ДОДАТОК А

Програмна реалізація

Нижче наведено класи мобільного додатку, який відповідає за обробку зображень з використанням Core ML та являються моделлю у архітектурі MVC:

```
import UIKit
import CoreML
import Vision
import ImageIO

class ImageClassificationViewController: UIViewController {
    // MARK: - IBOutlets

    @IBOutlet weak var imageView: UIImageView!
    @IBOutlet weak var cameraButton: UIBarButtonItem!
    @IBOutlet weak var classificationLabel: UILabel!

    // MARK: - Image Classification

    /// - Tag: MLModelSetup
    lazy var classificationRequest: VNCoreMLRequest = {
        do {
            /*
             Use the Swift class `MobileNet` Core ML generates from
the model.
             To use a different Core ML classifier model, add it to
the project
             and replace `MobileNet` with that model's generated
Swift class.
             */
            let model = try VNCoreMLModel(for: MobileNet().model)

            let request = VNCoreMLRequest(model: model,
completionHandler: { [weak self] request, error in
                self?.processClassifications(for: request, error:
error)
            })
            request.imageCropAndScaleOption = .centerCrop
            return request
        } catch {
            fatalError("Failed to load Vision ML model: \(error)")
        }
    }()
```

```
    /// - Tag: PerformRequests
    func updateClassifications(for image: UIImage) {
        classificationLabel.text = "Classifying..."

        let orientation =
CGImagePropertyOrientation(image.imageOrientation)
        guard let ciImage = CIImage(image: image) else {
fatalError("Unable to create \(CIImage.self) from \(image).") }

        DispatchQueue.global(qos: .userInitiated).async {
            let handler = VNImageRequestHandler(ciImage: ciImage,
orientation: orientation)
            do {
                try handler.perform([self.classificationRequest])
            } catch {
                /*
                 This handler catches general image processing
errors. The `classificationRequest`'s
                 completion handler
`processClassifications(_:error:)` catches errors specific
                 to processing that request.
                 */
                print("Failed to perform
classification.\n\(error.localizedDescription)")
            }
        }
    }

    /// Updates the UI with the results of the classification.
    /// - Tag: ProcessClassifications
    func processClassifications(for request: VNRequest, error:
Error?) {
        DispatchQueue.main.async {
            guard let results = request.results else {
                self.classificationLabel.text = "Unable to classify
image.\n\(error!.localizedDescription)"
                return
            }
            // The `results` will always be
`VNClassificationObservation`s, as specified by the Core ML model
in this project.
            let classifications = results as!
[VNClassificationObservation]

            if classifications.isEmpty {
                self.classificationLabel.text = "Nothing
recognized."
            } else {
```

```
                // Display top classifications ranked by confidence
in the UI.
                let topClassifications = classifications.prefix(2)
                let descriptions = topClassifications.map {
classification in
                    // Formats the classification for display; e.g.
"(0.37) cliff, drop, drop-off".
                    return String(format: "  (%.2f) %@",
classification.confidence, classification.identifier)
                }
                self.classificationLabel.text = "Classification:\n"
+ descriptions.joined(separator: "\n")
            }
        }
    }

    // MARK: - Photo Actions

    @IBAction func takePicture() {
        // Show options for the source picker only if the camera is
available.
        guard
UIImagePickerController.isSourceTypeAvailable(.camera) else {
            presentPhotoPicker(sourceType: .photoLibrary)
            return
        }

        let photoSourcePicker = UIAlertController()
        let takePhoto = UIAlertAction(title: "Take Photo", style:
.default) { [unowned self] _ in
            self.presentPhotoPicker(sourceType: .camera)
        }
        let choosePhoto = UIAlertAction(title: "Choose Photo",
style: .default) { [unowned self] _ in
            self.presentPhotoPicker(sourceType: .photoLibrary)
        }

        photoSourcePicker.addAction(takePhoto)
        photoSourcePicker.addAction(choosePhoto)
        photoSourcePicker.addAction(UIAlertAction(title: "Cancel",
style: .cancel, handler: nil))

        present(photoSourcePicker, animated: true)
    }

    func presentPhotoPicker(sourceType:
UIImagePickerController.SourceType) {
        let picker = UIImagePickerController()
        picker.delegate = self
```

```
        picker.sourceType = sourceType
        present(picker, animated: true)
    }
}

extension ImageClassificationViewController:
UIImagePickerControllerDelegate, UINavigationControllerDelegate {
    // MARK: - Handling Image Picker Selection

    func imagePickerController(_ picker: UIImagePickerController,
didFinishPickingMediaWithInfo info:
[UIImagePickerController.InfoKey: Any]) {
// Local variable inserted by Swift 4.2 migrator.
let info =
convertFromUIImagePickerControllerInfoKeyDictionary(info)

        picker.dismiss(animated: true)

        // We always expect
`imagePickerController(:didFinishPickingMediaWithInfo:)` to supply
the original image.
        let image =
info[convertFromUIImagePickerControllerInfoKey(UIImagePickerControl
ler.InfoKey.originalImage)] as! UIImage
        imageView.image = image
        updateClassifications(for: image)
    }
}

// Helper function inserted by Swift 4.2 migrator.
fileprivate func
convertFromUIImagePickerControllerInfoKeyDictionary(_ input:
[UIImagePickerController.InfoKey: Any]) -> [String: Any] {
  return Dictionary(uniqueKeysWithValues: input.map {key, value in
(key.rawValue, value)})
}

// Helper function inserted by Swift 4.2 migrator.
fileprivate func convertFromUIImagePickerControllerInfoKey(_ input:
UIImagePickerController.InfoKey) -> String {
  return input.rawValue
}


import AVFoundation
import MetalPerformanceShaders
import UIKit
```

```
class BaseController: UIViewController {

    @IBOutlet weak var fpsLabel: UILabel!

    var networkQueue: DispatchQueue
    var resultQueue: DispatchQueue
    var runQueue = RunQueue(maxConcurrent: 2)
    var timeChecker = TimeChecker()

    required init?(coder aDecoder: NSCoder) {
        networkQueue = DispatchQueue(label:
"com.envision.mlkittest.neuralqueue",
                                     qos: .userInitiated,
                                     target: nil)
        resultQueue = DispatchQueue(label:
"com.envision.mlkittest.resultqueue",
                                    qos: .userInitiated,
                                    target: nil)
        super.init(coder: aDecoder)
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        self.modelSetup()
        CaptureController.instance.capturedFrame = { [weak self]
buffer in
            guard let `self` = self, self.runQueue.push() else {
return }
            DispatchQueue.global().async { [unowned self] in
                    self.run(buffer: buffer)
            }
//            DispatchQueue.main.asyncAfter(deadline: .now() + 30)
{ [weak self] in
//                self?.run(buffer: buffer)
//            }
        }
        timeChecker.onClean = { [weak self] avg in
            DispatchQueue.main.async {
                self?.fpsLabel.text = "Avg time: \(String(format:
"%.4f", avg)) (\(String(format: "%.3f", 1/avg)) FPS)"
            }
        }

        // Wait a sec and start the camera
        DispatchQueue.main.asyncAfter(deadline:
DispatchTime(uptimeNanoseconds: UInt64(1e9))) {
            CaptureController.instance.startVideoCapture()
        }
    }
```

```
    deinit {
        CaptureController.instance.stopVideoCapture()
    }

    func modelSetup() {
        //Please override
    }

    func run(buffer: CMSampleBuffer?) {
        // Please override
    }

}
```

Наведено клас мобільного додатку, який являє собою контроллер та відповідає за роботу камери та з'єднанні моделі машинного навчання з нею:

```
import AVFoundation

class CaptureController: NSObject,
AVCaptureVideoDataOutputSampleBufferDelegate {

    static let instance = CaptureController()

    // AVFoundation variables
    var captureSession = AVCaptureSession()
    lazy var videoDevice: AVCaptureDevice! = {
        let x = AVCaptureDevice.DiscoverySession(deviceTypes:
[AVCaptureDevice.DeviceType.builtInDualCamera,

AVCaptureDevice.DeviceType.builtInWideAngleCamera,

AVCaptureDevice.DeviceType.builtInTelephotoCamera],
                                                 mediaType:
AVMediaType.video,
                                                 position:
AVCaptureDevice.Position.back).devices
        return x.first
    }()

    var movieOutput: AVCaptureVideoDataOutput!
    var capturedFrame: ((CMSampleBuffer) -> Void)!
    private var initialized = false
```

```
    private override init() {}

    func startVideoCapture() {
        if !initialized {
            initialized = true
            let capturePreset = AVCaptureSession.Preset.medium

            #if targetEnvironment(simulator)
            //Simulator, would crash, return
            return
            #endif

            guard videoDevice.supportsSessionPreset(capturePreset)
else {
                print("Device does not support medium quality")
                return
            }

            let outputSettings: [String: Any] = [String(describing:
kCVPixelBufferPixelFormatTypeKey): NSNumber(value:
kCVPixelFormatType_32BGRA)]

            captureSession.beginConfiguration()

            // Set video quality
            captureSession.sessionPreset = capturePreset

            //add device inputs (front camera and mic)
            try?
captureSession.addInput(AVCaptureDeviceInput(device: videoDevice))

            //add output to get the frames
            movieOutput = AVCaptureVideoDataOutput()
            movieOutput.videoSettings = outputSettings
            movieOutput.setSampleBufferDelegate(self, queue:
DispatchQueue.global(qos: .background))
            movieOutput.alwaysDiscardsLateVideoFrames = true
            captureSession.addOutput(movieOutput)

            //start session
            captureSession.commitConfiguration()
            setMirroring()
            setMaxFPS(fps: Constants.maxFPSCountForVideo)
        }
        captureSession.startRunning()

    }

    func stopVideoCapture() {
```

```
        captureSession.stopRunning()
    }

    func setMirroring() {
        for conn in movieOutput.connections {
            for port in conn.inputPorts {
                if port.mediaType == AVMediaType.video,
conn.isVideoMirroringSupported {
                    conn.isVideoMirrored = videoDevice.position ==
.front
                }
            }
        }
    }

    func setMaxFPS(fps maxFpsDesired: Double = 60.0) {
        var finalFormat: AVCaptureDevice.Format!
        var maxFps: Double = 0
        var width: Int32 = 0
        for vFormat in videoDevice.formats {
            var ranges      = vFormat.videoSupportedFrameRateRanges
            let frameRates  = ranges[0]
            let dims =
CMVideoFormatDescriptionGetDimensions(vFormat.formatDescription)
            if (frameRates.maxFrameRate > maxFps ||
(frameRates.maxFrameRate == maxFps && dims.width < width))
                && frameRates.maxFrameRate <= maxFpsDesired {
                maxFps = frameRates.maxFrameRate
                finalFormat = vFormat
                width =
CMVideoFormatDescriptionGetDimensions(finalFormat.formatDescription
).width
            }
        }
        if maxFps != 0 {
            let timeValue = Int64(1200.0 / maxFps)
            let timeScale: Int32 = 1200
            try! videoDevice.lockForConfiguration()
            videoDevice.activeFormat = finalFormat
            videoDevice.activeVideoMinFrameDuration =
CMTimeMake(value: timeValue, timescale: timeScale)
            videoDevice.activeVideoMaxFrameDuration =
CMTimeMake(value: timeValue, timescale: timeScale)
            videoDevice.unlockForConfiguration()
        }
    }

    // MARK: - AVCaptureVideoDataOutputSampleBufferDelegate
```

```
    var acct = 0
    func captureOutput(_ captureOutput: AVCaptureOutput, didOutput
sampleBuffer: CMSampleBuffer, from connection: AVCaptureConnection)
{
//        inc()
        guard 1 == CMSampleBufferGetNumSamples(sampleBuffer) else {
            return
        }
        guard connection.videoOrientation == .portrait else {
            connection.videoOrientation = .portrait
            return
        }

        capturedFrame(sampleBuffer)
    }

    func inc() {
        acct += 1
        if acct == 60 {
            print(Date().timeIntervalSinceReferenceDate)
            acct = 0
        }
    }

    func captureOutput(_ captureOutput: AVCaptureOutput, didDrop
sampleBuffer: CMSampleBuffer, from connection: AVCaptureConnection)
{
    }
}
```

Наведено клас контролеру для головного екрану:

```
import AVFoundation
import FirebaseMLVision
import UIKit
import MessageUI

class ViewController: BaseController,
UIImagePickerControllerDelegate, UINavigationControllerDelegate,
MFMailComposeViewControllerDelegate {

    @IBOutlet weak var toggleCameraButton: UIButton!
    @IBOutlet weak var imageView: UIImageView!
    @IBOutlet weak var videoView: UIView!
    @IBOutlet weak var resultLabel: UILabel!
    @IBOutlet weak var resultTranslateLabel: UILabel!

    let options = VisionLabelDetectorOptions(
```

```
        confidenceThreshold: Constants.labelConfidenceThreshold
    )
    lazy var vision = Vision.vision()
    var labelDetector: VisionLabelDetector!

    override func viewDidLoad() {
        super.viewDidLoad()
        labelDetector = vision.labelDetector(options: options)

        // Add Video sublayer
        let layer = AVCaptureVideoPreviewLayer(session:
CaptureController.instance.captureSession)
        layer.frame = videoView.bounds
        videoView.layer.addSublayer(layer)
    }

    func imagePickerController(_ picker: UIImagePickerController,
didFinishPickingMediaWithInfo info:
[UIImagePickerController.InfoKey : Any]) {
// Local variable inserted by Swift 4.2 migrator.
let info =
convertFromUIImagePickerControllerInfoKeyDictionary(info)

        if let uiImage =
info[convertFromUIImagePickerControllerInfoKey(UIImagePickerControl
ler.InfoKey.originalImage)] as? UIImage {

            imageView.image = uiImage
            let image = VisionImage(image: uiImage)
            detect(image: image)
            picker.dismiss(animated: true, completion: nil)
        }
    }

    override func run(buffer: CMSampleBuffer?) {
        if let buffer = buffer {
            let image = VisionImage(buffer: buffer)
            let metadata = VisionImageMetadata()
            metadata.orientation = .topLeft
            image.metadata = metadata
            detect(image: image)
        }
    }

    func detect(image: VisionImage) {
        labelDetector.detect(in: image) { (labels, error) in
            self.runQueue.pop()
            guard error == nil, let labels = labels,
!labels.isEmpty else {
```

```
                // Error.
                self.resultLabel.text = "No idea"
                return
            }
            self.resultLabel.text = labels.reduce("") { $0 +
"\($1.label) (\($1.confidence))\n" }

            //FIXME: This temporary method is replacing original
responce to  translation responce
            self.resultLabel.text = ""
            let recognitionLabels = self.resultTranslateLabel

            for recognitionWordList in labels {
                var label = recognitionWordList

TranslationManager.sharedInstance.translateDetectionObjects(
                        inputWord: recognitionWordList.label,
                    fromLanguage: ListOfLanguages.English.rawValue,
                      toLanguage: ListOfLanguages.Russian.rawValue,
                                callback: { translation in
                                    recognitionLabels!.text =
(recognitionLabels?.text)! + translation + "\n" })
            }
        }
    }

    @IBAction func selectImage(_ sender: Any) {
        stopVideo()
        let picker = UIImagePickerController()
        picker.sourceType = .photoLibrary
        picker.delegate = self
        present(picker, animated: true, completion: nil)
    }

    func stopVideo() {
        CaptureController.instance.stopVideoCapture()
        view.bringSubviewToFront(imageView)
        toggleCameraButton.setTitle("Start video", for: .normal)
    }

    func startVideo() {
        CaptureController.instance.startVideoCapture()
        view.bringSubviewToFront(videoView)
        toggleCameraButton.setTitle("Stop video", for: .normal)
    }

    @IBAction func toggleCamera(_ sender: Any) {
        if CaptureController.instance.captureSession.isRunning {
            stopVideo()
```

```
        } else {
            startVideo()
        }
    }

    //Feature for sending feedback on the email
*envision.mobile.feedback@gmail.com*
    @IBAction func sendFeedback(_ sender: UIButton) {
        let feedbackManager = FeedbackManager()
        let mailComposeViewController =
feedbackManager.configureMailController()
        if MFMailComposeViewController.canSendMail() {
            self.present(mailComposeViewController, animated: true,
completion: nil)
        } else {
            feedbackManager.showMailError()
        }
    }
}

// Helper function inserted by Swift 4.2 migrator.
fileprivate func
convertFromUIImagePickerControllerInfoKeyDictionary(_ input:
[UIImagePickerController.InfoKey: Any]) -> [String: Any] {
  return Dictionary(uniqueKeysWithValues: input.map {key, value in
(key.rawValue, value)})
}

// Helper function inserted by Swift 4.2 migrator.
fileprivate func convertFromUIImagePickerControllerInfoKey(_ input:
UIImagePickerController.InfoKey) -> String {
  return input.rawValue
}
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	113 с.
	Інші документи	
2.	Презентаційні матеріали	20 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Зареченський О.А.			Методи розпізнавання та ідентифікації об’єктів у доповненої реальності і переклади їх назви на іноземну мову
Перевірив		Шевченко О.Ю.			
Н.контр.		Малєєва І.А.			
Затв.		Філатов В.О.			

Шифр групи	Код напр./спец.
СШІм-19-2	122
ХНУРЕ кафедра ШІ	