МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт информационных технологий |
| наименование института (факультета) |
| Кафедра математического и программного обеспечения ЭВМ |
| наименование кафедры |

ОТЧЁТ

по Производственной практике (Технологической практике 2)

|  |  |
| --- | --- |
| Листов | 10 |

|  |  |
| --- | --- |
| Студента | Микуцких Григория Андреевича |
| группы | 1ПИб-02-1оп-22, |

|  |
| --- |
| Место прохождения практики: |
| АО «Апатит» |

Студент \_\_\_\_\_\_\_\_\_\_\_

                 (подпись)

|  |  |
| --- | --- |
| Руководитель практики от кафедры | МПО ЭВМ |
| Сальникова О.С., к.филол.н., доцент | |
| Руководитель практики от предприятия, организации, учреждения  (Ф.И.О., должность) | |
| Руфанов Александр Олегович, ведущий специалист | |

2025 г.

Оглавление

[Введение 2](#_Toc203139570)

[1. Общая характеристика практики 3](#_Toc203139571)

[2. Организация производства 4](#_Toc203139572)

[3. Структура и функции подразделения базы практики 5](#_Toc203139573)

[4. Реализация индивидуального задания 6](#_Toc203139574)

[5. Мероприятия по охране труда и технике безопасности 8](#_Toc203139575)

[Заключение 9](#_Toc203139576)

[Список литературы 10](#_Toc203139577)

[Приложение 1. Текст программы 11](#_Toc203139578)

# Введение

В ходе прохождения технологической практики была поставлена цель: получение основных навыков на этапах обучения модели детектирования объектов на видеоряде.

Несмотря на широкое использование систем видеонаблюдения на производственных участках предприятия АО «Апатит», они не всегда обеспечивают своевременное выявление нарушений со стороны персонала, поскольку требуется постоянное участие человека, не способного анализировать происходящее в реальном времени.

Использование современных технологий компьютерного зрения и нейросетевых моделей детектирования персонала позволяет автоматизировать этот процесс. Такие модели способны в режиме реального времени определять перемещение сотрудников и фиксировать отклонения от регламентов, как отсутствие СИЗ или проникновение в запрещённые зоны.

Таким образом, интеграция подобных решений в производственную инфраструктуру АО «Апатит» может повысить уровень промышленной безопасности и улучшить контроль за соблюдением требований.

В качестве примера поставлено задание распознавания работников предприятия АО «Апатит» и определения экипировки на них (каска, защитные очки, спецодежда).

Были поставлены следующие задачи:

1. подготовить графические данные (сделать разметку) для обучения нейронной модели;
2. обучить модель на основе подготовленных данных;
3. использовать полученную модель на примере тестового видеоряда;
4. измерить и привести качество моделей в соответствие со специальными метриками (precision – точность, recall – полнота).

В качестве инструмента разметки выступит «Bounding Box» (ограничивающая рамка), подходящий для решения задач «object detection» (распознавания объектов).

# Общая характеристика практики

«ФосАгро» — крупнейший российский производитель фосфорных удобрений, входящий в топ-5 мировых поставщиков. Компания занимается добычей апатитового сырья и выпуском экологичных удобрений (аммофос, NPK), которые экспортируются в более сотни стран.

АО «Апатит» (химический кластер группы «ФосАгро» в Череповце) – крупнейший в Европе производитель фосфорсодержащих удобрений, фосфорной и серной кислот, а также один из лидеров в России по объёмам выпуска NPK-удобрений, аммиака и аммиачной селитры [1].

Предприятие на 80% обеспечивает себя электроэнергией собственной генерации, в том числе за счёт её выработки при утилизации отходящего тепла в производстве серной кислоты, занимая по этому показателю лидирующее положение в отрасли [1].

Помимо российского рынка, продукция АО «Апатит» поставляется в страны Западной Европы, Азии, Африки и Америки. На территории предприятия расположен промышленный речной порт, который используется для перевозки сырья и готовой продукции. Вагонный парк АО «Апатит» насчитывает около 9000 вагонов разных типов [1].

Производственные мощности предприятия позволяют выпускать более 7,5 млн тонн удобрений в год [1].

Управление развития решений искусственного интеллекта предприятия АО «Апатит» занимается разработкой решений и программного обеспечения для компании, которое использует искусственный интеллект, в частности, анализом числовых показателей производства, трендов, решают задачи видеоаналитики и генеративного интеллекта.

# Организация производства

АО «Апатит», входящее в состав группы «ФосАгро», представляет собой одно из крупнейших химических производств в России, специализирующееся на выпуске фосфорсодержащих и азотных удобрений. Технологический процесс включает полный цикл переработки апатитового концентрата, поступающего с Кольского полуострова, и синтеза готовой продукции, востребованной в аграрной отрасли по всему миру [2].

На первом этапе концентрат поступает на переработку, где из него выделяется фосфор в виде фосфорной кислоты. Далее кислота используется для производства минеральных удобрений различных марок, таких как MAP, DAP, NPK. Параллельно с этим в азотном сегменте производятся аммиак, аммиачная селитра, карбамид, серную кислоту и фторид алюминия. Процесс сопровождается высокой степенью автоматизации и контролируется через систему автоматизированного управления технологическими процессами [2].

Методы управления на предприятии основаны на интеграции цифровых систем, сертифицированных по международным стандартам качества, экологической и промышленной безопасности. Производственные участки оснащены системой АСУТП, которая обеспечивает мониторинг параметров в режиме реального времени, включая температуру, давление, химический состав и другие критические показатели.

Особое внимание уделяется охране окружающей среды. На предприятии действуют современные очистные сооружения, которые позволяют повторно использовать сточные воды после биохимической и химической очистки. Также реализован проект по строительству шламонакопителя фосфогипса с высокими экологическими стандартами. Все отходы контролируются и утилизируются в соответствии с законодательством РФ [2]

В управлении персоналом используются современные HR-инструменты. Внедрены программы обучения, аттестации, внутренней мотивации и охраны труда. На предприятии активно развиваются культура информационной безопасности, система поощрений и внутренняя коммуникация [2].

# Структура и функции подразделения базы практики

Практика проходила под руководством ведущего специалиста отдела «Управления развития решений искусственного интеллекта» (подразделение дирекции по информационным технологиям или «ДИТ»).

Отдел аналитики данных был сформирован в рамках стратегии цифровой трансформации группы ФосАгро и отвечает за применение ИИ‑технологий в повседневных и производственных процессах предприятия [3] В отделе работают аналитики данных, ML‑инженеры, специалисты по промпт‑инжинирингу, DevOps‑инженеры и специалисты по BI.

Ключевые функции отдела: сбор и предобработка данных с АСУТП, MES и ERP систем, построение отчётов и визуализаций, разработка и обучение моделей машинного обучения для предсказания отказов оборудования, выявления отклонений в процессах, оптимизации логистики и мониторинга состояния железнодорожного подвижного состава [4].

# Реализация индивидуального задания

В ходе прохождения технологической практики было поставлено задание распознавания работников предприятия АО «Апатит» и определения экипировки на них (каска, защитные очки, спецодежда).

Для выполнения подготовки данных использовалось web-приложение «CVAT» [5] (рис. 1).

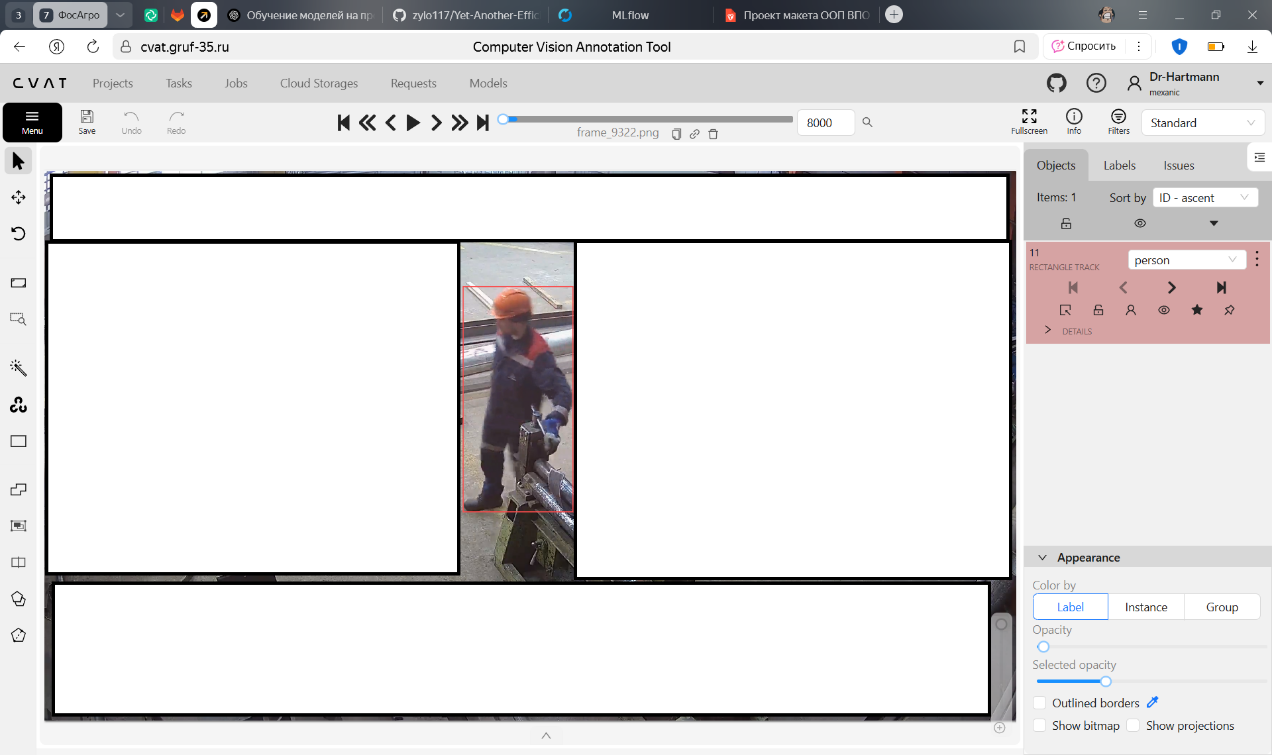


Рис. 1. Примерный интерфейс разметки «CVAT»

Для создания предварительной авторазметки данных в целях ускорения использовался проект на GitHub «AutoCVAT» [6] и ЯВУ «Python» [6].

Запись результатов экспериментов, используемых параметров и метрик проводилась с помощью пакета «MLflow» [8] (рис. 2).

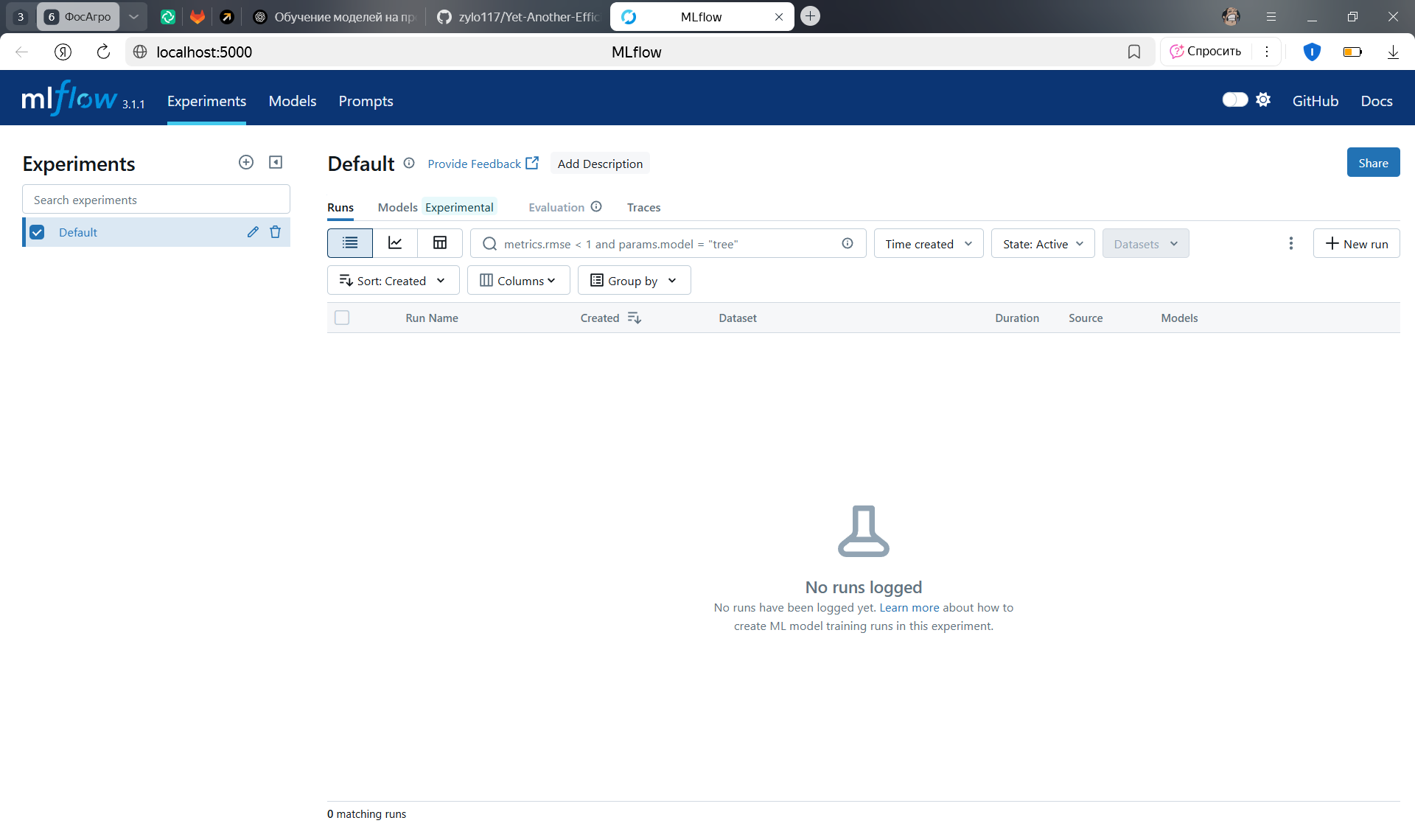


Рис. 2. Примерный интерфейс «MLflow UI»

Для обучения модели требовался персональный компьютер с дискретным графическим ускорителем (в частности, «Nvidia GeForce RTX 3050»).

Обучение происходило следующим образом:

1. экспортированы исходные графические данные (датасеты) с их разметкой (аннотациями);
2. для обучения использована модель типа «EfficientDet» с помощью фреймворка «PyTorch»;
3. написан скрипт, использующий данную модель;
4. обучена модель на основе более шести тысяч размеченных кадров.

Была проведена серия экспериментов с разными параметрами обучения, а для измерения качества обучения использовались метрики точности и полноты, которые составили следующее:

* optimizer=Adam, batch\_size=32, epochs=50, learning\_rate=0.001, точность=90%, полнота=92%;
* optimizer=SGD, batch\_size=64, epochs=40, learning\_rate=0.01, точность=88%, полнота=84%;
* optimizer=RMSprop, batch\_size=128, epochs=25, learning\_rate=0.005, точность=89%, полнота=80%;
* ptimizer=Adam, batch\_size=128, epochs=30, learning\_rate=0.01, точность=92%, полнота=80%;
* optimizer=Adam, batch\_size=64, epochs=40, learning\_rate=0.005, точность=94%, полнота=87%.

Для тестирования работы модели произведён инференс – процесс применения обученной модели к неизвестным для модели данным для получения предсказаний. Инференс производился на видеопотоке с предприятия.

Результаты показали достаточную точность детекции рабочих и их экипировки.

# Мероприятия по охране труда и технике безопасности

Отдел размещён в административно‑технических помещениях с современными инженерными системами, где создаются комфортный микроклимат и оптимальные условия труда [9].

Опасные и вредные производственные факторы:

* электромагнитные поля от серверов и оборудования;
* шум низкого уровня от систем охлаждения и кондиционирования;
* напряжения в электросети (рабочие места);
* повышенное время пребывания в статическом положении (офисная нагрузка).

Для снижения этих факторов осуществляются следующие мероприятия:

1. регулярные проверки заземления рабочих мест и серверных шкафов;
2. установка УЗО (устройств защиты от токов утечки) на все линии;
3. обучение сотрудников безопасной работе с техникой, включая инструктажи по безопасной эксплуатации розеток, ИБП и кабельных систем;
4. выделенная автоматизированная система пожарной сигнализации и оповещения, огнетушители и газовые/сплинкерные установки в серверных комнатах;
5. обучение сотрудников эвакуации и действиям при пожаре, регулярные тренировки и учения по эвакуации, согласно корпоративным стандартам ОТ и ПБ «Золотые правила», «Блокировка‑Маркировка‑Проверка»;
6. периодические медицинские осмотры, в том числе для офисных сотрудников, входят в корпоративный телемедицинский сервис «Телемед»;
7. организация лечебно‑профилактического питания (диета № 10 при необходимости) и проведение оздоровительных программ, а также доступ к психологической поддержке [10].

Был проведён вводный инструктаж, первичный инструктаж, инструктаж по технике безопасности и инструктаж антитеррористического поведения на предприятии. По причине высокого класса опасности предприятия было необходимо подписать документ о неразглашении внутренней информации.

# Заключение

В ходе прохождения технологической практики были освоены следующие компетенции:

* способен подготавливать данные для обучения;
* способен обучать модель типа «EfficientDet»;
* способен использовать модель в режиме реального времени.

Задачи разметки и обучения были выполнены в полной мере. Было размечено более 6000 кадров и обучена модель со следующими метриками предсказания:

* точность – 94%;
* полнота – 87%.

Также была определена предполагаемая тема дипломной работы «Система определения нарушения применения СИЗ на АО "Апатит"».

# Список литературы

1. АО «Апатит» (Череповец, Вологодская область) | Деятельность и продукция, адрес, телефоны – Электронный ресурс. – URL: «https://www.phosagro.ru/about/holding\_cherepovets/» (11.07.25).
2. Фосагро производство минеральных удобрений – Электронный ресурс. – URL: «https://concranes.ru/fosagro-proizvodstvo-mineralnyh-udobreniy/» (11.07.25).
3. Повседневные рабочие задачи поможет решить ИИ: корпоративный GPT-чат АО «Апатит» – Электронный ресурс. – URL: «https://up-pro.ru/library /information\_systems/automation\_management/korporativnyj-gpt-chat-ao-apatit/» (11.07.25).
4. Стратегия – Стратегический отчет – Интегрированный отчет ПАО «Фосагро» за 2024 г.– Электронный ресурс. – URL: «https://ar2024.phosagro.ru/strategic-report/strategy-2025» (11.07.25).
5. Leading Image & Video Data Annotation Platform | CVAT – Электронный ресурс. – URL: «https://www.cvat.ai/» (11.07.25).
6. BelickNicko/AutoCVAT: Use this project to automatically annotate your dataset for free in CVAT – Электронный ресурс. – URL: «https://github.com/BelickNicko/AutoCVAT » (11.07.25).
7. Welcome to Python.org – Электронный ресурс. – URL: «https://www.python.org/» (11.07.25).
8. MLflow: A Tool for Managing the Machine Learning Lifecycle | MLflow – Электронный ресурс. – URL: «https://mlflow.org/docs/latest/ml/» (11.07.25).
9. Безопасность производства – Обзор результатов – Интегрированный отчет ПАО «Фосагро» за 2023 г. – Электронный ресурс. – URL: «https://ar2023.phosagro.ru/operational-performance/industrial-safety» (11.07.25).
10. Безопасность производства – Обзор результатов – Интегрированный отчет ПАО «Фосагро» за 2024 г. – Электронный ресурс. – URL: «https://ar2024.phosagro.ru/perfomance-review/industrial-safety» (11.07.25).

# Приложение 1

Текст программы.

Файл «unpaker.py»

|  |
| --- |
| import os  import random  import shutil  import zipfile  import argparse  import xml.etree.ElementTree as ET  from tqdm import tqdm  from pathlib import Path  from natsort import natsorted  def get\_image\_paths(  path: Path | str,  extensions = (".jpg",".jpeg",".png",".bmp",".tif",".tiff",".webp")  ) -> list[Path]:  path = Path(path)  paths = [f for f in path.iterdir() if f.suffix.lower() in extensions]  if not paths:  raise FileExistsError(f"В {path} не найдено изображений!!!")  return paths  """Распаковать zip-архивы и объединить xml-аннотации в один файл"""  def unpack\_zip\_and\_join\_xml\_annotations(  arch\_path: Path | str,  out\_path: Path | str,  out\_img\_name: str,  out\_ann\_name: str,  ann\_name: str  ) -> None:  temp\_name = ".temp"  temp\_img\_path = get\_real\_path(temp\_name, out\_img\_name)  temp\_ann\_path = get\_real\_path(temp\_name, ann\_name)    out\_img\_path = get\_dpns\_path(out\_path, out\_img\_name)  os.makedirs(out\_path, exist\_ok=True)  os.makedirs(out\_img\_path, exist\_ok=True)  os.makedirs(get\_dpns\_path(out\_path, out\_ann\_name), exist\_ok=True)    root = ET.Element("annotations")  archives = os.listdir(arch\_path)    for arch in tqdm(archives, desc="Распаковка zip-архивов"):  if arch.endswith(".zip"):  arch\_file = get\_dpns\_path(arch\_path, arch)  with zipfile.ZipFile(arch\_file, 'r') as zip:  zip.extractall(temp\_name)  if os.path.exists(temp\_img\_path):  for file in os.listdir(temp\_img\_path):  src = get\_dpns\_path(temp\_img\_path, file)  dst = get\_dpns\_path(out\_img\_path, file)  shutil.move(src=src, dst=dst)    if os.path.exists(temp\_ann\_path):  root.extend(ET.parse(temp\_ann\_path).getroot())  shutil.rmtree(temp\_name, ignore\_errors=True)    tree = ET.ElementTree(root)  out\_ann\_path = get\_dpns\_path(out\_path, out\_ann\_name, ann\_name)  tree.write(out\_ann\_path, encoding="utf-8", xml\_declaration=True)  """Перемешать изображения на 'train', 'val', 'test'"""  def split\_images\_to\_train\_val\_test(  img\_path: Path | str,  train: str = "train",  val: str = "val",  test: str = "test"  ) -> None:  dataset = natsorted(get\_image\_paths(img\_path))  amt\_total = len(dataset)  amt\_train = int(amt\_total \* 0.7)  amt\_val = int(amt\_total \* 0.2)    def move\_files(files: list[Path], trg\_path: Path | str) -> None:  if os.path.exists(trg\_path):  shutil.rmtree(trg\_path)  os.makedirs(trg\_path, exist\_ok=True)    for file in tqdm(files, desc=f"Создание {trg\_path}"):  shutil.move(src=file, dst=trg\_path)      test\_dataset = dataset[amt\_train + amt\_val:]  move\_files(test\_dataset, get\_dpns\_path(img\_path, test))    dataset = dataset[: amt\_train + amt\_val]  random.shuffle(dataset)  train\_dataset = dataset[: amt\_train]  val\_dataset = dataset[amt\_train : amt\_train + amt\_val]    move\_files(train\_dataset, get\_dpns\_path(img\_path, train))  move\_files(val\_dataset, get\_dpns\_path(img\_path, val))  def split\_annotations\_to\_train\_val\_test(  img\_path: Path | str,  ann\_path: Path | str,  ann\_file: Path | str,  train: str = "train",  val: str = "val",  test: str = "test"  ) -> None:  train\_dataset = os.listdir(get\_dpns\_path(img\_path, train))  val\_dataset = os.listdir(get\_dpns\_path(img\_path, val))  test\_dataset = os.listdir(get\_dpns\_path(img\_path, test))    tree = ET.parse(ann\_file)  root = tree.getroot()    new\_roots = {  train: ET.Element("annotations"),  val: ET.Element("annotations"),  test: ET.Element("annotations")  }    dic = {  train: train\_dataset,  val: val\_dataset,  test: test\_dataset  }    if (labels := root.find(".//labels")) is not None:  [new\_roots[path].extend(labels) for path in new\_roots]    for image in root.findall("image"):  for key in dic:  if image.attrib["name"] in dic[key]:  new\_roots[key].append(image)  break    for path in new\_roots:  ET.ElementTree(new\_roots[path]).write(  get\_dpns\_path(ann\_path, path + ".xml"),  encoding="utf-8",  xml\_declaration=True  )  # TODO - сранить производительность с обычным list.append  """Получить список классов из xml-аннотаций"""  def get\_classes(  ann\_path: Path,  ) -> list[str]:  root = ET.parse(ann\_path).getroot()  tree = root.findall(".//box")  boxes = [box.get("label") for box in tree]  return list(lb for lb in set(boxes) if lb)  def get\_real\_path(\*path: Path | str) -> Path:  return Path(f"{os.getcwd()}/{"/".join([str(p) for p in path])}")  def get\_dpns\_path(\*path: Path | str) -> Path:  return Path("/".join([str(p) for p in path]))  if \_\_name\_\_ == '\_\_main\_\_':  out\_path = "dataset"  out\_img\_name = "images"  out\_ann\_name = "annotations"    parser = argparse.ArgumentParser()  parser.add\_argument("--archpath", type=str)  parser.add\_argument("--annfile", type=str)  args = parser.parse\_args()    if args.archpath is None:  print(f"Не передан путь к архиву, по умолчанию используется 'archives'")  args.archpath = "archives"  if args.annfile is None:  print(f"Не передано имя для аннотаций, по умолчанию используется 'annotations.xml'")  args.annfile = "annotations.xml"  unpack\_zip\_and\_join\_xml\_annotations(  get\_real\_path(args.archpath),  get\_real\_path(out\_path),  out\_img\_name,  out\_ann\_name,  args.annfile  )  split\_images\_to\_train\_val\_test(get\_real\_path(out\_path, out\_img\_name))  split\_annotations\_to\_train\_val\_test(  get\_real\_path(out\_path, out\_img\_name),  get\_real\_path(out\_path, out\_ann\_name),  get\_real\_path(out\_path, out\_ann\_name, args.annfile)  )  print(f"Классы:\n{get\_classes(get\_real\_path(out\_path, out\_ann\_name, args.annfile))}") |

Файл «inference.py»

|  |
| --- |
| import os  import cv2  import torch  import hashlib  import datetime  from torch import nn  from tqdm import tqdm  from pathlib import Path  from ultralytics import YOLO  from natsort import natsorted  from dataclasses import dataclass  # TODO - что это  torch.backends.cudnn.benchmark = True  CURRENT\_DIR = os.getcwd()  # TODO - сделать заменяемые модели, get\_YOLO, get\_fasterCNN ... pred\_func()...  # TODO - сделать функцию, ведущую обработку для разных типов моделей  # from torchvision.utils import draw\_bounding\_boxes  def get\_image\_paths(  path: Path,  extensions = (".jpg",".jpeg",".png",".bmp",".tif",".tiff",".webp")  ) -> list[Path]:  paths = [f for f in path.iterdir() if f.suffix.lower() in extensions]  if not paths:  raise FileExistsError(f"В {path} не найдено изображений!!!")  return paths  def get\_bb\_frame(path: Path, model: nn.Module | YOLO, score\_thresh: float) -> cv2.typing.MatLike:  frm = cv2.imread(str(path))  if frm is None:  raise FileNotFoundError(f"{frm} не найдено.")  # model.to(cfg.device)  # model.eval()    # tensor = transforms.ToTensor()(cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)).to(cfg.device)  # pred = model([tensor])[0]  # boxes = pred["boxes"].cpu().numpy()  # labels = pred["labels"].cpu().numpy()  # scores = pred["scores"].cpu().numpy()  # for box, label, score in zip(boxes, labels, scores):  # if float(score) < cfg.score\_thresh:  # continue  # if str(label) != "person":  # continue    if isinstance(model, YOLO):  pred = model(frm, verbose=False)[0]  for box in pred.boxes:  cls\_id = int(box.cls[0])  cls\_name = pred.names[cls\_id]  conf = float(box.conf[0])  if cls\_name != "person" or conf < score\_thresh:  continue    x1, y1, x2, y2 = map(int, box.xyxy[0])  cv2.rectangle(frm, (x1, y1), (x2, y2), (0, 255, 0), 2)  # cv2.putText(bb\_frm, f"{label} {score:.2f}", (x1, y1 - 10),  cv2.putText(frm, f"{cls\_name} {conf:.2f}", (x1, y1 - 10),  cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (0, 255, 0), 2)    return frm  def get\_video\_writer(  save: bool,  paths: list[Path],  trgt\_vid\_h: int | None,  codec: str = "FFV1",  fps: int = 2,  prefix: str = "VID",  ext: str = "mkv"  ) -> tuple[cv2.VideoWriter | None, tuple[int, int]]:  frm = cv2.imread(str(paths[0]))  if frm is None:  raise FileNotFoundError(f"{frm} не найдено.")    (h, w) = frm.shape[:2]  if trgt\_vid\_h is not None:  (h, w) = int(w \* trgt\_vid\_h / h), trgt\_vid\_h    writer = None  if save:  now = datetime.datetime.now().strftime("%Y%m%d\_%H%M%S")  raw = str(datetime.datetime.now().timestamp()).encode()  hsh = hashlib.md5(raw).hexdigest()[:6]  out\_path = Path(f"{CURRENT\_DIR}/{prefix}\_{now}\_{hsh}.{ext}")  writer = cv2.VideoWriter(str(out\_path), cv2.VideoWriter.fourcc(\*codec), fps, (h, w))  return (writer, (h, w))  def write\_video(writer: cv2.VideoWriter | None, frm, out\_size: tuple[int, int]) -> None:  if writer is not None:  writer.write(cv2.resize(frm, out\_size, interpolation=cv2.INTER\_AREA))  def stop\_video\_writing(wtr: cv2.VideoWriter | None) -> None:  if wtr is not None:  wtr.release()  def show\_window(show: bool) -> None:  if show:  cv2.namedWindow("Inference", cv2.WINDOW\_NORMAL | cv2.WINDOW\_KEEPRATIO)  cv2.setWindowProperty("Inference", cv2.WND\_PROP\_FULLSCREEN, cv2.WINDOW\_FULLSCREEN)  def show\_frame(show\_window: bool, path: Path, frm: cv2.typing.MatLike | None) -> None:  if frm is None:  raise FileNotFoundError(f"{frm} не найдено.")  if show\_window:  cv2.putText(frm, os.path.basename(path), (20, 40),  cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE\_AA)  cv2.imshow("Inference", frm)  def check\_break() -> bool:  return cv2.waitKey(1) & 0xFF == 27 # Esc = выход  def destroyAllWindows(show\_window) -> None:  if show\_window:  cv2.destroyAllWindows()      @dataclass  class Config:  images\_path: Path = Path(f"{CURRENT\_DIR}/dataset/images/test/")  fps: int = 20  codec: str = "FFV1"  score\_thresh: float = 0.6  show\_window: bool = True  save\_video: bool = True  target\_video\_heigth: int | None = None  device: str = "cuda" if torch.cuda.is\_available() else "cpu"  # model: nn.Module = torchvision.models.detection.fasterrcnn\_resnet50\_fpn(weights="COCO\_V1")  model: YOLO = YOLO("yolov8n.pt")  if \_\_name\_\_ == "\_\_main\_\_":  ut\_dataset\_path = "dataset"  out\_img\_name = "images"  out\_ann\_name = "annotations"    cfg = Config(  model=YOLO("best.pt"),  target\_video\_heigth = 720,  fps=10,  device="cuda",  show\_window=True  )    paths = natsorted(get\_image\_paths(cfg.images\_path))  writer, out\_size = get\_video\_writer(  save=cfg.save\_video,  paths=paths,  trgt\_vid\_h=cfg.target\_video\_heigth,  codec=cfg.codec,  fps=cfg.fps  )  show\_window(cfg.show\_window)  for path in tqdm(paths, desc="Прогресс инференса"):  bb\_frm = get\_bb\_frame(path, cfg.model, cfg.score\_thresh)  write\_video(writer, bb\_frm, out\_size)  show\_frame(cfg.show\_window, path, bb\_frm)  if check\_break():  break  stop\_video\_writing(writer)  destroyAllWindows(cfg.show\_window) |

Файл «convert\_to\_yolo.py»

|  |
| --- |
| import os  import xml.etree.ElementTree as ET  # словарь классов (твои 12 классов)  classes = ['nothing','helmet','overalls','helmet\_overalls','glasses',  'glasses\_helmet','glasses\_overalls','glasses\_helmet\_overalls',  'back','back\_helmet','back\_overalls','back\_helmet\_overalls']  def convert\_bbox(size, box):  """Преобразует CVAT bbox в YOLO формат (x\_center, y\_center, w, h)"""  dw = 1.0 / size[0]  dh = 1.0 / size[1]  x\_center = ((box[0] + box[2]) / 2.0) \* dw  y\_center = ((box[1] + box[3]) / 2.0) \* dh  w = (box[2] - box[0]) \* dw  h = (box[3] - box[1]) \* dh  return (x\_center, y\_center, w, h)  def convert\_cvat\_xml(xml\_file, images\_dir):  if not os.path.exists(images\_dir):  os.makedirs(images\_dir)  tree = ET.parse(xml\_file)  root = tree.getroot()  for image\_tag in root.findall("image"):  file\_name = image\_tag.attrib["name"]  img\_w = int(image\_tag.attrib["width"])  img\_h = int(image\_tag.attrib["height"])  txt\_name = os.path.splitext(file\_name)[0] + ".txt"  txt\_path = os.path.join(images\_dir, txt\_name)  with open(txt\_path, "w") as out\_file:  for box in image\_tag.findall("box"):  list\_name = [att.get("name") for att in box.findall("attribute") if att.text == "true"]  list\_name = [n for n in list\_name if n is not None]  cls = "\_".join(sorted(list\_name))  if cls not in classes:  continue  cls\_id = classes.index(cls)  xtl = float(box.attrib["xtl"])  ytl = float(box.attrib["ytl"])  xbr = float(box.attrib["xbr"])  ybr = float(box.attrib["ybr"])  bb = convert\_bbox((img\_w, img\_h), (xtl, ytl, xbr, ybr))  out\_file.write(f"{cls\_id} {' '.join([str(round(a,6)) for a in bb])}\n")  print(f"Аннотации {xml\_file} сконвертированы в {images\_dir}")  # --- Использование ---  splits = ["train", "val", "test"]  for split in splits:  xml\_file = os.getcwd() + f"\\dataset\\annotations\\{split}.xml"  images\_dir = os.getcwd() + f"\\dataset\\labels\\{split}"  convert\_cvat\_xml(xml\_file, images\_dir) |

Файл «test\_phosagro\_classes.ipynb»

|  |
| --- |
| !pip install torch torchvision opencv-python tqdm natsort  # TODO - разобраться с пакетами  # !python.exe -m pip install --upgrade pip  # !python.exe -m pip install pipreqs  # !pipreqs . --force  # !pip freeze > file.txt  in\_arch\_path = "archives"  out\_ann\_file = "annotations.xml"  # распаковка архивов  %run ./scripts/unpacker.py --archpath $in\_arch\_path --annfile $out\_ann\_file  %run ./convert\_to\_yolo.py  import os  import torch  import itertools  import xml.etree.ElementTree as ET  from PIL import Image  from torch.utils.data import Dataset, DataLoader  import torchvision  from torchvision import transforms  class CVATDataset(Dataset):  def \_\_init\_\_(  self,  images\_path: str,  ann\_path: str,  cost: dict[str, int] = {  'nothing': 0,  'helmet': 1,  'overalls': 2,  'glasses': 4,  'back': 8  },  transform: transforms.Compose | None = None  ):  self.img\_path = images\_path  self.transforms = transform  self.ann\_path = ann\_path  tree = ET.parse(self.ann\_path)  root = tree.getroot()    attr = [n.text for n in root.findall(".//label//attribute/name")  if n is not None and n.text]  attr = list(dict.fromkeys(attr))    all\_combo: list[list[str]] = []  if not attr:  all\_combo = [["nothing"]]  else:  for combo in itertools.product(("False", "True"), repeat=len(attr)):  values = dict(zip(attr, combo))  if values.get("glasses") == "True" and values.get("back") == "True":  continue  combo = [k for k, v in values.items() if v == "True"]  all\_combo.append(combo if combo else ["nothing"])    hash\_map = {combo: sum(cost[p] for p in combo.split("\_")) for combo in ["\_".join(sorted(i)) for i in all\_combo]}    self.label\_map = {  k: (v - min\_v) / (max\_v - min\_v) for k, v in hash\_map.items()  } if (max\_v := max(hash\_map.values())) > (min\_v := min(hash\_map.values())) else {  k: 0.0 for k in hash\_map  }  self.samples = []  for img in root.findall("image"):  name = img.attrib["name"]  width, height = int(img.attrib["width"]), int(img.attrib["height"])  boxes, labels = [], []  for box in img.findall("box"):  xtl, ytl = float(box.attrib["xtl"]), float(box.attrib["ytl"])  xbr, ybr = float(box.attrib["xbr"]), float(box.attrib["ybr"])  boxes.append([xtl, ytl, xbr, ybr])    list\_name = [att.get("name") for att in box.findall("attribute") if att.text == "true"]  list\_name = [n for n in list\_name if n is not None]  labels.append(self.label\_map["\_".join(sorted(list\_name))]) ###  self.samples.append({  "name": name,  "width": width,  "height": height,  "boxes": boxes,  "labels": labels  })    def \_\_len\_\_(self):  return len(self.samples)  def \_\_getitem\_\_(self, idx):  sample = self.samples[idx]  img = Image.open(os.path.join(self.img\_path, sample["name"])).convert("RGB")  boxes = torch.tensor(sample["boxes"], dtype=torch.float32)  labels = torch.tensor(sample["labels"], dtype=torch.float32)  # image\_id = torch.tensor([idx])  area = (boxes[:, 2] - boxes[:, 0]) \* (boxes[:, 3] - boxes[:, 1])  iscrowd = torch.zeros((len(boxes),), dtype=torch.int64)  target = {  "boxes": boxes,  "labels": labels,  # "image\_id": image\_id,  "area": area,  "iscrowd": iscrowd  }  if self.transforms:  img = self.transforms(img)  return img, target  train\_dataset = CVATDataset(  images\_path="dataset/images/test",  ann\_path="dataset/annotations/test.xml",  transform = transforms.Compose([  # transforms.Resize((28, 28)),  transforms.RandomRotation(90),  transforms.ToTensor(),  transforms.Normalize((0.1307,), (0.3081,))  ])  )  train\_loader = DataLoader(  train\_dataset,  batch\_size=4,  shuffle=True,  # shuffle=False,  collate\_fn=lambda batch: tuple(zip(\*batch)), # важно для detection моделей  )  keys = {v: k for k, v in train\_dataset.label\_map.items()}  # model: YOLO = cfg.model  # in\_features = model.roi\_heads.box\_predictor.cls\_score.in\_features  def get\_label\_by\_pred(values: list[float], mapping: dict[str, float]) -> list[str]:  return [min(mapping, key=lambda k: abs(mapping[k] - x)) for x in values]  from pprint import pprint  for images, targets in train\_loader:  pprint(get\_label\_by\_pred(targets[3]["labels"], train\_dataset.label\_map))  pprint(targets[3])  # pprint(train\_dataset.label\_map)  pprint(keys)  break  from ultralytics import YOLO  model = YOLO('yolov8n.pt')  results = model.train(  data="data.yaml", # путь к yaml  epochs=10, # количество эпох  imgsz=640, # размер изображения  batch=4, # размер батча  workers=2, # количество потоков  device=0 # GPU (0), CPU ("cpu"), или несколько "0,1"  )  # # Training.  # results = model.train(  # data='data.yaml',  # imgsz=1280,  # epochs=50,  # batch=8,  # name='yolov8n\_v8\_50e'  # )  # model = YOLO("runs/detect/train3/weights/best.pt")  # results = model.predict(source="dataset/images/test", save=True)  import cv2  import glob  import os  from ultralytics import YOLO  # Загружаем обученную модель  model = YOLO("runs/detect/train3/weights/best.pt")  # Папка с тестовыми изображениями  image\_folder = "dataset/images/test"  image\_paths = sorted(glob.glob(os.path.join(image\_folder, "\*.\*")))  # Проверяем первое изображение для параметров видео  first\_frame = cv2.imread(image\_paths[0])  if first\_frame is None:  raise KeyError()  h, w = first\_frame.shape[:2]  # Настройка видеозаписи (mp4, 30 FPS)  out = cv2.VideoWriter(  "output.mp4", cv2.VideoWriter.fourcc(\*"mp4v"), 30, (w, h)  )  # Открываем окно на полный экран  cv2.namedWindow("YOLOv8 Real-Time Detection", cv2.WND\_PROP\_FULLSCREEN)  cv2.setWindowProperty("YOLOv8 Real-Time Detection", cv2.WND\_PROP\_FULLSCREEN, cv2.WINDOW\_FULLSCREEN)  for img\_path in image\_paths:  frame = cv2.imread(img\_path)  if frame is not None:  results = model.predict(frame, conf=0.5, verbose=False)  annotated\_frame = results[0].plot()  # Записываем кадр в видео  out.write(annotated\_frame)  # Показываем на экране  cv2.imshow("YOLOv8 Real-Time Detection", annotated\_frame)  # Ждём 30 мс между кадрами (ESC или Q для выхода)  key = cv2.waitKey(30) & 0xFF  if key == 27 or key == ord("q"):  break  out.release()  cv2.destroyAllWindows()  # инференс  %run ./scripts/inference.py |