**СОДЕРЖАНИЕ**

Введение ...................................................................................................................................4

[1. Анализ предметной области. Постановка задачи 5](#_Toc135198584)

[2. Разработка и описание алгоритма детекции и распознавания номерных знаков автомобилей 6](#_Toc135198585)

[2.1. Получение разрешений устройства на камеру и др. 6](#_Toc135198586)

[2.2. Активность камеры 7](#_Toc135198587)

[2.3. Активность загрузки изображения 7](#_Toc135198588)

[2.4. Активность сохранённых распознаваний 7](#_Toc135198589)

[3. Разработка программы 8](#_Toc135198590)

[4. Тестирование 16](#_Toc135198591)

[4.1. Подготовка к тестированию 16](#_Toc135198592)

[1.2. Тестирование 16](#_Toc135198593)

[Заключение 20](#_Toc135198594)

[Список использованной литературы 21](#_Toc135198595)

**ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ**

**ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ. СХЕМА ТРЕКИНГА ВЫБРАННЫХ ОБЪЕКТОВ В ВИДЕОПОТОКЕ**

**ВВЕДЕНИЕ**

Автоматическое распознавание номерных знаков (Automatic License Plate Recognition - ALPR) — это технология, использующая оптическое распознавание символов на изображениях для считывания номерных знаков транспортных средств для создания данных о местоположении транспортных средств. Технология может использовать как существующие камеры, обеспечивающие соблюдение правил дорожного движения, так и камеры, специально разработанные для этой задачи. ALPR используется полицией по всему миру в правоохранительных целях, в том числе для проверки того, зарегистрировано ли транспортное средство или лицензировано. Она также используется для электронного сбора платы за проезд по дорогам с оплатой по мере использования и как метод каталогизации движения транспорта, например, дорожными агентствами.

Автоматическое распознавание номерных знаков можно использовать для хранения изображений, снятых камерами, а также текста с номерного знака, с некоторыми настройками для хранения фотографии водителя. В системах обычно используется инфракрасное освещение, позволяющее камере делать снимки в любое время дня и ночи. Технология ALPR должна учитывать различия в номерных знаках от места к месту.

Некоторая часть общества всё же выражает опасения по поводу ALPR, так как массовое отслеживание правительством передвижения граждан и неправильная идентификация не по нраву народу. Также среди проблем выражают высокий уровень ошибок и увеличение государственных расходов. Отдельные критики описывают эту технологию как форму массовой слежки.

Также ALPR (ANPR) известен в других формах, таких как:

* Automatic vehicle identification (AVI)
* Automatisk nummerpladegenkendelse (ANPG)
* Car-plate recognition (CPR)
* License-plate recognition (LPR)
* Lecture automatique de plaques d'immatriculation (LAPI)
* Mobile license-plate reader (MLPR)
* Vehicle license-plate recognition (VLPR)
* Vehicle recognition identification (VRI)

# Анализ предметной области. Постановка задачи

В данной курсовой работе описан процесс разработки системы автоматического распознавания номерных знаков автомобилей в виде мобильного приложения для устройств на ОС Android. Система функционирует при помощи использования принципов машинного обучения и нейронных сетей, запускаемых сразу на мобильном устройстве, которые обрабатывают изображение с камеры устройства и записывают всё в локальную базу данных.

В данном курсовом проекте необходимо разработать мобильное приложение (Android/iOS) для обнаружения и распознавания автомобильных знаков со следующим функционалом:

1. Возможность обнаружения и распознавания автомобильного номера в реальном времени с камеры
2. Возможность распознать автомобильный номер по загруженному изображению с карты памяти устройства
3. Сохранение распознанных номеров в БД
4. Хранение статистики по распознанным номерам (дата и время распознавания, общее количество распознаваний данного номера за все время и т.д.), возможность редактировать и удалять записи

# Разработка и описание алгоритма детекции и распознавания номерных знаков автомобилей

Каждый раз когда задачей стоит что-то определять или читать с картинок или камер с использованием компьютерного зрения, разумней всего использовать для решения таких задач нейронные сети.

Чтение содержимого номерных знаков будет проходить в несколько этапов. Самым первым является обнаружение – сначала автомобиля, а в нём и номерного знака. Для детекции будет использоваться сеть типа CNN (свёрточная нейронная сеть) используемая в архитектуре SSD и построенная при помощи TensorFlow 2.3 Object Detection API.

После обнаружения на изображении автомобиля и номерного знака, необходимо распознать текст, который содержит этот самый знак. Одной из архитектур, с помощью которой можно реализовать распознавание текста, является так называемая CRNN (сверточно-рекуррентная нейронная сеть) . Модель представляет из себя комбинацию сверточной нейронной сети (CNN) для извлечения локальных признаков из изображения и рекуррентной нейронной сети (RNN), представленной двумя слоями двунаправленных LSTM, которая занимается обработкой последовательности. Для данного проекта возьмём готовую сеть из открытых источников GitHub’a.

В ходе разработки программной системы были использованы следующие процедуры, функции и фрагменты:

1. Получение разрешений устройства на камеру и др
2. Активность камеры
3. Активность загрузки изображения
4. Активность сохранённых распознаваний

## 2.1. Получение разрешений устройства на камеру и др.

Входные данные: ввод пользователя.

Выходные данные: продолжение выполения программы.

1. Инициализация окна
2. Открытие окна с запросом разрешений
3. Реакция приложения на результат опроса (выход либо продолжение)

## 2.2. Активность камеры

Входные данные: изображение с камеры.

Выходные данные: изображение с выделенными автомобилями и распознанными номерными знаками.

1. Инициализация камеры
   1. Проверка разрешений
   2. Закрепление Listener’ов и создание потоков для обработки
2. Получение последующего кадра с камеры устройства
3. Отправка кадра в один из свободных потоков (всего 4) обработки сетями
4. Получение ответа в виде обработанного изображения и отображение его в активности вместе с распознанными объектами и текстом

## Активность загрузки изображения

Входные данные: выбранное пользователем изображение.

Выходные данные: изображение с выделенными автомобилями и распознанными номерными знаками.

1. Инициализация окна
2. Ожидание нажатия по кнопке пользователя
3. После нажатия открывается диалог выбора файла изображения
4. Выбранное изображение проходит через функцию обработки сетями
5. Изображение возвращается в активность и отображается с уже выделенными автомобилями и разспознанными номерными знаками

## Активность сохранённых распознаваний

Выходные данные: активность, в которой отображены сохранённые распознавания – автомобили, знаки, время сохранения, дата, местоположение и оставленное сообщение.

1. Инициализация окна
   1. Чтение списка распознаваний из локальной БД (если нет то создаётся)
   2. Создание элементов распознаваний

# Разработка программы

Среда разработки: Android Studio Electric Eel 2022.1.1.

Задание выполнялось на языке программирования Kotlin.

Операционная система: Windows 10 x64 Home.

Для выполнения задачи потребовалось подключение tensorflow 2.3.

TensorFlow - программная библиотека для машинного обучения, предназначенная для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия.

Разработка модуля:

**Получение разрешений устройства на камеру и др.**

Назначение: получение доступа к камере устройства, геолокации, сети интернет и внутреннему хранилищу устройства;

Входные данные: ввод пользователя.

Выходные данные: продолжение выполения программы.

class PermissionsFragment(finished: () -> Unit = {}) : AppFragment() {

companion object {

fun hasPermissions(context: Context) = PERMISSIONS\_REQUIRED.all {

ContextCompat.checkSelfPermission(context, it) == PackageManager.PERMISSION\_GRANTED

}

}

val actionFinished = finished

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

if (!hasPermissions(requireContext())) {

// Request permissions

requestPermissions(PERMISSIONS\_REQUIRED, PERMISSIONS\_REQUEST\_CODE)

}

else {

removeThisFragment()

}

}

override fun appearingAnimations(){}

override fun subscribeToViewModel(){}

override fun subscribeListeners(){}

override fun unsubscribe(){}

override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<String>, grantResults: IntArray) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults)

if (requestCode == PERMISSIONS\_REQUEST\_CODE) {

if(grantResults.isEmpty()){

exitProcess(0)

}

if (grantResults.contains(PackageManager.PERMISSION\_DENIED)) {

exitProcess(0)

}

else {

//granted

}

removeThisFragment()

}

}

private fun removeThisFragment(){

requireActivity().supportFragmentManager.beginTransaction().remove(this).commit()

actionFinished()

}

}

**Активность камеры**

Назначение: главная активность приложения;

Входные данные: изображение с камеры.

Выходные данные: изображение с выделенными автомобилями и распознанными номерными знаками.

class CameraActivity : AppActivity() {

override lateinit var viewModel: CameraViewModel

private lateinit var container: FrameLayout

lateinit var deviceOrientationListener: OrientationEventListener

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_camera)

container = findViewById(R.id.camera\_container)

val drawer = findViewById<DrawerLayout>(R.id.cameraActivityDrawerLayout)

val navigation = findViewById<NavigationView>(R.id.camera\_navigation)

initUi(drawer, navigation)

viewModel = ViewModelProvider(this).get(CameraViewModel::class.java)

showCameraFragment()

deviceOrientationListener = object : OrientationEventListener(this,

SensorManager.SENSOR\_DELAY\_NORMAL) {

override fun onOrientationChanged(orientation: Int) {

CameraViewModel.deviceOrientation = orientation

}

}

}

override fun subscribeToViewModel() {

val settings = PreferenceManager.getDefaultSharedPreferences(applicationContext)

val settingsNumRecognitionsKey = getString(R.string.SETTINGS\_NUM\_RECOGNITIONS)

val settingsMinimumPredictionCertaintyKey = getString(R.string.SETTINGS\_MINIMUM\_PREDICTION\_CERTAINTY)

val settingsShowReceptiveFieldKey = getString(R.string.SETTINGS\_SHOW\_RECEPTIVE\_FIELD)

val numRecognitionsToShow = settings.getInt(settingsNumRecognitionsKey, resources.getInteger(R.integer.settings\_num\_recognitions\_default))

val minimumPredictionCertaintyToShow = settings.getInt(settingsMinimumPredictionCertaintyKey, resources.getInteger(R.integer.settings\_minimum\_prediction\_certainty\_default))

val settingsShowReceptiveField = settings.getBoolean(settingsShowReceptiveFieldKey, resources.getBoolean(R.bool.settings\_receptive\_field\_default))

CameraViewModel.numRecognitionsToShow = numRecognitionsToShow

CameraViewModel.minimumPredictionCertaintyToShow = minimumPredictionCertaintyToShow.toFloat()

CameraViewModel.settingsShowReceptiveField = settingsShowReceptiveField

}

override fun subscribeListeners() {

deviceOrientationListener.enable()

container.postDelayed({

container.systemUiVisibility = (SYSTEM\_UI\_FLAG\_LAYOUT\_FULLSCREEN and SYSTEM\_UI\_FLAG\_LAYOUT\_STABLE)

appearingAnimations()

}, ApplicationRoot.IMMERSIVE\_FLAG\_TIMEOUT)

}

override fun unsubscribe() {

deviceOrientationListener.disable()

}

private fun showCameraFragment(){

supportFragmentManager

.beginTransaction()

.replace(R.id.camera\_container, CameraFragment())

.commit()

}

override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean {

return when (keyCode){

KeyEvent.KEYCODE\_VOLUME\_DOWN -> {

val intent = Intent(CameraViewModel.KEY\_EVENT\_ACTION).apply {

putExtra(CameraViewModel.KEY\_EVENT\_EXTRA, keyCode)

}

LocalBroadcastManager.getInstance(this).sendBroadcast(intent)

true

}

else -> {

super.onKeyDown(keyCode, event)

}

}

}

override fun appearingAnimations() {

camera\_switch\_button?.overshootAppearingAnimation(this)

camera\_capture\_button?.overshootAppearingAnimation(this)

}

}

**Активность загрузки изображения**

Назначение: распознавание знаков и детекция автомобилей на изображениях с внутреннего хранилища устройства;

Входные данные: выбранное пользователем изображение.

Выходные данные: изображение с выделенными автомобилями и распознанными номерными знаками.

class LoadActivity : AppActivity() {

override lateinit var viewModel: LoadViewModel

private lateinit var container: ConstraintLayout

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_load)

container = findViewById(R.id.loaded\_image\_container)

val drawer = findViewById<DrawerLayout>(R.id.loadActivityDrawerLayout)

val navigation = findViewById<NavigationView>(R.id.load\_navigation)

viewModel = ViewModelProvider(this).get(LoadViewModel::class.java)

initUi(drawer, navigation)

ivLoadedImage.clipToOutline = true

Glide

.with(this)

.load(R.drawable.icon\_photo\_library)

.into(ivLoadedImage)

extendedFabLoadHelp.setOnClickListener {

loadImage()

}

extendedFabLoadHelp.text = getString(R.string.load\_an\_image)

extendedFabLoadHelp.icon = ContextCompat.getDrawable(this, R.drawable.icon\_photo\_library)

extendedFabLoadHelp.iconTint = ContextCompat.getColorStateList(this, R.color.selector\_ic)

extendedFabLoadHelp.backgroundTintList = ContextCompat.getColorStateList(this, R.color.selector\_fab\_normal\_color)

val myUri = intent.getParcelableExtra<Uri>("fromCameraUri")

if (myUri != null) {

viewModel.loadImage(myUri){isSuccess ->

if(!isSuccess){

showErrorSnackBar(getString(R.string.image\_load\_failed))

}

}

}

}

override fun onResume() {

super.onResume()

val metrics = DisplayMetrics()

windowManager.defaultDisplay.getMetrics(metrics)

viewModel.setScreenProperties(metrics.widthPixels, metrics.heightPixels)

container.postDelayed({

container.systemUiVisibility = (View.SYSTEM\_UI\_FLAG\_LAYOUT\_FULLSCREEN and View.SYSTEM\_UI\_FLAG\_LAYOUT\_STABLE)

}, ApplicationRoot.IMMERSIVE\_FLAG\_TIMEOUT)

val settings = PreferenceManager.getDefaultSharedPreferences(applicationContext)

val settingsNumRecognitionsKey = getString(R.string.SETTINGS\_NUM\_RECOGNITIONS)

val settingsMinimumPredictionCertaintyKey = getString(R.string.SETTINGS\_MINIMUM\_PREDICTION\_CERTAINTY)

val numRecognitionsToShow = settings.getInt(settingsNumRecognitionsKey, resources.getInteger(R.integer.settings\_num\_recognitions\_default))

val minimumPredictionCertaintyToShow = settings.getInt(settingsMinimumPredictionCertaintyKey, resources.getInteger(R.integer.settings\_minimum\_prediction\_certainty\_default))

LoadViewModel.numRecognitionsToShow = numRecognitionsToShow

LoadViewModel.minimumPredictionCertaintyToShow = minimumPredictionCertaintyToShow.toFloat()

}

override fun subscribeToViewModel() {

val imageObserver = Observer<Bitmap> { newImage ->

ivLoadedImage.disappearingAnimation(this)

Glide

.with(this)

.load(newImage)

.centerCrop()

.error(R.drawable.icon\_photo\_library)

.placeholder(R.drawable.icon\_photo\_library)

.into(ivLoadedImage)

ivLoadedImage.appearingAnimation(this)

}

val boundingBoxImageObserver = Observer<Bitmap> { newImage ->

Glide

.with(this)

.load(newImage)

.centerCrop()

.into(ivLoadedImageBoundingBoxes)

}

val recognitionsObserver = Observer<Array<UserRecognition>> { recognitions ->

if(recognitions.isNotEmpty()){

alertLoadedButton.setOnClickListener {

viewModel.setAlertActivityParams()

val intent = Intent(this, AlertActivity::class.java)

startActivity(intent)

}

extendedFabLoadHelp.setOnClickListener {

viewModel.setAlertActivityParams()

val intent = Intent(this, AlertActivity::class.java)

startActivity(intent)

}

if(alertLoadedButton.visibility == View.GONE){

alertLoadedButton.appearingAnimation(this)

}

extendedFabLoadHelp.text = getString(R.string.view\_alert)

extendedFabLoadHelp.icon = ContextCompat.getDrawable(this, android.R.drawable.ic\_dialog\_alert)

extendedFabLoadHelp.iconTint = ContextCompat.getColorStateList(this, R.color.selector\_ic)

extendedFabLoadHelp.backgroundTintList = ContextCompat.getColorStateList(this, R.color.selector\_fab\_alert\_color)

}

else{

if(alertLoadedButton.visibility == View.VISIBLE){

alertLoadedButton.disappearingAnimation(this)

}

if(viewModel.loadedImage.value != null){

extendedFabLoadHelp.setOnClickListener {}

extendedFabLoadHelp.text = getString(R.string.inspected)

extendedFabLoadHelp.icon = ContextCompat.getDrawable(this, R.drawable.icon\_done)

}

else{

extendedFabLoadHelp.setOnClickListener {

loadImage()

}

extendedFabLoadHelp.text = getString(R.string.load\_an\_image)

extendedFabLoadHelp.icon = ContextCompat.getDrawable(this, R.drawable.icon\_photo\_library)

}

extendedFabLoadHelp.iconTint = ContextCompat.getColorStateList(this, R.color.selector\_ic)

extendedFabLoadHelp.backgroundTintList = ContextCompat.getColorStateList(this, R.color.selector\_fab\_normal\_color)

}

}

viewModel.loadedImage.observe(this, imageObserver)

viewModel.boundingBoxImage.observe(this, boundingBoxImageObserver)

viewModel.recognitions.observe(this, recognitionsObserver)

}

override fun subscribeListeners() {

ivLoadedImage.setOnClickListener {

if(viewModel.loadedImage.value == null){

loadImage()

}

}

load\_image\_button.setOnClickListener {

loadImage()

}

loaded\_image\_rotate\_button.setOnClickListener {

viewModel.rotateImage()

}

}

override fun unsubscribe() {}

private fun loadImage(){

val intent = Intent(Intent.ACTION\_PICK)

intent.type = "image/\*"

intent.putExtra(Intent.EXTRA\_MIME\_TYPES, viewModel.imageMimeTypes)

startActivityForResult(intent, LoadViewModel.GALLERY\_REQUEST\_CODE)

}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {

super.onActivityResult(requestCode, resultCode, data)

if (resultCode == RESULT\_OK) {

when (requestCode) {

LoadViewModel.GALLERY\_REQUEST\_CODE -> {

val selectedImageUri = data?.data ?: return

viewModel.loadImage(selectedImageUri){isSuccess ->

if(!isSuccess){

showErrorSnackBar(getString(R.string.image\_load\_failed))

}

}

}

}

}

}

override fun appearingAnimations(){

load\_image\_button.overshootAppearingAnimation(this)

loaded\_image\_rotate\_button.overshootAppearingAnimation(this)

extendedFabLoadHelp.overshootAppearingAnimation(this)

}

}

**Активность сохранённых распознаваний**

Назначение: отображение избранных распознаваний в виде удобно читаемого списка;

Выходные данные: активность, в которой отображены сохранённые распознавания – автомобили, знаки, время сохранения, дата, местоположение и оставленное сообщение.

class RecognitionActivity : MasterDetailActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

viewModel = ViewModelProvider(this).get(RecognitionViewModel::class.java)

(viewModel as RecognitionViewModel).updateDataFromDb()

super.onCreate(savedInstanceState)

listName = getString(R.string.user\_recognition\_list)

detailName = getString(R.string.user\_recognition\_details)

sendSucceed = getString(R.string.recognition\_sent)

sendFailed = getString(R.string.recognition\_sending\_failed)

deleted = getString(R.string.deleted)

deleteFailed = getString(R.string.delete\_failed)

alreadySent = getString(R.string.recognition\_already\_sent)

updateSucceed = getString(R.string.updated)

updateFailed = getString(R.string.update\_failed)

}

override fun onResume() {

super.onResume()

(viewModel as RecognitionViewModel).updateDataFromDb()

}

}

# 4. Тестирование

## 4.1. Подготовка к тестированию

Минимальные требования для Android Studio: Процессор x64 или x32 структуры с минимальной тактовой частотой 1.6 гигагерц или быстрее, а также установленное ОЗУ с объёмом памяти не менее 2 гигабайт.

Android Studio позволяет проводить отладку приложения через внутренний эмулятор мобильного устройства, однако учитывая отсутствие возможности использования камеры и относительно медленной скорости работы, было решено проводить отладку и тестирование на настоящем устройстве под ОС Android – Xiaomi Mi 9T 6GB 2.2MHz 8 Core.

Сборка для тестирования проводилась на 64-битной операционной системе Windows 10 Домашняя версии 21H2. Аппаратное обеспечение устройства, на котором происходило тестирование:

* Процессор: Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 GHz
* Оперативная память: 8,00 ГБ (доступно: 7,78 ГБ)
* Тип системы: 64-разрядная операционная система Windows 10 Домашняя версии 21H2, процессор x64



## Тестирование

Программа запускается путём установки на мобильное устройство и последующего нажатия на иконку запуска. При запуске первоначально пользователя встречает окно запроса разрешений на съёмку видео и доступ к внутреннему хранилищу, а также местоположению. Для продолжения работы необходимо предоставить необходимые разрешения. После предоставления разрешений перед пользователем открывается главная активность – активность камеры, на которой будут видны кнопки управления и изображение с наложенными поверх цветными квадратами – ими выделяются автомобили и номерные знаки на них.

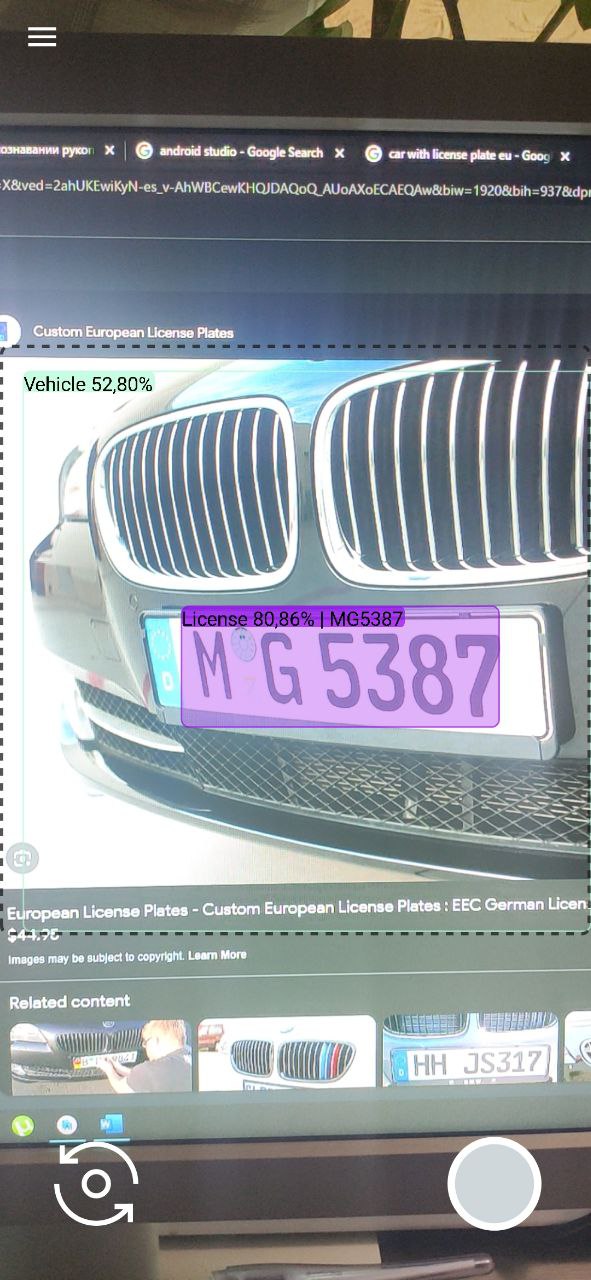
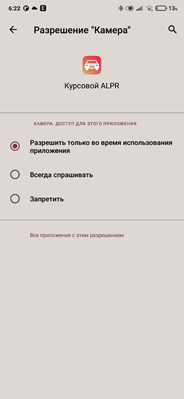


Рисунок 4.2.1 – Меню запроса разрешений (слева) и главная активность приложения (справа). В главной активности видно выделение автомобиля и номерного знака, вместе с уверенностью сети в этом.

На главной активности можно нажать кнопку открытия меню сверху слева. Открывается боковое меню (его так же можно вызвать смахиванием вправо).

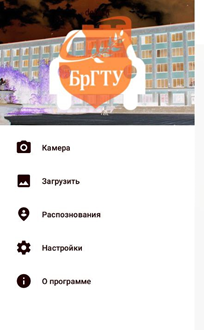


Рисунок 4.2.2 – Боковое меню.

При выборе пункта меню Загрузить открывается активность распознавания по картинке, загруженной с галереи устройства. Для этого пользователь нажимает кнопку галереи и выбирает необходимое фото. После этого происходит обработка изображения сетями.



Рисунок 4.2.3 – Видна кнопка галереи слева снизу и кнопка поворота изображения снизу справа. С каждым поворотом детекция происходит заново.

Также в приложении можно просмотреть сохранённые распознавания, которые загружаются из базы данных, создаваемой на основе действий пользователя. Для этого необходимо открыть пункт меню Распознавания. Пользователя встретит список состоящий из предыдущих распознаваний, в которых определены номера, дата, место определения а также показана картинка автомобиля. Рядом стоит подпись которую можно изменить после открытия. Подробности об элементе видны после нажатия.

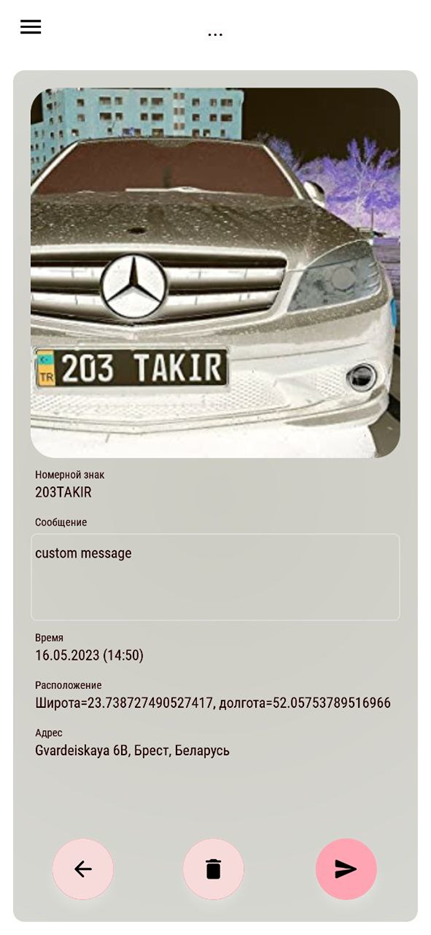


Рисунок 4.2.4 – Список распознанных номерных знаков и информация о них (слева) и подробности об элементе списка (справа).

Также присутствует активность настроек. Можно изменить количество распознаваний максимально отображаемых на одном кадре, необходимый порог уверенности сети в объекте для отображения, переключатель удержания яркости экрана от потухания и др. Настройки применяются сразу же после применения.



Рисунок 4.2.5 – Активность настроек.

# Заключение

Сейчас большую роль в развитии ПО в области общественной безопасности играют технологии computer vision. Основными задачами такого ПО являются распознавание и слежение за объектами, последняя из которых представляется более сложной, т.к. условия съемки часто бывают неблагоприятными.

В данном курсовом проекте мною были получены базовые теоретические и практические знания применения алгоритмов детекции и распознавания различных объектов и текста на изображениях, применены знания по детекционным нейронным сетям.

В результате выполнения курсового проекта были выполнены все необходимые условия, проведено тестирование программного продукта.

# Список использованной литературы

1. ГОСТ 19.103-77. ЕСПД. Обозначения программ и программных документов.
2. ГОСТ 19.504-79. Единая система программной документации ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
3. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.
4. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов.
5. ГОСТ 19.102-77. ЕСПД. Стадии разработки.
6. ГОСТ 2.105-95. Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам.
7. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.
8. ГОСТ 19.402-78. ЕСПД. Описание программы.
9. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.
10. ГОСТ 19.005-85. ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.