Инженерный вестник Дона, №6 (2025)
ivdon.ru/ru/magazine/archive/n6y2025/10153

Программное обеспечение подсистемы идентификации оператора в мобильном тренажере на основе нейронной сети

М.П. Василевский[1], *И.С. Полевщиков*[1,2]

[1]*Российский биотехнологический университет*
[2]*Пермский национальный исследовательский политехнический университет*

Аннотация: Представлены результаты исследования, посвященного разработке подсистемы идентификации оператора производственного процесса в мобильном тренажере, используемом для обучения и контроля профессиональных навыков. Формализовано на основе визуальных моделей в нотации UML описаны функциональные требования к подсистеме идентификации оператора на основе нейросетевых технологий, процессы взаимодействия пользователя с подсистемой распознавания личности, загрузки эталонного изображения для дальнейшей идентификации оператора при обучении и контроле на тренажере. Разработан прототип подсистемы на основе языка программирования Kotlin и библиотеки TensorFlow. Разработанная подсистема анализа изображений обладает высокой скоростью обнаружения и инициализации лиц, достигающей менее 0,5 с, что делает ее особенно эффективной для задач реального времени, где быстродействие играет ключевую роль. Локальная обработка данных на мобильных устройствах обеспечивает защиту конфиденциальности пользователей, исключая передачу данных на удаленные серверы, что минимизирует риски утечек информации. Оптимизация энергопотребления обеспечивает продолжительную работу на устройствах с ограниченной емкостью батареи, что делает систему удобной и практичной в использовании. Рассмотренную подсистему планируется адаптировать для контроля формирования навыков работы на оборудовании при обучении операторов на мобильных тренажерах. Подсистема, основанная на технологиях VR/AR, а также обученной нейронной сети, будет осуществлять анализ данных о реакциях пользователей в режиме реального времени.
Ключевые слова: мобильные тренажеры, нейронные сети, идентификация пользователя, профессиональное обучение, диаграммы UML, TensorFlow.

1 Введение

Современные мобильные устройства являются неотъемлемой частью повседневной жизни и профессиональной деятельности специалистов в различных отраслях (в частности, промышленность, образование), их функциональность постоянно расширяется.

Одной из задач, требующей практического применения мобильных технологий, является разработка подсистемы идентификации оператора производственного процесса в мобильном тренажере. Подсистема

обеспечивает решение задачи прокторинга в образовательных и корпоративных структурах (например, при внутрифирменном обучении сотрудников на предприятии) – автоматической идентификации участников прохождения практических заданий (упражнений) и/или тестирования (в частности, в рамках инструктажей по технике безопасности), отслеживания их присутствия в кадре. Рассматриваемая подсистема минимизирует возможность мошенничества, обеспечивает точный учет прохождения, фиксируя личности сотрудников, автоматизирует регистрацию присутствия, что освобождает преподавателей и других сотрудников от рутинных задач.

Современные мобильные устройства стали достаточно мощными, чтобы выполнять сложные вычисления, такие как распознавание объектов, классификация изображений и анализ сцены, прямо на устройстве. Одной из ключевых возможностей таких устройств является способность обрабатывать и анализировать изображения в реальном времени. Указанные задачи выполняются локально, без необходимости передачи данных на удаленные серверы. Это особенно важно в условиях строгих требований к конфиденциальности данных и быстродействию приложений.

Решению актуальной для предприятий и учебных заведений задачи обеспечения быстрой и надежной идентификации операторов посредством разработки подсистемы, интегрированной в мобильный тренажер, посвящено настоящее исследование, основные результаты которого изложены далее.

2 Обзор программных систем для идентификации личности

Разработке и применению моделей, алгоритмов и программных продуктов, релевантных решению задачи идентификации личности, посвящены труды различных ученых.

В статье [1] рассмотрены современные подходы к распознаванию лиц, включая метод гибкого сравнения на графах, метод главных компонент,

активные модели внешнего вида и нейронные сети. Авторы исследуют ключевые характеристики каждого метода, выявляют их преимущества и недостатки, а также определяют основные аспекты, влияющие на точность и эффективность распознавания лиц. Однако, применительно к теме настоящего исследования, данная работа не фокусируется непосредственно на использовании этих методов в мобильных приложениях.

Работа [2] посвящена анализу современных методов распознавания лиц с использованием нейронных сетей. Подробно рассматриваются архитектуры MTCNN (Multi-task Cascaded Convolutional Networks – каскадная сверточная сеть для одновременного обнаружения лиц и определения их ключевых точек) и FaceNet (архитектура глубокого машинного обучения для преобразования изображений лиц в компактные эмбеддинги), их особенности и применение в реальных задачах. Преимуществами разработки являются высокая точность и возможность работы в реальном времени. Применительно к теме настоящего исследования, реализация этих моделей на мобильных устройствах может потребовать значительных вычислительных ресурсов и оптимизации.

В исследовании [3] обсуждаются возможности применения искусственного интеллекта, в частности машинного обучения, в разработке мобильных приложений. Авторы рассматривают примеры использования технологий распознавания изображений и речи, обсуждают перспективы интеграции этих технологий в мобильные устройства. Относительно темы настоящего исследования статья носит общий характер и не предоставляет детальных технических решений для распознавания лиц.

В работе [4] описано создание обучаемой системы распознавания лиц с использованием алгоритмов сверточной нейронной сети. Автор представил модель, способную распознавать новые лица в реальном времени, автоматически выделяя черты лица и сравнивая их с ранее известными данными для идентификации. Отличительными особенностями модели

являются использование сверточной нейронной сети и реализация на языке программирования Python. Преимуществами разработки являются возможность работы в реальном времени и перспективное применение в областях безопасности, видеонаблюдения, автоматизации систем контроля доступа. Относительно темы настоящего исследования, данная модель демонстрирует сравнительно низкую точность по сравнению с современными алгоритмами распознавания лиц, что может ограничивать ее практическое использование.

В статье [5] описана разработка интеллектуальной системы для распознавания лиц с использованием нейронных сетей. Авторами предложен метод создания сложных архитектур с использованием различных признаков и дополнительных алгоритмов. Отличительной особенностью системы является возможность анализа нескольких кадров, подтверждающих микродвижения головы или моргание, что повышает точность идентификации. Преимуществами разработки являются улучшенная надежность системы и снижение количества ложных срабатываний. Применительно к теме настоящего исследования, система может быть чувствительна к качеству входных данных и требовать значительных вычислительных ресурсов.

Авторами работы [6] представлены три подхода к распознаванию лиц, реализованных в программном комплексе визуальной идентификации: алгоритмы Eigenface (метод распознавания лиц с помощью анализа главных компонент изображения), Fisherface (метод распознавания лиц с использованием линейного дискриминантного анализа) и LBP (Local Binary Patterns – метод описания текстуры изображения по локальным бинарным шаблонам). Описаны соответствующие алгоритмы, приводятся графики и фрагменты их программной реализации, а также анализируются результаты их использования. Отличительными особенностями исследования являются

сравнительный анализ эффективности методов и демонстрация их работы на практике. Преимуществами работы являются детальное изучение различных алгоритмов и их практическое применение. Относительно темы настоящего исследования, статья не фокусируется на использовании этих методов в мобильных приложениях.

В статье [7] рассмотрена разработка и оптимизация модели глубокого обучения с использованием TensorFlow Lite для распознавания активности человека на смартфоне. Авторами предложена модель на основе многослойной сети долгой краткосрочной памяти (LSTM), предназначенная для работы на мобильных устройствах. Отличительными особенностями исследования являются фокус на оптимизации модели для ограниченных ресурсов смартфонов и использование квантования для уменьшения размера модели. Преимуществами разработки являются высокая точность распознавания активности (92,7%) и небольшой размер модели (27 КБ), что способствует эффективной работе на мобильных устройствах. Однако, применительно к теме настоящего исследования, данная модель не может быть непосредственно интегрирована в приложения, требующие обработки видео в реальном времени.

В настоящем исследовании будут учтены особенности, преимущества и недостатки научных трудов и разработок в области распознавания образов для создания эффективного решения по идентификации личности операторов производственного процесса в мобильном тренажере.

3 Основные требования к подсистеме идентификации оператора в мобильном тренажере

Функциональные требования к подсистеме идентификации оператора в мобильном тренажере представлены визуально диаграммой Use Case UML (рис. 1). Инструктор загружает эталонное изображение в систему, чтобы

использовать его как базу для распознавания лиц. Оператор активирует камеру смартфона, чтобы зафиксировать изображение для анализа. Система отправляет захваченное изображение в обученную модель, чтобы выполнить сравнение с эталоном. Инструктор и оператор получают уведомление с результатом анализа: совпадение лица с эталоном или несовпадение.

Рис. 1. – Функциональные требования к подсистеме идентификации оператора в мобильном тренажере

Процесс взаимодействия пользователя с подсистемой идентификации личности представлен на рис. 2 диаграммой состояний UML.

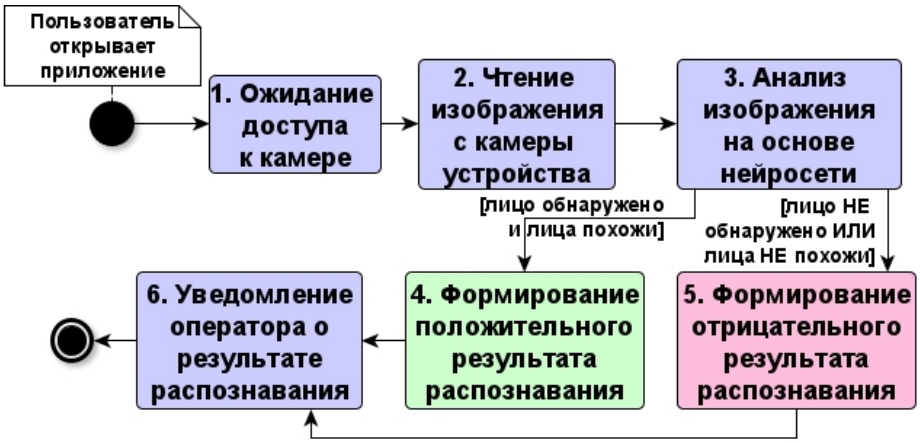

Рис. 2. – Взаимодействие пользователя с подсистемой идентификации личности

Процесс загрузки инструктором эталонного изображения для дальнейшей идентификации оператора в процессе обучения и контроля на тренажере показан на рис. 3 диаграммой состояний UML.

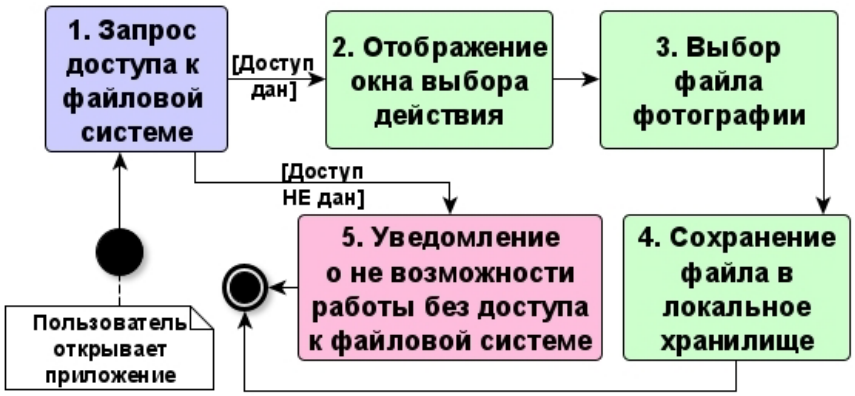

Рис. 3. – Процесс загрузки эталонного изображения для дальнейшей идентификации оператора при обучении и контроле на тренажере

При разработке подсистемы идентификации оператора на основе анализа изображений в мобильных устройствах необходимо учитывать ограничения по вычислительным ресурсам, энергопотреблению и разнообразию платформ. Для достижения требуемого уровня производительности предлагается использовать оптимизированные алгоритмы и фреймворки, такие как TensorFlow Lite (облегченная версия платформы TensorFlow для выполнения моделей машинного обучения на мобильных и встраиваемых устройствах) [8-11]. Данный инструмент обеспечивает использование обученных нейросетей, что позволяет реализовать сложные функции, такие как распознавание объектов, классификация изображений и сегментация [12, 13].

Преимущества подсистемы идентификации личности оператора в мобильном тренажере, с учетом обозначенных выше возможностей подсистемы, обусловлены несколькими факторами:

1) выполнение задач обработки изображений и видео без необходимости передачи данных в облако;

2) выполнение анализа изображений локально на устройстве обеспечивает большую защиту (конфиденциальность) личных данных, чем облачные решения;

3) эффективность работы в режиме реального времени, поскольку локальный анализ изображений минимизирует задержки.

4 Программная реализация прототипа подсистемы идентификации оператора в мобильном тренажере

Рассмотрим программную реализацию прототипа подсистемы идентификации оператора в мобильном тренажере на базе операционной системы Android. Для достижения указанных выше функциональных и нефункциональных требований к подсистеме использован следующий стек технологий разработки приложений: язык программирования Kotlin 1.9.0; библиотека TensorFlow – для сравнения изображений; Kotlin Coroutines – для асинхронной работы; Google MlKit – для обнаружения лица в кадре (обрезка всего изображения в зоне лица); Google CameraX – для работы с камерой; Jetpack Compose – для работы с UI.

Для выполнения задачи сравнения лиц была выбрана предварительно обученная модель нейросети, специализирующаяся на задачах распознавания лиц. Модель прошла обучение на большом наборе данных и демонстрирует высокую точность в определении сходства между лицами.

Модуль захвата изображений реализован таким образом, чтобы обеспечить плавное перемещение «окошки» по экрану. Это достигается за счет использования специальных алгоритмов для управления движением «окошки», которые учитывают положение пальцев пользователя на экране.

После того как изображение захвачено и обработано, оно передается в блок сравнения с эталонным образом. Здесь происходит применение выбранной модели машинного обучения, которая анализирует сходства между текущим изображением и эталонным примером.

Создадим виджет DraggableItem, который отвечает за плавающее по всему экрану окошко (рис. 4). Данный компонент представляет собой Composable – функцию FloatingDraggableItem, которая позволяет пользователю свободно перемещать элемент внутри заданного контейнера. Внутри реализована обработка жестов с помощью pointerInput, что

обеспечивает интерактивность элемента и возможность его перемещения путем касания и перетаскивания.

```
@Composable
fun FloatingDraggableItem(
    initialOffset: (FloatingDraggableItemState) -> IntOffset,
    content: @Composable BoxScope.(FloatingDraggableItemState) -> Unit,
) {
    val state = remember { mutableStateOf(FloatingDraggableItemState()) }

    Box(
        modifier = Modifier
            .fillMaxSize()
            .onGloballyPositioned { state.updateContainerSize(size = it.size) },
    ) {
        Box(
            modifier = Modifier
                .width(120.dp)
                .height(200.dp)
                .background(
                    color = Color.Transparent,
                    shape = RoundedCornerShape(16.dp)
                )
                .offset { state.value.offset }
                .onGloballyPositioned {
                    state.updateContentSizeSize(
                        size = it.size,
                        initialOffset = initialOffset,
                    )
                }
                .pointerInput(Unit) {
                    detectDragGestures { _, dragAmount ->
                        val calculatedX = state.value.offset.x + dragAmount.x.roundToInt()
                        val calculatedY = state.value.offset.y + dragAmount.y.roundToInt()

                        val offset = IntOffset(
                            calculatedX.coerceIn(0, state.value.dragAreaSize.width),
                            calculatedY.coerceIn(0, state.value.dragAreaSize.height),
                        )
                        state.updateOffset(offset = offset)
                    }
                }
                .clip(shape = RoundedCornerShape(16.dp))
        ) {
            content(state.value)
        }

        DisposableEffect(true) {
            onDispose {
                val offset = initialOffset(state.value)
                state.updateOffset(offset = offset)
            }
        }
    }
}
```

Рис. 4. – Виджет FloatingDraggableItem

Далее создадим виджет, который будет считывать изображение с камеры (рис. 5, 6). Виджет реализуется с помощью CameraView (рис. 5), который отвечает за обработку видеопотока и анализ изображений. В основе компонента лежит CameraViewModel (рис. 6), управляющий состоянием анализа изображений и взаимодействием с моделью обработки лиц.

В CameraView создается AndroidView (компонент для внедрения нативных Android View в Compose-приложение), оборачивающий PreviewView для отображения видеопотока. Камера управляется через ProcessCameraProvider (API для управления жизненным циклом камеры в приложениях Android), который настраивается в factory. При получении изображения выполняется его анализ, а состояние обновляется через

analyseState. В зависимости от результата анализа виджет меняет цвет фона: зеленый фон указывает на успешное распознавание лица (UserRecognized); красный фон с таймером отображается при отсутствии лица, после чего вызывается onFaceError(), если лицо не обнаружено в течение заданного времени.

При отсутствии распознанного лица запускается LaunchedEffect (эффект побочного действия в Jetpack Compose, запускаемый при изменении состояния), в котором реализован таймер обратного отсчета (инициализирован с mutableIntStateOf(5)). Если в течение 5 секунд лицо не обнаружено, вызывается onFaceError.invoke(), сигнализируя об ошибке.

```
@Composable
fun CameraView(modifier: Modifier = Modifier, onFaceError: () -> Unit) {
    val viewModel: CameraViewModel = viewModel()

    val lifecycleOwner = LocalLifecycleOwner.current
    val state by viewModel.analiseState.collectAsState()
    AndroidView(
        factory = { context ->
            PreviewView(context).apply {
                ProcessCameraProvider.getInstance(context).apply {
                    addListener(
                        {
                         //Реализация
                        },
                        ContextCompat.getMainExecutor(context)
                    )
                }
            }
        },
        modifier = Modifier.fillMaxSize()
    )

    when (state) {
        AnalyseUserImageState.UserRecognized -> {
            Box(
                modifier = Modifier
                    .fillMaxSize()
                    .background(
                        Color.Green.copy(0.2f)
                    )
            )
        }

        AnalyseUserImageState.Init -> {}
        else -> {
            var timer by remember {
                mutableIntStateOf(5)
            }
            Box(
                modifier = Modifier
                    .fillMaxSize()
                    .background(
                        Color.Red.copy(0.2f)
                    ),
                contentAlignment = Alignment.Center
            ) {
                Text(
                    text = timer.toString(),
                    style = MaterialTheme.typography.titleLarge,
                    color = Color.White
                )
            }
            LaunchedEffect(key1 = Unit) {
                while (timer > 0) {
                    delay(1000)
                    timer--
                }
                onFaceError.invoke()
            }
        }
    }
}
```

Рис. 5. – Виджет CameraView

Основной метод, который будет использоваться в классе FileReader (рис. 7), – это метод run. В его рамках с помощью приватного метода scanImage извлекаются эмбеддинги изображения. Данный метод принимает в качестве аргумента список, содержащий пары, где первый элемент представляет название изображения, а второй – само изображение в формате Bitmap.

```
class CameraViewModel(
    private val applicationContext: Application
) : AndroidViewModel(applicationContext) {
    private val _analiseState = MutableStateFlow<AnalyseUserImageState>(AnalyseUserImageState.Init)
    val analiseState = _analiseState.asStateFlow()

    private val faceNetModel: FaceNetModel =
        FaceNetModel(applicationContext, ModelInfo.FACENET_512_QUANTIZED)
    val frameAnalyser: FrameAnalyser = FrameAnalyser(faceNetModel) {
        _analiseState.value = it
    }
    private val fileReader: FileReader = FileReader(faceNetModel)

    private val fileReaderCallback = object : FileReader.ProcessCallback {
        override fun onProcessCompleted(
            data: ArrayList<Pair<String, FloatArray>>,
            numImagesWithNoFaces: Int
        ) {
            frameAnalyser.faceList = data
        }
    }

    init {
        loadImgDataSet()
    }

    private fun loadImgDataSet() {
        viewModelScope.launch {
            fileReader.run(
                arrayListOf(
                    //reference img
                    "" to BitmapFactory.decodeResource(applicationContext.resources, R.mipmap.ivan)
                ), callback = fileReaderCallback
            )
        }
    }


}
```

Рис. 6. – Виджет CameraViewModel

После этого выполняется предварительная обработка изображения для последующего анализа. Также метод принимает в качестве аргумента функцию обратного вызова, назначение которой будет рассмотрено далее.

```
class FileReader(private var faceNetModel: FaceNetModel) {
    fun run(data: ArrayList<Pair<String, Bitmap>>, callback: ProcessCallback) {
        numImages = data.size
        this.data = data
        this.callback = callback
        scanImage(data[imageCounter].first, data[imageCounter].second)
    }
}
```

Рис. 7. – Класс FileReader

Далее рассмотрим класс FrameAnalyzer (рис. 8), работающий с подготовленным изображением. Для того чтобы передать в него данные используем функцию обратного вызова fileReaderCallBack из CameraViewModel (рис. 6).

```
class FrameAnalyser() : ImageAnalysis.Analyzer {
  override fun analyze(image: ImageProxy) {
        if (isProcessing || faceList.size == 0) {
            image.close()
            return
        } else {
            isProcessing = true
            val cameraXImage = image.image!!
            var frameBitmap = Bitmap.createBitmap(
                cameraXImage.width,
                cameraXImage.height,
                Bitmap.Config.ARGB_8888
            )
            frameBitmap.copyPixelsFromBuffer(image.planes[0].buffer)
            frameBitmap =
                BitmapUtils.rotateBitmap(frameBitmap, image.imageInfo.rotationDegrees.toFloat())

            val inputImage = InputImage.fromBitmap(frameBitmap, 0)
            detector.process(inputImage)
                .addOnSuccessListener { faces ->
                    CoroutineScope(Dispatchers.Default).launch {
                        runModel(faces, frameBitmap)
                    }
                }
                .addOnCompleteListener {
                    image.close()
                }
        }
    }

}
```

Рис. 8. – Класс FrameAnalyzer

На рис. 9 и 10 представлены примеры работы алгоритма распознавания лиц. В случае успешного обнаружения лица (рис. 9) область отображения камеры окрашивается в зеленый цвет, что сигнализирует о корректной идентификации пользователя. В противном случае (рис. 10), если лицо не распознано, область окрашивается в красный цвет, а в центре отображается обратный отсчет, указывающий на оставшееся время перед вызовом обработчика ошибки. По истечении заданного времени, если распознавание так и не произошло, система инициирует соответствующее действие, например, уведомление пользователя или повторный запуск процесса идентификации. Такой механизм позволяет визуально показать статус обработки изображения и информировать пользователя о результате работы алгоритма.

Рис. 9. – Успешное распознавание лица

Рис. 10. – Неудачное распознавание лица

5 Заключение

Разработанная система анализа изображений на базе TensorFlow Lite демонстрирует значительные преимущества и широкий спектр возможностей практического применения. Ее высокая скорость обнаружения и инициализации лиц, достигающая менее 0,5 секунды, делает ее особенно эффективной для задач реального времени, где быстродействие играет ключевую роль.

Локальная обработка данных на мобильных устройствах обеспечивает защиту конфиденциальности пользователей, исключая передачу данных на удаленные серверы, что минимизирует риски утечек информации. Кроме того, система обладает высокой степенью кроссплатформенной совместимости, включая Android и iOS. Оптимизация энергопотребления обеспечивает продолжительную работу на устройствах с ограниченной емкостью батареи, что делает систему удобной и практичной в использовании.

Рассмотренную подсистему планируется адаптировать для контроля формирования навыков работы на оборудовании при обучении операторов на мобильных тренажерах. Подсистема, основанная на технологиях VR/AR, а

также обученной нейронной сети, будет осуществлять анализ данных о реакциях пользователей в режиме реального времени.

Исследование выполнено за счет гранта Российского научного фонда № 23-79-10162, rscf.ru/project/23-79-10162/.

Литература

1. Натолина С.А., Кузнецова О.Ю. Анализ существующих подходов к распознаванию лиц // Научный журнал. 2024. № 2 (69). С. 5-10.

2. Безгачев Ф.В. Распознавание лиц на основе нейронных сетей: современные технологии // Научный компонент. 2021. № 4 (12). С. 56-63.

3. Денисенко В.В., Ященко А.С., Чесников Л.С. Применение искусственного интеллекта в разработке мобильных приложений // Международный журнал гуманитарных и естественных наук. 2023. № 2-2 (77). С. 18-21.

4. Койшыбаев Р.Ж. Разработка обучаемой системы распознавания лиц с помощью алгоритмов нейронной сети // Символ науки: международный научный журнал. 2023. № 6-1. С. 23-33.

5. Истратова Е.Е., Достовалов Д.Н., Бухамер Е.А. Разработка интеллектуальной системы для распознавания лиц на основе нейронных сетей // International Journal of Open Information Technologies. 2021. Т. 9. № 4. С. 41-45.

6. Кадров М.С., Туркевич А.С., Князева А.А. Анализ методов распознавания лиц людей // Международный журнал прикладных наук и технологий Integral. 2019. № 3. С. 26.

7. Salah Eddin Adi, Alexander J. Casson. Design and Optimization of a TensorFlow Lite Deep Learning Neural Network for Human Activity Recognition on a Smartphone // Proceedings of the 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), 2021. URL:

ieeexplore.ieee.org/document/9629549 (Дата обращения: 25.03.2025).

8. TensorFlow Lite: A Guide for Efficient Mobile Machine Learning. Официальная документация. URL: tensorflow.org/lite (Дата обращения: 08.12.2024).

9. TensorFlow Lite for Mobile and Embedded Devices. Официальная документация. URL: tensorflow.org/lite/guide (Дата обращения: 09.12.2024).

10. Real-Time Object Detection with TensorFlow Lite on Android. URL: blog.tensorflow.org/2021/01/real-time-object-detection-with-tensorflow-lite-on-android.html (Дата обращения: 09.12.2024).

11. Introduction to Mobile Vision with TensorFlow Lite. URL: towardsdatascience.com/introduction-to-mobile-vision-with-tensorflow-lite-bbcfc95cbb7b (Дата обращения: 12.12.2024).

12. Евдокимова Т.С. Влияние замены и расширения данных с применением преобразований на точность распознавания глубокой нейронной сети ResNet - 50 // Инженерный вестник Дона. 2025. №1. URL: ivdon.ru/ru/magazine/archive/n1y2025/9764.

13. Корчагин С.А., Сердечный Д.В. Алгоритмы компьютерного зрения для распознавания объектов в условиях плохой видимости // Инженерный вестник Дона. 2024. №10. URL: ivdon.ru/ru/magazine/archive/n10y2024/9577.

References

1. Natolina S.A., Kuznetsova O.Yu. Nauchnyj zhurnal. 2024. № 2 (69). pp. 5-10.

2. Bezgachev F.V. Nauchnyj komponent. 2021. № 4 (12). pp. 56-63.

3. Denisenko V.V., Yashchenko A.S., Chesnikov L.S. Mezhdunarodnyy zhurnal gumanitarnykh i estestvennykh nauk. 2023. № 2-2 (77). pp. 18-21.

4. Koyshybaev R.Zh. Simvol nauki: mezhdunarodnyj nauchnyj zhurnal. 2023. № 6-1. pp. 23-33.

5. Istratova E.E., Dostovalov D.N., Bukhamer E.A. International Journal of Open Information Technologies. 2021. Vol. 9. № 4. pp. 41-45.

6. Kadrov M.S., Turkevich A.S., Knyazeva A.A. Mezhdunarodnyj zhurnal prikladnykh nauk i tekhnologij Integral. 2019. № 3. p. 26.

7. Salah Eddin Adi, Alexander J. Casson. Design and Optimization of a TensorFlow Lite Deep Learning Neural Network for Human Activity Recognition on a Smartphone. Proceedings of the 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), 2021. URL: ieeexplore.ieee.org/document/9629549 (Date accessed 25.03.2025).

8. TensorFlow Lite: A Guide for Efficient Mobile Machine Learning. Ofitsial'naya dokumentatsiya [TensorFlow Lite: A Guide for Efficient Mobile Machine Learning. Official documentation]. URL: tensorflow.org/lite (Date accessed 08.12.2024).

9. TensorFlow Lite for Mobile and Embedded Devices. Ofitsial'naya dokumentatsiya [TensorFlow Lite for Mobile and Embedded Devices. Official documentation]. URL: tensorflow.org/lite/guide (Date accessed 09.12.2024).

10. Real-Time Object Detection with TensorFlow Lite on Android. URL: blog.tensorflow.org/2021/01/real-time-object-detection-with-tensorflow-lite-on-android.html (Date accessed 09.12.2024).

11. Introduction to Mobile Vision with TensorFlow Lite. URL: towardsdatascience.com/introduction-to-mobile-vision-with-tensorflow-lite-bbcfc95cbb7b (Date accessed 12.12.2024).

12. Evdokimova T.S. Inzhenernyj vestnik Dona, 2025, №1. URL: ivdon.ru/ru/magazine/archive/n1y2025/9764.

13. Korchagin S.A., Serdechnyy D.V. Inzhenernyj vestnik Dona, 2024, №10. URL: ivdon.ru/ru/magazine/archive/n10y2024/9577.

Дата поступления: 27.03.2025

Дата публикации: 25.05.2025