|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Информатика и вычислительная техника \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_Автоматизированные системы обработки информации и управления\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***Распознавание пола на изображении при помощи инструментов Swift и Xcode***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_\_ИУ5-65Б\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_Шкарин Е.А \_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Гапанюк Ю.Е. \_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2022 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме \_\_\_Распознавание пола на изображении при помощи инструментов Swift и Xcode\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_\_ИУ5-65Б\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шкарин Егор Алексеевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

\_Исследовательская\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к \_\_\_ нед., 50% к \_\_\_ нед., 75% к \_\_ нед., 100% к \_\_\_ нед.

Техническое задание:\_\_создать модель с помощью Swift, внедрить ее в приложение, классифицирующее изображения людей по полу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на \_\_\_\_\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_20\_\_ » \_\_\_\_\_марта\_\_\_\_\_\_\_ 2022 г.

**Руководитель НИР**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Гапанюк Ю.Е \_\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шкарин Е.А. \_\_\_**

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

[Введение 4](#_Toc105327069)

[Ход работы 4](#_Toc105327070)

[Тесты. 6](#_Toc105327071)

[Выводы. 16](#_Toc105327072)

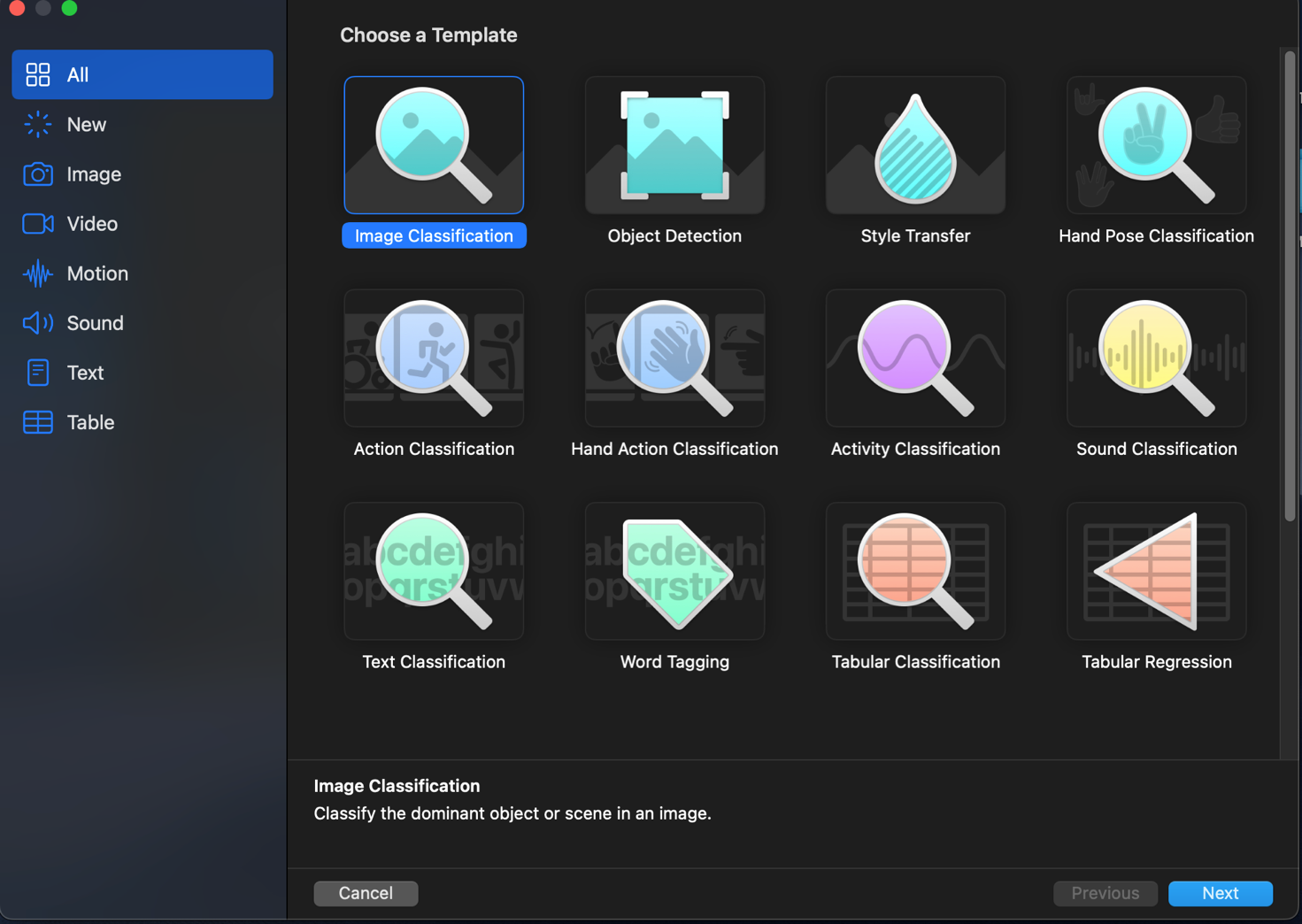
[Литература 16](#_Toc105327073)

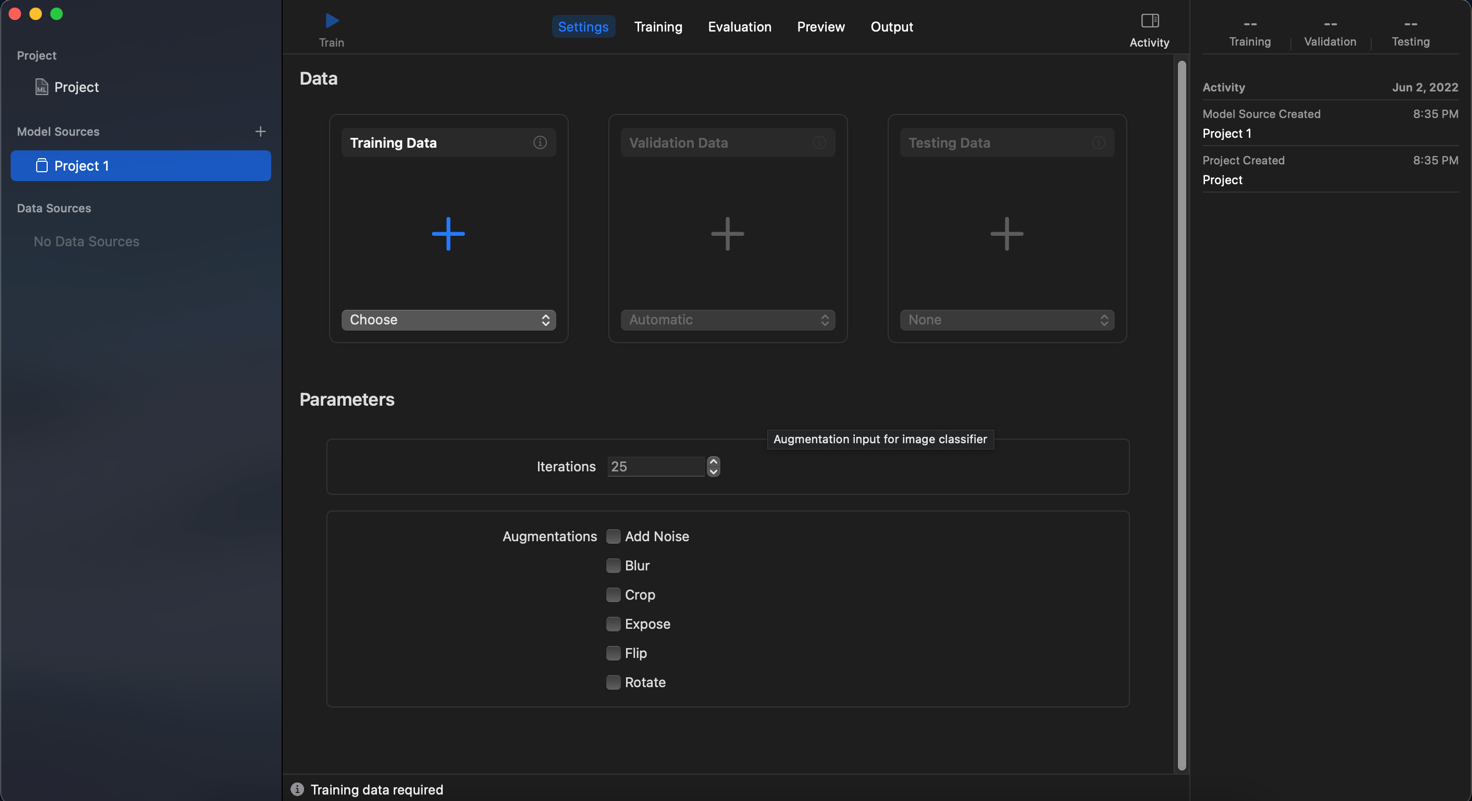
# Введение

В современном мире машинное обучение одно из самых популярных направлений. Данное направление нашло реализацию также и в таком языке, как Swift. В 2018 году Apple представила инструменты для работы с машинным обучением Create ML, поэтому я решил воспользоваться данным инструментом и проверить, как можно интегрировать машинное обучение в приложения под IOS.

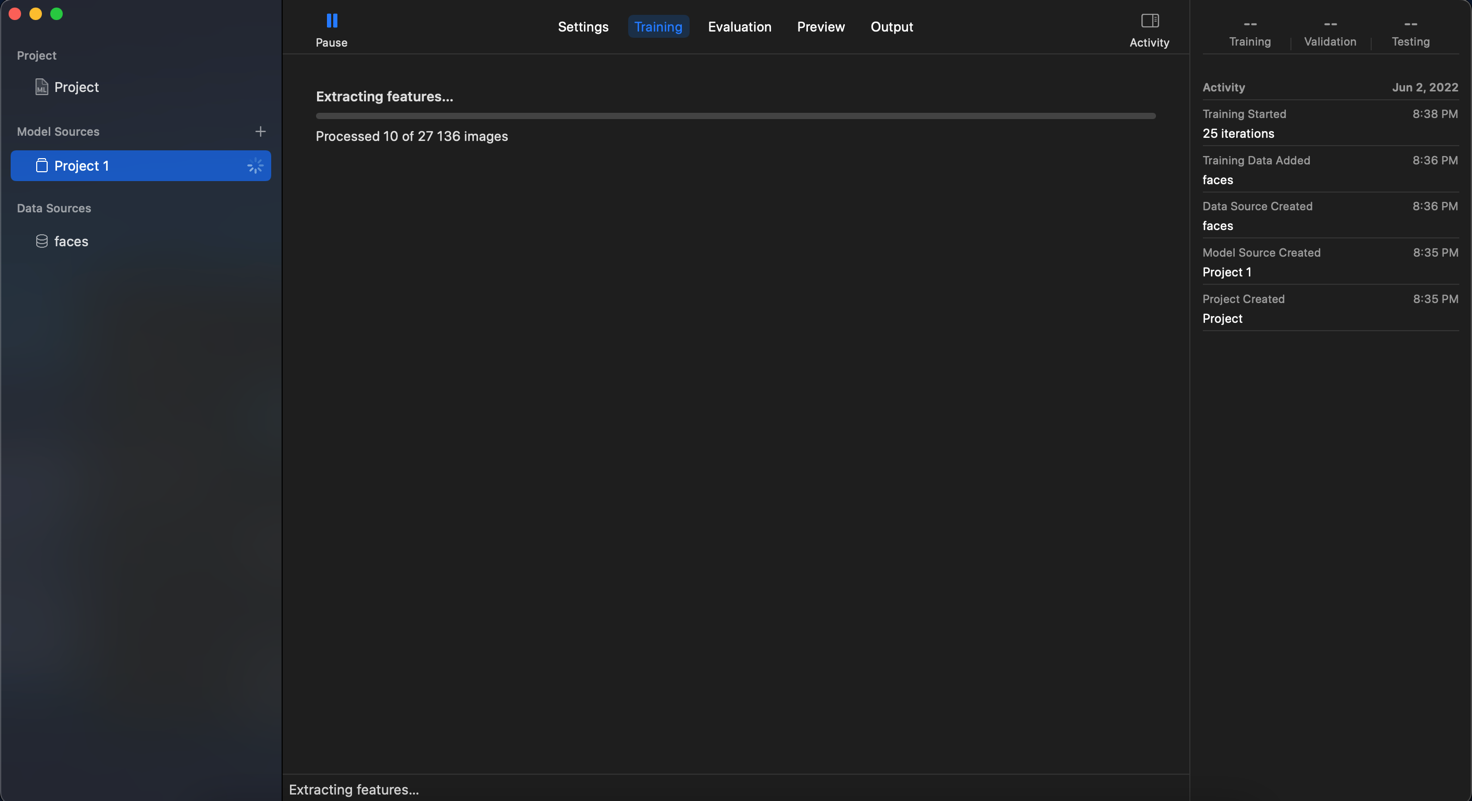
Create ML использует метод трансферного обучения, который направлен на упрощение обучения моделей. Для этого подхода не нужны большие объемы данных. Достаточно всего 10 изображений. Однако, чем больше, тем лучше. Также важно уточнить, что у этого подхода есть недостаток, в частности при перевешивании в один признак, может быть некорректное определения признаков.

# Ход работы

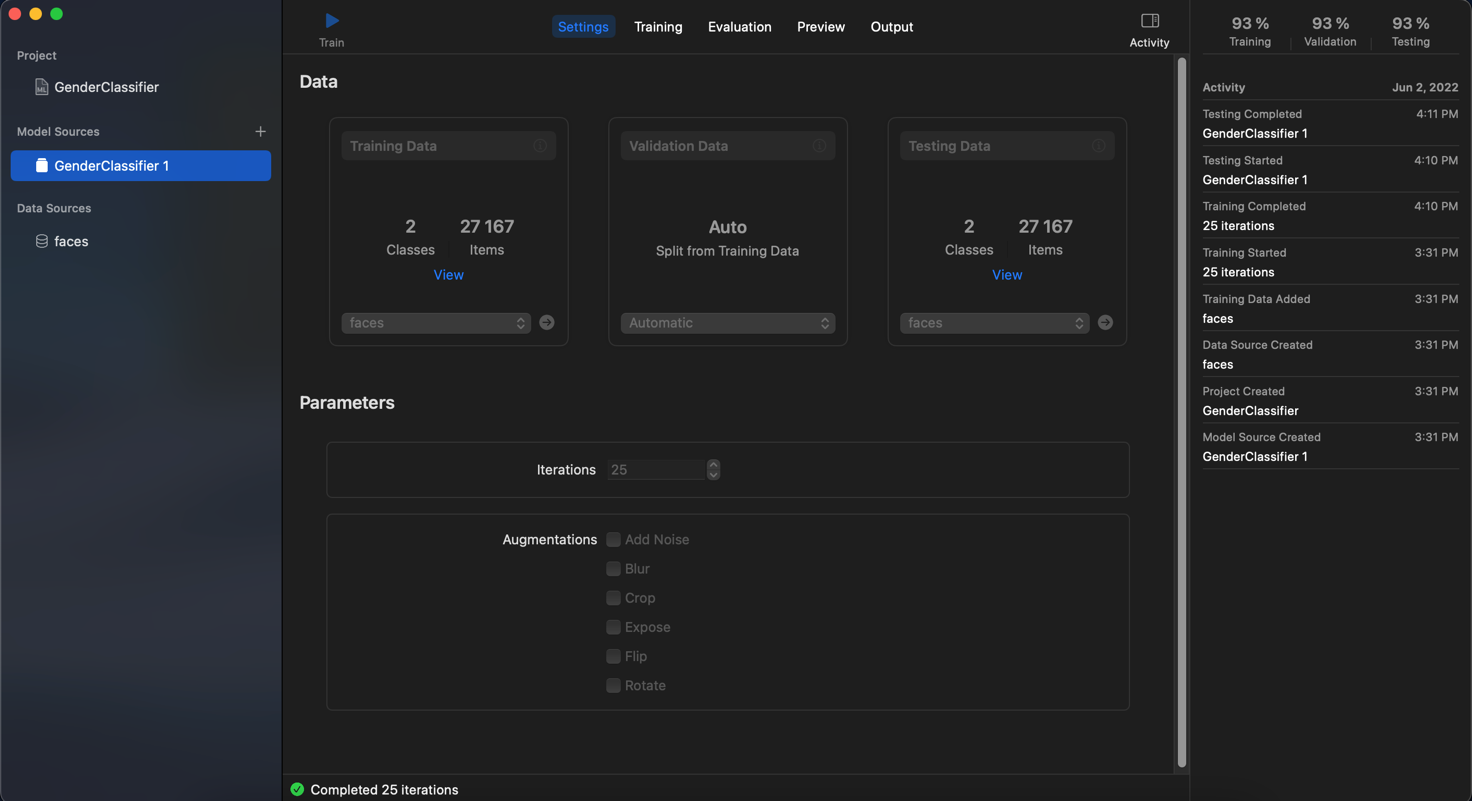
В начале необходимо создать проект Create ML. 

Здесь есть выбор между различными уже проработанными моделями, нам необходима Image Classification. 

После создания проекта, у нас появляется окно, в котором необходимо добавить датасет при помощи которого мы хотим обучать модель.



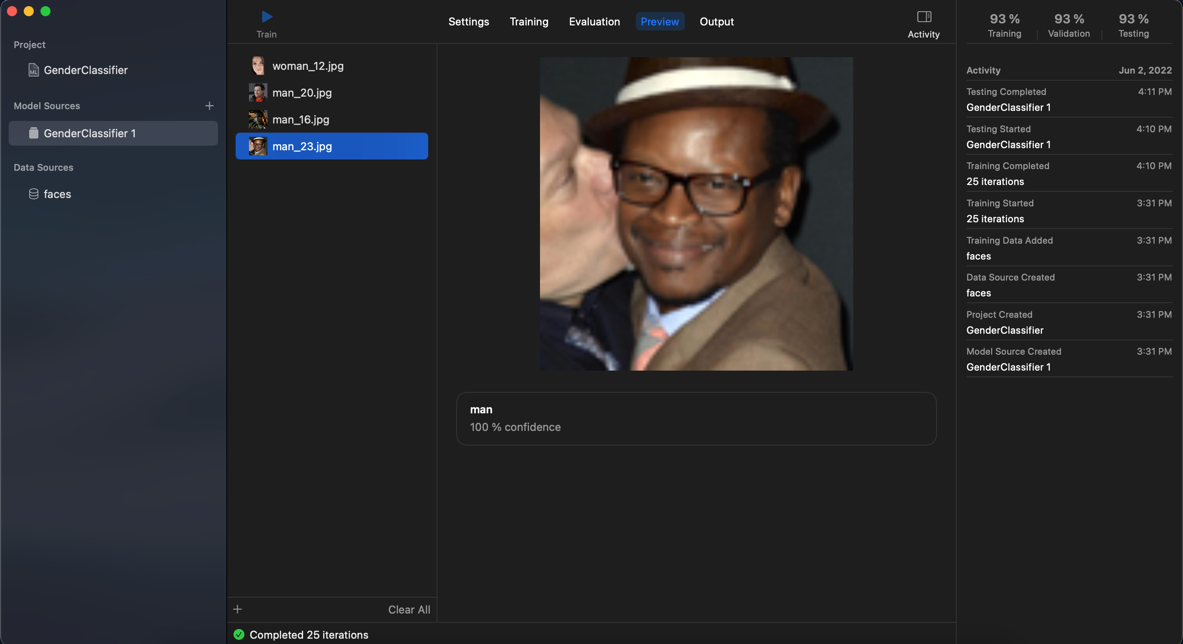
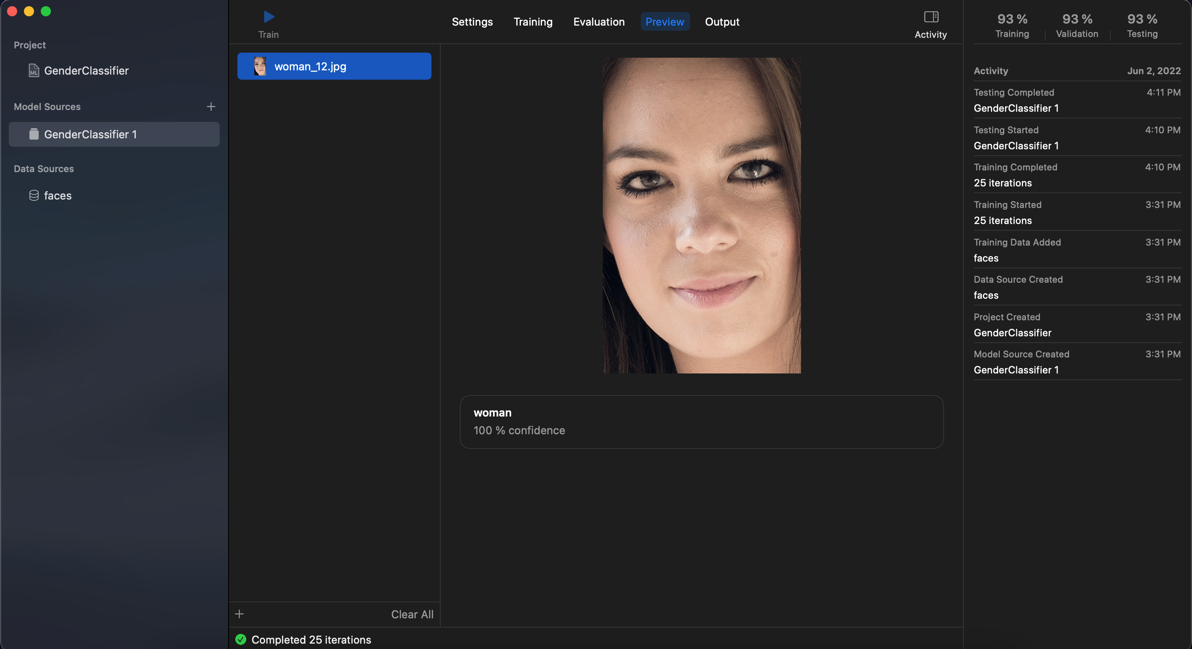
Далее нам необходимо нажать на кнопку “Train”, после чего запустится процесс обучения. Он может быть долгим.



После того, как закончится процесс обучения можно посмотреть на сколько классов разбит наш датасет и сколько в общей сложности картинок было обработано.

Теперь можно приступать к тестам.

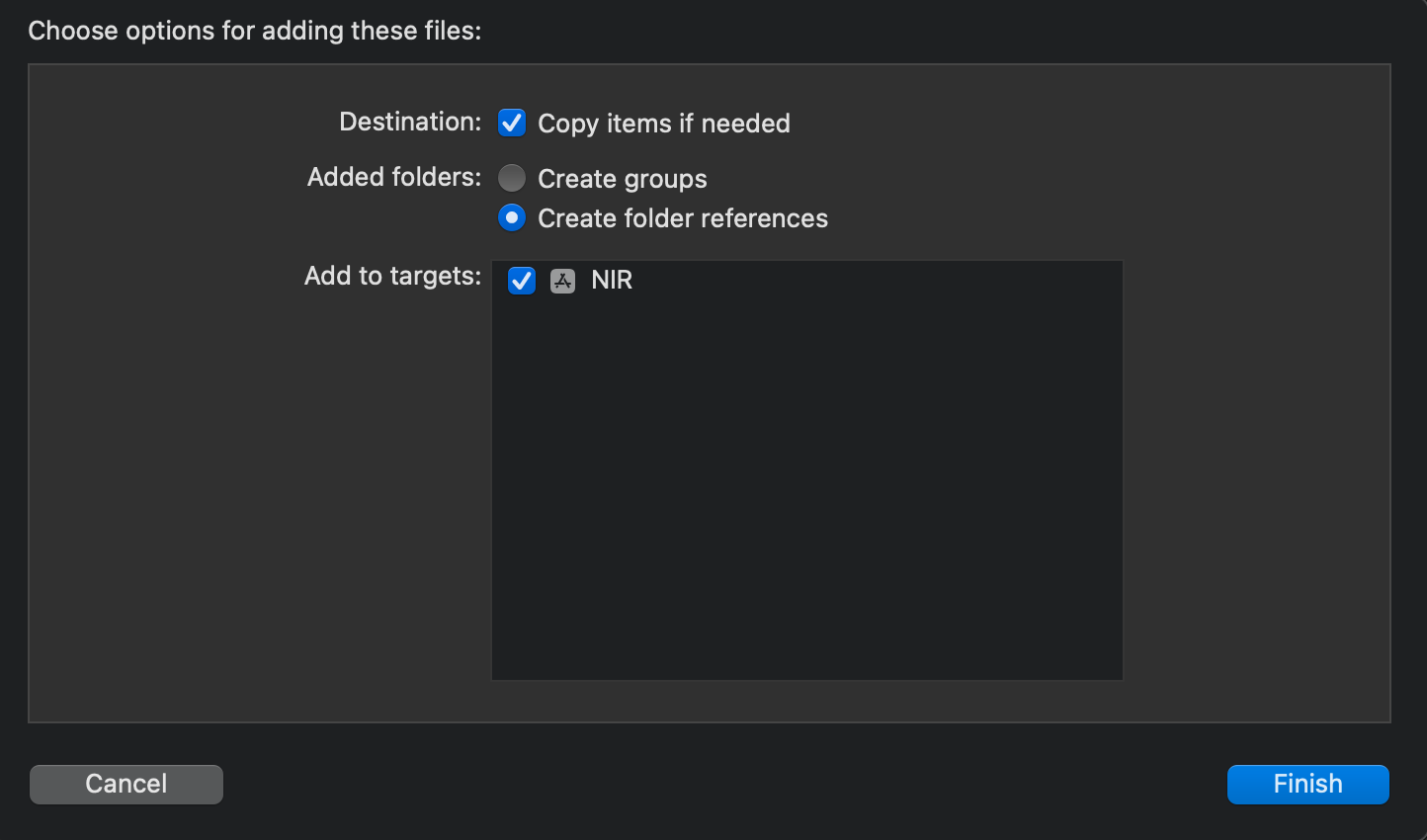
# Тесты.

Во-первых, стоит проверить непосредственно в самом Create ML. Результат: 

Теперь можно приступать к разработке приложения, с использованием этой модели.

Перед этим нам нужно сохранить модель.

В первую очередь нам надо добавить модель в Bundle проекта и написать класс, который будет отвечать за работу с моделью. Это необязательно, но так приятнее.



//

// ImagePredictor.swift

// NIR

//

// Created by Егор Шкарин on 02.06.2022.

//

import Foundation

import UIKit

import Vision

final class ImagePredictor {

static func createImageClassifier() -> VNCoreMLModel {

let defaultConfig = MLModelConfiguration()

let imageClassifierWrapper = try? GenderClassifier\_1(configuration: defaultConfig)

guard let imageClassifier = imageClassifierWrapper else {

fatalError("App failed to create an image classifier model instance.")

}

let imageClassifierModel = imageClassifier.model

guard let imageClassifierVisionModel = try? VNCoreMLModel(for: imageClassifierModel) else {

fatalError("App failed to create a VNCoreMLModel instance.")

}

return imageClassifierVisionModel

}

private static let imageClassifier = createImageClassifier()

struct Prediction {

let classification: String

let confidencePercentage: String

}

typealias ImagePredictionHandler = (\_ predictions: [Prediction]?) -> Void

private var predictionHandlers = [VNRequest: ImagePredictionHandler]()

private func createImageClassificationRequest() -> VNImageBasedRequest {

let imageClassificationRequest = VNCoreMLRequest(model: ImagePredictor.imageClassifier,

completionHandler: visionRequestHandler)

imageClassificationRequest.imageCropAndScaleOption = .centerCrop

return imageClassificationRequest

}

func makePredictions(for photo: UIImage, completionHandler: @escaping ImagePredictionHandler) throws {

let orientation = CGImagePropertyOrientation(photo.imageOrientation)

guard let photoImage = photo.cgImage else {

return

}

let imageClassificationRequest = createImageClassificationRequest()

predictionHandlers[imageClassificationRequest] = completionHandler

let handler = VNImageRequestHandler(cgImage: photoImage, orientation: orientation)

let requests: [VNRequest] = [imageClassificationRequest]

try handler.perform(requests)

}

private func visionRequestHandler(\_ request: VNRequest, error: Error?) {

guard let predictionHandler = predictionHandlers.removeValue(forKey: request) else {

fatalError("Every request must have a prediction handler.")

}

var predictions: [Prediction]? = nil

defer {

predictionHandler(predictions)

}

if let error = error {

print("Vision image classification error...\n\n\(error.localizedDescription)")

return

}

if request.results == nil {

print("Vision request had no results.")

return

}

guard let observations = request.results as? [VNClassificationObservation] else {

print("VNRequest produced the wrong result type: \(type(of: request.results)).")

return

}

predictions = observations.map { observation in

Prediction(classification: observation.identifier,

confidencePercentage: observation.confidencePercentageString)

}

}

}

extension VNClassificationObservation {

var confidencePercentageString: String {

let percentage = confidence \* 100

switch percentage {

case 100.0...:

return "100%"

case 10.0..<100.0:

return String(format: "%2.1f", percentage)

case 1.0..<10.0:

return String(format: "%2.1f", percentage)

case ..<1.0:

return String(format: "%1.2f", percentage)

default:

return String(format: "%2.1f", percentage)

}

}

}

import ImageIO

extension CGImagePropertyOrientation {

init(\_ orientation: UIImage.Orientation) {

switch orientation {

case .up: self = .up

case .down: self = .down

case .left: self = .left

case .right: self = .right

case .upMirrored: self = .upMirrored

case .downMirrored: self = .downMirrored

case .leftMirrored: self = .leftMirrored

case .rightMirrored: self = .rightMirrored

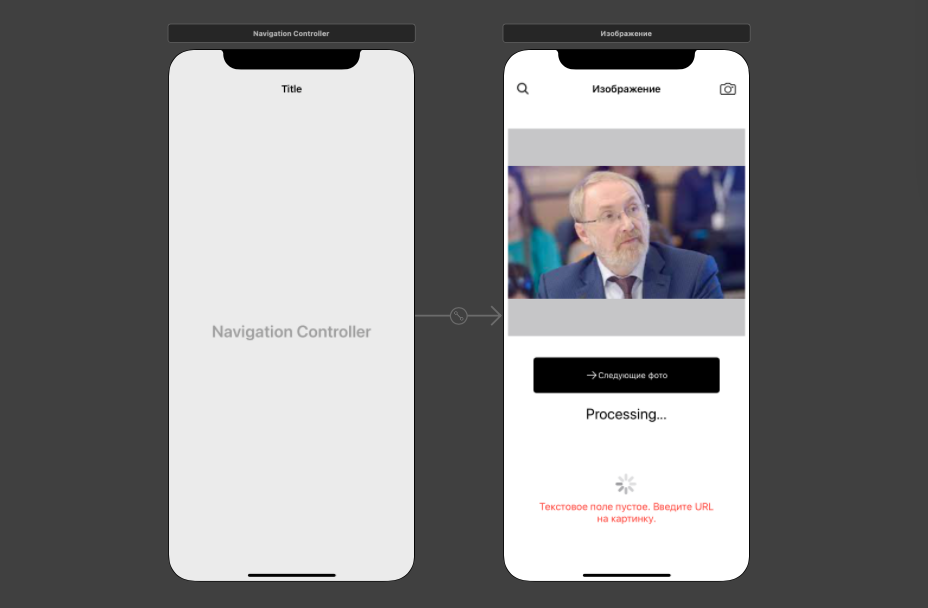
@unknown default: self = .up

}

}

}

Далее идет работа с Storyboard для создания интерфейса.



После этого настраиваем нам контроллер.

//

// NavalNIRa.swift

// NIR

//

// Created by Егор Шкарин on 02.06.2022.

//

import UIKit

class NavalNIRa: UIViewController, ImagePickerDelegate {

@IBOutlet weak var imageView: UIImageView!

@IBOutlet weak var nextButton: UIButton!

private var imagePicker: ImagePicker!

@IBOutlet weak var errorLabel: UILabel!

@IBOutlet weak var activity: UIActivityIndicatorView!

@IBOutlet weak var percentLabel: UILabel!

let imagePredictor = ImagePredictor()

let predictionsToShow = 2

override func viewDidLoad() {

super.viewDidLoad()

view.backgroundColor = .white

errorLabel.isHidden = true

activity.isHidden = true

self.imagePicker = ImagePicker(presentationController: self, delegate: self)

let startImage = UIImage(named: "face9")

imageView.image = startImage

checkImage(image: startImage ?? UIImage())

imageView.layer.cornerRadius = 20

imageView.contentMode = .scaleAspectFill

imageView.backgroundColor = .gray

}

override func viewWillLayoutSubviews() {

super.viewWillLayoutSubviews()

percentLabel.layer.cornerRadius = 10

}

func didSelect(image: UIImage?) {

self.imageView.image = image

checkImage(image: image ?? UIImage())

}

@IBAction func takePhoto(\_ sender: UIBarButtonItem) {

if !errorLabel.isHidden {

errorLabel.isHidden = true

}

self.imagePicker.present(from: view)

}

@IBAction func nextButtonAction(\_ sender: UIButton) {

let randNum = Int.random(in: 9...29)

let randImage = UIImage(named: "face\(randNum)") ?? UIImage()

self.imageView.image = randImage

DispatchQueue.global(qos: .userInitiated).async {

self.classifyImage(randImage)

}

if !errorLabel.isHidden {

errorLabel.isHidden = true

}

}

@IBAction func newPhoto(\_ sender: UIBarButtonItem) {

if !errorLabel.isHidden {

errorLabel.isHidden = true

}

let alertController = UIAlertController(title: "Найти фото",

message: "Вставте сслыку на фото из интернета",

preferredStyle: .alert)

alertController.addTextField()

let action = UIAlertAction(title: "Найти", style: .default) {action in

guard let text = alertController.textFields?.first?.text, !text.isEmpty else {

self.errorLabel.isHidden = false

self.setErrorMessage(text: "Вставте ссылку на картинку")

return

}

Task {

do {

self.activity.isHidden = false

self.activity.startAnimating()

let data = try await NetworkManager.shared.downloadImage(urlString:text)

self.setImage(data: data)

} catch NetworkErrors.badUrl {

self.setErrorMessage(text: "Неверный URL, пожалуйста проверьте URL и повторите попвтку")

self.activity.stopAnimating()

self.activity.isHidden = true

} catch NetworkErrors.internalError {

self.setErrorMessage(text: "Внутренняя ошибка, повторите попытку позже")

self.activity.stopAnimating()

self.activity.isHidden = true

} catch NetworkErrors.noInternetConnection {

self.setErrorMessage(text: "Нет подключения к интернету, проверьте подлючение и повторите снова")

self.activity.stopAnimating()

self.activity.isHidden = true

} catch {

self.setErrorMessage(text: "Неизвестная ошибка")

self.activity.stopAnimating()

self.activity.isHidden = true

}

}

}

alertController.addAction(action)

present(alertController, animated: true)

}

func updatePredictionLabel(\_ message: String) {

DispatchQueue.main.async {

self.percentLabel.text = message

}

}

private func classifyImage(\_ image: UIImage) {

do {

try self.imagePredictor.makePredictions(for: image,

completionHandler: imagePredictionHandler)

} catch {

print("Vision was unable to make a prediction...\n\n\(error.localizedDescription)")

}

}

/// The method the Image Predictor calls when its image classifier model generates a prediction.

/// - Parameter predictions: An array of predictions.

/// - Tag: imagePredictionHandler

private func imagePredictionHandler(\_ predictions: [ImagePredictor.Prediction]?) {

guard let predictions = predictions else {

updatePredictionLabel("No predictions. (Check console log.)")

return

}

let formattedPredictions = formatPredictions(predictions)

let predictionString = formattedPredictions.joined(separator: "\n")

updatePredictionLabel(predictionString)

}

private func formatPredictions(\_ predictions: [ImagePredictor.Prediction]) -> [String] {

// Vision sorts the classifications in descending confidence order.

let topPredictions: [String] = predictions.prefix(predictionsToShow).map { prediction in

var name = prediction.classification

// For classifications with more than one name, keep the one before the first comma.

if let firstComma = name.firstIndex(of: ",") {

name = String(name.prefix(upTo: firstComma))

}

return "\(name) - \(prediction.confidencePercentage)%"

}

return topPredictions

}

@MainActor

private func setErrorMessage(text: String) {

self.errorLabel.text = text

self.errorLabel.isHidden = false

}

@MainActor

private func setImage(data: Data) {

let recivedImage = UIImage(data: data) ?? UIImage()

imageView.image = recivedImage

if imageView.image != nil {

print("sfsdfs")

} else {

self.setErrorMessage(text: "Неизвестная ошибка")

imageView.backgroundColor = .opaqueSeparator

}

activity.stopAnimating()

activity.isHidden = true

self.errorLabel.isHidden = true

checkImage(image: recivedImage)

}

func checkImage(image: UIImage) {

DispatchQueue.global(qos: .userInitiated).async {

self.classifyImage(image)

}

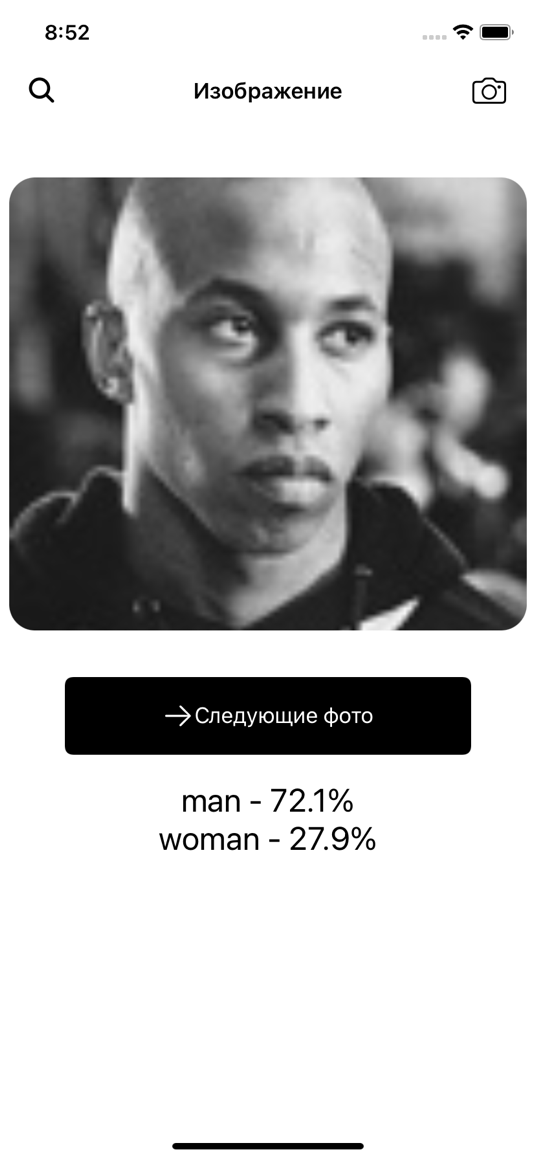
}

}

(существуют еще несколько классов помощников, в частности для работы с библиотекой фото и сетью, но листинг не будет приведен, так как это не относится к работе)

Теперь можно переходит к тестам.





# Выводы.

Из результатов работы видно, что есть некоторая неточность в определении. Это произошло как раз из-за того, что фотографий женщин было меньше, чем мужчин. Однако как инструмент для обучения моделей, Create ML очень удобен и эффективен для IOS разработки.

# Литература

Документация Apple.

<https://developer.apple.com/machine-learning/create-ml/>

Документация Apple.

<https://developer.apple.com/documentation/createml>

Статья на интернет-ресурсе «Хабр»

<https://habr.com/ru/post/490936/>