**Ministry of Science and Higher Education**

**of the Russian Federation**

**ITMO University**

Faculty of Digital Transformations

Educational program 01.04.02 Applied mathematics and informatics\_\_\_\_

Subject area (major) Big Data and Machine Learning\_\_\_\_

REPORT

on practical training Technical Internship

Task topic: Investigation of the influence of environmental factors on the psychoemotional state of a person

Student: Yamoldin Alexander, J42321c

Head of Practice from the trainee’s host organization: Basov Oleg, National Center for Cognitive Development, Senior Researcherhead of the laboratory «Center Information Optical Technologies»

Head of Practice from ITMO University: Alexandra Karabintseva, Faculty of Digital Transformation, Specialist in Educational and Methodological Work

|  |  |
| --- | --- |
|  | Practice completed with grade \_\_\_\_  Commission member signatures:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *full name*  (signature)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *full name*  (signature)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *full name*  (signature)  Date \_\_\_\_\_\_\_\_ |

St. Petersburg  
2023

# Annotation

34 pages, 51 pictures, 0 tables, 14 references

MACHINE LEARNING, CONVOLUTIONAL NEURONNET, OBJECT DETECTION, INDOOR AIR QUALITY, PSYCHO-EMOTIONAL STATE, WELL-BEING

**Object of research**: Changing the psycho-emotional state of people from Indoor Air Quality factors~~.~~

**Target of research:** The aim of the work is to determine the existence of a correlation between the psycho-emotional state of a person and the IAQ index.

**Methodology:** Application of Machine Learning and Artificial Intelligence for Dynamic Carbon Dioxide Level and Indoor Temperature Prediction.

**Results:** In this work have reviewed the theoretical aspects of solving the problems of object detection, selected a model for further study, converted the sensor data to the IAQ index, collected experimental data, selected behavioral markers, selected tools for data markup, and performed initial manual markup of the data

Table of contents

[Annotation 1](#_Toc124948421)

[1. The theoretical part of the computer vision concept 3](#_Toc124948422)

[1.1 Brief part of detection models 3](#_Toc124948423)

[1.2 YOLO model 5](#_Toc124948424)

[2. Installation and configuring YOLOv4 on Darknet Framework in local PC 7](#_Toc124948425)

[2.1 Git installation 7](#_Toc124948426)

[2.2 CMake installation 14](#_Toc124948427)

[2.3 Microsoft Visual Studio installation 18](#_Toc124948428)

[2.4 CUDA installation 20](#_Toc124948429)

[2.5 cuDNN installation 23](#_Toc124948430)

[2.6 OpenCV installation 24](#_Toc124948431)

[2.7 Darknet Framework installation 28](#_Toc124948432)

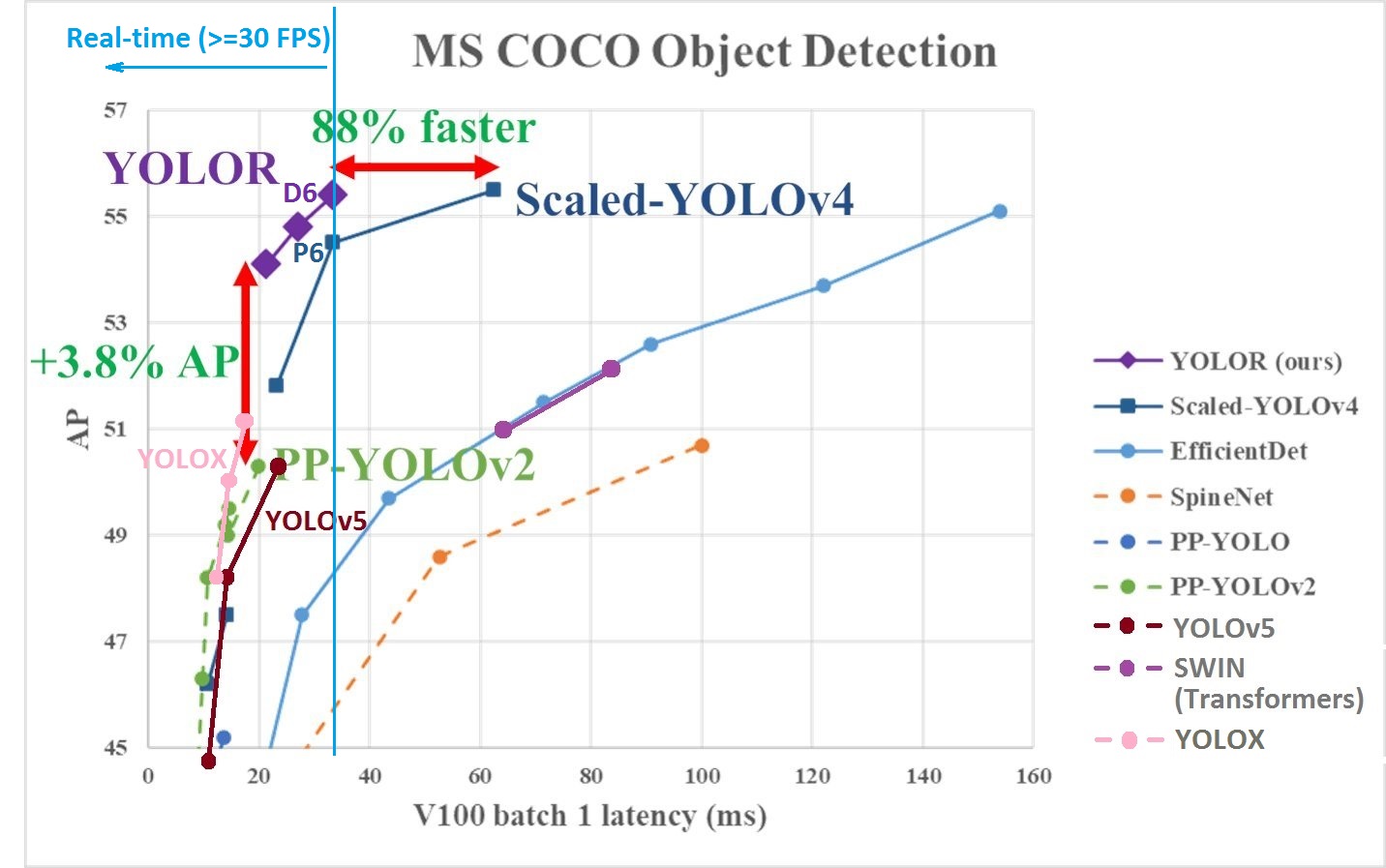
[3. Conclusion 34](#_Toc124948433)

[References 35](#_Toc124948434)

# 1. The theoretical part of the computer vision concept

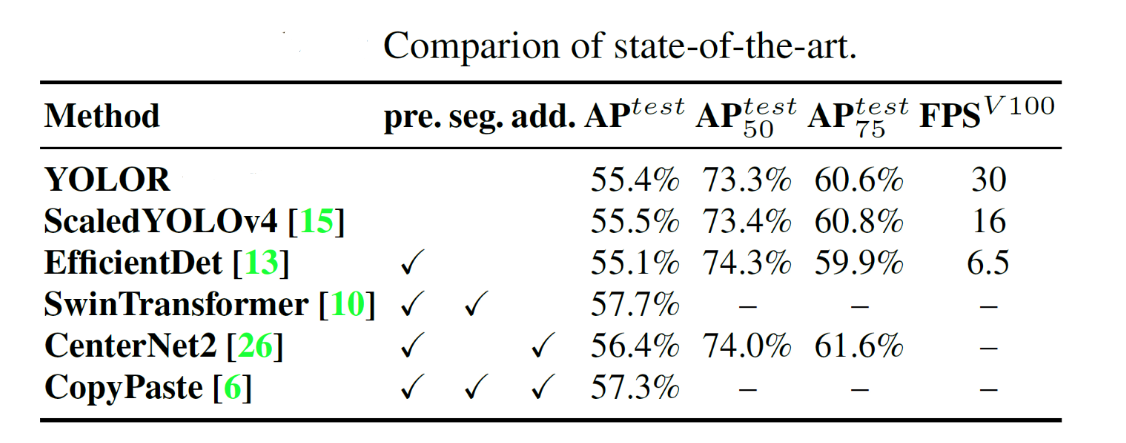
## 1.1 Brief part of detection models

Object detection models are currently being tested on the Microsoft COCO dataset [1]. This dataset contains 328,000 images with 2.5 million labeled instances of 91 object types that would be easily recognizable by a 4 year old. As of 2022, the best way to detect image objects is to YOLOR [2] and Scaled-YOLOv4 [3] (picture 1).



Picture 1 – Models comparing on MS COCO dataset [2]

This paper will consider the world’s best solutions available at the time of writing. Based on the work of [4], it is obvious that the best choice for the problem we are solving is Scaled-YOLOv4 (picture 2).



Picture 2 – Comparison state-of-the-art object detection models

ScaledYOLOv4 loses in FPS, but has the highest recognition accuracy. Since this work analyzes the change of IAQ in the room, which is a highly inertial parameter, we do not need to require the neural network to work directly in real time. Therefore, in this study priority to the quality of object recognition.

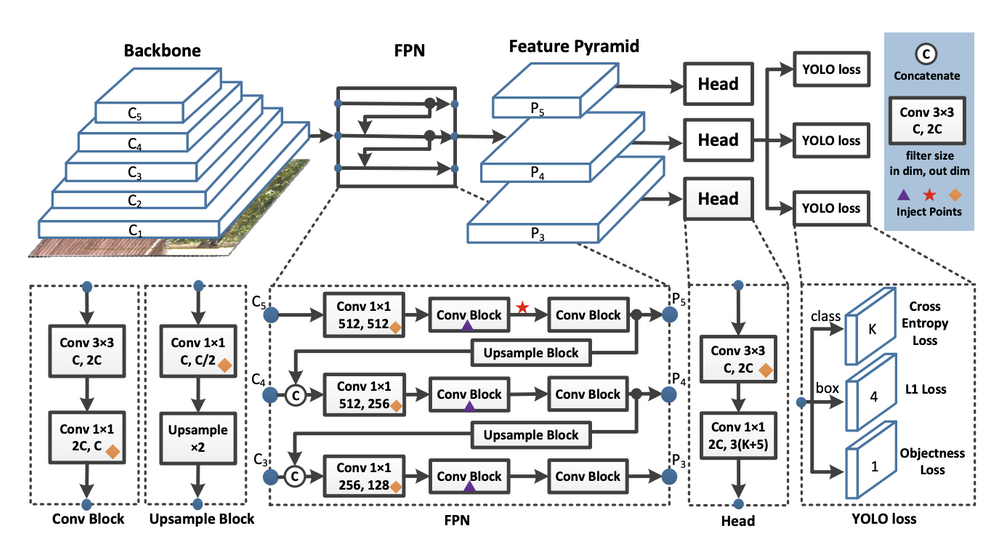
## 1.2 YOLO model

The YOLO (You Only Look Once) model was first published (by Joseph Redmon et al.) in 2015. The original YOLO network uses a modified GoogLeNet as a reference network [4]. Later, a new model called DarkNet-19 was created, which follows the general design of 3 × 3 filters, doubling the number of channels at each merging step. The 1 × 1 filters are also used to periodically compress the representation of an object across the network. YOLOv4 is made on the basis of YOLOv3 with the following changes: YOLOv4 uses CSPDarknet53 convolutional neural network as the backbone, SPP (spatial pyramid pooling) intermediate stage (neck) and (PAN) Path Aggregation Network, YOLOv3 full link final classifier (Dense Prediction). The basic models (Backbone) were pre-trained as image classifiers before being adapted to the detection task. To adapt the classification network into a network for object detection, it is necessary to remove the last few layers of the network and add a convolution layer with filters to produce Bbox predictions. A grid is superimposed on the input image, dividing it into S×S regions. For each region, the neural network determines 5 bounding boxes for the object, a confidence score for frame detection, which reflects the degree of confidence of the model that the field contains an object:

|  |  |
| --- | --- |
|  | (1) |

As a result, the network selects the area with the highest level of network response reliability.

Unlike the parent YOLOv4 (picture 3) model the Scaled-YOLOv4 authors apply the concepts laid out in the Cross-Stage Partial Networks [5].



Picture 3 – YOLOv4 architecture

To detect large objects in large images, the authors increase the depth and number of stages in in the CNN backbone and neck. This allows them to first scale up input size and number of stages, and dynamically adjust width and depth according to realtime inference speed requirements.

However, a special difficulty is the installation and configuration of this architecture on Windows 10. The installation and configuration of the YOLOv4 neural network on a personal computer under Microsoft Windows 10 was performed during the technological practice.

# 2. Installation and configuring YOLOv4 on Darknet Framework in local PC

## 2.1 Git installation

The first thing you need to install on your personal computer is git. It is necessary to download the executable file [6] in our case for the 64-bit operating system.

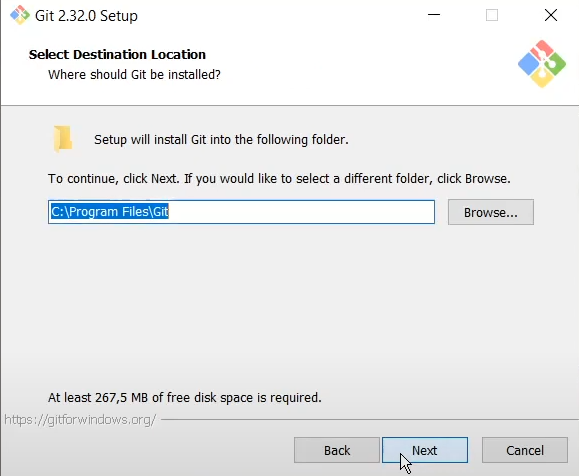
After downloading the file, run the executable file and follow the instructions.

The first window you will see is the welcome screen (picture 4) and click *Next*

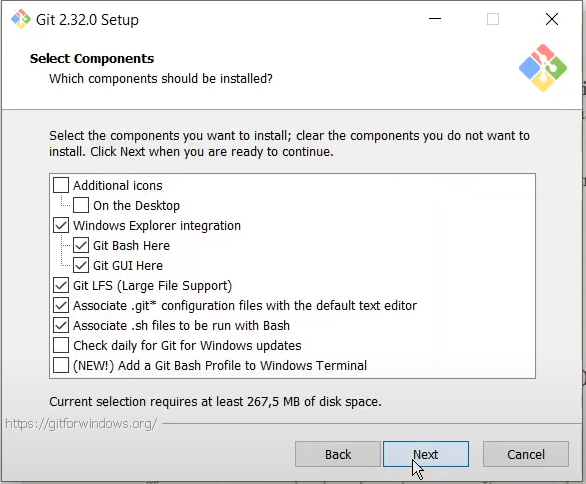


Picture 4 – Git welcome screen

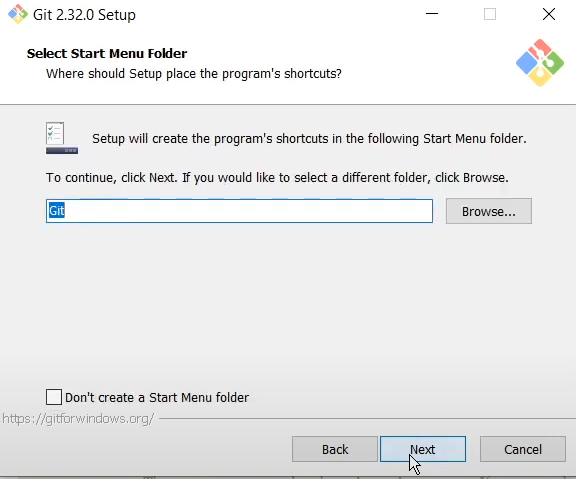
On the next window (picture 5) keep the default values and click *Next*



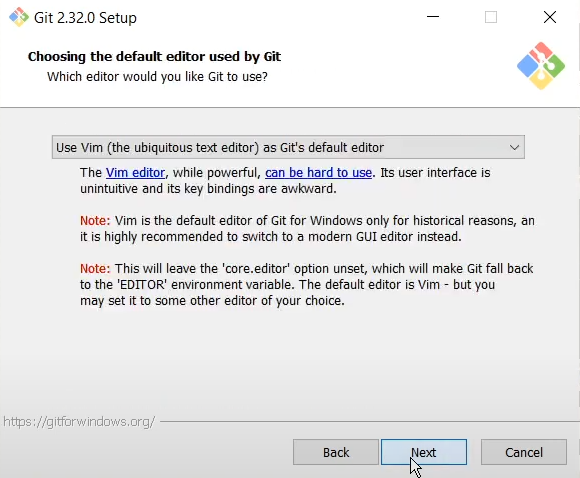
Picture 5 – The window for selecting the git installation path



Picture 6 – Selecting components



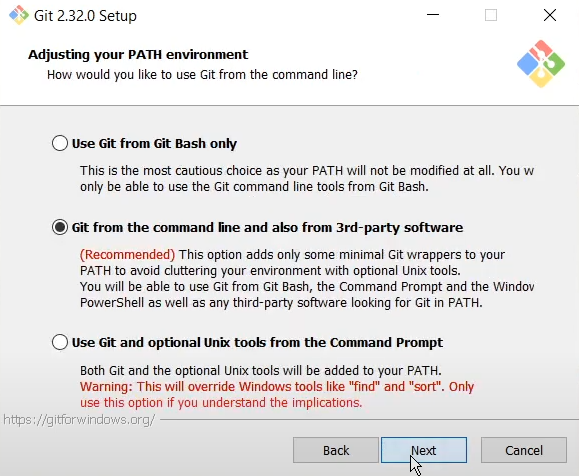
Picture 7 – Selecting start menu folder



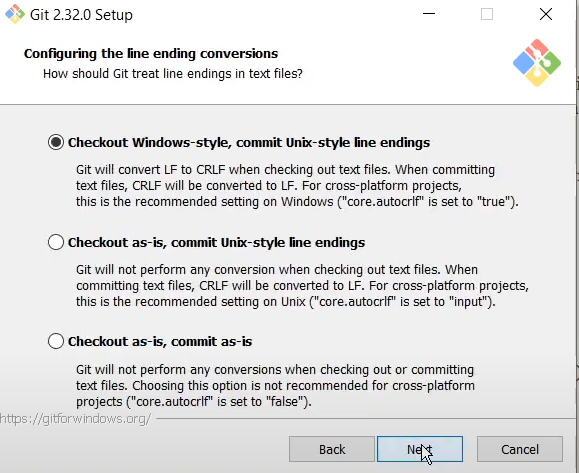
Picture 8 – Selecting default editor



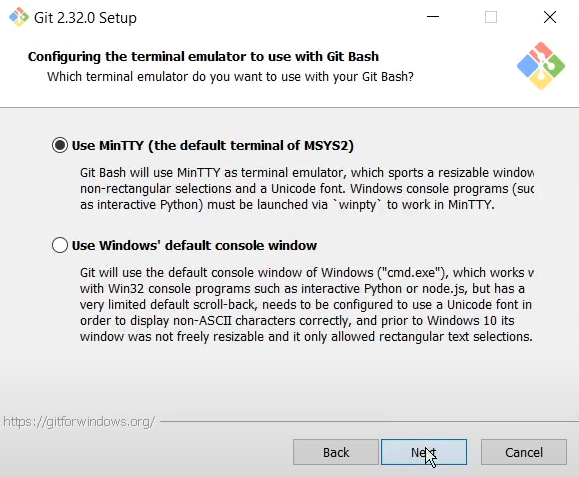
Picture 9 – Adjusting the name of the initial branch in new repositories



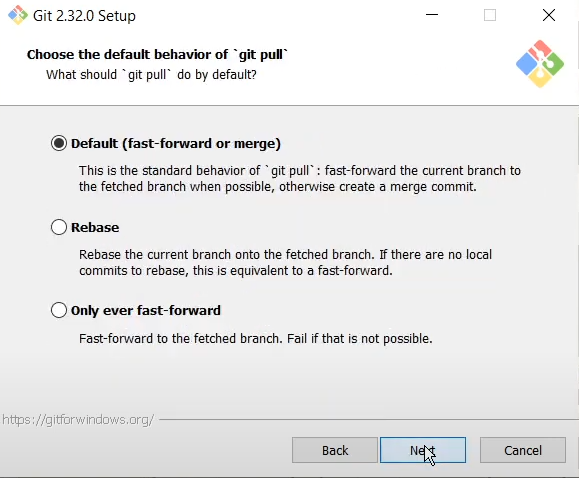
Picture 10 – Adjusting PATH environment



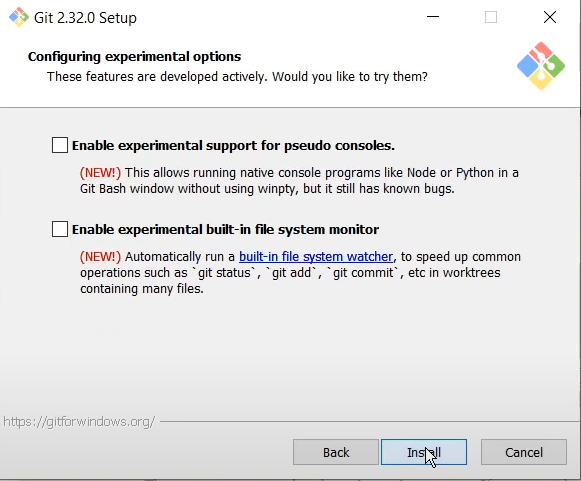
Picture 11 – Configuring the line ending conversions



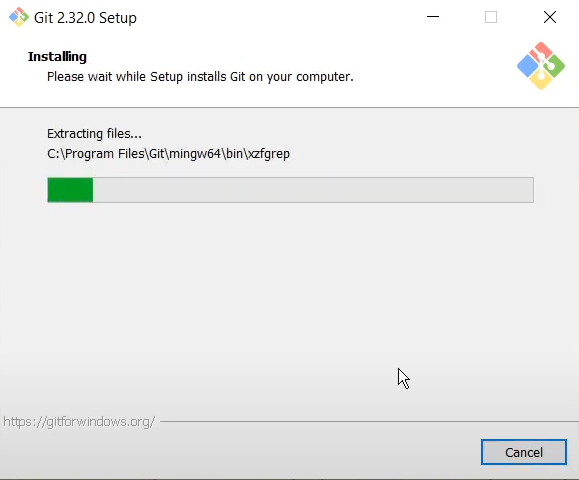
Picture 12 – Configuring the terminal emulator



Picture 13 – Choose the default behavior of Git Pull



Picture 14 – Configuring experimental options

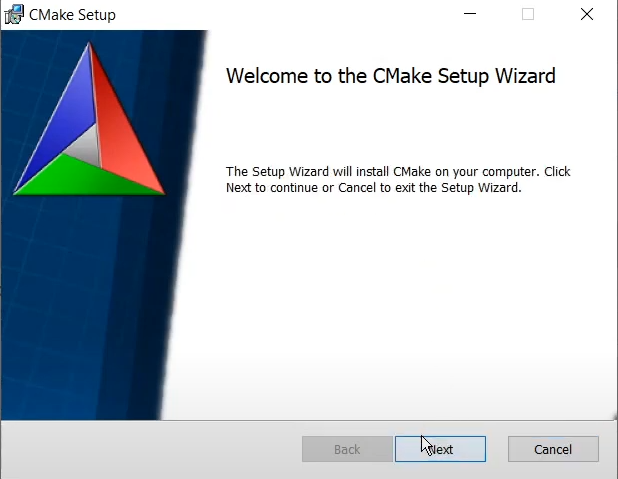


Picture 15 – Install git

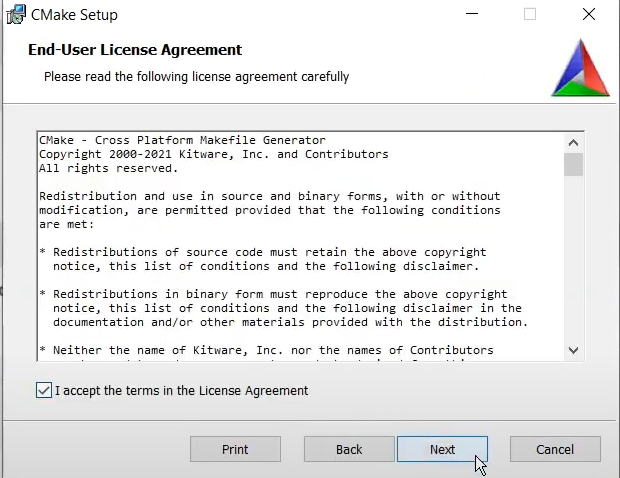
## 2.2 CMake installation

It is necessary to download the cmake-3.21.0-rc2-windows-x86\_64.msi package [7] in our case for the 64-bit operating system.

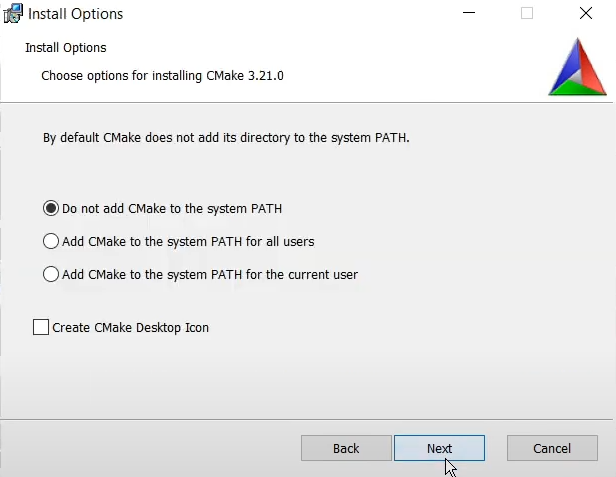
After downloading the file, run the .msi package and follow the instructions.



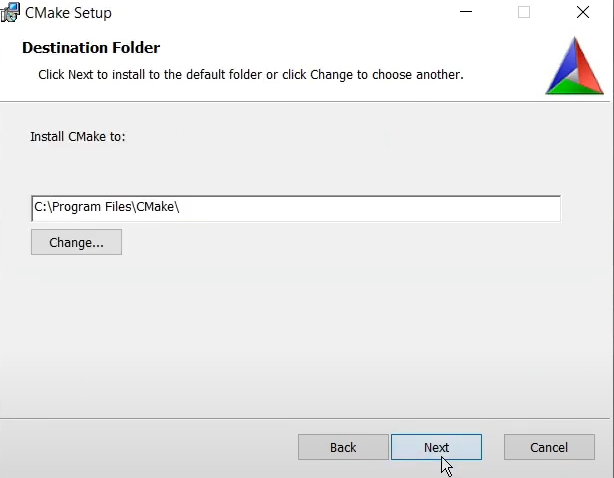
Picture 18 – CMake setup



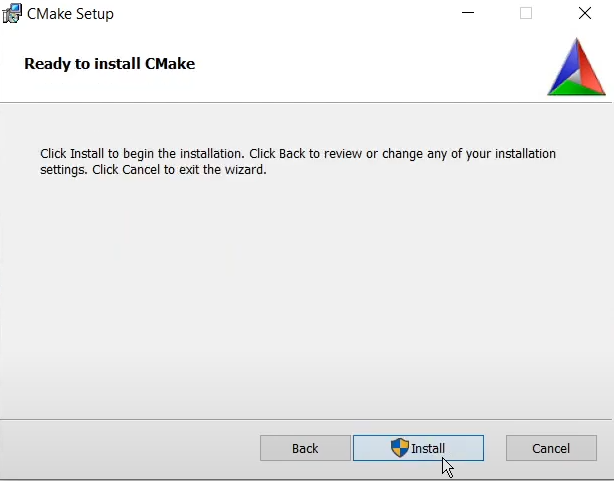
Picture 19 – License Agreement



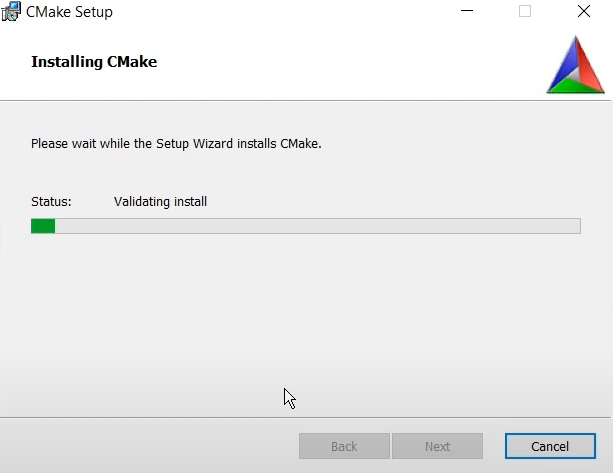
Picture 20 – Install options



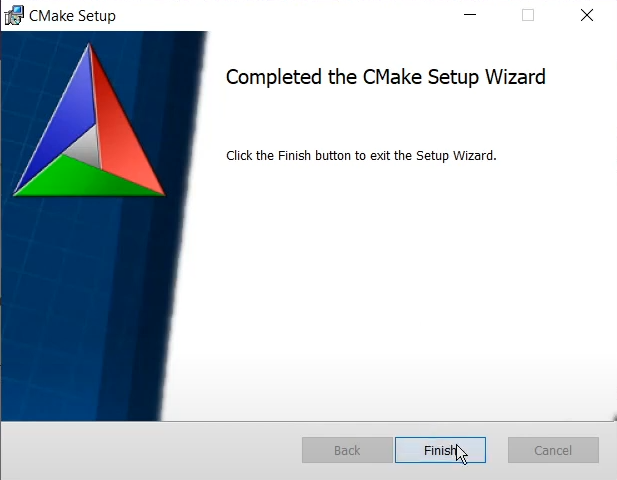
Picture 20 – Destination Folder



Picture 21 – Ready to install



Picture 22 – Installing CMake

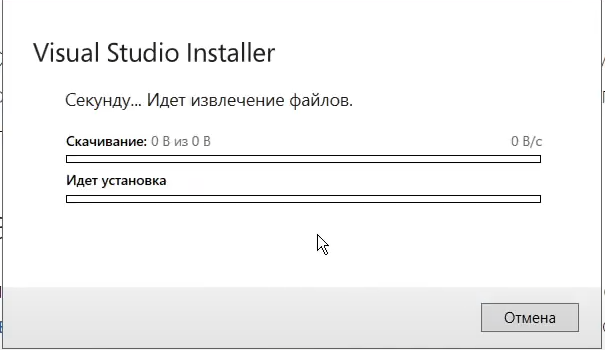


Picture 23 – Installation CMake succesfull

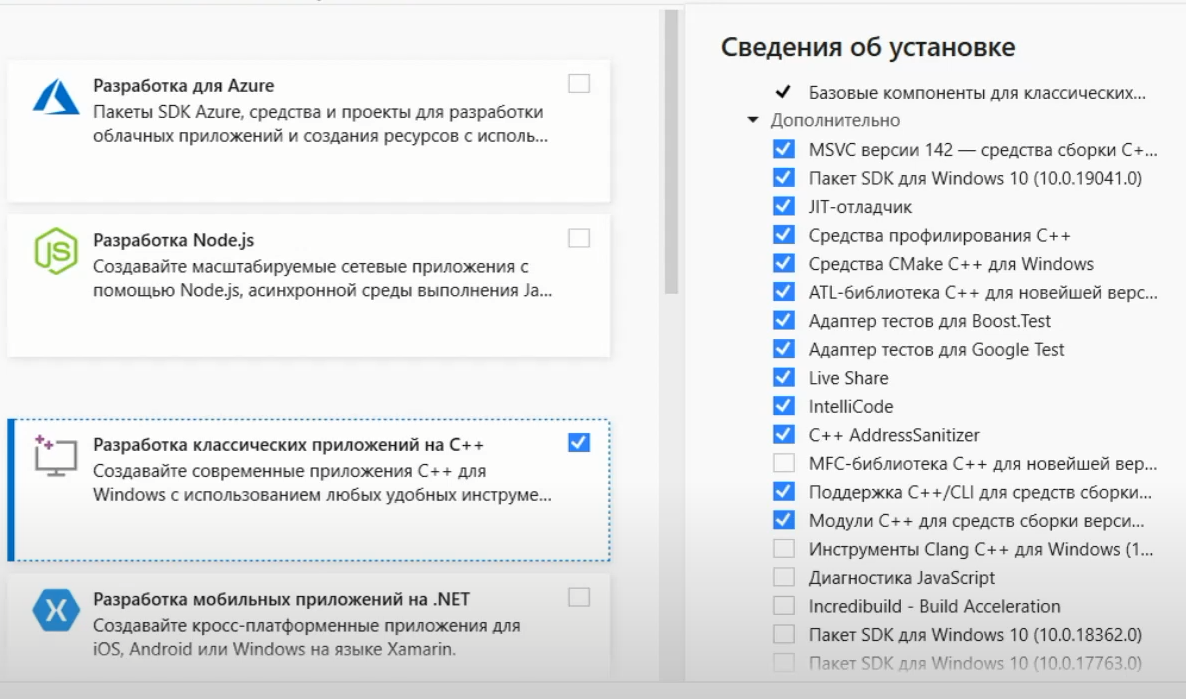
## 2.3 Microsoft Visual Studio installation

It is necessary to download the Microsoft Visual Studio execution file [8] in our case for the 64-bit operating system.

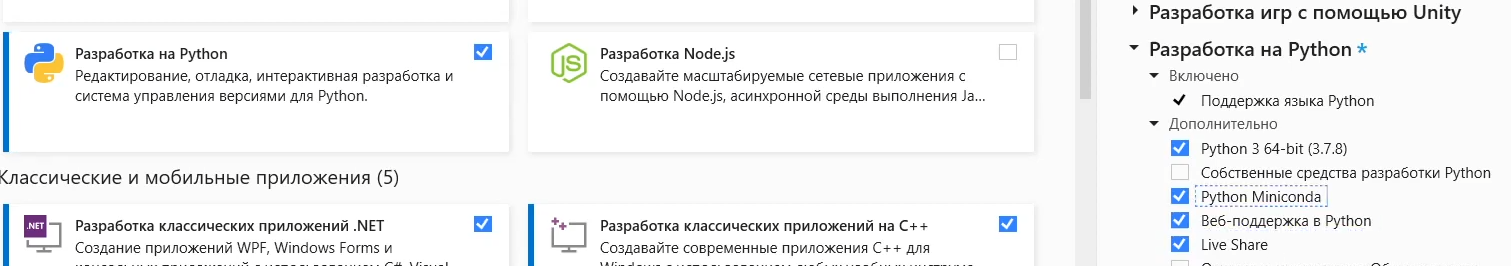
After downloading the file, run the execution file and follow the instructions.



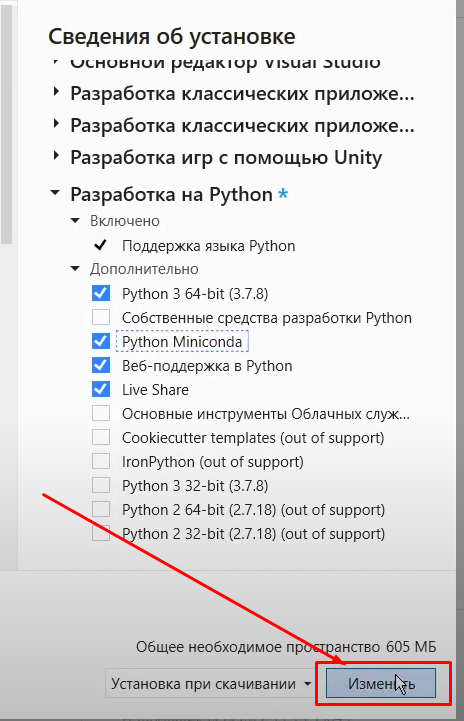
Picture 24 – Wait for file extraction



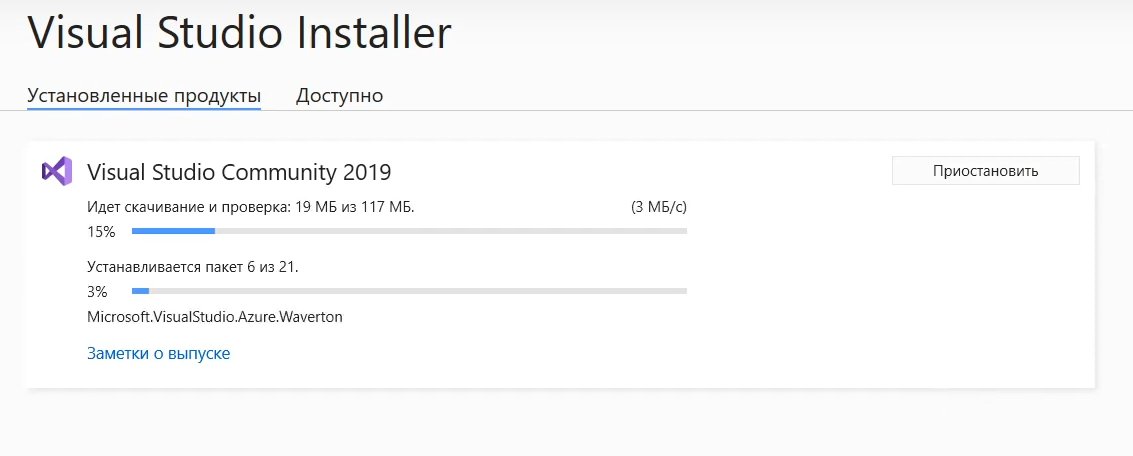
Picture 25 – Choose the following modules for C++



Picture 26 – Choose the following modules for python



Picture 27 – Click to the button in right down corner

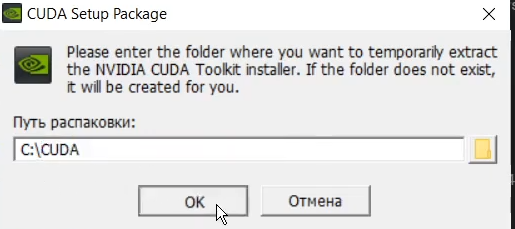


Picture 28 – Redirecting to the installing process

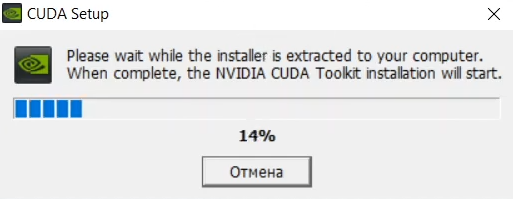
## 2.4 CUDA installation

It is necessary to download the CUDA Toolkit execution file [9] in our case for the 64-bit operating system.

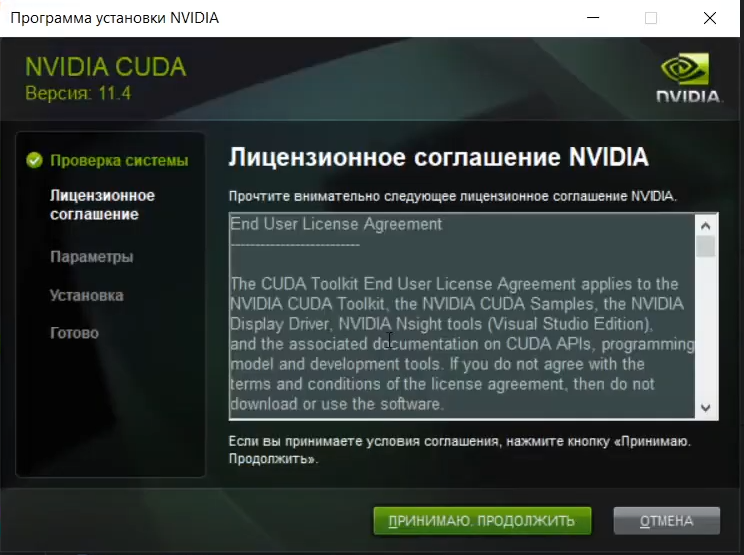
After downloading the file, run the execution file and follow the instructions.



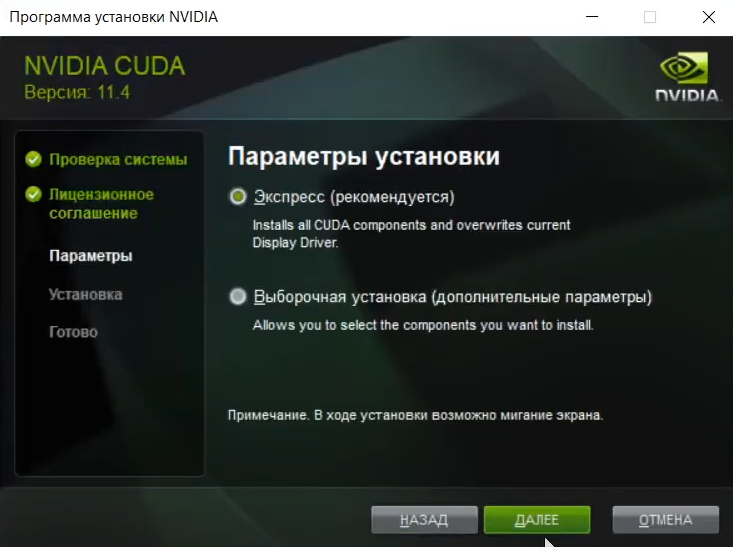
Picture 30 – Path to CUDA



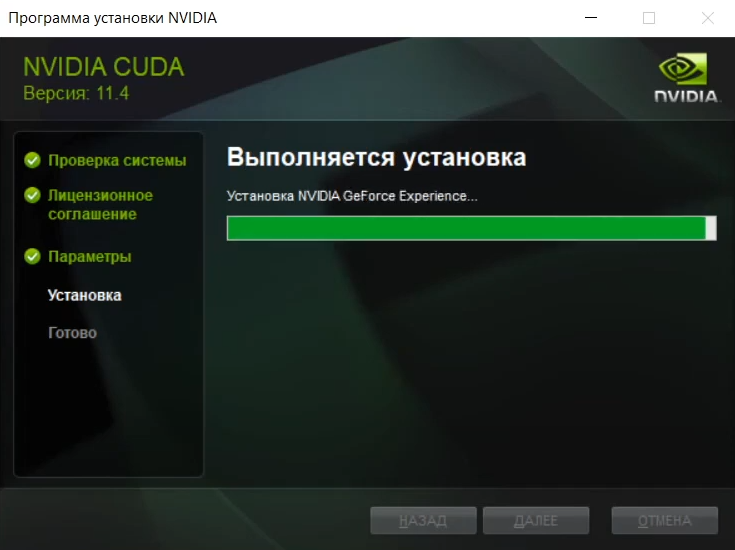
Picture 31 – CUDA extraction



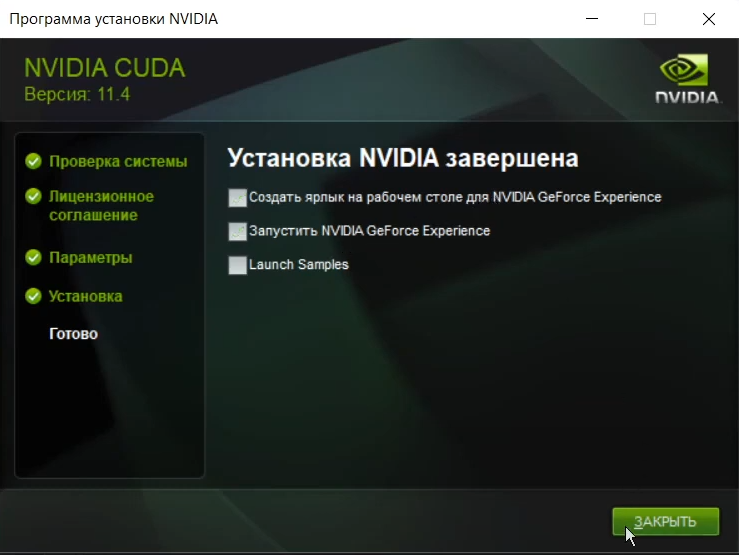
Picture 32 – CUDA license agreement



Picture 33 – CUDA parameters installation



Picture 34 – CUDA installation



Picture 35 – CUDA installation completed

## 2.5 cuDNN installation

Pay attention! You must be a registered user in developer.nvidia.com community. It is necessary to download the cuDNN 11.x archive [10] in our case for the 64-bit operating system.

After downloading the file, unpackage the cuDNN archive to C:\cuda and follow the instructions.

Copy all files from

C:\cuda\bin and past them in C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.4\bin

The same way copy files from C:\cuda\include and past them in C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.4\include

The same way copy files from C:\cuda\lib\x64 and past them in C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.4\lib\x64

## 2.6 OpenCV installation

It is necessary to download the OpenCV source code archive [11] in our case for the 64-bit operating system.

Create directory named “opencv” on path C:\opencv and in the new directory create another one directory C:\opencv\build

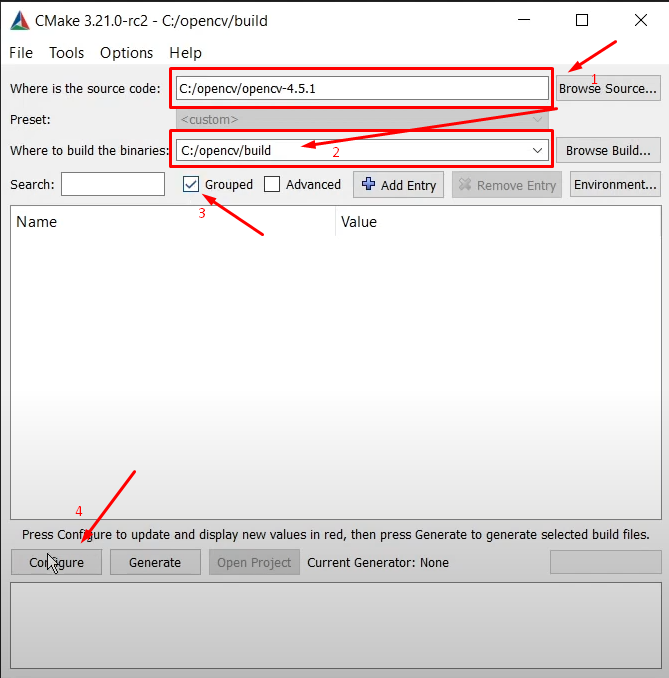
After downloading the file, unpackage OpenCV-4.5.1.zip archive in C:\opencv.

Also, it is necessary to download the open\_contrib source code archive [12].

After downloading the file, unpackage opencv\_contrib-4.5.1.zip archive in C:\opencv.

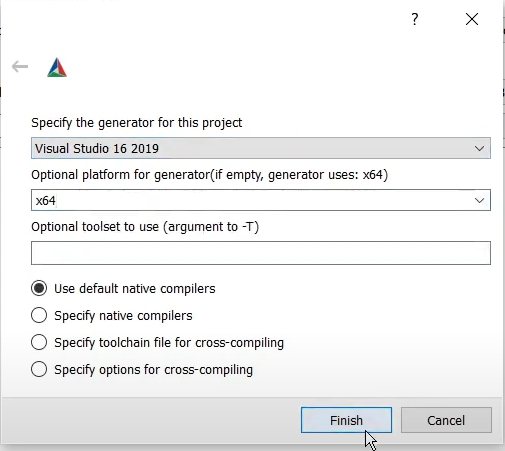
Now we should build OpenCV with CMake. Start CMake application and follow the instructions.

Browse for the source code in C:\opencv\opencv-4.5.1 and build in C:\opencv\build. Stay checkbox to Grouped and click Configure



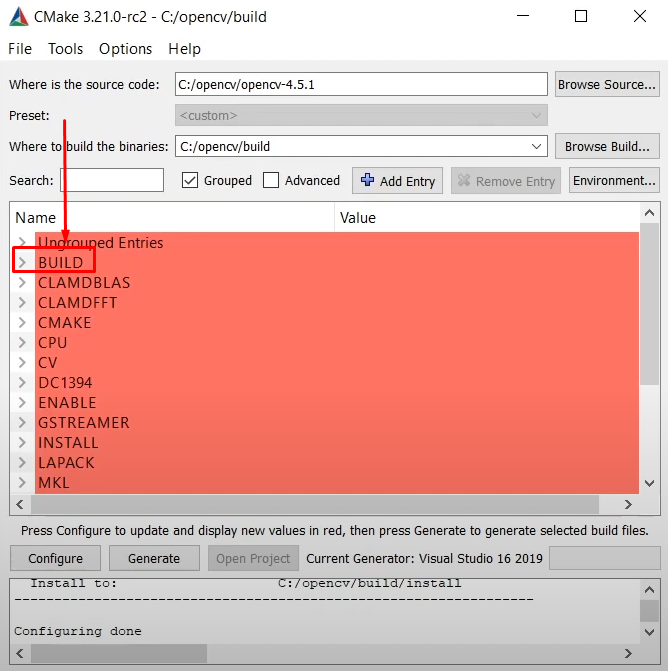
Picture 36 – Configuration build OpenCV

Choose configuration as on the picture 37 and click Finish.



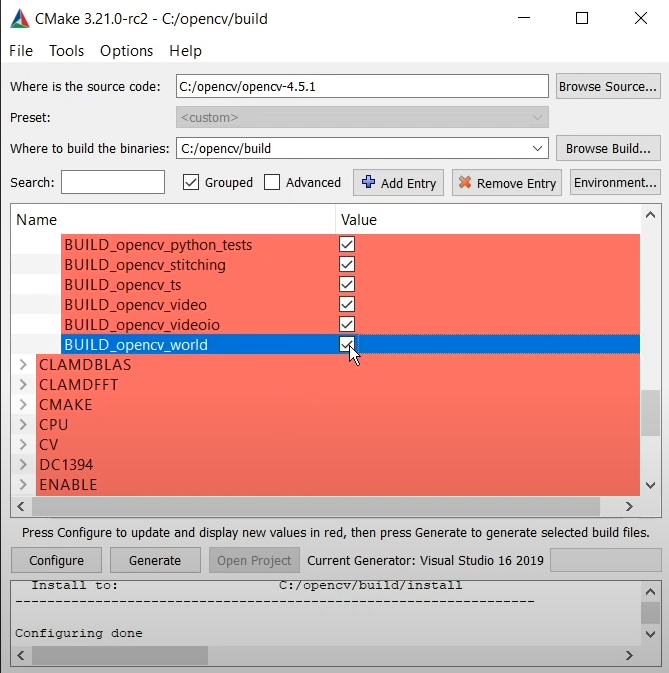
Picture 37 – Finish project configuration

Wait while OpenCV configurating. When configurating done, you must go to BUILD (picture 38)



Picture 38 – Go to BUILD after configurating

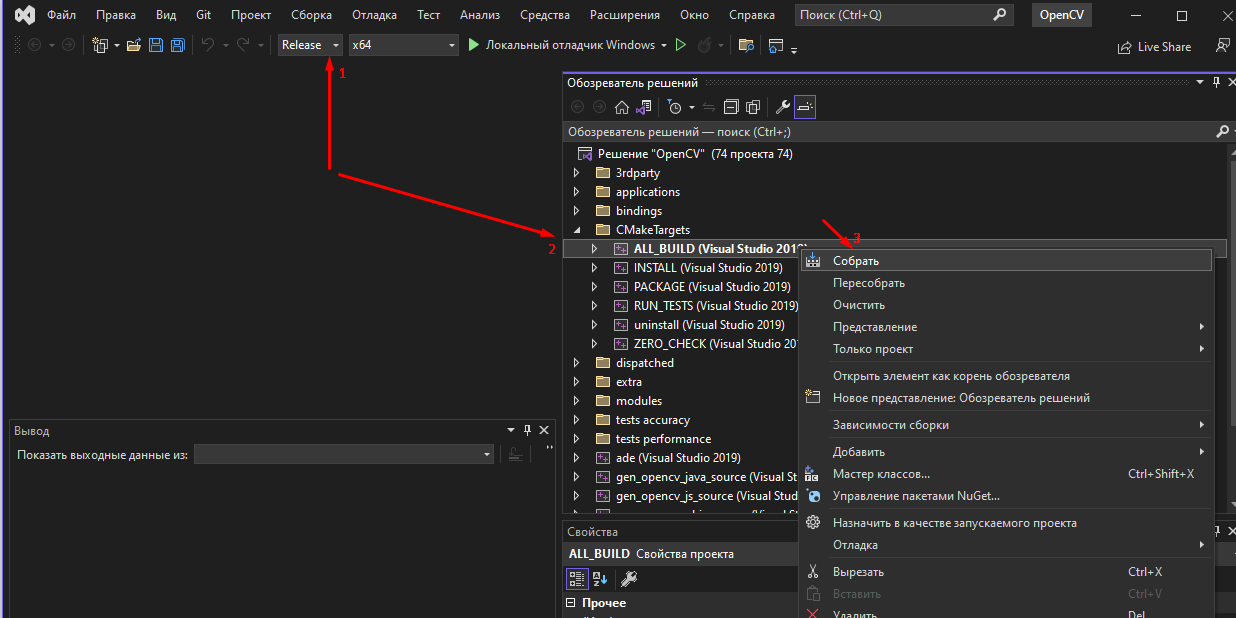
And stay checkbox BUILD\_opencv\_world as in picture 39. and click to the Generate button in the down of window.



Picture 39 – Add BUILD\_opencv\_world checkbox

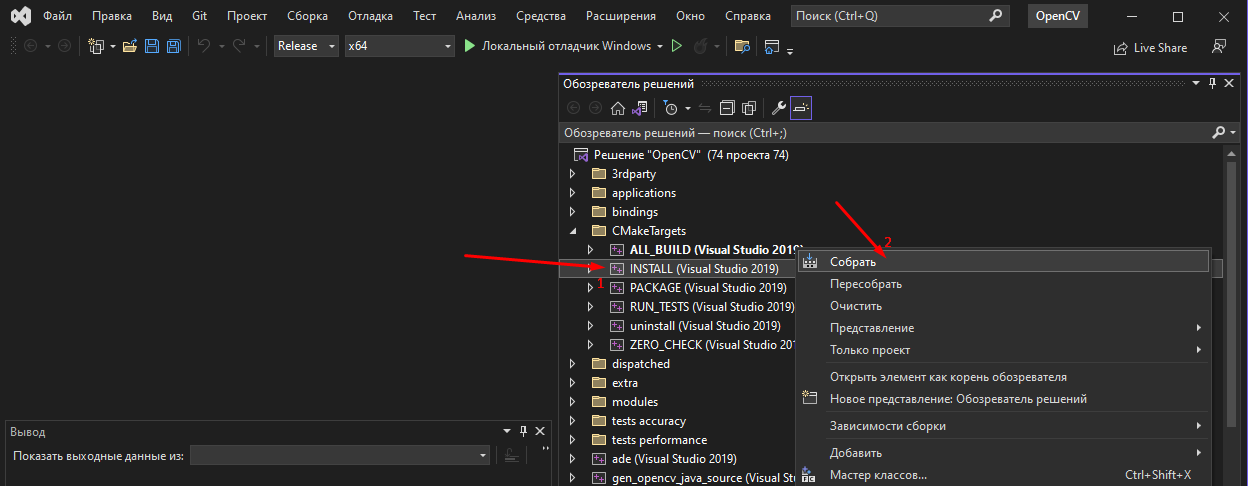
When build of OpenCV is finished you must open with Microsoft Visual Studio file C:\opencv\build\ALL\_BUILD.vcxproj

In the Microsoft Visual Studio (picture 40) choose Release (step 1), go to CMakeTargets, choose ALL\_BUILD, do mouse right button click and choose Build.



Picture 40 – Build ALL\_BUILD

When build is done, you should choose INSTALL, do mouse right button click and choose Build (picture 41).



Picture 41 – Build INSTALL

Now, OpenCV is successfully installed.

## 2.7 Darknet Framework installation

In this step we will install Darknet Framework. Сreate a directory wherever you want. For example, I create ITMO directory in C:\Users\gto\_n\Documents\ITMO.

In the directory open command interpreter cmd.exe [13] and print into command

git clone <https://github.com/AlexeyAB/darknet.git>

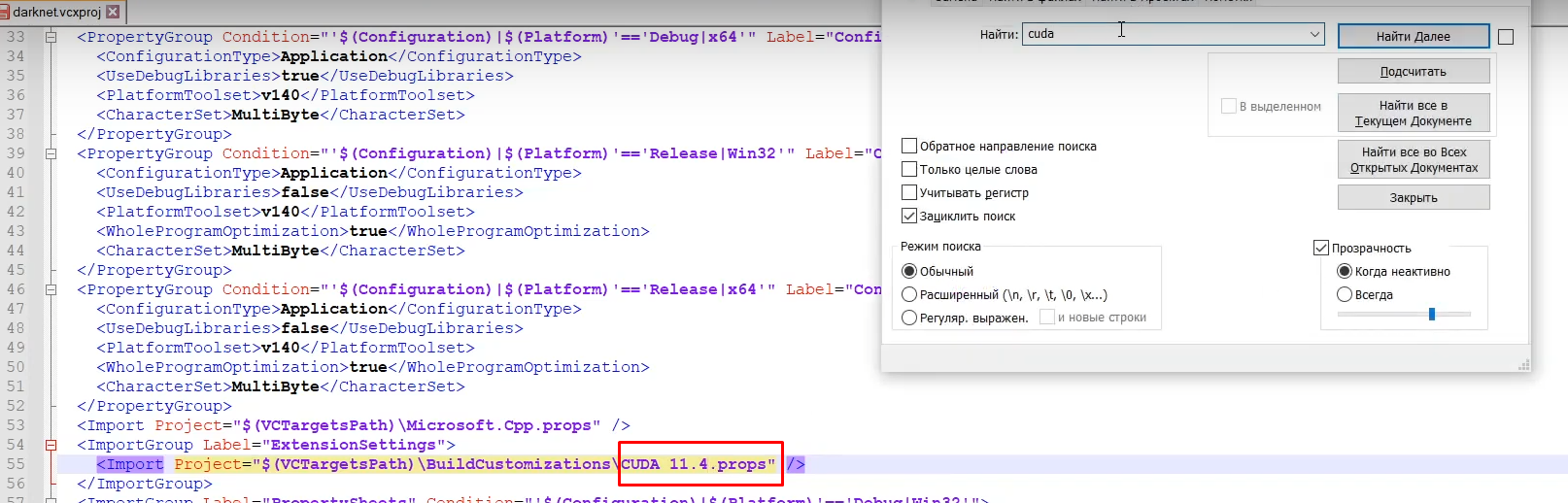
After cloning the repository, copy 2 files from OpenCV build:

1. C:\opencv\build\bin\Release\opencv\_videoio\_ffmpeg451\_64.dll
2. C:\opencv\build\bin\Release\opencv\_world451.dll

Past copied files in C:\Users\gto\_n\Documents\ITMO\darknet\build\darknet\x64.

After that copy file C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.4\bin\cudnn64\_8.dll and past copied file in C:\Users\gto\_n\Documents\ITMO\darknet\build\darknet\x64.

Open with text editor (Notepad, Notepad++, Visual Studio Code and e.g.) file C:\Users\gto\_n\Documents\ITMO\darknet\build\darknet\darknet.vcxproj. Open «Search in documents», find all «cuda» instances and replace in text cuda version from 11.2 to 11.4 (picture 42).

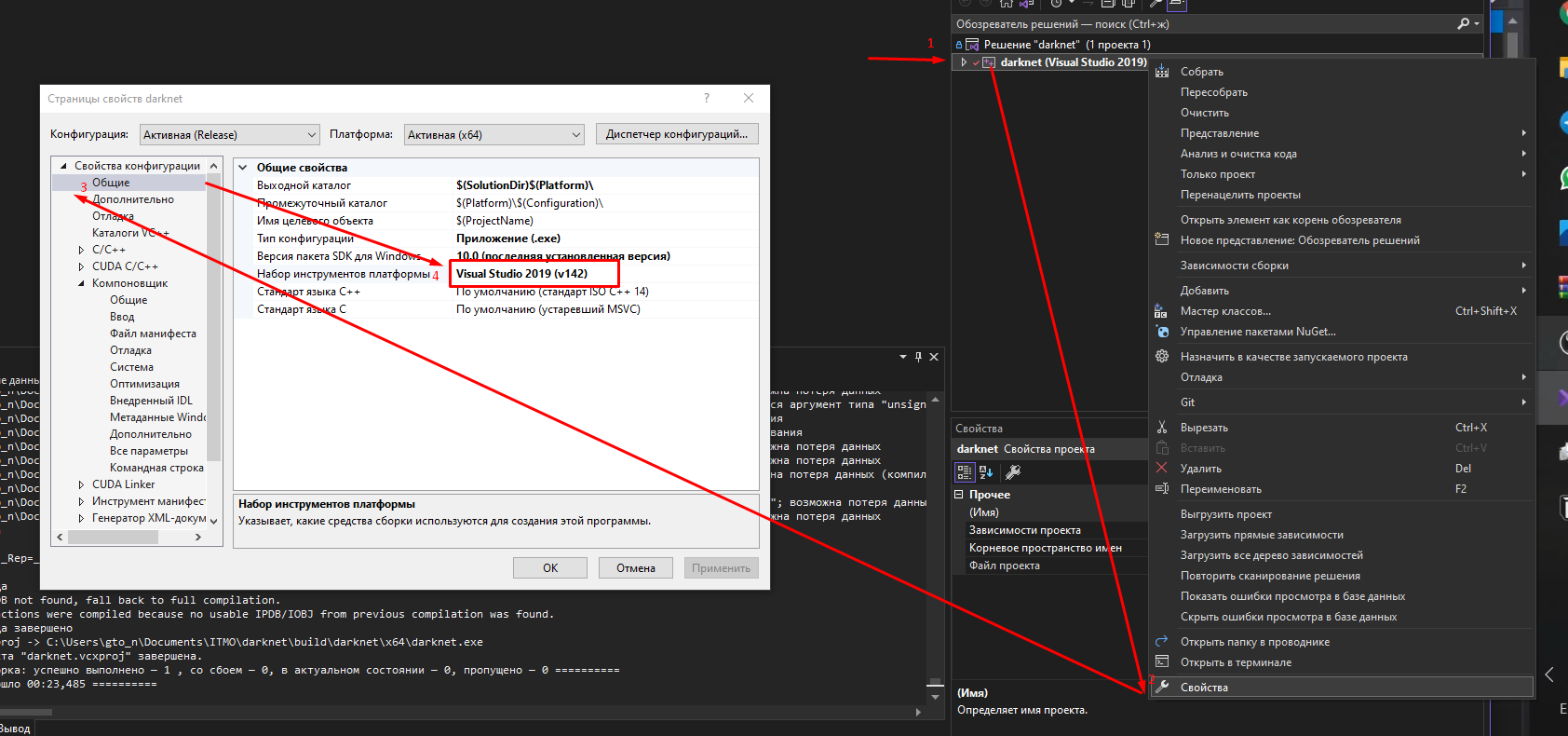


Picture 42 – Changing cuda version from 11.2 to 11.4

Absolutely the same operations you must do with file C:\Users\gto\_n\Documents\ITMO\darknet\build\darknet\yolo\_cpp\_dll.vcxproj.

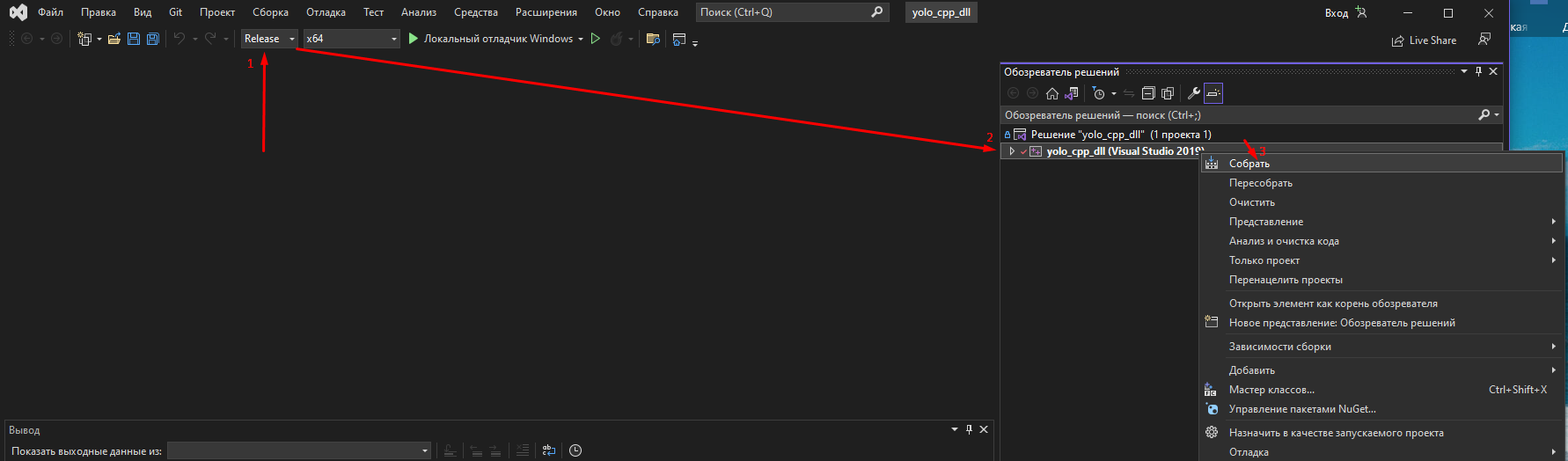
Open with Microsoft Visual Studio file C:\Users\gto\_n\Documents\ITMO\darknet\build\darknet\yolo\_cpp\_dll.vcxproj.

In the Microsoft Visual Studio do mouse right click to «yolo\_cpp\_dll», choose «properties», choose «Platform instruments», choose «Visual Studio 2019 (v142)» (picture 43).



Picture 43 – Choosing Visual Studio 2019 configuration

After that we can build «yolo\_cpp\_dll». Do mouse right click to «yolo\_cpp\_dll» and click to «build» (picture 44).



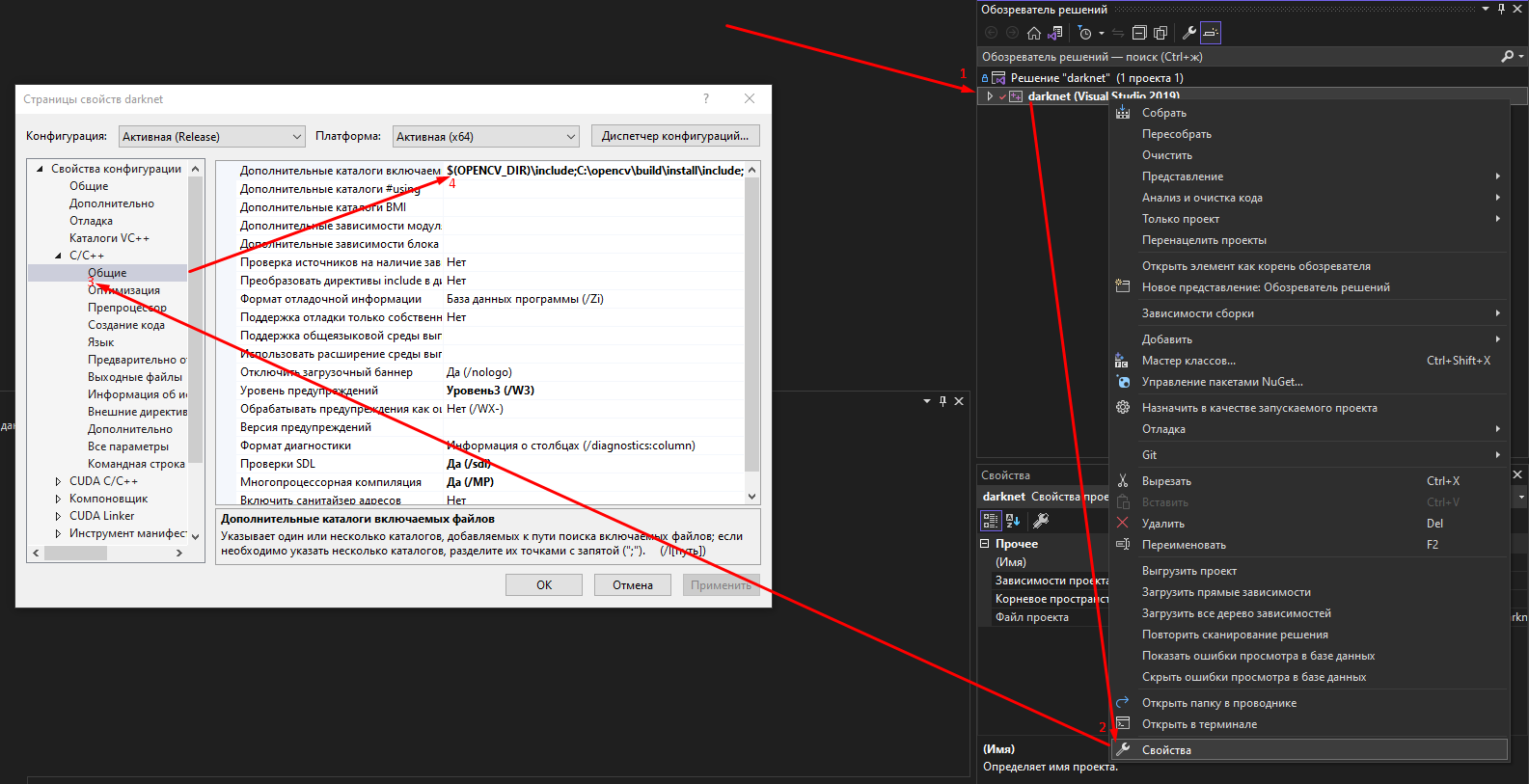
Picture 44 – Building «yolo\_cpp\_dll»

If you have an error «Ошибка : произошел сбой сборки Designtime для конфигурации "C:\Users\gto\_n\Documents\darknet\build\darknet\yolo\_cpp\_dll.vcxproj" проекта "Release|x64". Компонент IntelliSense может быть недоступен. Задайте для переменной среды TRACEDESIGNTIME значение True и перезапустите Visual Studio, чтобы найти причины.»

You must close the Microsoft Visual Studio and delete hidden directory «.vs» from «yolo\_cpp\_dll» starts directory and repeat previous step (picture 44).

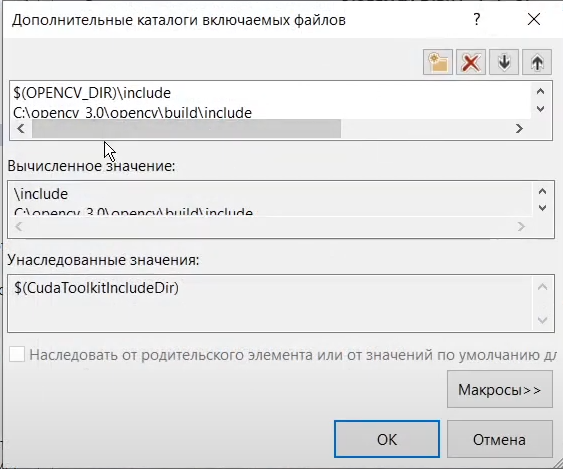
Finally, we start to build Darknet Framework.

Open with Microsoft Visual Studio file C:\Users\gto\_n\Documents\ITMO\darknet\build\darknet\darknet.sln. In the Microsoft Visual Studio do mouse right click to «darknet», choose «properties», choose «C/C++», choose «Common» and choose «Additional catalogs include»(picture 45).



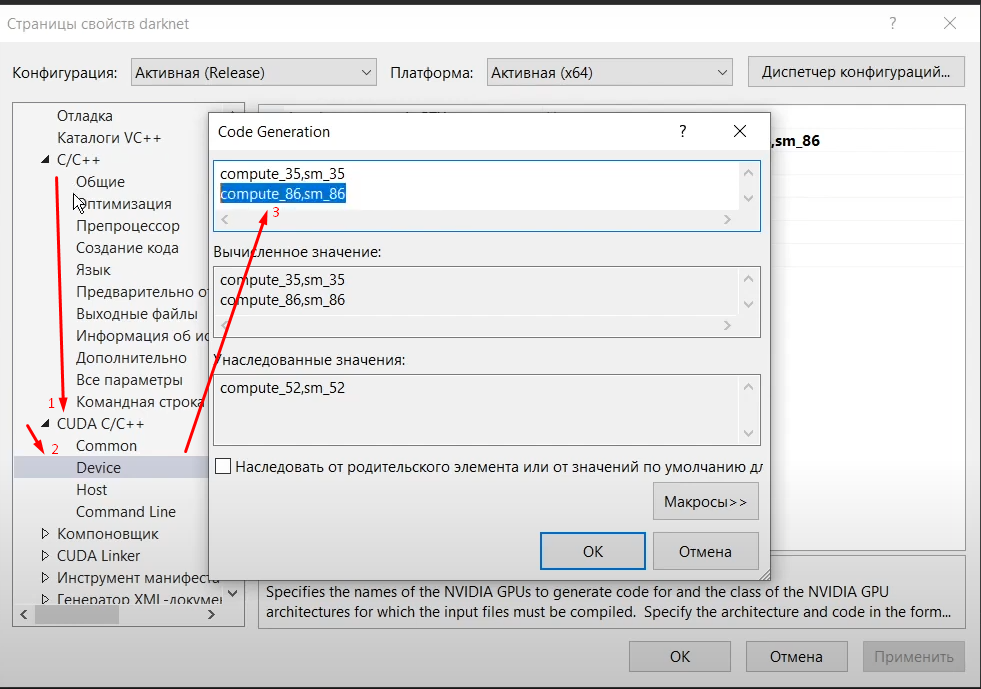
Picture 45 – Go to additional catalogs

We will see window (picture 46) and in the first sub-window we need to change path to OpenCV. In our case OpenCV is on C:\opencv\build\install\include so this path we need to replace from C:\opencv\_3.0\opencv\build\install\include



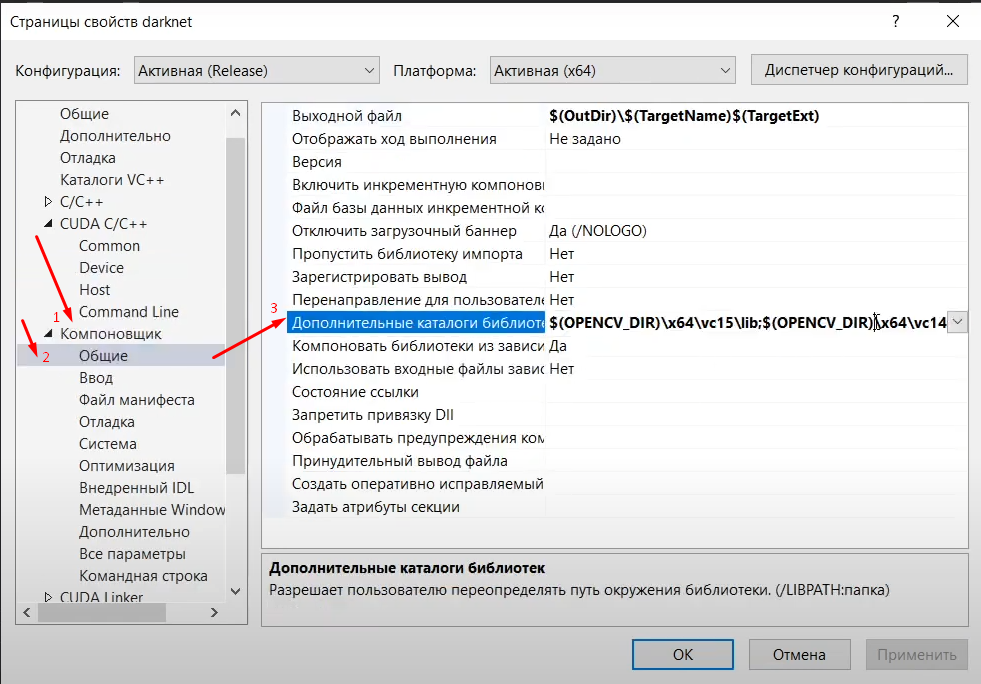
Picture 46 – Replacing OpenCV path

After that we need to go to «CUDA C/C++», choose «Device» and in the first sub-window delete second string «compute\_86,sm\_86» (picture 47).



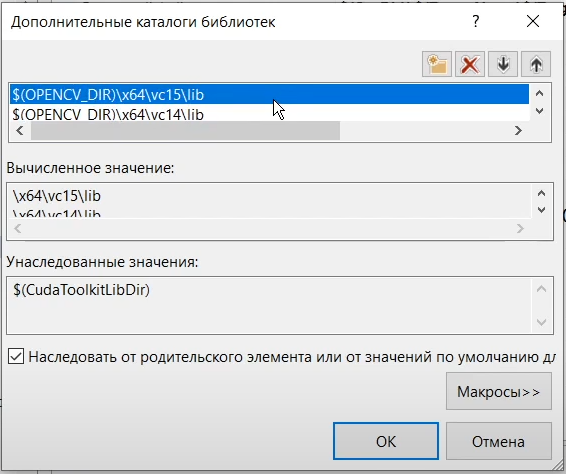
Picture 47 – Deleting second string

After that we should go to «Linker», choose «Common» and select «Additional catalogs of libraries» and



Picture 48 – Go to OpenCV directory

We will see window (picture 49) and in the first sub-window we need to change path to OpenCV lib. In our case OpenCV lib is on C:\opencv\build\install\x64\vc16\lib so this path we need to replace from $(OPENCV\_DIR)\x64\vc15\lib



Picture 49 – Replacing $(OPENCV\_DIR)\x64\vc15\lib to C:\opencv\build\install\x64\vc16\lib

Click to «Apply» button on right down corner in Microsoft Visual Studio and close properties window.

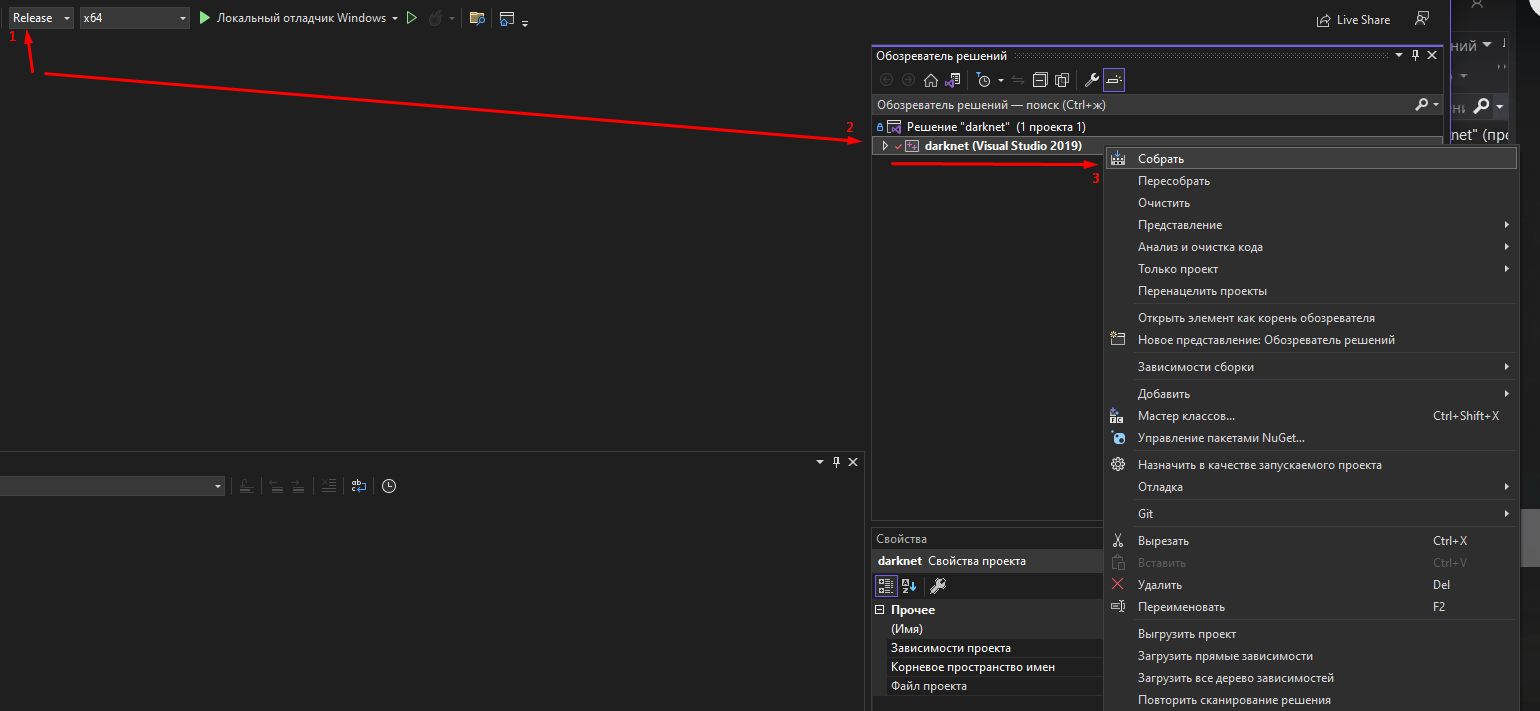
After that, according forum developer nvidia [14] you must copy all files from this path (depends on the path you installed CUDA in) in our case:

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.4\extras\visual\_studio\_integration\MSBuildExtensions

to this path:

C:\Program Files\Microsoft Visual Studio\2022\Community\MSBuild\Microsoft\VC\v170\BuildCustomizations

And start Darknet building (picture 50).

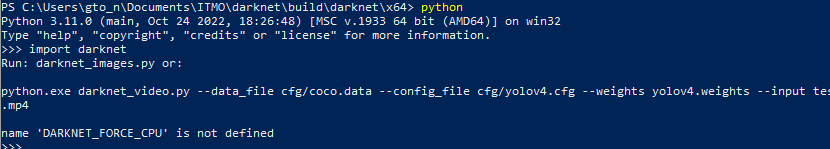


Picture 50 – Building the Darknet

If you have python > 3.8 you must go to darknet folder C:\Users\gto\_n\Documents\ITMO\darknet\darknet.py and adding winmode=0 to

lib = CDLL(winGPUdll, RTLD\_GLOBAL, winmode=0).

To check the installation in C:\Users\gto\_n\Documents\ITMO\darknet\ directory open cmd and past «python» click to Enter and past «import darknet»



Picture 51 – Importing darknet

As we can see darknet import is successful.

So we completely build darknet framework and YOLOv4 could be used.

# 3. Conclusion

In the tech practice the theoretical foundations of detection models were briefly discussed. We compared the accuracy and performance of modern object detection models and showed that scaled-YOLOv4 is the most balanced by the criterion of detection quality/speed of work.

The YOLOv4 neural network is shared on the darknet framework. Building and configuring this framework on a local computer is very difficult. Therefore, during the technological practice it was decided to build and configure the framework for further work with the training of the neural network YOLOv4.

In the work done, we installed git, CMake, and Microsodt Visual Studio. We specified the details of the configuration for further successful builds. We also installed CUDA on our computer and configured it cuDNN for further work.

A special difficulty was the building and installation of OpenCV. To install OpenCV we had to swap its files with opencv contrib, build and configure it with CMake, rebuild it twice with Microsoft Visual Studio and change its configuration.

After installing OpenCV, we proceeded to install the Darknet Framework. During the installation we had to manually replace the build files from OpenCV, CUDA. Manually edit the darknet.vcxproj project code to change the CUDA version. We made a special configuration of the project in Microsoft Visual Studio to build the framework successfully. We rebuilt the build files three times, changing the project configuration for each of them. In addition we had to correct the python code to successfully import darknet.

As a result, we successfully installed and configured the necessary components, which will allow us to train the YOLOv4 neural network on the local computer.

The main goal of technical practice has been successfully achieved.

# References

1. Lin T.-Y. et al. Microsoft COCO: Common Objects in Context // Computer Vision – ECCV 2014. Cham: Springer International Publishing, 2014. P. 740–755.
2. Wang C.-Y., Yeh I.-H., Liao H.-Y.M. You Only Learn One Representation: Unified Network for Multiple Tasks [Electronic resource] // arXiv.org. 2021. URL: https://arxiv.org/abs/2105.04206.
3. Wang C.-Y., Bochkovskiy A., Liao H.-Y.M. Scaled-YOLOv4: Scaling Cross Stage Partial Network [Electronic resource] // arXiv.org. 2020. URL: https://arxiv.org/abs/2011.08036.
4. Redmon J. et al. You Only Look Once: Unified, Real-Time Object Detection [Electronic resource] // arXiv.org. 2015. URL: https://arxiv.org/abs/1506.02640.
5. Wang C.-Y. et al. CSPNet: A New Backbone that can Enhance Learning Capability of CNN [Electronic resource] // arXiv.org. 2019. URL: https://arxiv.org/abs/1911.11929.
6. Download git 2.39.02-64-bit. – URL: <https://github.com/git-for-windows/git/releases/download/v2.39.0.windows.2/Git-2.39.0.2-64-bit.exe> (reference date 17.01.2023)
7. Download cmake-3.21.0-rc2-windows-x86\_64. – URL: <https://cmake.org/files/v3.21/cmake-3.21.0-rc2-windows-x86_64.msi> (reference date 17.01.2023)
8. Download Microsoft Visual Studio. – URL: <https://visualstudio.microsoft.com/ru/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSLandingPage&cid=2030&passive=false> (reference date 17.01.2023)
9. Download cuda\_11.4.0\_471.11\_win10. – URL: <https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda_11.4.0_471.11_win10.exe> (reference date 17.01.2023)
10. Download cudnn-windows-8664-87084cuda11-. – URL: <https://developer.nvidia.com/downloads/c118-cudnn-windows-8664-87084cuda11-archivezip> (reference date 17.01.2023)
11. Download OpenCV 4.5.1. – URL: <https://github.com/opencv/opencv/archive/4.5.1.zip> (reference date 17.01.2023)
12. Download opencv\_contrib v 4.5.1. – URL: <https://github.com/opencv/opencv_contrib/archive/refs/tags/4.5.1.zip> (reference date 17.01.2023)
13. All about cmd. – URL: <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/cmd> (reference date 17.01.2023)
14. Cannot run samples on MS Visual Studi 2019. – URL: <https://forums.developer.nvidia.com/t/cannot-run-samples-on-ms-visual-studi-2019/72472> (reference date 17.01.2023)