|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
|  |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |

|  |
| --- |
| **КУРСОВОЙ ПРОЕКТ** |
|  |
| по междисциплинарному курсу: МДК.03.01. Технология разработки программного обеспечения. |
|  |
| на тему: Разработка прототипа программы «Формирование векторной карты на основе спутниковых снимков». |
|  |
| студента группы ПКС-405  специальности 09.02.03 Программирование в компьютерных системах |
| Масленникова Михаила Владимировича |

|  |  |  |
| --- | --- | --- |
| Студент |  | М.В. Масленников |
| Руководитель курсового проекта |  | И.А. Бикжанов |
| Председатель ПЦК специальности 09.02.03 Программирование в компьютерных системах |  | А.И. Глускер |

|  |  |  |
| --- | --- | --- |
| Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2021 г. | | |
| Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Заведующий отделением № 3 |  | И.А. Миланова |

Москва

2021

Оглавление

[Введение 3](#_Toc94727610)

[1 Общая часть 6](#_Toc94727611)

[1.1 Назначение и цели создания системы 6](#_Toc94727612)

[1.2 Обзор и анализ предметной области 6](#_Toc94727613)

[1.3 Выбор и характеристика среды разработки прототипа приложения 6](#_Toc94727614)

[2.1 Постановка задачи 7](#_Toc94727615)

[2.2 Архитектура информационной системы 7](#_Toc94727616)

[Приемущества: 8](#_Toc94727617)

[2.3 Логическая модель изображений в папке images 9](#_Toc94727618)

[2.4 Разработка прототипа программы 10](#_Toc94727619)

[2.5.1 Диаграмма вариантов использования 10](#_Toc94727620)

[2.5.2 Решение главной задачи проекта 11](#_Toc94727621)

[2.6.1 Требования к техническим средствам 26](#_Toc94727622)

[2.6.2 Требования к программным средствам 26](#_Toc94727623)

[Заключение 27](#_Toc94727624)

[Список источников 28](#_Toc94727625)

# **Введение**

Курсовой проект был разработке прототипа программы «Формирование векторных карт на основе спутниковых снимков». Основная цель проекта – создание прототипа программы, который позволит создать векторную карту, на базе спутниковых снимков, также прилагается уже готовый DataSet.

Обзор существующих прототипов приложений в рассматриваемой предметной области показал, что универсальных решений данной задачи на рынке не имеется и под каждую организацию разрабатываются прототипы приложений, соответсвующие требуемым критериям компании.

Анализ создания векторных карт из спутниковых снимков показывает, что это трудоемкий и времязатратный процесс, к чему и было приведено решение, создать прототип нейронной сети, для выполнения этой задачи.

Задачи проекта:

1. Анализ предметной области,
2. Поиск наиболее подходящего для выполнения этой задачи Dataset’а,
3. Разработка прототипа программы всеми возможными средствами,

Структура пояснительной записки состоит из этапов разработки программного продукта, оформленных по ГОСТам и объединенных в одном документе.

Составление пояснительной записки:

1. Титульный лист,
2. Содержание,
3. Введение,
4. Основная часть,
5. Заключение – описание того, что было сделано,
6. Список источников,
7. Приложение.

Содержние включает в себя названия разделов и номера страниц этих

разделов.

Введение описывает общую суть курсового проекта, цель курсового проекта, и задачи, которые он ставит.

В основную часть входят раздел спецификации, описывающий формирование структуры программного продукта, раздел реализации программного изделия на языке программирования, описывающего создание работающей программы на выбранном языке программирования, с тестированием и отладкой и раздел тестирования программного продукта, выявляющего степени соответствия готового программного продукта со спецификацией, разработанной на первом этапе.

В прототипах приложений находится исходный код программы.

Основные методы, которые были использованы:

1. анализ,
2. синтез,
3. моделирование,
4. императивный метод.

Частные методы исследования:

1. документирование,
2. метод объектно-ориентированного программирования,
3. формализация.

Основание структуры и содержание пояснительной записки:

Содержание пояснительной записки основывается на модели технического задания реальной модели разработки.

Возможности прототипа программы показать формирование векторных карт на основе спутниковых снимков.

# **1 Общая часть**

## **Назначение и цели создания системы**

Основной целью курсового проекта – создание прототипа программы,

Который позволит создать векторную карту.

## **Обзор и анализ предметной области**

DataSet был найдет на сайте

https://www.kaggle.com/arka47/satellitegooglemapsmasks

А также работа с Dataset’ми была изучена самостоятельна, а также эффективное использование этих данных. После анализа различных частичных продуктов, были выделены следующие особенности:

1. В DataSet можно вносить особенности пользователю напрямую для быстрого устранения проблем, для быстрого решения.
2. Пользователь сам может запускать прототип и к нему прилагающие файлы.
3. Пользователю прилагается код, для дальнейших изменений прототипа для своего пользования.

## **1.3 Выбор и характеристика среды разработки прототипа приложения**

Для разработки прототипа программы, было использовано Microsoft

Visual Studio Code, языки программирования Python, также библиотеки os – для работы с директориями в Windows, numpy – для математических рассчетов, pandas – для работы с данными, matplotlib, для вывода графиков и фотографий, cv2 – для разрисовки масок и чтения фотографий, а также FrameWork под названием tensorflow, который поддерживает также keras.

#### Специальная часть

## **Постановка задачи**

Найти наиболее подходящий DataSet в сети интернет или создать

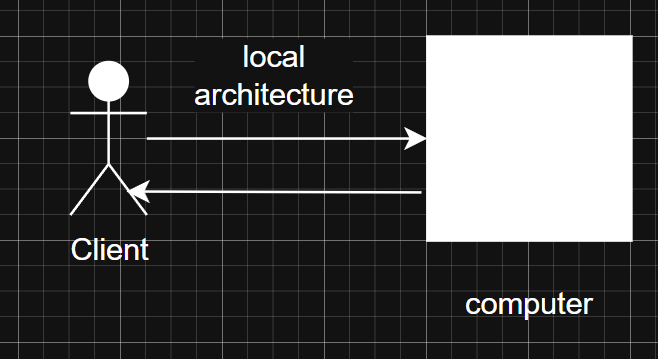
его самому и создать прототип программы средствами Python, позволяющий:

1. Добавление изображений,
2. Обучение нейронной сети,
3. Формирование новых изображений с использованием маски.

## **Архитектура информационной системы**

Архитектура прототипа программы является локальная

Рисунок 1-локальная архитектура



## **Приемущества:**

1. Выполнение всей работы не требует никаких иных устройств.
2. Основная часть данных хранится на одном устройстве и потеря их невозможна
3. Кроссплатформенность, данный прототип позволяет запуск на любой операционной системе.
4. Нет никакой нагрузки на сеть, так как идет работа напрямую с памятью вашего компьютера.

**Недостатки:**

1. При потере данных на вашем персональном комьютере, будет необходимость скачивать все необходимое ПО снова, а также все данные придется восстанавливать через git clone
2. При необходимости покупки сервера для решения более крупных задач это будет очень дорого.

## **Логическая модель изображений в папке images**

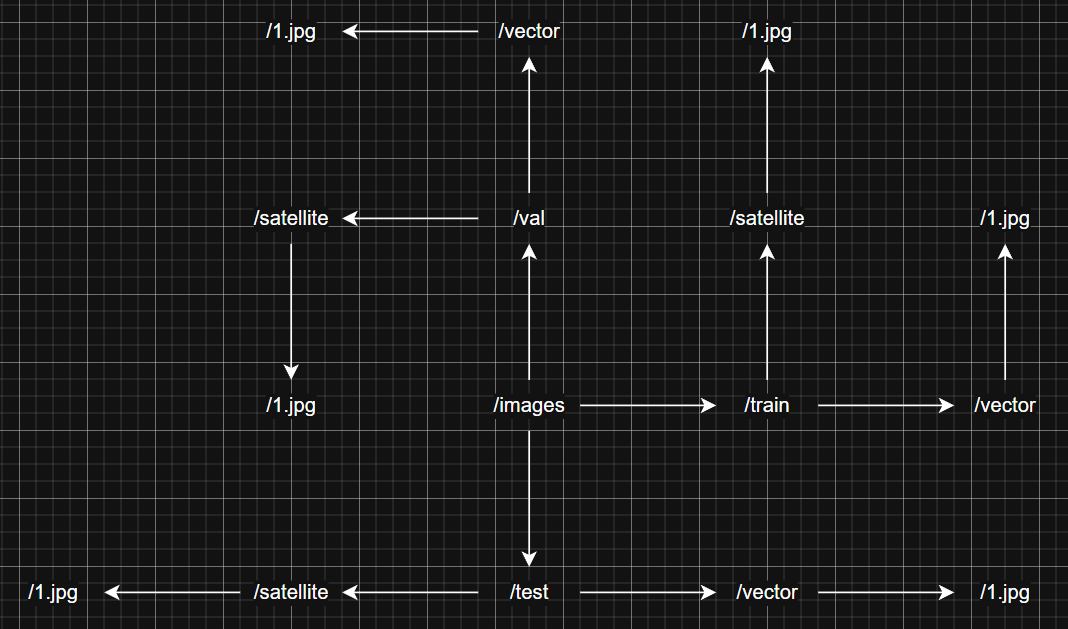


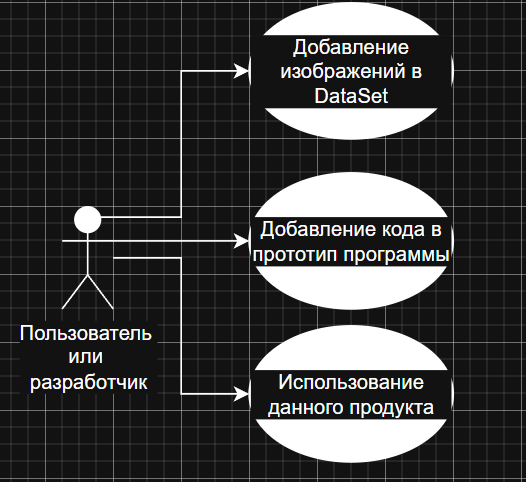
Рисунок 2 – логическая модель изображений в папке images

## **Разработка прототипа программы**

Данный прототип разрабатывался на долгосрочную перспективу,

он имеет огромнный потенциал для дальнейшей разработки и улучшения. В данный момент прототип программы представляет весь необходимый список требований. Первым этапом был поиск или создание своего DataSet’а, вторым этапом было создание скрипта для рассинхронности этого DataSet’а, на 3 различные папки, для обучение, для тестирования и для проверки, третьим этапом было создание нейронной сети для формирования из спутниновых снимков в векторные и соотвественно четвертым, написание документации.

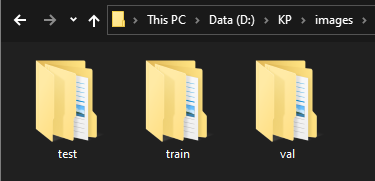
## **Диаграмма вариантов использования**



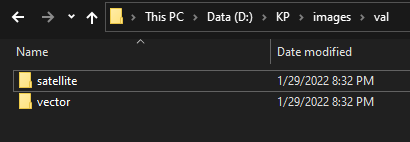
## **Решение главной задачи проекта**

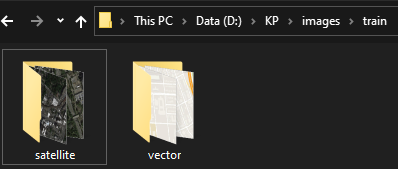
Согласно с главной задачей проекта были реализованны функции:

1. Поиск готового DataSet’а или создание собственного

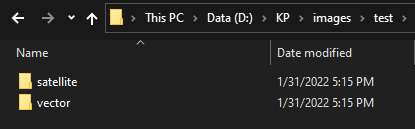


Images/val



Images/train  
 

Images/test

  
В подпапке train и val по 1033 изображений vector и satellite

В подпапке test 15% от папки train

1. Скрипт для рассределения по папкам эти данные

import shutil

import os

import pathlib

# Каталог с набором данных спутниковых снимков

data\_dir = r'D:\KP\images\train'

# Каталог с данными для обучения

train\_dir = r'D:\KP\images\train'

# Каталог с данными для проверки

val\_dir = r'D:\KP\images\val'

# Каталог с данными для тестирования

test\_dir = r'D:\KP\images\test'

# Часть набора данных для тестирования

test\_data\_portion = 0.15

# Часть набора данных для проверки

val\_data\_portion = 0.15

# Количество элементов данных в одном классе

nb\_images = 1033

# Функция создания каталога с двумя подкаталогами по названию классов: satellite и vector

def create\_directory(dir\_name):

if os.path.exists(dir\_name):

shutil.rmtree(dir\_name)

os.makedirs(dir\_name)

os.makedirs(os.path.join(dir\_name, "satellite"))

os.makedirs(os.path.join(dir\_name, "vector"))

# Создание структуры каталогов для обучающего, проверочного и тестового набора данных

create\_directory(train\_dir)

create\_directory(val\_dir)

create\_directory(test\_dir)

# Функция копирования изображений в заданный каталог. Изображения спутниковых карт и вектроных карт копируются в отдельные подкаталоги

def copy\_image(start\_index, end\_index,source\_dir, dest\_dir):

images = 1

try:

for i in range(start\_index, end\_index):

shutil.copy2(pathlib.Path(source\_dir)/"satellite"/(str(i+images) + ".jpg"),

pathlib.Path(dest\_dir)/"satellite")

except shutil.SameFileError:

pass

try:

for i in range(start\_index, end\_index):

shutil.copy2(pathlib.Path(source\_dir)/"vector"/(str(i+images) + ".jpg"),

pathlib.Path(dest\_dir)/"vector")

except shutil.SameFileError:

pass

# Расчет индексов наборов данных для обучения, приверки и тестирования

start\_val\_data\_idx = int(nb\_images \* (1 - val\_data\_portion - test\_data\_portion))

start\_test\_data\_idx = int(nb\_images \* (1 - test\_data\_portion))

print(start\_val\_data\_idx)

print(start\_test\_data\_idx)

# Копирование изображений

copy\_image(1, start\_val\_data\_idx, data\_dir, train\_dir)

copy\_image(start\_val\_data\_idx, start\_test\_data\_idx, data\_dir, val\_dir)

copy\_image(start\_test\_data\_idx, nb\_images, data\_dir, test\_dir)

1. Создание прототипа программы для решения этой задачи

# Подключение библиотек

import os

import numpy as np

import pandas as pd

import re

import matplotlib.pyplot as plt

import cv2

import tensorflow as tf

# определние рамки для храния масок и изображений

framObjTrain = {'img' : [],

'mask' : []

}

# определение функции загрузки данных

def LoadData( frameObj = None, imgPath = None, maskPath = None, shape = 128):

imgNames = os.listdir(imgPath)

names = []

maskNames = []

unames = []

for i in range(len(imgNames)):

unames.append(imgNames[i].split('\_')[0])

unames = list(set(unames))

for i in range(len(unames)):

names.append(unames[i])

maskNames.append(unames[i])

imgAddr = imgPath + '/'

maskAddr = maskPath + '/'

for i in range (len(names)):

img = plt.imread(imgAddr + names[i])

mask = plt.imread(maskAddr + maskNames[i])

img = cv2.resize(img, (shape, shape))

mask = cv2.resize(mask, (shape, shape))

frameObj['img'].append(img)

frameObj['mask'].append(mask)

return frameObj

framObjTrain = LoadData( framObjTrain, imgPath = 'images/train/satellite'

, maskPath = 'images/train/vector'

, shape = 128)

# Отображение данных, загруженных нашей функцией

plt.figure(figsize = (7, 5))

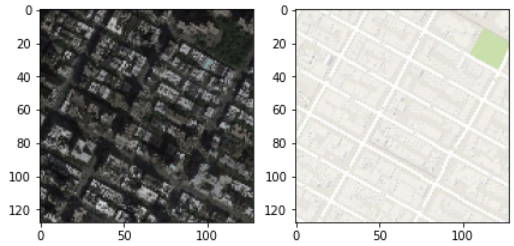
plt.subplot(1,2,1)

plt.imshow(framObjTrain['img'][1])

plt.subplot(1,2,2)

plt.imshow(framObjTrain['mask'][1])

plt.show()



# выполним фактически 2 свертки

def Conv2dBlock(inputTensor, numFilters, kernelSize = 3, doBatchNorm = True):

# Первая свертка

x = tf.keras.layers.Conv2D(filters = numFilters, kernel\_size = (kernelSize, kernelSize),

kernel\_initializer = 'he\_normal', padding = 'same') (inputTensor)

if doBatchNorm:

x = tf.keras.layers.BatchNormalization()(x)

x =tf.keras.layers.Activation('relu')(x)

# Вторая свертка

x = tf.keras.layers.Conv2D(filters = numFilters, kernel\_size = (kernelSize, kernelSize),

kernel\_initializer = 'he\_normal', padding = 'same') (x)

if doBatchNorm:

x = tf.keras.layers.BatchNormalization()(x)

x = tf.keras.layers.Activation('relu')(x)

return x

# Определим Unet - модель

def GiveMeUnet(inputImage, numFilters = 16, droupouts = 0.1, doBatchNorm = True):

# Определим пути кодирования

c1 = Conv2dBlock(inputImage, numFilters \* 1, kernelSize = 3, doBatchNorm = doBatchNorm)

p1 = tf.keras.layers.MaxPooling2D((2,2))(c1)

p1 = tf.keras.layers.Dropout(droupouts)(p1)

c2 = Conv2dBlock(p1, numFilters \* 2, kernelSize = 3, doBatchNorm = doBatchNorm)

p2 = tf.keras.layers.MaxPooling2D((2,2))(c2)

p2 = tf.keras.layers.Dropout(droupouts)(p2)

c3 = Conv2dBlock(p2, numFilters \* 4, kernelSize = 3, doBatchNorm = doBatchNorm)

p3 = tf.keras.layers.MaxPooling2D((2,2))(c3)

p3 = tf.keras.layers.Dropout(droupouts)(p3)

c4 = Conv2dBlock(p3, numFilters \* 8, kernelSize = 3, doBatchNorm = doBatchNorm)

p4 = tf.keras.layers.MaxPooling2D((2,2))(c4)

p4 = tf.keras.layers.Dropout(droupouts)(p4)

c5 = Conv2dBlock(p4, numFilters \* 16, kernelSize = 3, doBatchNorm = doBatchNorm)

# Определим пути декодирования

u6 = tf.keras.layers.Conv2DTranspose(numFilters\*8, (3, 3), strides = (2, 2), padding = 'same')(c5)

u6 = tf.keras.layers.concatenate([u6, c4])

u6 = tf.keras.layers.Dropout(droupouts)(u6)

c6 = Conv2dBlock(u6, numFilters \* 8, kernelSize = 3, doBatchNorm = doBatchNorm)

u7 = tf.keras.layers.Conv2DTranspose(numFilters\*4, (3, 3), strides = (2, 2), padding = 'same')(c6)

u7 = tf.keras.layers.concatenate([u7, c3])

u7 = tf.keras.layers.Dropout(droupouts)(u7)

c7 = Conv2dBlock(u7, numFilters \* 4, kernelSize = 3, doBatchNorm = doBatchNorm)

u8 = tf.keras.layers.Conv2DTranspose(numFilters\*2, (3, 3), strides = (2, 2), padding = 'same')(c7)

u8 = tf.keras.layers.concatenate([u8, c2])

u8 = tf.keras.layers.Dropout(droupouts)(u8)

c8 = Conv2dBlock(u8, numFilters \* 2, kernelSize = 3, doBatchNorm = doBatchNorm)

u9 = tf.keras.layers.Conv2DTranspose(numFilters\*1, (3, 3), strides = (2, 2), padding = 'same')(c8)

u9 = tf.keras.layers.concatenate([u9, c1])

u9 = tf.keras.layers.Dropout(droupouts)(u9)

c9 = Conv2dBlock(u9, numFilters \* 1, kernelSize = 3, doBatchNorm = doBatchNorm)

output = tf.keras.layers.Conv2D(3, (1, 1), activation = 'sigmoid')(c9)

model = tf.keras.Model(inputs = [inputImage], outputs = [output])

return model

# Создание экземпляра модели

inputs = tf.keras.layers.Input((128, 128, 3))

myTransformer = GiveMeUnet(inputs, droupouts= 0.07)

myTransformer.compile(optimizer = 'Adam', loss = 'binary\_crossentropy', metrics = ['accuracy'] )

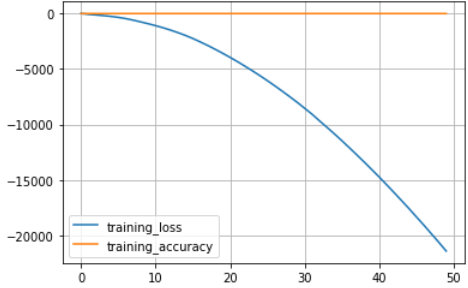
retVal = myTransformer.fit(np.array(framObjTrain['img']), np.array(framObjTrain['mask']), epochs = 50, verbose = 0)

plt.plot(retVal.history['loss'], label = 'training\_loss')

plt.plot(retVal.history['accuracy'], label = 'training\_accuracy')

plt.legend()

plt.grid(True)



# функция для получения 16 прогнозов

def predict16(valMap, model, shape=256):

# получение и обработка данных val

img = valMap['img']

mask = valMap['mask']

mask = mask[0:16]

imgProc = img[0:16]

imgProc = np.array(img)

predictions = model.predict(imgProc)

for i in range(len(predictions)):

predictions[i] = cv2.merge((predictions[i,:,:,0], predictions[i,:,:,1],predictions[i,:,:,2]))

return predictions, imgProc, mask

def Plotter(img, predMask, groudTruth):

plt.figure(figsize=(20,10))

plt.subplot(1,3,1)

plt.imshow(img)

plt.title('Изображение территории')

# добавление фильтра вектора на изображение здесь

filter = np.array([[-1,-1,-1], [-1,-1,-1]])

imgSharpen = cv2.filter2D(predMask, -1, filter)

plt.subplot(1,3,3)

plt.imshow(groudTruth)

plt.title('Актуальная маска местности')

sixteenPrediction, actuals, masks = predict16(framObjTrain, myTransformer)

Plotter(actuals[1], sixteenPrediction[1], masks[1])



Plotter(actuals[2], sixteenPrediction[2], masks[2])



Plotter(actuals[3], sixteenPrediction[3], masks[3])



Plotter(actuals[4], sixteenPrediction[4], masks[4])



Plotter(actuals[5], sixteenPrediction[5], masks[5])



Plotter(actuals[6], sixteenPrediction[6], masks[6])



Plotter(actuals[7], sixteenPrediction[7], masks[7])



Plotter(actuals[8], sixteenPrediction[8], masks[8])



myTransformer.save('KP.h5')

## **2.6.1 Требования к техническим средствам**

состав технических средств должен входить IBM-совместимый компьютер, включающий:

– процессор AMD FX 6100 или совместимый с ним,

– клавиатуру,

– мышь,

– видеокарту,

– монитор,

– жесткий диск,

– оперативная память размеров от 8 Гб.

## **2.6.2 Требования к программным средствам**

В состав программных средств должно входить:

– Операционная система Windows 8.1 или новее, также может подойти любой Linux.

– visual studio code, requirements.txt, python 3.8.8.

# **Заключение**

Курсовой проект посвящен прототипу программы «Формирование векторных карт на основе спутниковых снимков», которое помогло усвоить новое. Итог данной работы, есть готовый продукт. По мнению автора проект выполнен в приличном, с точки зрения профессионализма, уровне на текущем этапе развития студента, что спокойно может дать возможность без особых трудностей продолжить улучшать данный продукт, каждая часть кода, которая могла быть вынесена в отдельный модуль, была отделена для удобства, большая часть была написана следуя документации Microsoft, это во многом позволило привести код в правильный и легко читаемый вид, что в конечном счете даст возможность другому разработчику без особых усилий, следуя документации разобраться как устроена отдельная часть.

В процессе выполнения курсового проекта:

1. Изучена предметная область
2. Найден подходящий DataSet
3. Разработан прототип программы «Формирование векторных карт на основе спутниковых снимнов»
4. Освоены методы разработки

# **Список источников**

1. https://neurohive.io/ru/vidy-nejrosetej/u-net-image-segmentation/
2. https://habr.com/ru/post/332464/
3. https://www.youtube.com/watch?v=OWK8VlgJM4I
4. https://www.youtube.com/watch?v=OWK8VlgJM4I&t=475s
5. https://www.kaggle.com/arka47/satellitegooglemapsmasks