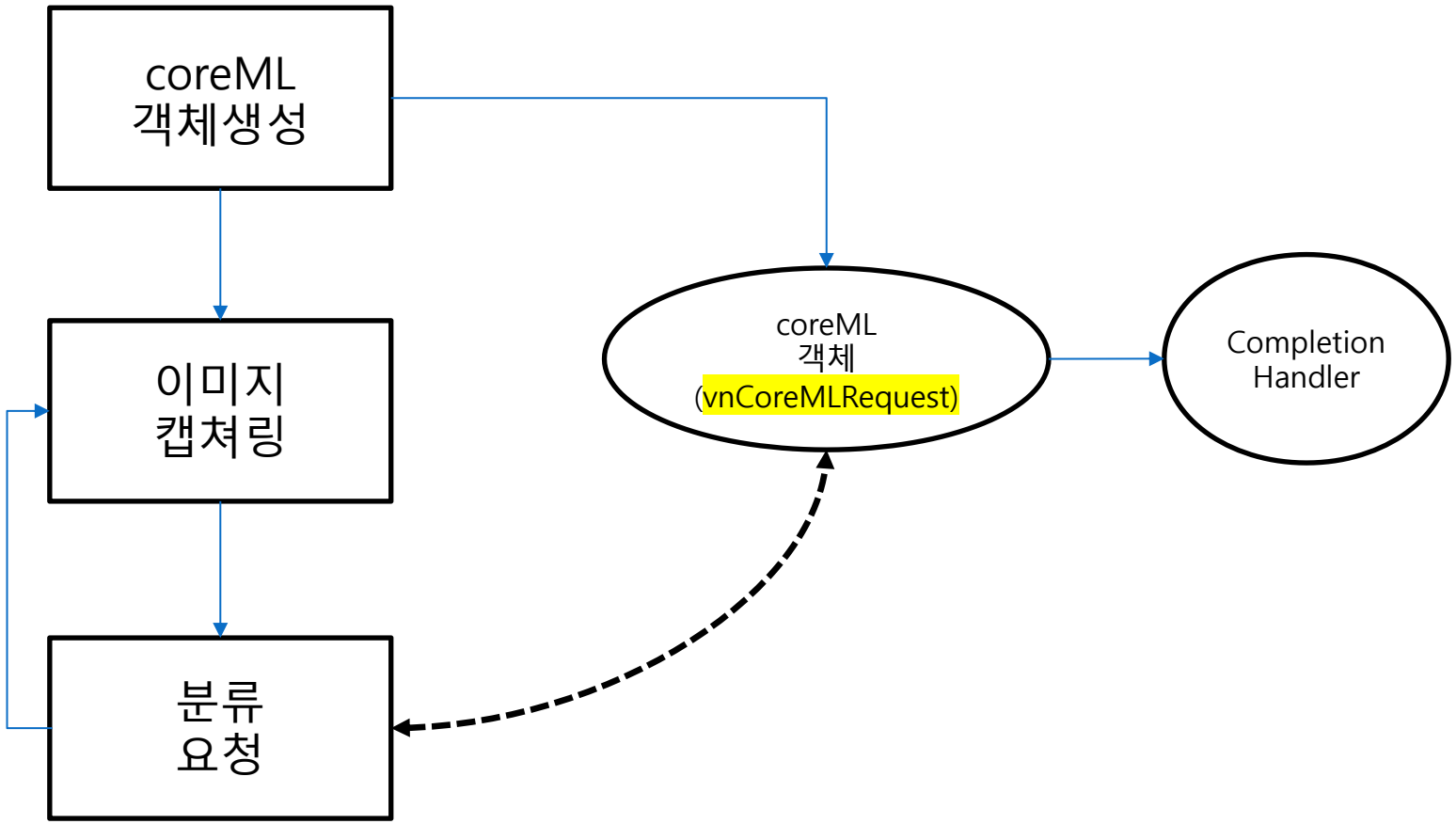
Browse models ›

내부 모델(Pretrained Models)

- **FCRN-DepthPrediction:** Predict the depth from a single image.
- **MNIST:** Classify a single handwritten digit (supports digits 0-9)
- **UpdatableDrawingClassifier:** Drawing classifier that learns to recognize new drawings based on a K-Nearest Neighbors model (KNN).
- **MobileNetV2:** The MobileNetv2 architecture trained to classify the dominant object in a camera frame or image.
- **Resnet50:** A Residual Neural Network that will classify the dominant object in a camera frame or image.
- **SqueezeNet:** A small Deep Neural Network architecture that classifies the dominant object in a camera frame or image.
- **DeeplabV3:** Segment the pixels of a camera frame or image into a predefined set of classes.
- **YOLOv3:** Locate and classify 80 different types of objects present in a camera frame or image.
- **PoseNet:** Estimates up to 17 joint positions for each person in an image.
- **BERT-SQuAD:** Find answers to questions about paragraphs of text.

coreML 사용하기

- 통상적 프로그래밍 방법



coreML 사용하기

- **Model 객체 생성**

```
func createCoreML(modelName: String, modelExt: String, completionHandler: @escaping (VNRequest, Error?) -> Void) -> VNCoreMLRequest?{
    guard let modelURL = Bundle.main.url(forResource: modelName, withExtension: modelExt) else {
        return nil
    }
    guard let vnCoreMLModel = try? VNCoreMLModel(for: MLModel(contentsOf: modelURL)) else{
        return nil
    }
    return VNCoreMLRequest(model: vnCoreMLModel, completionHandler: completionHandler)
}
```

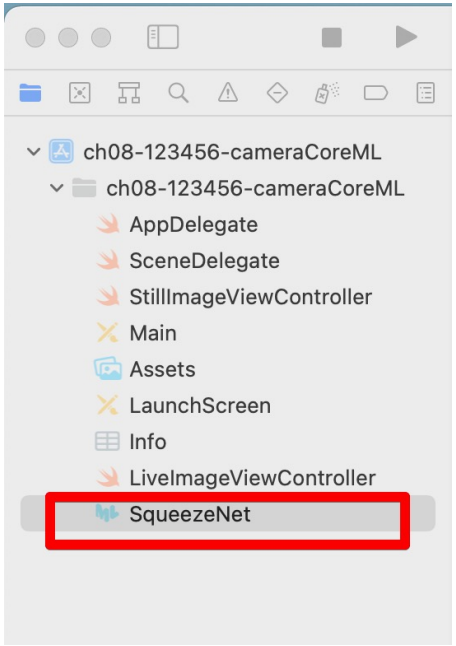
```
func handleResults(request: VNRequest, error: Error?){
    guard let results = request.results as? [VNClassificationObservation] else{ return }
    // 여기서 results(분류등의 결과)의 내용을 사용자에게 알린다.
}
```

- **분류요청**

```
let handler = VNImageRequestHandler(ciImage: CIImage(image: image)!)
try! handler.perform([vnCoreMLRequest])
```

카메라를 이용한 이미지 분류하기

- **프로젝트 변경**
  - Ch08-cameraCoreML에서 다시 StillImageViewController을 스토리보드의 ViewController과 연결하라.
  - StillImageViewController.swift에서 다음을 import하라
    - import Vision
    - import CoreML
- **CoreML model 추가**
  - https://developer.apple.com/kr/machine-learning/models/
  - **SqueezeNet을 선택하고 다운을 받아라**
    - SqueezeNet.mlmodel을 다운받아 프로젝트에 포함시켜라



카메라를 이용한 이미지 분류하기

- **coreML 객체 생성**

```
extension StillImageViewController{

    func createCoreML(modelName: String, modelExt: String, completionHandler: @escaping (VNReques
t, Error?) -> Void) -> VNCoreMLRequest?{
        guard let modelURL = Bundle.main.url(forResource: modelName, withExtension: modelExt) else {
            return nil
        }
        guard let vnCoreMLModel = try? VNCoreMLModel(for: MLModel(contentsOf: modelURL)) else{
            return nil
        }
        return VNCoreMLRequest(model: vnCoreMLModel, completionHandler: completionHandler)
    }
}
```

```
extension StillImageViewController{
    func handleImageClassifier(request: VNRequest, error: Error?){
        guard let results = request.results as? [VNClassificationObservation] else{ return }
        if let topResult = results.first{
            DispatchQueue.main.async {
                self.messageLabel.text = "₩(topResult.identifier)입니다. 아무곳이나 클릭하세요"
            }
        }
    }
}
```

카메라를 이용한 이미지 분류하기

- **coreML 객체 생성**

```
class StillImageViewController: UIViewController {

    @IBOutlet weak var imageView: UIImageView!
    @IBOutlet weak var messageLabel: UILabel!

    var vnCoreMLRequest: VNCoreMLRequest!

    override func viewDidLoad() {
        super.viewDidLoad()

        vnCoreMLRequest = createCoreML(modelName: "SqueezeNet", modelExt: "mlmodelc", completi
onHandler: handleImageClassifier)

        let tapGesture = UITapGestureRecognizer(target: self, action: #selector(takePicture))
        imageView.addGestureRecognizer(tapGesture)

        // 반드시 설정하여야 한다
        imageView.isUserInteractionEnabled = true
    }
}
```

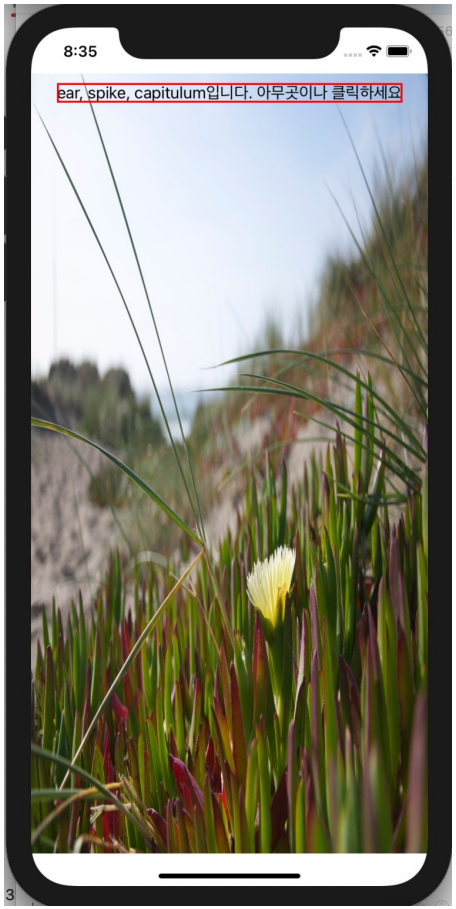
카메라를 이용한 이미지 분류하기

- **분류 요청**

```
extension StillImageViewController: UINavigationControllerDelegate, UIImagePickerControllerDelegate{

    // 사진을 캡쳐하는 경우 호출 함수
    func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [U
IImagePickerController.InfoKey : Any]) {

        // 사진을 가져온다
        let image = info[UIImagePickerController.InfoKey.originalImage] as! UIImage

        // 가져온 사진을 UIImageView에 나타나도록 한다
        imageView.image = image
        picker.dismiss(animated: true, completion: nil)

        let handler = VNImageRequestHandler(ciImage: CIImage(image: image)!)
        try! handler.perform([vnCoreMLRequest])
    }

    // 사진 캡쳐를 취소하는 경우 호출 함수
    func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
        picker.dismiss(animated: true, completion: nil)
    }
}
```

카메라를 이용한 이미지 분류하기

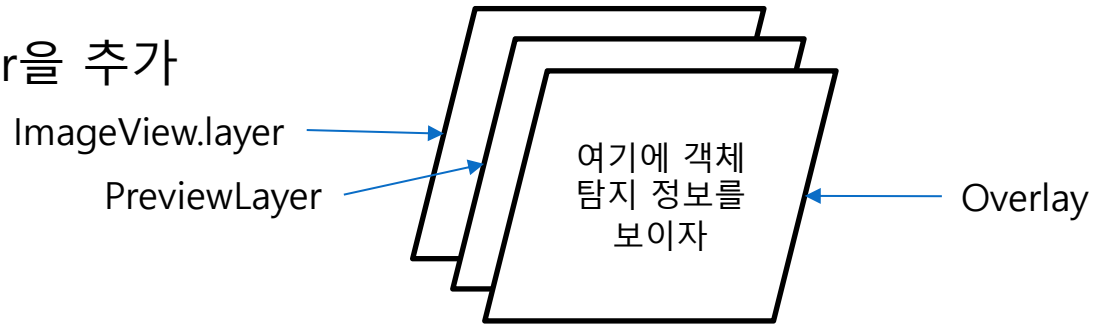
- 실행



카메라를 이용한 객체 탐지

- **프로젝트 변경**
  - Ch08-cameraCoreML에서 다시 LiveImageViewController을 스토리보드의 ViewController과 연결하라.
  - LiveImageViewController.swift에서 다음을 import하라
    - import Vision
    - import CoreML
  - LiveImageViewController.swift에서 클래스 변수를 선언하라
    - var request: VNCoreMLRequest!
    - var detectionOverlay: CALayer!
    - var videoBufferSize = CGSize()
  - YOLOv3.mlmodel 다운로드
    - https://developer.apple.com/machine-learning/models에서 YOLOv3.mlmodel을 다운받아 프로젝트에 포함하라

- **Overlay를 위한 준비**
  - Preview위에 하나의 CALayer을 추가




```
extension LiveImageViewController{
    func attachPreviewer(captureSession: AVCaptureSession){

        let avCaptureVideoPreviewLayer = AVCaptureVideoPreviewLayer(session: captureSession)
        avCaptureVideoPreviewLayer.frame = imageView.layer.bounds
        avCaptureVideoPreviewLayer.videoGravity = .resize
        imageView.layer.addSublayer(avCaptureVideoPreviewLayer)

        detectionOverlay = CALayer() // container layer that has all the renderings of the observations
        detectionOverlay.name = "DetectionOverlay"
        detectionOverlay.bounds = CGRect(x: 0, y: 0, width: videoBufferSize.width, height: videoBufferSize.
height)
        detectionOverlay.position = CGPoint(x: detectionOverlay.bounds.midX, y: detectionOverlay.bounds.
midY)
        avCaptureVideoPreviewLayer.addSublayer(detectionOverlay)
    }
}
```

카메라를 이용한 객체 탐지

- **디바이스 Demension 저장**
  - 디바이스 Demension이 있어야 탐지된 사각형 영역과 실제 화면의 비율을 계산하여 Overlay에 탐지 객체를 표시할 수 있음

```
extension LiveImageViewController{
    func createVideoInput() -> AVCaptureDeviceInput? {

        guard let device = AVCaptureDevice.default(.builtInWideAngleCamera, for: .video, position: .back)
else{
            return nil
        }
        do {
            try  device.lockForConfiguration()
            let dimensions = CMVideoFormatDescriptionGetDimensions((device.activeFormat.formatDescription))
            videoBufferSize.width = CGFloat(dimensions.width)
            videoBufferSize.height = CGFloat(dimensions.height)
            device.unlockForConfiguration()
        } catch {
            return nil
        }
        return try? AVCaptureDeviceInput(device: device)
    }
}
```

카메라를 이용한 객체 탐지

- **coreML 객체 생성 // 이미지 분류와 동일**

```
extension LiveImageViewController{

    func createCoreML(modelName: String, modelExt: String, completionHandler: @escaping (VNReques
t, Error?) -> Void) -> VNCoreMLRequest?{
        guard let modelURL = Bundle.main.url(forResource: modelName, withExtension: modelExt) else {
            return nil
        }
        guard let vnCoreMLModel = try? VNCoreMLModel(for: MLModel(contentsOf: modelURL)) else{
            return nil
        }
        return VNCoreMLRequest(model: vnCoreMLModel, completionHandler: completionHandler)
    }
}
```

```
extension LiveImageViewController{
    func handleObjectDetection(request: VNRequest, error: Error?){

        guard let results = request.results else{ return }
        DispatchQueue.main.async {
            self.drawVisionRequestResults(results)
        }
    }
}
```

카메라를 이용한 객체 탐지

- **탐지 객체 화면에 표시하기**

  이 코드는 https://developer.apple.com/documentation/vision/recognizing_objects_in_live_capture 에서의 코드를 간략한 것임

```
extension LiveImageViewController{
    func drawVisionRequestResults(_ results: [Any]) {
        CATransaction.begin()

        CATransaction.setValue(kCFBooleanTrue, forKey: kCATransactionDisableActions)
        detectionOverlay.sublayers = nil // remove all the old recognized objects

        for observation in results{

            guard let objectObservation = observation as? VNRecognizedObjectObservation else {
                continue
            }

            // Select only the label with the highest confidence.
            let topLabelObservation = objectObservation.labels[0]
            let objectBounds = VNImageRectForNormalizedRect(objectObservation.boundingBox, Int(videoBufferSize.width), Int(videoBufferSize.height)
)

            let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
            let text = String(format: "₩(topLabelObservation.identifier)₩nconfidence:  %.2f", topLabelObservation.confidence)

            let textLayer = self.createTextSubLayerInBounds(objectBounds, text: text)
            shapeLayer.addSublayer(textLayer)
            detectionOverlay.addSublayer(shapeLayer)
        }
        updateLayerGeometry()

        CATransaction.commit()
    }
}
```

카메라를 이용한 객체 탐지

- **탐지 객체 화면에 표시하기**

```
extension LiveImageViewController{

    func createTextSubLayerInBounds(_ bounds: CGRect, text: String) -> CATextLayer {
        let textLayer = CATextLayer()
        textLayer.string = text
        textLayer.bounds = CGRect(x: 0, y: 0, width: bounds.size.height - 10, height: bounds.size.width - 10)
        textLayer.position = CGPoint(x: bounds.midX, y: bounds.midY)
        textLayer.foregroundColor = UIColor.blue.cgColor

        // rotate the layer into screen orientation and scale and mirror
        textLayer.setAffineTransform(CGAffineTransform(rotationAngle: CGFloat(.pi / 2.0)).scaledBy(x: 1.0, y: -1.0))
        return textLayer
    }

    func createRoundedRectLayerWithBounds(_ bounds: CGRect) -> CALayer {
        let shapeLayer = CALayer()
        shapeLayer.bounds = bounds
        shapeLayer.position = CGPoint(x: bounds.midX, y: bounds.midY)
        shapeLayer.backgroundColor = CGColor(colorSpace: CGColorSpaceCreateDeviceRGB(), components: [1.0, 1.0, 0.2, 0.4
])
        shapeLayer.cornerRadius = 7
        return shapeLayer
    }
}
```

카메라를 이용한 객체 탐지

- **탐지 객체 화면에 표시하기**

```
extension LiveImageViewController{
    func updateLayerGeometry() {

        let rootLayer = detectionOverlay.superlayer!
        let bounds = rootLayer.bounds

        let xScale: CGFloat = bounds.size.width / videoBufferSize.height
        let yScale: CGFloat = bounds.size.height / videoBufferSize.width

        var scale = fmax(xScale, yScale)
        if scale.isInfinite {
            scale = 1.0
        }
        // rotate the layer into screen orientation and scale and mirror
        detectionOverlay.setAffineTransform(CGAffineTransform(rotationAngle: CGFloat(.pi / 2.0)).scaledBy(x: scale, y: -scale)
)
        detectionOverlay.position = CGPoint(x: bounds.midX, y: bounds.midY)
    }
}
```

카메라를 이용한 객체 탐지

- **모델 객체 생성**

```
class LiveImageViewController: UIViewController {
    ...
    var request: VNCoreMLRequest!
    override func viewDidLoad() {
        super.viewDidLoad()

        captureSession = AVCaptureSession()
        ...

        captureSession.commitConfiguration()

        request = createCoreML(modelName: "YOLOv3", modelExt: "mlmodelc", completionHandler: hand
leObjectDetection)

        let tapGesture = UITapGestureRecognizer(target: self, action: #selector(takePicture))
        ...
    }
}
```

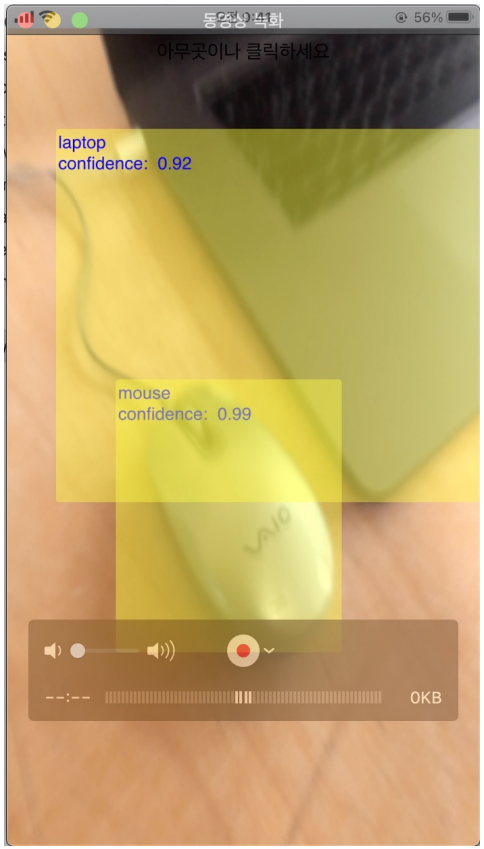
카메라를 이용한 객체 탐지

- **탐지 요청**

```
extension    LiveImageViewController: AVCaptureVideoDataOutputSampleBufferDelegate{

    func captureOutput(_ output: AVCaptureOutput, didOutput sampleBuffer: CMSampleBuffer, from con
nection: AVCaptureConnection) {

        // 여기서 이미지가 담겨져 온 sampleBuffer에 대한 처리를 하면된다.
        let imageBuffer = CMSampleBufferGetImageBuffer(sampleBuffer)
        let imageRequestHandler = VNImageRequestHandler(cvPixelBuffer: imageBuffer!, orientation: .up,
options: [:])
        if let request = request{
            try! imageRequestHandler.perform([request])
        }
    }
}
```

카메라를 이용한 객체 탐지

- 실행



Xcode 업그레드시 유의사항

- **Xcode와 iOS지원 관계**
  - Xcode 12.3 → iOS 14.3
  - Xcode 12.2 → iOS 14.2
  - Xcode 12.1 → iOS 14.1
  - Xcode 12 → iOS 14
  - Xcode 11.7 → iOS 13.7
  - Xcode 11.6 → iOS 13.6
  - Xcode 11.5 → iOS 13.5
  - Xcode 11.4 → iOS 13.4