Журнал итераций

Бароцкий Глеб Алексеевич

Группа: ПШ-310101

В рамках создания нейронной сети для анализа маркировки нам поручено вести журнал итерации, в котором мы документируем каждый этап процесса разработки. Этот журнал поможет отслеживать прогресс, выявлять возможные проблемы и улучшать нашу работу по мере продвижения.

Журнал я буду вести в виде таблицы, где в одном блоке будет дата, а в другом то, что я сделал (моя работа)

|  |  |
| --- | --- |
| 9 Апреля 2024г. | На консультации нам было сказано, что нам необходимо будет сделать нейронную сеть (далее НС) для анализа маркировки, для нее можно использовать выбранный нами язык программирования (далее ЯП), а также любую среду разработку (далее IDE) |
| 10 Апреля 2024г | У меня уже был установлен «Pycharm», т.к. в рамках обучения у нас был курс по «Python», установлен сам «Python» версии 3.9, а также есть Visual studio (Далее VS), т.к. также там делали проекты по «Python», «C++», и сам я программирую на этом ЯП. |
| 14-19 апреля 2024г. | Нашел различные видео-гайды по НС, сидел, разбирался с ними, пытался сам делать вместе с авторами видео задания в «Pycharm». Затем решил посмотреть курсы на «Степике», наткнулся на данный курс: <https://clck.ru/3BBpga>, но он оказался недешевым, поэтому я отложил его. Поэтому я вернулся к видео на ютубе, наткнулся на вот такой плейлист: <https://clck.ru/3BBpqc>. В целом, видео были интересные, можно многому научиться |
| 22 апреля-7мая 2024г. | Решил узнать, какие еще есть курсы по НС, почитать про библиотеки, нашел для себя статьи на «Хабр»: <https://habr.com/ru/articles/765176/>, <https://habr.com/ru/articles/312450/>. Дальше нашел курсы: <https://stepik.org/course/50352/promo>,  <https://edu.sirius.online/course/aiintro>,  <https://journal.tinkoff.ru/pro/ai/?internal_source=tj_uch_uchebnik.ai_how-to-chat-with-ai>;  Это были курсы по нейронным сетям, отдельно решил пройти курсы по «Python», хоть и программирую на нем немного, но явно моих знаний точно не хватает, нужно их дополнять. Выбрал следующие курсы:<https://practicum.yandex.ru/python-free/?from=catalog>,  <https://practicum.yandex.ru/data-analysis-basic/?from=catalog>,  <https://stepik.org/course/100707/promo?search=4374640843>,  <https://stepik.org/course/81846/promo?search=4374640852>; |
| 21 мая 2024 г. | Был получен файл с изображениями, в нем находится 2906 фотографий, а также видео, из которого впоследствии также будут нарезаны изображения для разметки |
| 22 мая 2024 г. | Решил просмотреть видео, чтобы нарезать хорошие четкие фото, это было выполнено, после недолгого перерыва по планам было изучение изображений маркировки, с уже имеющимися названиями файлов, больше всего фотографий было ММК, в основном это были ММК 10Г2ФБЮ, ММК 17Г1С-У, фотографий от других поставщиков было меньше, есть поставщик, где маркировка находится в верху листа, есть тот, где маркировка занимает почти весь лист, исходя из фотографий. |
| 23 мая 2024 г. | Далее я поставил задачу разметить изображения, в качестве платформы я решил выбирать между Makesense: <https://www.makesense.ai/>,VGG Image Annotator: <https://annotation-tool.github.io/>,  Google Colab: [https: //colab.research.google.com/](https://colab.research.google.com/). Посмотрев и изучив каждый из предложенных вариантов, я остановился на Makesense, т.к. у него простой и удобной интерфейс, сразу понятно, как экспортировать и импортировать файлы, а также делать саму разметку |
| 29 мая 2024г. | Решил сделать разметку из проанализированных фотографий из файла, а также из некоторых изображений, нарезанных из видео, решил делать разметку полигонами, она выдает пять координат, я решил выделять производителя (в случае ММК 1-ая строка, а также 2-ую строку). Пример разметки представлен на рисунке 1:    Рисунок 1  Далее я разметил так в общей сложности 678 изображений, т.к. на них можно было как-то выделить границы двух строк, а также задать, что там написано, затем я экспортировал это в json-формат. Работа была продолжительной, но спустя пару сотен фотографий уже привык делать разметку, начал делать быстрее. Результаты представлены на рисунке 2 и 3:    Рисунок 2    Рисунок 3    Рисунок 4  На рисунке 2-3 представлено, как экспортировать файл в json-формат, а на рисунке 4 его содержимое (на рисунке оно представлено не полностью) |
| 4 июня | Была проведена встреча с ML-специалистом, на которой я задавал вопросы, касаемо того, правильно ли я начал делать разметку, в каком направлении мы двигаемся. Я получил ответ, что я делал разметку, в целом, верно, но важной частью является третья строка. Я спросил, что я могу просто вбить каждый номер (до этого маркировку было делать легче, т.к. первая и вторая строки могли повторяться), а вот третья, как было сказано на данной встрече, нет. Специалист посчитал, что это будет дольше по времени, предложил пойти другим путем, а затем показал, что значит сама маркировка:  **Уральская Сталь:**  cid:image001.jpg@01D9BA68.D7535C70  4801190119 – номер заказа между УЗСР ООО «ТМК ТР» и АО «Уральская Сталь»  2 – номер позиции заказа  Z31397 – номер плавки  3402 – номер партии в плавке  4 – номер листа  К56-2 – класс прочности или марка стали, в зависимости от заказа  С.41 – углеродный эквивалент Сэкв  Р.21 – параметр стойкости против растрескивания при сварке Pcm  11 – толщина листа, мм  2518 – ширина листа, мм  12020 – длина листа, мм  **ММК:**  cid:image002.jpg@01D9BA68.D7535C70  ММК – наименование поставщика  10Г2ФБЮ – марка стали или класс прочности, в зависимости от заказа  2023 – год плавки  В112021 – номер плавки  100 – номер партии в плавке  19080 – номер раската  1001 – номер листа (номер раската и номер листа в раскате образуют итоговый номер листа, т.е. фактический номер листа для задачи в производство 19080-1001)  12,0х3134х12040 – толщина х ширина х длина листа, мм  **Северсталь:**  cid:image003.jpg@01D9BA68.D7535C70  ПАО «Северсталь» - поставщик листа  ЛПЦ-3 – номер цеха поставщика листа  0313402612 – внутренний номер заказа в ПАО «Северсталь»  75/3-2111 – номер заказа или спецификации между УЗСР ООО «ТМК ТР» и ПАО «Северсталь»  643/1000-22101265 – внешний номер договора между ПАО «ТМК» и ПАО «Северсталь»  К56 – класс прочности или марка стали, в зависимости от заказа  333452 – номер плавки  6705301 – номер партии в плавке  96622 – номер раската  2 – номер листа в раскате (номер раската и номер листа в раскате образуют итоговый номер листа, т.е. фактический номер листа для задачи в производство 96622-2)  14,0х3131х12020мм – толщина х ширина х длина листа, мм  4150 – теоретический вес листа, кг  ООО «ТМК Трубопроводные решения» - грузополучатель листа  СЭКВ=0,35 – углеродный эквивалент Сэкв  PCM=0,19 – параметр стойкости против растрескивания при сварке Pcm  01.06.2023 – дата производства листа  Мне сказали, чтобы я продолжил работу в Makesense, но выделял прямоугольником, а не полигонами, а затем уже работал с ним. Добавили, что можно выделять три первые строки, а сама третья строка является важной. Далее мы получили архив с сегодняшней консультации, где была также презентация с шагами работы. Специалист сказал мне, что моя работа не была сделана просто так, потому что я получил опыт, а также, если захочу, смогу сравнить результаты между разными способами разметки. |
| 5 июня | Начал делать разметку, т.к. нам порекомендовали прямоугольниками, выделил 574 изображения, которые нормально помещались в прямоугольник, и на которых можно было прочитать то, что написано, таким образом я размечал прямоугольниками, времени ушло намного меньше, чем я делал полигонами, т.к. можно было сделать одно название для всей разметки, а также прямоугольником выделять быстрее, чем полигонами:    Рисунок 5  Затем я экспортировал их, получилось следующие файлы в формате .txt. В текстовых файлах формат аннотаций YOLO (разметка прямоугольником MakeSense):    Рисунок 6  Каждый файл содержит пять цифр, первая указывает на то, есть ли маркировка (1-да, 0-нет), остальные четыре цифры координаты прямоугольника:    Рисунок 7  Далее я создал папку datasets, в ней создал три папки:    Рисунок 8  Я назвал их valid,train(папки валидации и обучающий модели), содержимое в них одинаковое:    Рисунок 9  В папке images представлены размеченные прямоугольником изображения в формате .jpg:    Рисунок 10  В папке labels содержатся файлы формата .txt размеченных изображений из папки images, как выглядят эти файлы представлены на рисунке 6.  Также в datasets существует папка vse\_ostalnyie, где просто содержатся все изображения, которые у нас были:    Рисунок 11 |
| 6 июня | Далее была найдена статья, с ее помощью написал код в «Pycharm» для обучения модельки: <https://docs.ultralytics.com/ru/quickstart/#use-ultralytics-with-python>. Так же нам из материалов встречи от 4 июня был предоставлен текстовый файл. Сам код выглядит следующим образом:  Рисунок 12  Здесь указаны пути к файлам, а также классы.  Далее я решил обновить версию питона с 3.9 до 3.12, т.к. это более новая версия, в File-Settings-Python Interpreter изменил версию питона на 3.12. Начал писать код, используя материал из статьи, которую указал выше.  Сам код с комментариями выглядит следующим образом:    Рисунок 13  Файл datasets.yaml, здесь указаны пути к файлам (картинки), а также представлены классы, где 1- есть маркировка (marking), 0-ее нет (empty):    Рисунок 14  Сначала возникли проблемы с путем к файлу datasets, т.к. я его указывал из  «Pycharm», а нужно было с самого компьютера.  Само дерево проекта, где я перенес папку datasets с ее содержимым в виртуальное окружение Python (.venv), а также создал файл datasets.yaml выглядит следующим образом:    Рисунок 15  Сначала мне выдавало ошибку, но это было связано с тем, что в начале я использую:  from ultralytics import YOLO  # Создание модели с использованием конфигурационного файла model = YOLO("yolov8n.yaml")  Но у меня не была установлена библиотека ultralytics, установил я ее через pip:    Рисунок 16  Далее программа запускалась, но не хотела мне что-то выдавать, погуглив, я понял, что у меня было написано изначально epochs=1, затем я изменил на epochs=10. "Эпоха" (epoch) в машинном обучении обозначает один проход через все обучающие данные в модели машинного обучения.  Затем модель начала обучаться:    Рисунок 17  Спустя 1ч 43 мин. Модель обучилась, выдала такой результат:    Рисунок 18  Затем в дереве создался каталог runs, этот каталог обычно содержит подкаталоги для каждого запуска обучения, каждый из которых может содержать результаты, модели, журналы и другую информацию, связанную с конкретным запуском, в нем находится подкаталог detect, в котором есть train, train2, … train7, train8. "Train" в контексте обучения нейронной сети обычно означает процесс обучения модели на тренировочных данных:    Рисунок 19  В каждом из этих train содержится подкаталог «weights» («Веса»). Веса модели - это настройки, которые определяют, как модель принимает входные данные и делает предсказания:    Рисунок 20  В train8-weights находятся два файла: best.pt и last.pt.  "Best" обычно относится к лучшей модели, которая достигла наилучших результатов по метрикам валидации.  "Last" относится к последней сохраненной модели в конце обучения:    Рисунок 21  Далее я решил написать скрипт для того, чтобы проверить, что файлы в images и labels совпадали, полный код будет представлен в Приложении 1, вот фрагмент кода с его результатами:    Рисунок 22    Рисунок 23  Далее я написал код, чтобы посмотреть результат, что мне моя моделька может вывести:    Рисунок 24  Она выводит следующее:    Рисунок 25  Класс empty: Обозначает, что на данном участке изображения нет маркировки.  Вероятность 0.64 (или 64%) означает, что модель уверена на 64% в том, что в этой области нет маркировки.  Думал, почему это происходит так. Оказалось, что через ноутбук и google у меня через makesense экспортируются файлы формата .txt, где в начале стоит 1 (маркировка есть), а вот с компьютера с Яндекс браузера получается 0. Пришлось писать скрипт, который изменяет содержимое файлов и меняет 0 на 1. Код представлен в приложении 2:    Рисунок 26  Далее я перезапустил модель, чтобы она заново обучалась, на это ушло около 1,5 часов, теперь она выдает такой результат:    Рисунок 27  marking 0.67" означает, что модель обнаружила на изображении маркировку и уверена в этом на 67%.  Код, который я использовал для обучения и для просмотра результата представлены в приложении 3 и 4: |
| 7 июня | В ТЗ было сказано, что можно использовать jupyter lab. Решил использовать его, потому что наткнулся на статью: <https://medium.com/@paul_lefevre/id-documents-detection-with-yolov8-plus-rotation-e991192e74d2>.  Затем я установил данную среду разработки. Для этого я зашел на официальный сайт: <https://jupyter.org/>. Установил pipы: pip install jupyterlab и pip install notebook. Затем вводим в консоль: jupyter lab и открываем нашу среду разработки, нажали на Notebook-Python 3:    Рисунок 28  Далее я выполнил следующие шаги:   1. Создание и активация виртуального окружения: cv\_env\Scripts\activate, python -m venv cv\_env , активируем через консоль. 2. Добавление нового ядра для окружения и установка библиотек: pip3 install ipykernel, python3 -m ipykernel install --user --name=cv\_env --display-name "Python 3.12 (cv\_env)", также активируем через консоль.   Термин "ядро" обычно относится к виртуальному окружению Python, которое используется для выполнения кода в ноутбуке. После этой команды виртуальное окружение cv\_env будет доступно в JupyterNotebook и мы можем выбрать его при создании или открытии ноутбука.   1. Установка зависимостей в созданное виртуальное окружение: pip3 install -r requirements.txt   В самом начале у меня есть папка datasets с папками test, train и valid, причем содержание train и valid - одинаковое.  В datasets/test - только картинки (без подпапок)  папка vse\_ostalnyie была изменена на test, где хранятся изображения, которых нет в train и valid.  В datasets/train (и аналогично datasets/valid) - подпапки images (с фотографиями в формате .jpg) и labels (с labels в формате .txt). В текстовых файлах формат аннотаций YOLO (разметка прямоугольником MakeSense): <object-class> <x\_center> <y\_center> <width> <height>    Рисунок 29  Далее я поменял название папки valid на val, потому что постоянно писал val, но приходилось еще дописывать id, чтобы не запутаться назвал ее все-таки val, потом установил еще оставшиеся pipы, вот их список:  pip install pyyaml  pip3 install pandas  pip3 install keras  pip3 install keras\_preprocessing  pip3 install scikit-learn  pip3 install opencv-python  pip3 install opencv-python-headless  pip3 install ultralytics pillow  pip3 install tqdm  pip3 install easyocr  pip3 install transformers  Затем я импортирую библиотеки:    Рисунок 30  Выдает: Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU. Это не является ошибкой, просто оно просто информирует вас о том, что PyTorch не нашел доступных графических ускорителей (GPU) и будет использовать CPU для всех операций. CPU(Central Processing Unit) — это центральный процессор.  Затем я написал RANDOM\_STATE = 42 — это параметр, часто используемый в машинном обучении и статистике для обеспечения воспроизводимости результатов при случайных процессах, таких как разбиение данных на обучающую и тестовую выборки, перемешивание данных или генерация случайных чисел. 42 просто выбрал в качестве числа.  Затем написал: «Чтение файлов. Создание csv с разметкой прямоугольниками. Создадим CSV файл с изображениями и их метками. Это позволит нам систематически проверить и проанализировать данные», затем указал Путь к папкам с обучающим и валидационным наборами (изменить при необходимости), а также создадим csv файлы для обучающего и валидационного набора:    Рисунок 31  После этого сравним два полученных csv:    Рисунок 32  Файлы совпадают, значит, будем использовать одну из папок и ее разбивать на обучающую (train) и валидационную выборки. Далее я все-таки решил объединить формат json и csv, потому что я решил сделать заново разметку полигонами, выделять на изображениях те строки, которые нам важны, т.е., производителя, номер плавки и номер листа, а затем сравнить, сколько у меня совпадают изображений, выделенных прямоугольником и маркировкой:   Рисунок 33  На рисунке 32 представлено объединение двух форматов:    Рисунок 34  data - объединение разметки всего текста прямоугольниками + разметка полигонами части текста + сама часть этого текста.  Далее я сделал разделение данных на train и val Разделим данные на обучающуюю и валидационную выборки (валидационная выборка = 25% от общего количества изображений).  Валидационная выборка используется для оценки качества модели на данных, которых не было в обучающем наборе. Она дает представление о том, как модель будет работать на новых данных.    Рисунок 35  Далее создал новые папки для хранения разделенных данных и скопировали файлы в соответствующе директории:    Рисунок 36  Вызываем функции, а затем выделяем тестовую выборку:    Рисунок 37  data - данные, содержащие информацию о файлах изображений и их разметке  Создаем папку для тестовых фотографий, а затем копируем изображения для тестовой выборки:    Рисунок 38  Чтобы улучшить детекцию текста, сначала приведем все изображения к черно-белой цветовой гамме:    Рисунок 39  Далее указываем путь к исходным изображениям и путь к папке для сохранения черно-белых изображений:    Рисунок 40 |
| 8 июня | Затем начал делать детекцию текста на фотографиях train, val, test. Воспользовался документацией: <https://github.com/ultralytics/ultralytics>.Потом нормализируем изображения и переведем значения пикселей из диапазона [0,255] в [0,1], чтобы модель воспринимала данные: Рисунок 41 Создание yaml (это язык разметки данных, который часто используется для конфигурационных файлов в различных приложениях.), а также удалил ненужные нам папки:   Рисунок 42  Далее мы определили структуру данных для yaml. Файл datasets.yaml, здесь указаны пути к файлам (картинки), а также представлены классы, где 1- есть маркировка (marking), 0-ее нет (empty):    Рисунок 43  Затем приступаем к обучению YOLOv8:    Рисунок 44  Загрузим предобученную COCO-pretrained YOLOv8n модель и дообучим модель на нашем датасете на 10 эпохах  "Эпоха" (epoch) в машинном обучении обозначает один проход через все обучающие данные в модели машинного обучения:    Рисунок 45  Затем модель начинает обучаться, сначала возникли проблемы с тем, что файл не детектит, пришлось заново пересоздавать такой же data.yaml файл, и менять к нему путь, это не помогло, вылезла ошибка, пришлось удалить папку data, заново пройтись по коду, проверяя каждую ячейку, оказалось, что у меня в yaml файле было: train:C:/Users/1/data/train  Val:C:/Users/1/data/train/val, а надо: train: C:/Users/1/data/train  val: C:/Users/1/data/val.    Рисунок 46  Модель обучилась спустя 24 минуты:    Рисунок 47  Box (P): 0.983 - Это Precision (точность), показывающая долю правильно предсказанных положительных примеров среди всех предсказанных положительных примеров.  R: 0.65 - Это Recall (полнота), показывающая долю правильно предсказанных положительных примеров среди всех реальных положительных примеров.  Затем я сохранил лучшую модель, она находилась по пути:  model = YOLO('runs/detect/train/weights/best.pt')  Запустим модель на примере, получаем:    Рисунок 48  Далее отложил работу, сохранил файл. |
| 10 июня | Запускаю jupyter через jupyter lab в консоли.  В программировании, я узнал, используют шрифт SpaceMono-Regular, решил его скачать, чтобы визуализировать результаты обнаружения объектов:    Рисунок 49  Потом я делал детекцию и обрезку всех фото train и val.  Но столкнулся с проблемой, что мне выводит, что фото не обнаружено:    Рисунок 50 |
| 11 июня | Пытался решить проблему вчера, но как-то не получилось, решил заново пройтись по коду и что-то изменить.  Постепенно буду прописывать все изменения. То, что изменил, добавил, буду вносить, что не трогал, то не буду.  Решил сразу записать директории к файлам, в том числе и сразу путь yaml файл с каталогом EasyOCR. Для EasyOCR нашел туториал: <https://jaided.ai/easyocr/tutorial/>    Рисунок 51  Далее я за комментировал все то, что было у меня связано с JSON файлом, т.к. я его и не использовал в предыдущий раз, посмотрим, что будет, если пока его не использовать:    Рисунок 52  Далее посмотрел, что из себя представляет разметка прямоугольником.  Датафрейм состоит из двух столбцов:   1. image, который содержит название jpg файла с расширением 2. label, который содержит непосредственно разметку (название файла label в формате txt совпадает с названием из столбца image)     Рисунок 53 Потом я все-таки решил заняться проверкой имеющейся текстовой разметки, т.е., поработать с моим JSON.   Рисунок 54 В задачах компьютерного зрения первая цифра в файлах аннотаций обычно представляет собой класс объекта. Иногда требуется изменить класс всех объектов на единый, например, класс 0, если мы хотим унифицировать или переопределить аннотации, поэтому делаем Корректировку rectangle\_class:   Рисунок 55  Оставил тоже самое, что было на рисунке 33: Далее я сделал разделение данных на train и val Разделим данные на обучающуюю и валидационную выборки (валидационная выборка = 20% от общего количества изображений).  Валидационная выборка используется для оценки качества модели на данных, которых не было в обучающем наборе. Она дает представление о том, как модель будет работать на новых данных.  Создал новые папки для хранения разделенных данных    Рисунок 56  Затем мы копируем файлы в новые, соответствующие директории:    Рисунок 57  Проверим количество файлов в каждой папке, чтобы количество совпадало в train, val и test:    Рисунок 58  Проверим, нет ли совпадений train, val с test:    Рисунок 59  Совпадений нет, идем дальше.  Дальше также оставляет предобработку изображений.  Чтобы улучшить детекцию текста, сначала приведем все изображения к черно-белой цветовой гамме:    Рисунок 60  Нормализацию оставляем, чтобы модель воспринимала данные:    Рисунок 61  Детекция текста на фотографиях train, val, test. Воспользовался статьей, которую уже приводил: <https://medium.com/@paul_lefevre/id-documents-detection-with-yolov8-plus-rotation-e991192e74d2>. Создание yaml yaml файл должен содержать описание **только** структуры папок train и val:    Рисунок 62  Загрузим предобученную COCO-pretrained YOLOv8n модель и дообучим модель на нашем датасете на 10 эпохах  "Эпоха" (epoch) в машинном обучении обозначает один проход через все обучающие данные в модели машинного обучения:  Рисунок 63  Далее модель обучается:    Рисунок 64  Модель обучилась спустя 1,5 часа:    Рисунок 65  Box (P): 0.974 - Это Precision (точность), показывающая долю правильно предсказанных положительных примеров среди всех предсказанных положительных примеров.  R: 0.982 - Это Recall (полнота), показывающая долю правильно предсказанных положительных примеров среди всех реальных положительных примеров.  model = YOLO('runs/detect/train2/weights/best.pt')  Запустим модель на примере, получаем:    Рисунок 66  Визуализировать результаты обнаружения объектов с использованием функции с помощью шрифта SpaceMono-Regular:    Рисунок 67  Потом я делал детекцию и обрезку всех фото train и val при помощи функций crop\_images\_by\_largest\_object(new\_val\_images\_path, new\_val\_images\_path) и  crop\_images\_by\_largest\_object(new\_train\_images\_path, new\_train\_images\_path) :    Рисунок 68  Затем рассчитал угол наклона текста:    Рисунок 69  Дальше занимаемся инверсией цветов. Инвертируем цвета (чтобы текст был черным, фон белым):    Рисунок 70 Далее приступил к работе OCR – EasyOCR (ru)Нашел примеры работ: <https://colab.research.google.com/github/vistec-AI/colab/blob/master/easyocr.ipynb#scrollTo=lIYdn1woOS1n>    И еще один пример:  [https://jaided.ai/easyocr/tutorial/](https://jaided.ai/easyocr/tutorial/%20) Предсказание на одном фото. Получаем:    Рисунок 71  Выводит следующее:    Рисунок 72 |
| 12 июня | Пробуем другую модель - OCR - trocr-base-ru (HuggingFace) Рисунок 73 Выводит слово «статистики», не понятно, откуда оно взялось: Рисунок 74 Дальше нашел OCR - trocr-base-printedПопробую исходник предыдущей модели от Microsoft: Рисунок 75 Теперь используем OCR – EasyOSR(ru+en) as it is на одном фото Получаем такой результат:    Рисунок 76  Дальше решил изменить предобработку, заново производим детекцию и обрезку:    Рисунок 77    Рисунок 78  Предсказание на одном фото после изменения предобработки  Отображаем новое изображение:    Рисунок 79  Выводит следующее:  Рисунок 80  Сохранение зависимостей:  ! pip freeze > requirements.txt  Модель сохранена. Подводя итог, можно выделить следующие шаги, которые необходимо сделать до того, как запустить код (он будет выслан отдельно в файле) |

Версия Python

Python 3.12.3

Создание и активация виртуального окружения: cv\_env\Scripts\activate, python -m venv cv\_env , активируем через консоль.

Добавление нового ядра для окружения и установка библиотек: pip3 install ipykernel, python3 -m ipykernel install --user --name=cv\_env --display-name "Python 3.12 (cv\_env)", также активируем через консоль.

После этой команды виртуальное окружение cv\_env будет доступно в Jupyter Notebook и мы можем выбрать его при создании или открытии ноутбука

Установка зависимостей в созданное виртуальное окружение

pip3 install -r requirements.txt

В самом начале у меня есть папка datasets с папками test, train и val, причем содержание train и val - одинаковое.

В datasets/test - только картинки (без подпапок)

В datasets/train (и аналогично datasets/val) - подпапки images (с фотографиями в формате .jpg) и labels (с labels в формате .txt). В текстовых файлах формат аннотаций YOLO (разметка прямоугольником MakeSense): <object-class> <x\_center> <y\_center> <width> <height>

Установка pip:

pip install pyyaml

pip3 install pandas

pip3 install keras

pip3 install keras\_preprocessing

pip3 install scikit-learn

pip3 install opencv-python

pip3 install opencv-python-headless

pip3 install ultralytics pillow

pip3 install tqdm

pip3 install easyocr

pip3 install transformers

Все зависимости сохранил в requirements.txt (будет выслан файл). И при последующих прогонах уже устанавливаю библиотеки через это файл.

Кратко:

1) все зависимости, которые получил в requirements.txt в конце кода, появились после того, как прогнал все пипы в терминале;

2) Прогнал код;

3) Сохранил все зависимости в файл requirements.txt;

Чтобы запустить модель, нужно:

1. Установить зависимости из уже готового requirements

pip install virtualenv  
 virtualenv venv  
 .\venv\Scripts\activate  
 pip install -r requirements.txt  
 pip install jupyter  
 jupyter notebook

2) Прогнать код;

OCR - Вывод

Мы попробовали несколько предобученных моделей ImageToText, которые считывают текст с фотографий:

EasyOCR (для русского языка)

trocr-base-ru (с сайта [HuggingFace](https://huggingface.co/raxtemur/trocr-base-ru) - это модель trocr-base-printed от Microsoft, но дообученная на русском языке)

Непосредственно саму модель trocr-base-printed

EasyOCR (русский + английский)

Качество моделей проверяли выборочно на определенных фотографий. В первом приближении этого достаточно, чтобы увидеть, выдает модель что-то вменяемое или нет.

После испытания четырех предобученных моделей EasyOCR (ru), trocr-base-ru, trocr-base-printed, EasyOCR (ru+en) с предобработкой из раздела "Предобработка изображений" было решено скорректировать предобработку, так как в большей степени именно она влияет на качество моделей. Предобработка была изменена и модель EasyOCR (ru+en) была проверена на новых изображениях (предобработанных по-новому). Модель показала неплохой результат для изображения "Труба №30760 Плавка №2023-В306000 Лист №190702000.jpg", но при этом результат не постоянен и очень зависит от качества исходного изображения.

Для более стабильной работы модели в будущем могут быть следующие действия:

1. Классификация изображений в зависимости от их качества и специфичная предобработка изображений с хорошим и плохим качеством. То есть фото, для которых предсказания были неплохие, предобрабатываем одним образом, а "плохие" фото предобрабатываем иначе.
2. Использование разных моделей для фотографий разного качества.
3. Дообучение модели на наших данных (требует тонкой настройки).

Обученная модель, сохраненная в виде файла - это best.pt

Качество решения - это precision/recall в случае модели yolo. Проверил качество модели на валидационной выборке

Box (P): 0.974 - Это Precision (точность), показывающая долю правильно предсказанных положительных примеров среди всех предсказанных положительных примеров.

R: 0.982 - Это Recall (полнота), показывающая долю правильно предсказанных положительных примеров среди всех реальных положительных примеров.

Для детекции текста на фотографиях выбрал предобученную модель YOLOv8 (библиотека ultralytics).  
Для задачи ImageToText проверил несколько предобученных моделей:  
1) EasyOCR (ru) - библиотека easyocr  
2) trocr-base-ru c HuggingFace  
3) trocr-base-printed от Microsoft  
4) EasyOCR (ru+en)  
В итоге выбрал последнюю.

Приложение 1

import os  
  
# Укажите пути к папкам с изображениями и метками  
images\_path = r'C:\Users\1\Desktop\Урфу\Проект(Константин)\datasets\train\images'  
labels\_path = r'C:\Users\1\Desktop\Урфу\Проект(Константин)\datasets\train\labels'  
  
# Получить списки файлов  
images\_files = set(os.listdir(images\_path))  
labels\_files = set(os.listdir(labels\_path))  
  
# Функция для получения основного имени файла без расширения  
def get\_base\_name(file):  
 return os.path.splitext(file)[0]  
  
# Получить базовые имена файлов  
images\_base\_names = {get\_base\_name(file) for file in images\_files}  
labels\_base\_names = {get\_base\_name(file) for file in labels\_files}  
  
# Найти отсутствующие файлы  
missing\_images = labels\_base\_names - images\_base\_names  
missing\_labels = images\_base\_names - labels\_base\_names  
  
# Вывести результаты  
if missing\_images:  
 print("Отсутствующие файлы изображений для следующих меток:")  
 for file in missing\_images:  
 print(f"{file}.txt")  
  
if missing\_labels:  
 print("Отсутствующие файлы меток для следующих изображений:")  
 for file in missing\_labels:  
 print(f"{file}.jpg")  
  
if not missing\_images and not missing\_labels:  
 print("Все файлы соответствуют друг другу")

Приложение 2

import os  
  
# Путь к папке с текстовыми файлами  
folder\_path = r"C:\Users\1\Desktop\Урфу\Проект(Константин)\datasets\valid\labels"  
  
# Цикл по всем файлам в папке  
for filename in os.listdir(folder\_path):  
 # Проверка, что файл - текстовый файл  
 if filename.endswith(".txt"):  
 file\_path = os.path.join(folder\_path, filename)  
 # Открытие файла для чтения и чтение всех строк  
 with open(file\_path, "r") as file:  
 lines = file.readlines()  
 # Изменение первой координаты в каждой строке на 1  
 modified\_lines = [line.strip().split() for line in lines]  
 for i, line in enumerate(modified\_lines):  
 if len(line) >= 5:  
 line[0] = "1"  
 modified\_lines[i] = " ".join(line) + "\n"  
 # Запись измененных строк обратно в файл  
 with open(file\_path, "w") as file:  
 file.writelines(modified\_lines)  
  
print("Готово!")

Приложение 3

from ultralytics import YOLO  
  
# Создание модели с использованием конфигурационного файла  
model = YOLO("yolov8n.yaml")  
  
# Обучение модели с использованием пользовательского датасета  
results = model.train(data="C:/Users/1/PycharmProjects/markirovka\_1/.venv/datasets/datasets.yaml", epochs=10)  
  
# Печать результатов обучения  
print(results)

Приложение 4

from ultralytics import YOLO  
  
# Путь к тестовому изображению  
test\_image\_path = r"C:\Users\1\PycharmProjects\markirovka\_1\.venv\datasets\test\Труба №19009 Плавка №2023-В303501 Лист №192501001.jpg"  
  
# Загрузка модели YOLO  
model = YOLO("C:/Users/1/PycharmProjects/markirovka\_1/runs/detect/train9/weights/best.pt")  
  
# Предсказание на тестовом изображении  
results = model.predict(source=test\_image\_path)  
  
# Отображение результата с разметкой  
results[0].show()