МИРЭА – РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий

Кафедра промышленной информатики

Тема выпускной квалификационной работы:

«Информационно-аналитическая система процесса сортировки пластикового мусора с применением системы машинного зрения»

Выполнил бакалавр Казанцева А. Г. группы ИВБО-05-20

Научный руководитель: Благовещенский В. Г.

Выбор программных средств для реализации

Скачивание изображений по ссылкам

Листинг 8 — Скачивание изображений по ссылкам

```
from imutils import paths
import argparse
import requests
import cv2
import os

...

rows = open(args["urls"]).read().strip().split("\n")
total = 0
folder_path = args["output"]
filename_format = "{}.jpg"

def check_and_rename_file(folder_path, filename_format):
    i = 1
    while True:
        filename = filename_format.format(str(i).zfill(8))
        file_path = os.path.join(folder_path, filename)
        if not os.path.exists(file_path):
            return file_path
        i += 1

for url in rows:
    try:
        r = requests.get(url, timeout=60)
        p = check_and_rename_file(args["output"],
filename_format)
        f = open(p, "wb")
        f.write(r.content)
        f.close()
    except Exception as e:
        print("[INFO] ошибка при загрузке
{}...пропуск".format(p))
```

Продолжение Листинга 8

```
for imagePath in paths.list_images(args["output"]):
        delete = False
        try:
                image = cv2.imread(imagePath)
                if image is None:
                        delete = True
        except:
                print("Except")
                delete = True
        if delete:
                print("[INFO] deleting
{}".format(imagePath))
                os.remove(imagePath)
```

Удаление дубликатов по хэш-сумме

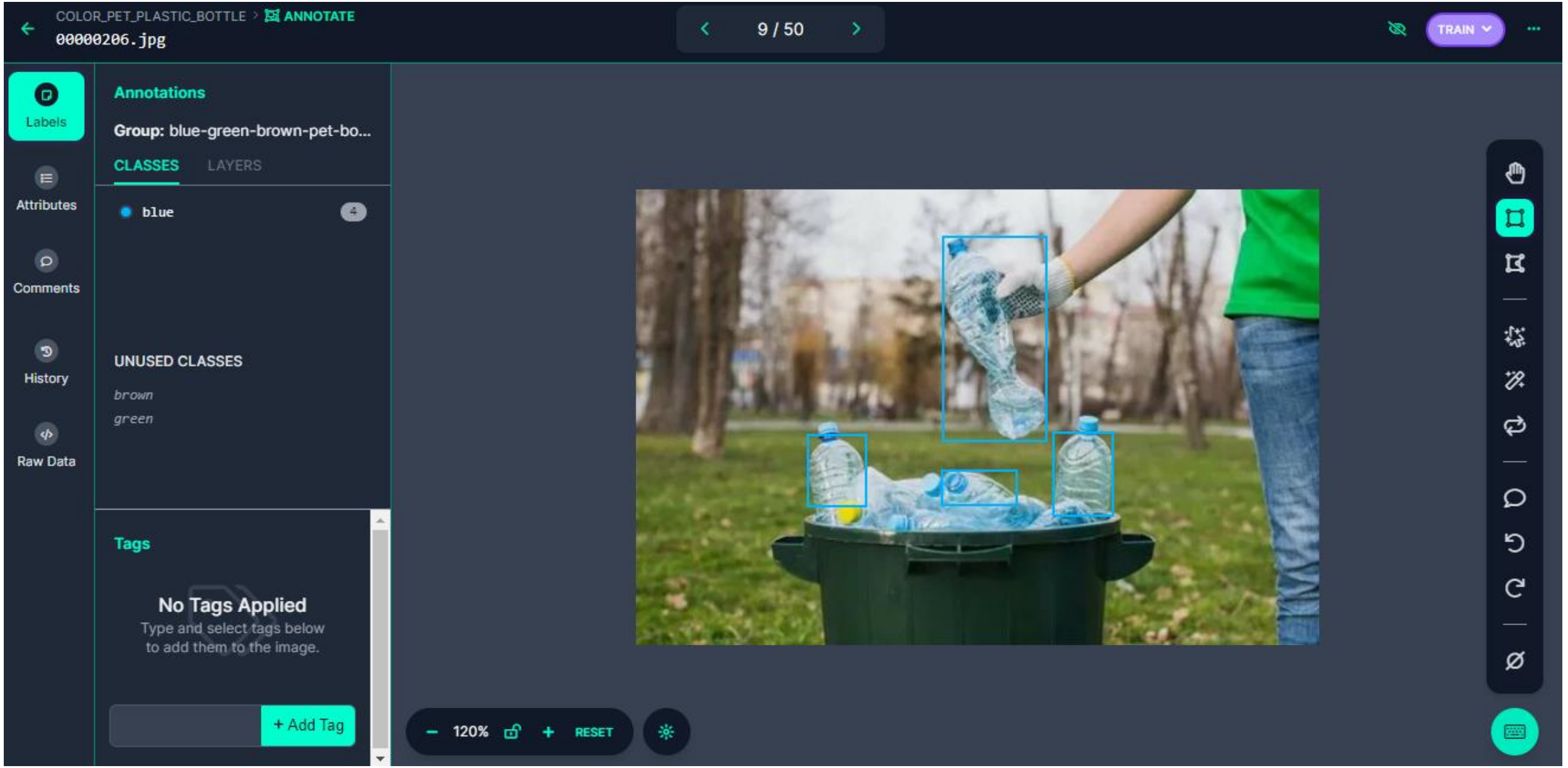
Листинг 9 — Удаление дубликатов

```
from imutils import paths
import numpy as np
import argparse
import cv2
import os

def dhash(image, hashSize=8):
          gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
          resized = cv2.resize(gray, (hashSize + 1, hashSize))
          diff = resized[:, 1:] > resized[:, :-1]
          return sum([2 ** i for (i, v) in
enumerate(diff.flatten()) if v])
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True, help="путь к
датасету")
ap.add_argument("-r", "--remove", type=int, default=-1,
help="выполнять пробный запуск")
args = vars(ap.parse_args())

print("[INFO] computing image hashes...")
imagePaths = list(paths.list_images(args["dataset"]))
hashes = {}

for imagePath in imagePaths:
          image = cv2.imread(imagePath)
          h = dhash(image)
          p = hashes.get(h, [])
          p.append(imagePath)
          hashes[h] = p
```

Продолжение Листинга 9

```
for (h, hashedPaths) in hashes.items():
          if len(hashedPaths) > 1:
                    # проверка на пробный запуск
                    if args["remove"] <= 0:
                              montage = None
                              for p in hashedPaths:
                                        image =
cv2.imread(p)
                                        image =
cv2.resize(image, (150, 150))
                                        if montage is
None:

          montage = image
                                        else:

          montage = np.hstack([montage, image])
                              print("[INFO] hash:
{}".format(h))
                              cv2.imshow("Montage",
montage)
                              cv2.waitKey(0)
                    else:
                              for p in hashedPaths[1:]:
                                        os.remove(p)
```

Разметка изображений в Roboflow



Разработка веб-интерфейса с видеовещанием

Листинг 3 — views.py Видеовещание с камеры

```
from django.shortcuts import render, redirect,
get_object_or_404
from django.http import StreamingHttpResponse
import cv2
from .forms import ImageForm
from .models import *
from ultralytics import YOLO

model = YOLO('../detect_pet.pt')

def video_filter(request):
    def stream_rgb_video():
        cam = cv2.VideoCapture(0)
        while cam.isOpened():
            ret, frame = cam.read()
            res_img = model(source=frame, show=False, conf=0.7)
            anno_frm = res_img[0].plot()
            #score = res_img.boxes.length()
            img_bytes = cv2.imencode('.jpg',
anno_frm)[1].tobytes()
            cv2.waitKey(24)
            yield (b'--frame\r\n'b'Content-type:
image/jpg\r\n\r\n' + img_bytes + b'\r\n')
            #b'--frame\r\n'b'Content-type: text/plain\r\n\r\n'
+ str(score).encode() + b'\r\n')
    return StreamingHttpResponse(stream_rgb_video(),
content_type='multipart/x-mixed-replace; boundary=frame')
```

Разработка веб-интерфейса с видеовещанием

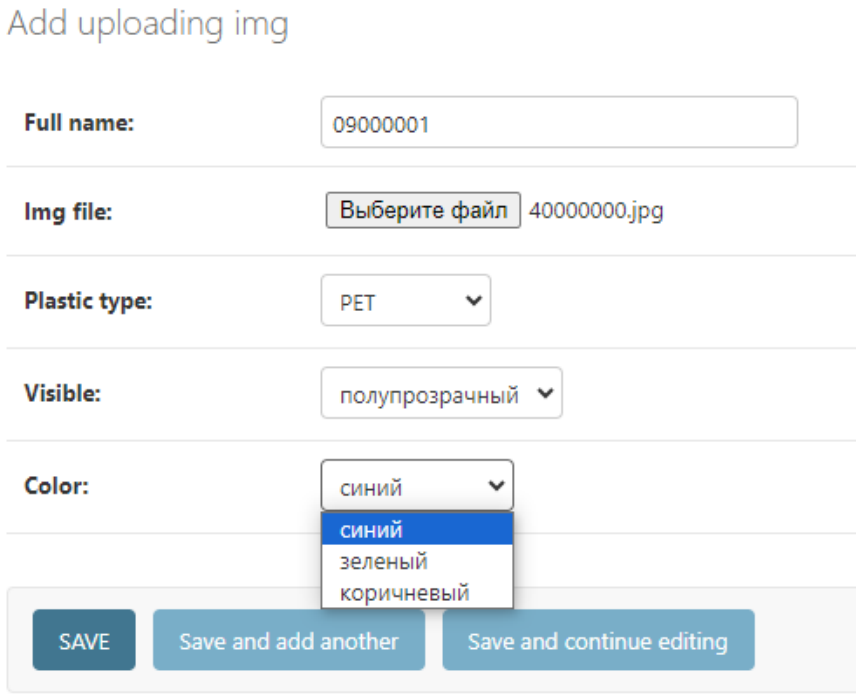
Листинг 4 — views.py, Добавление изображений

```
def upload_image(request):
    if request.method == 'POST':
        form = ImageForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            img_object = form.instance
            return render(request, 'opencv/upload_image.html',
{'form': form, 'img_obj': img_object})
    else:
        form = ImageForm()
    return render(request, 'opencv/upload_image.html', {'form':
form})
```

Листинг 5 — urls.py, Связь ссылок и методов из views.py

```
from django.urls import path
from .views import *
urlpatterns = [
    path('', image_list, name='image_list'),
    path('media_files/<int:id>', media_file_YOLO_opencv,
name='media_file_YOLO_opencv'),
path('video_filter/', video_filter, name='video_filter.url'),
    path('post/',UploadImg.as_view(), name='add')
]
```

Разработка веб-интерфейса для добавления изображений



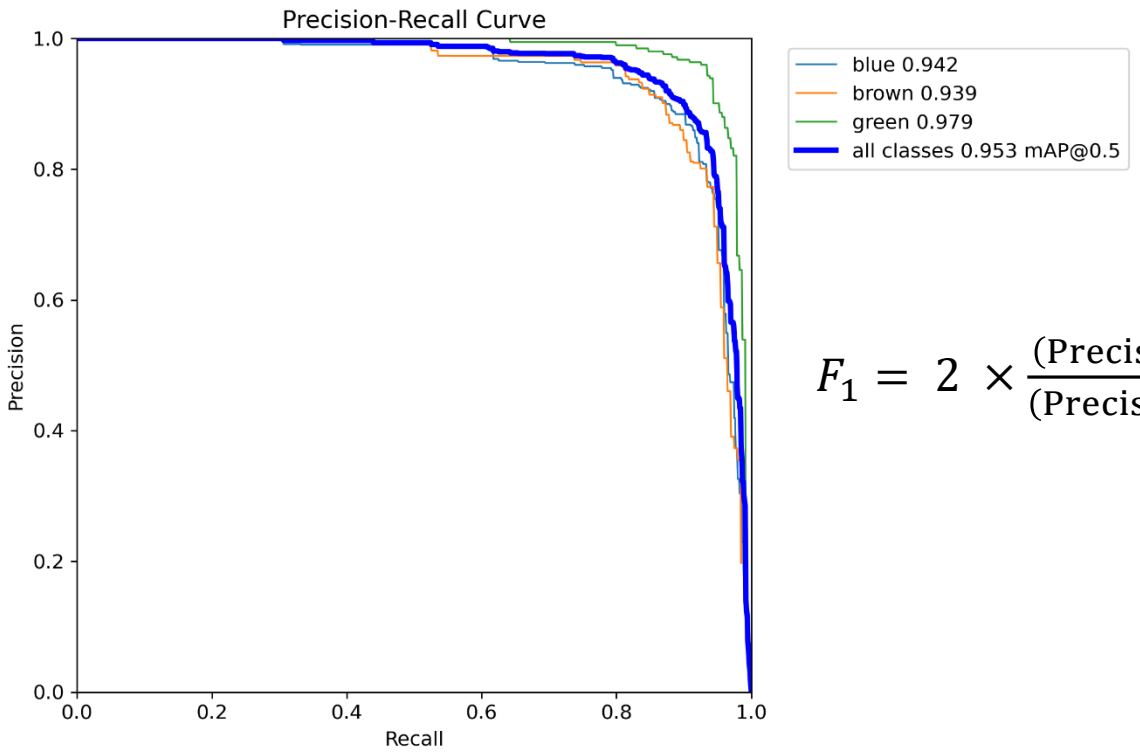
Листинг 6 — forms.py, Представление полей для добавления изображений

```
from django import forms
from .models import UploadingImg

class ImageForm(forms.ModelForm):
    class Meta:
        model = UploadingImg
        fields = ['title_text_img',
'img_file','plastic_type','visible','color']
```

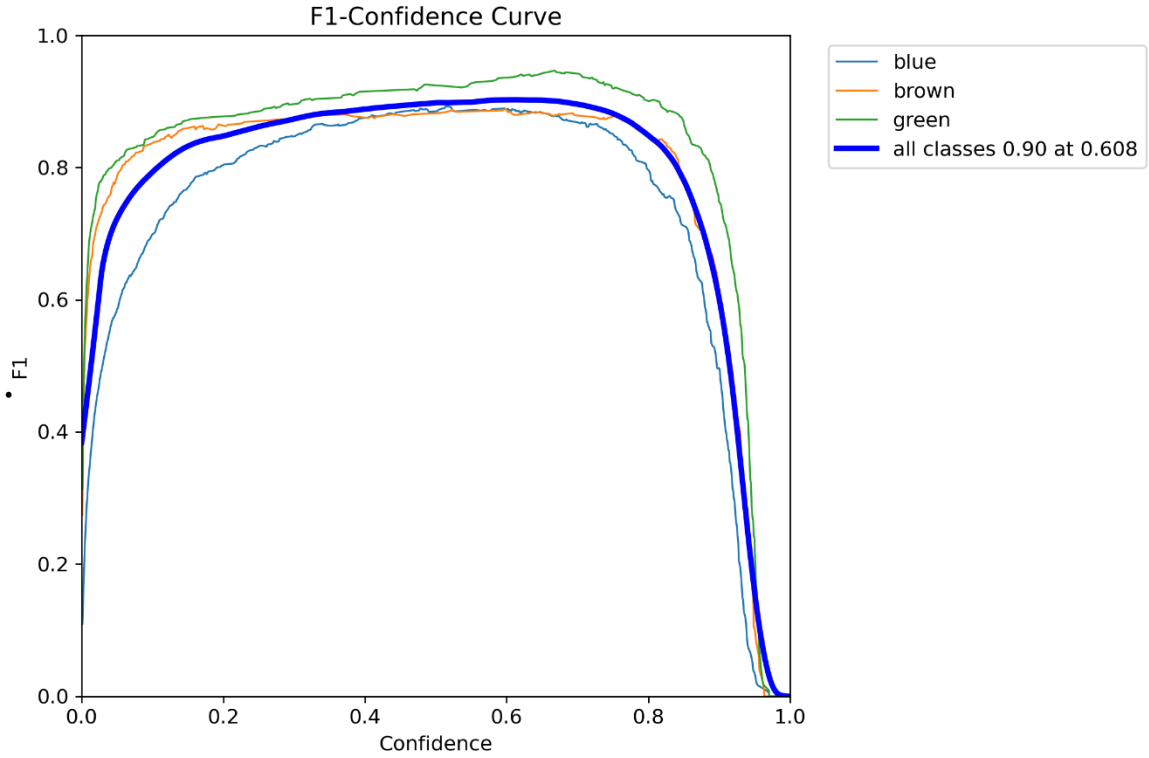
Листинг 7 — models.py, атрибуты базы данных для добавления изображений

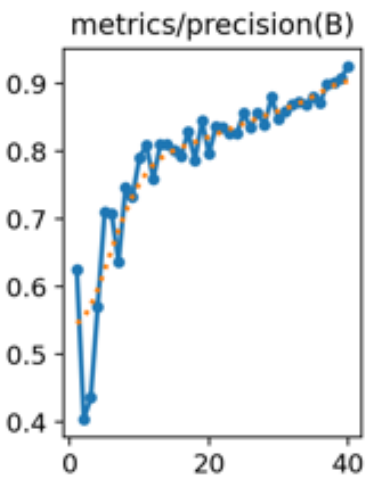
```
    from django.db import models
    class UploadingImg(models.Model):
    ...
    id_img = models.AutoField(primary_key=True)
        title_text_img=models.CharField(max_length=15, verbose_name='full
name')
        img_file = models.ImageField(upload_to='images/')
        uploaded_at = models.DateTimeField(auto_now=True)
        plastic_type=models.CharField(max_length=10,
choices=Plastic_types, default=Plastic_types[7][0])
        visible = models.CharField(max_length=14, choices=Visibility,
default=Visibility[1][0])
        color = models.CharField(max_length=10, choices=Color_plastic,
null=True, default=Color_plastic[0][0])

          def __str__(self):
            return self.title_text_img
```

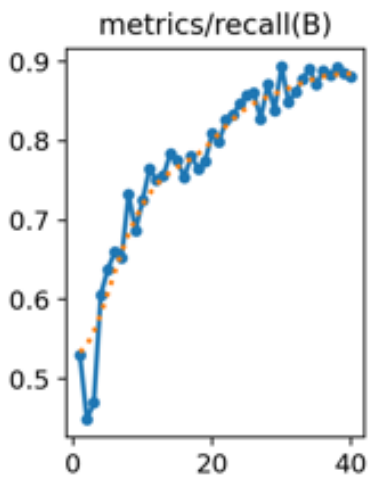
Анализ графиков

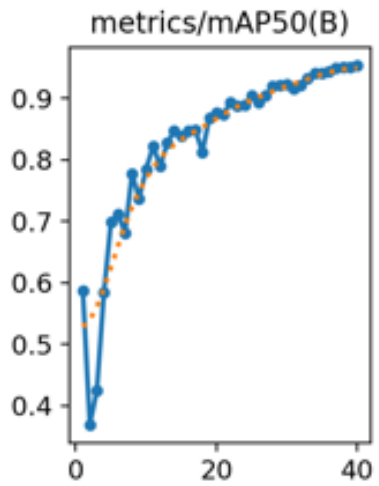


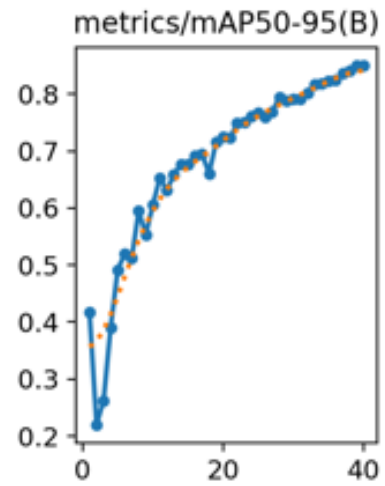
$$F_1 = 2 \times \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$



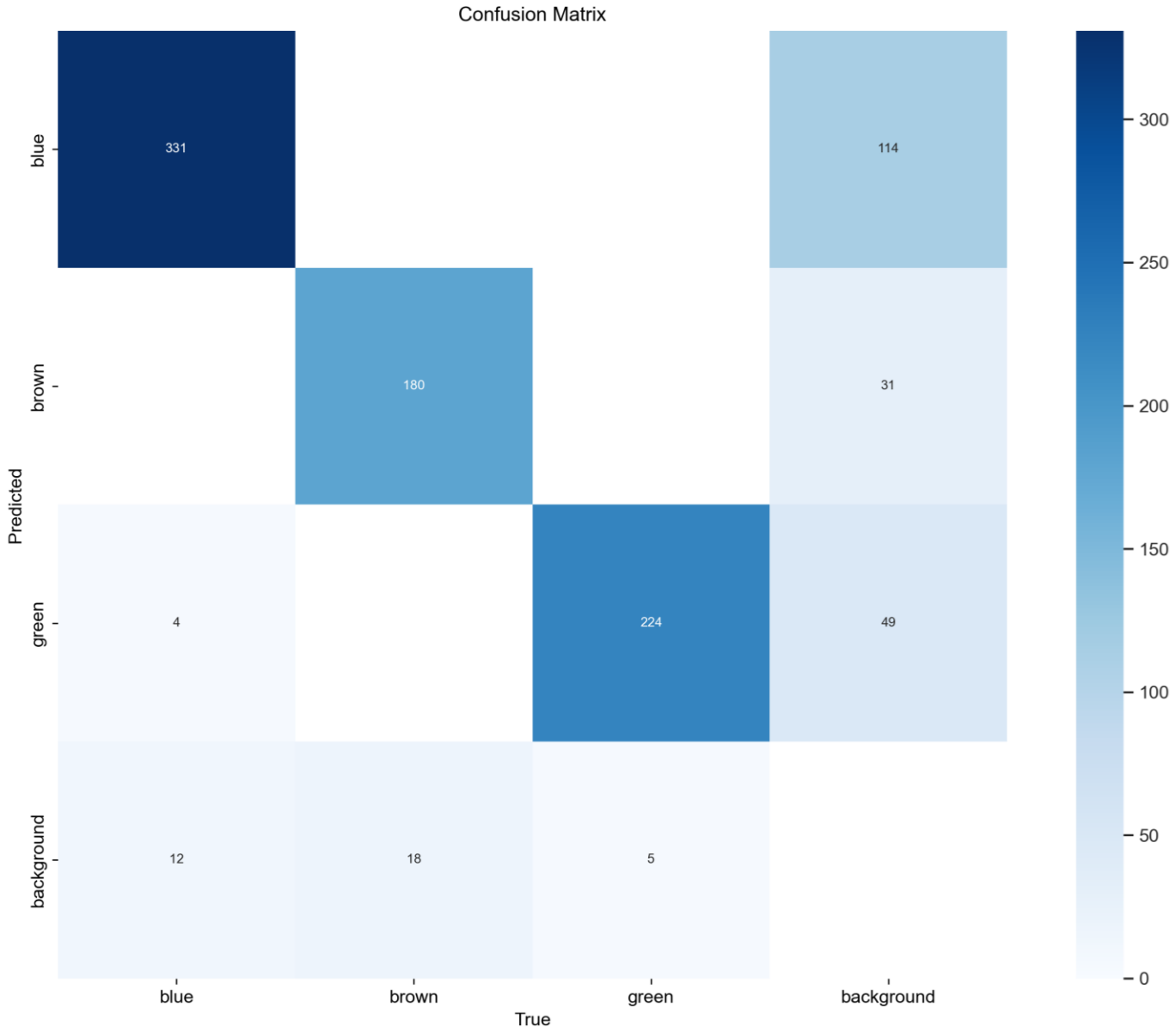








Матрица неточностей в абсолютных значениях



Запуск на испытательном наборе

Тестовый запуск

Из результатов вывод по метрикам модели следующий: точность составляет 0.93, полнота 0.88. Таким образом, метрика F1 составляет 0.9. Метрика mAP50-95 составляет 95%, mAP50 — около 85%. Архитектура состоит из 168 слоев. Алгоритмом оптимизации является AdamW. Количество эпох 40. Батчи по 16 изображений, размер входного изображения составляет 640 пикселей по ширине и высоте. Скорость обучения lr0 = 0.01. Изображений для обучения использовалось 1180, для испытаний 303.

Заключение

Основной целью являлось повышение эффективности процесса сортировки, поэтому в ходе выполнения выпускной квалифицированной работы была разработана информационно-аналитическая система процесса сортировки пластикового мусора с применением системы машинного зрения, способная автоматически обнаруживать и классифицировать пластиковые ПЭТ-бутылки по цвету, что существенно упрощает и ускоряет процесс их переработки.

В процессе работы над проектом были выполнены следующие функции системы:

- предобработка поступающих изображений и запись данных в базу данных;
- взаимодействие с пользователем через загрузку изображений;
- вывод видео с камер, определение границ объектов и их классификация;
- дополнение набора данных и запуск обучения;

Таким образом, поставленные задачи были успешно решены, а цели проекта достигнуты. Разработанная информационно-аналитическая система с функциями машинного зрения показала высокую эффективность и может быть рекомендована для внедрения на предприятиях, занимающихся переработкой пластиковых отходов. Это позволит значительно повысить качество и скорость сортировки, уменьшить затраты и внести вклад в решение проблемы пластикового загрязнения окружающей среды.