ДОДАТОК А

Графічні матеріали

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНИКИ

КАФЕДРА КІТС
КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЛАСИФІКАЦІЇ ТА РОЗПІЗНАВАННЯ СМІТТЯ ДЛЯ ПОДАЛЬШОГО СОРТУВАННЯ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ

Виконав:

Ст. гр. КІУКІ-21-10
Гришин Микола Миколайович

Керівник:

ас. каф. КІТС Кирило Олійник

1

МЕТА ТА ПОСТАНОВКА ЗАВДАННЯ

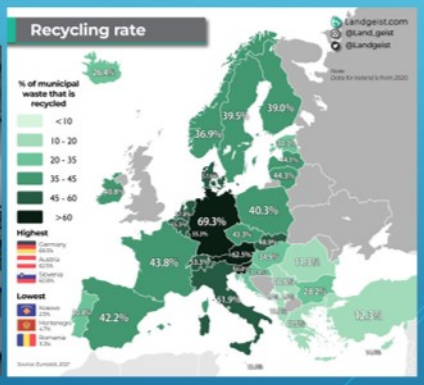
- Метою кваліфікаційної роботи є розробка системи автоматичного розпізнавання та класифікації побутових відходів на зображеннях із використанням комп'ютерного зору та глибокого навчання, а також інтеграція цієї системи з Telegram-ботом для зручної взаємодії користувачів.

Завдання кваліфікаційної роботи:

– Аналіз сучасних підходів до сортування відходів і методівдетекції;
– Формування та анотація датасету зображень сміття;
– Розробка та навчання моделі YOLOv5 для детекції шести категорій відходів;
– Інтеграція моделі з Telegram-ботом;
– Оцінка якості роботи системи та аналіз результатів.

2

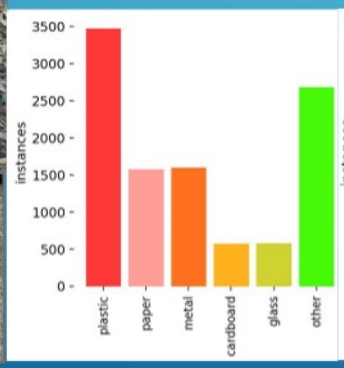
АКТУАЛЬНІСТЬ ПРОБЛЕМИ

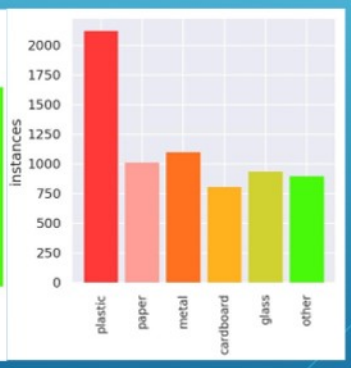


ЗАГАЛЬНИЙ ПІДХІД І АРХІТЕКТУРА СИСТЕМИ

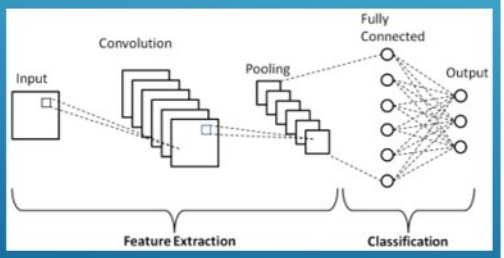
Система складається з двох основних частин:
1.Модель детекції YOLOv5
2.Telegram-бот для користувачів

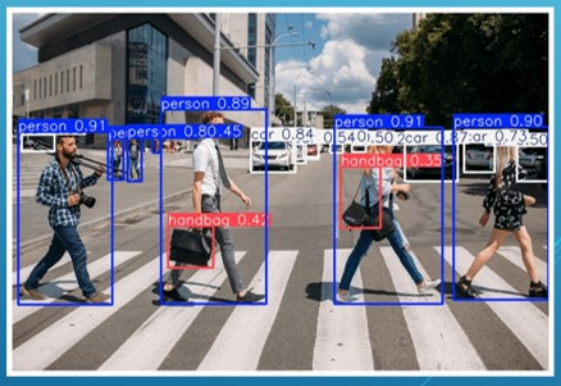
ОГЛЯД ДАТАСЕТУ ТА ПІДГОТОВКА ДАНИХ





АРХІТЕКТУРА ТА ПРИНЦИП РОБОТИ YOLOV5





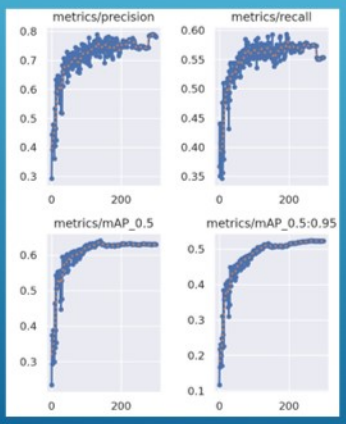
TELEGRAM-БОТ: ІНТЕГРАЦІЯ ТА ВИКОРИСТАННЯ

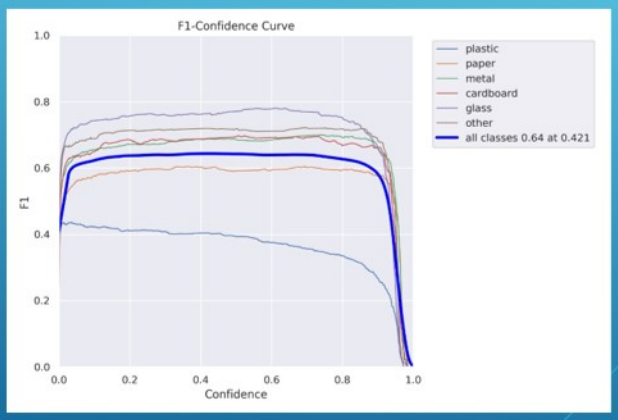
Для зручності використання моделі був розроблений телеграм бот. Користувач надсилає фото зі сміттям і одразу отримує результат з виділеними об'єктами та їх категоріями.

ГРИШИН МИКОЛА, КІУКІ21-10, ХНУРЕ

7

ОСНОВНІ РЕЗУЛЬТАТИ І МЕТРИКИ

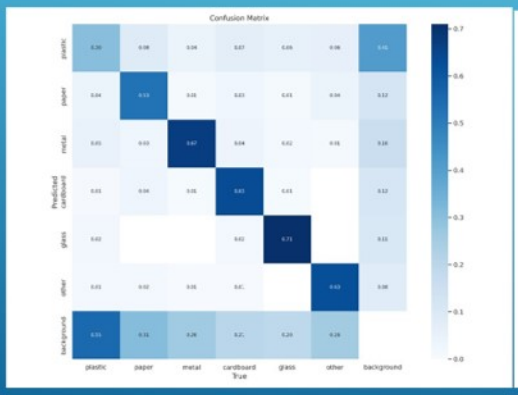


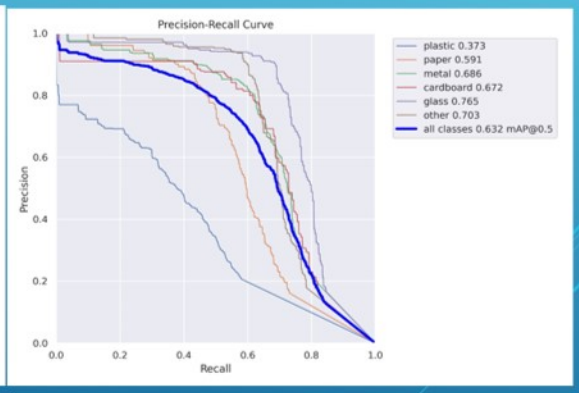


ГРИШИН МИКОЛА, КІУКІ21-10, ХНУРЕ

8

АНАЛІЗ ПОМИЛОК І СЛАБКИХ КЛАСІВ





ВИСНОВКИ

- ПРОВЕДЕНО АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ
- РОЗРОБЛЕНО ТА РЕАЛІЗОВАНО АРХІТЕКТУРУ
- СТВОРЕНО ПРОГРАМНУ РЕАЛІЗАЦІЮ
- ПРОВЕДЕНО ТЕСТУВАННЯ МОДЕЛІ
- ВИЗНАЧЕНО НЕДОЛІКИ МОДЕЛІ

ДОДАТОК Б

Лістинг програми Telegram бота

```
import os
import logging
import datetime

from telegram import Update
from telegram.ext import ApplicationBuilder, ContextTypes,
CommandHandler, MessageHandler
from telegram.ext.filters import PHOTO

import torch
import torchvision.transforms as transforms

from PIL import Image

import yolov5


import cv2



logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    level=logging.INFO
)

classes = {0: 'plastic',
           1: 'paper',
           2: 'metal',
           3: 'cardboard',
           4: 'glass',
           5: 'other'}


async def start(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    await
context.bot.send_message(chat_id=update.effective_chat.id,
                                     text="hi, i can tell what kind
of trash do you have."
                                          "just send me a picture of
it")

async def trash(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    image_id = update.message.photo[-1].file_id
    image = await context.bot.get_file(image_id)
    save_dir =
"D:\\Programming\\Projects\\ml_diploma\\trash_sorting\\media"
    img_time = datetime.datetime.now()
    image_name = img_time.strftime("%Y-%m-%d_%H-%M")
```

```
    image_path = os.path.join(save_dir, f"{image_name}.jpg")

    await image.download_to_drive(image_path)

    model =
yolov5.load("D:\Programming\Projects\ml_diploma\\models\\best3.pt"
, device='cpu')

    results = model(image_path)

    detections = results.xyxy[0]

    if detections.shape[0] == 0:
        await context.bot.send_message(update.effective_chat.id,
"Ничего не найдено 🤷")
    else:
        bbox_img = draw_detection_box(image_path, detections,
save_path='output.jpg')
        response = ""
        for *box, conf, cls in detections:
            class_name = model.names[int(cls)]
            response += f"{class_name}: {conf:.2%}\n"

        await context.bot.send_photo(update.effective_chat.id,
bbox_img)


import cv2

def draw_detection_box(
    image_path,
    detections,     # tensor или list: [x1, y1, x2, y2, conf,
class_id]
    save_path='output.jpg',
    color=(0, 255, 0),
    thickness=2,
    font_scale=1,
    font_thickness=2
):

    img = cv2.imread(image_path)
    cv2.imwrite(save_path, img)
    for detection in detections:

        x1, y1, x2, y2, conf, cls = detection
        x1, y1, x2, y2 = map(int, [x1, y1, x2, y2])
        cls = int(cls)
        label = f"{classes[cls]} {conf:.2f}"

        cv2.rectangle(img, (x1, y1), (x2, y2), color, thickness)
```

```
        font = cv2.FONT_HERSHEY_SIMPLEX
        text_size, _ = cv2.getTextSize(label, font, font_scale,
font_thickness)
        text_w, text_h = text_size

        cv2.rectangle(img, (x1, y1 - text_h - 5), (x1 + text_w,
y1), color, -1)
        cv2.putText(img, label, (x1, y1 - 5), font, font_scale,
(0, 0, 0), font_thickness, lineType=cv2.LINE_AA)

        cv2.imwrite(save_path, img)



def image_preparation(image_path):
    mean = [0.6732, 0.6399, 0.6049]
    std = [0.1805, 0.1797, 0.1902]
    user_transforms = transforms.Compose([
        transforms.Resize((224, 224)),  # same size as training
        transforms.ToTensor(),  # convert to [0, 1] tensor
        transforms.Normalize(mean, std)
    ])

    device = set_device()
    img = Image.open(image_path).convert("RGB")  # always RGB
    tensor = user_transforms(img).unsqueeze(0).to(device)  # add
batch dim
    return tensor



def set_device():
    if torch.cuda.is_available():
        dev = torch.device("cuda:0")
    else:
        dev = torch.device("cpu")
    return torch.device(dev)


if __name__ == '__main__':
    application =
ApplicationBuilder().token('7560188861:AAEzaDlQZzywzht6996LNOieP1a
CiiKY7cY').build()

    start_handler = CommandHandler('start', start)
    trash_handler = MessageHandler(filters=PHOTO, callback=trash)
    application.add_handler(start_handler)
    application.add_handler(trash_handler)

    application.run_polling()
```