`

Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Отчет по лабораторной работе 3

Специальность ИИ-23

Выполнил:

Гавришук В.Р.

Студент группы ИИ-23

Проверил:

Андренко К. В.
Преподаватель-стажёр
Кафедры ИИТ,
«___» _____2025 г.

Брест 2025

`

Цель: осуществлять обучение нейросетевого детектора для решения задачи обнаружения заданных объектов

Общее задание

1. Базируясь на своем варианте, ознакомится с выборкой для обучения детектора, выполнить необходимые преобразования данных для организации процесса обучения (если это нужно!);
2. Для заданной архитектуры нейросетевого детектора организовать процесс обучения для своей выборки. Оценить эффективность обучения на тестовой выборке (mAP);
3. Реализовать визуализацию работы детектора из пункта 1 (обнаружение знаков на отдельных фотографиях из сети Интернет);
4. Оформить отчет по выполненной работе, залить исходный код и отчет в соответствующий репозиторий на github.

Задание по вариантам

№ варианта	Детектор	Датасет
4	YOLOv10s	**Коты:** https://universe.roboflow.com/mohamed-traore-2ekkp/cats-n9b87/dataset/3

Ход работы:

```
def ensure_yolov10_installed():
    try:
        import ultralytics  # type: ignore
    except Exception:
        print("[INFO] ultralytics/yolov10 не найден — ставим из репозитория YOLOv10 (THU-MIG)...")
        pip_install("git+https://github.com/THU-MIG/yolov10.git")

    try:
        from ultralytics import YOLO  # type: ignore
        return True
    except Exception as e:
        print("[ERROR] Не удалось импортировать ultralytics.YOLO после установки:", e)
        return False

def create_data_yaml(data_dir: str, names: List[str]=["cat"]) -> str:
    data_dir = Path(data_dir)
    out = data_dir / "data.yaml"
    content = {
        "path": str(data_dir.resolve()),
        "train": "train/images",
        "val": "valid/images",
        "test": "test/images",
        "nc": len(names),
        "names": names
    }
```

`

```
    with open(out, "w", encoding="utf-8") as f:
        yaml_text = json.dumps(content, indent=2)
        f.write(f"path: {content['path']}\n")
        f.write(f"train: {content['train']}\n")
        f.write(f"val: {content['val']}\n")
        f.write(f"test: {content['test']}\n")
        f.write(f"nc: {content['nc']}\n")
        f.write("names:\n")
        for i, n in enumerate(names):
            f.write(f"  {i}: {n}\n")
    print(f"[INFO] Создан {out}")
    return str(out)

def run_training(data_yaml: str, epochs: int=50, batch: int=16, imgsz: int=640, project: str="runs/train",
name: str="yolov10s_cats"):
    ok = ensure_yolov10_installed()
    if not ok:
        raise RuntimeError("Требуется установить yolov10/ultralytics. Смотрите логи выше.")

    from ultralytics import YOLO

    model_cfgs = ["yolov10s.yaml", "yolov10s.pt", "yolov10n.yaml", "yolov10n.pt"]
    model_source = None
    for m in model_cfgs:
        try:
            model = YOLO(m)
            model_source = m
            break
        except Exception:
            continue

    if model_source is None:
        raise RuntimeError("Не найдено заранее доступных конфигов/весов yolov10s. Установите репозиторий
yolov10 и убедитесь, что файлы yolov10s.yaml / yolov10s.pt доступны.")

    print(f"[INFO] Использую модель/конфиг: {model_source}")
    model.train(data=data_yaml, epochs=epochs, imgsz=imgsz, batch=batch, project=project, name=name)
    print("[INFO] Обучение завершено (или прервано).")

def run_validation(weights_path: str, data_yaml: str, imgsz: int=640):
    from ultralytics import YOLO
    model = YOLO(weights_path)
    print("[INFO] Запускаю валидацию на тестовом наборе...")
    res = model.val(data=data_yaml, imgsz=imgsz)
    print("[INFO] Результат валидации:")
    print(res)
    return res

def download_image(url: str, out_dir: str="tmp_images") -> str:
    Path(out_dir).mkdir(parents=True, exist_ok=True)
    local_name = Path(out_dir) / Path(url.split("?")[0].split("/")[-1])
    if not local_name.exists():
        print(f"[INFO] Скачиваем {url} -> {local_name}")
        r = requests.get(url, stream=True, timeout=15)
        r.raise_for_status()
        with open(local_name, "wb") as f:
            for chunk in r.iter_content(chunk_size=8192):
                f.write(chunk)
    return str(local_name)

def visualize_predictions(weights_path: str, image_urls: List[str], imgsz: int=640, conf: float=0.25,
save_dir: str="predictions"):
    from ultralytics import YOLO
    model = YOLO(weights_path)
    Path(save_dir).mkdir(parents=True, exist_ok=True)
    for url in image_urls:
        try:
            img_path = download_image(url)
```

`

```
        except Exception as e:
            print("[WARN] Не удалось скачать", url, e)
            continue
        results = model(img_path, imgsz=imgsz, conf=conf)
        r = results[0]
        pil = Image.open(img_path).convert("RGB")
        draw = ImageDraw.Draw(pil)
        try:
            boxes = r.boxes.xyxy.cpu().numpy()
            scores = r.boxes.conf.cpu().numpy()
            classes = r.boxes.cls.cpu().numpy().astype(int)
        except Exception:
            boxes = []
            scores = []
            classes = []
        names = getattr(r, "names", None) or getattr(model, "names", None) or {}
        font = ImageFont.load_default()
        for (xy, sc, cl) in zip(boxes, scores, classes):
            x1, y1, x2, y2 = map(int, xy.tolist())
            label = f"{names.get(cl, str(cl))} {sc:.2f}"
            draw.rectangle([x1, y1, x2, y2], outline="red", width=2)
            text_w, text_h = font.getsize(label)
            draw.rectangle([x1, y1 - text_h, x1 + text_w, y1], fill="red")
            draw.text((x1, y1 - text_h), label, fill="white", font=font)
        out_path = Path(save_dir) / ("pred_" + Path(img_path).name)
        pil.save(out_path)
        print(f"[INFO] Сохранено предсказание: {out_path}")

def check_dataset_structure(data_dir: str) -> bool:
    data_dir = Path(data_dir)
    req = [
        data_dir / "train" / "images",
        data_dir / "train" / "labels",
        data_dir / "valid" / "images",
        data_dir / "valid" / "labels",
    ]
    ok = True
    for p in req:
        if not p.exists():
            print(f"[WARN] Отсутствует {p}")
            ok = False
    if ok:
        print("[INFO] Структура датасета выглядит корректно.")
    else:
        print("[ERROR] Проверьте, что вы скачали Roboflow датасет и распаковали в:", data_dir)
    return ok

def main():
    import torch
    print("CUDA доступна:", torch.cuda.is_available())
    if torch.cuda.is_available():
        print("Используется видеокарта:", torch.cuda.get_device_name(0))
    parser = argparse.ArgumentParser()
    parser.add_argument("--data-dir", type=str, default="./datasets/cats", help="путь к распакованному
датасету (YOLO формат)")
    parser.add_argument("--epochs", type=int, default=50)
    parser.add_argument("--batch", type=int, default=16)
    parser.add_argument("--imgsz", type=int, default=640)
    parser.add_argument("--train-only", action="store_true", help="только тренировать (без
валидации/визуализации)")
    parser.add_argument("--weights", type=str, default="", help="если указать путь к весам для
валидации/визуализации")
    parser.add_argument("--visualize-urls", type=str, nargs="*", default=[], help="URL-изображения для
отрисовки предсказаний")
    args = parser.parse_args()

    data_dir = args.data_dir
    if not check_dataset_structure(data_dir):
```

`

```
        print("\n[INSTR] Скачайте датасет YOLO с Roboflow и распакуйте в", data_dir)
        print("Roboflow dataset page:", "https://universe.roboflow.com/mohamed-traore-2ekkp/cats-
n9b87/dataset/3")
        sys.exit(1)

    data_yaml = create_data_yaml(data_dir, names=["cat"])

    run_training(data_yaml, epochs=args.epochs, batch=args.batch, imgsz=args.imgsz)

    run_dir = Path("runs/train")
    last_run = None
    if run_dir.exists():
        runs = sorted([d for d in run_dir.iterdir() if d.is_dir()], key=lambda x: x.stat().st_mtime)
        if runs:
            last_run = runs[-1]
    if last_run:
        candidate_weights = last_run / "weights" / "best.pt"
        if candidate_weights.exists():
            best_weights = str(candidate_weights)
        else:
            pts = list(last_run.rglob("*.pt"))
            best_weights = str(pts[-1]) if pts else ""
    else:
        best_weights = ""

    if best_weights:
        print("[INFO] Найдены веса для валидации/визуализации:", best_weights)
    else:
        print("[WARN] Не найдены веса автоматически. Укажите --weights путь к .pt если хотите запустить
валидацию/визуализацию.")
        if args.train_only:
            print("[DONE] Завершено (только тренировка).")
            return

    if best_weights:
        run_validation(best_weights, data_yaml, imgsz=args.imgsz)

    if args.visualize_urls:
        weights_for_vis = args.weights if args.weights else best_weights
        if not weights_for_vis:
            print("[WARN] Нет весов для визуализации предсказаний.")
        else:
            visualize_predictions(weights_for_vis, args.visualize_urls, imgsz=args.imgsz)
```

Результат работы программы:

Ultralytics 8.3.227 Python-3.12.6 torch-2.5.1+cu121 CUDA:0 (NVIDIA GeForce RTX 3050 Laptop GPU, 4096MiB)

engine\trainer: agnostic_nms=False, amp=True, augment=False, auto_augment=randaugment, batch=16, bgr=0.0, box=7.5, cache=False, cfg=None, classes=None, close_mosaic=10, cls=0.5, compile=False, conf=None, copy_paste=0.0, copy_paste_mode=flip, cos_lr=False, cutmix=0.0, data=datasets\cats\data.yaml, degrees=0.0, deterministic=True, device=None, dfl=1.5, dnn=False, dropout=0.0, dynamic=False, embed=None, epochs=50, erasing=0.4, exist_ok=False, fliplr=0.5, flipud=0.0, format=torchscript, fraction=1.0, freeze=None, half=False, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, imgsz=640, int8=False, iou=0.7, keras=False, kobj=1.0, line_width=None, lr0=0.01, lrf=0.01, mask_ratio=4, max_det=300, mixup=0.0, mode=train, model=yolov10s.yaml, momentum=0.937, mosaic=1.0, multi_scale=False, name=yolov10s_cats3, nbs=64, nms=False, opset=None, optimize=False, optimizer=auto, overlap_mask=True, patience=100, perspective=0.0, plots=True, pose=12.0, pretrained=True, profile=False, project=runs/train, rect=False, resume=False, retina_masks=False, save=True, save_conf=False, save_crop=False, save_dir=C:\Users\ASUS\PycharmProjects\oiis2\runs\train\yolov10s_cats3, save_frames=False, save_json=False, save_period=-1, save_txt=False, scale=0.5, seed=0, shear=0.0, show=False, show_boxes=True, show_conf=True, show_labels=True, simplify=True, single_cls=False, source=None, split=val, stream_buffer=False, task=detect, time=None, tracker=botsort.yaml, translate=0.1, val=True, verbose=True, vid_stride=1, visualize=False,

`

warmup_bias_lr=0.1, warmup_epochs=3.0, warmup_momentum=0.8, weight_decay=0.0005, workers=8, workspace=None

Overriding model.yaml nc=80 with nc=1

from n params module arguments

0 -1 1 928 ultralytics.nn.modules.conv.Conv [3, 32, 3, 2]

50/50 4.95G 1.466 0.941 2.855 11 640: 100% ━━━━━━━━━━━━ 51/51 0.6it/s 1:26

Class Images Instances Box(P R mAP50 mAP50-95): 100% ━━━━━━━━━━━━ 8/8 2.1it/s 3.9s

all 232 235 0.902 0.872 0.95 0.705

50 epochs completed in 1.274 hours.

Optimizer stripped from C:\Users\ASUS\PycharmProjects\oiis2\runs\train\yolov10s_cats3\weights\last.pt, 16.5MB

Optimizer stripped from C:\Users\ASUS\PycharmProjects\oiis2\runs\train\yolov10s_cats3\weights\best.pt, 16.5MB

Validating C:\Users\ASUS\PycharmProjects\oiis2\runs\train\yolov10s_cats3\weights\best.pt...

Ultralytics 8.3.227 Python-3.12.6 torch-2.5.1+cu121 CUDA:0 (NVIDIA GeForce RTX 3050 Laptop GPU, 4096MiB)

YOLOv10s summary (fused): 106 layers, 7,218,387 parameters, 0 gradients, 21.4 GFLOPs

Class Images Instances Box(P R mAP50 mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.6it/s 2.2s

all 232 235 0.892 0.845 0.944 0.708

Speed: 0.2ms preprocess, 5.6ms inference, 0.0ms loss, 0.3ms postprocess per image

Results saved to C:\Users\ASUS\PycharmProjects\oiis2\runs\train\yolov10s_cats3

[INFO] Запускаю валидацию на тестовом наборе...

Ultralytics 8.3.227 Python-3.12.6 torch-2.5.1+cu121 CUDA:0 (NVIDIA GeForce RTX 3050 Laptop GPU, 4096MiB)

YOLOv10s summary (fused): 106 layers, 7,218,387 parameters, 0 gradients, 21.4 GFLOPs

val: Fast image access (ping: 0.00.0 ms, read: 492.9613.6 MB/s, size: 70.2 KB)

val: Scanning C:\Users\ASUS\PycharmProjects\oiis2\datasets\cats\valid\labels.cache... 232 images, 0 backgrounds, 0 corrupt: 100% ━━━━━━━━━━━━ 232/232 0.0s

Class Images Instances Box(P R mAP50 mAP50-95): 100% ━━━━━━━━━━━━ 15/15 4.6it/s 3.3s

all 232 235 0.88 0.845 0.941 0.706

Speed: 2.3ms preprocess, 8.8ms inference, 0.0ms loss, 0.2ms postprocess per image

Results saved to C:\Users\ASUS\PycharmProjects\oiis2\runs\detect\val

[INFO] Результат валидации:

ultralytics.utils.metrics.DetMetrics object with attributes:

`

ap_class_index: array([0])

box: ultralytics.utils.metrics.Metric object

confusion_matrix: <ultralytics.utils.metrics.ConfusionMatrix object at 0x000001CA07CE1520>

curves: ['Precision-Recall(B)', 'F1-Confidence(B)', 'Precision-Confidence(B)', 'Recall-Confidence(B)']

curves_results: [[array([0, 0.001001, 0.002002, 0.003003, 0.004004, 0.005005, 0.006006, 0.007007, 0.008008, 0.009009, 0.01001, 0.011011, 0.012012, 0.013013, 0.014014, 0.015015, 0.016016, 0.017017, 0.018018, 0.019019, 0.02002, 0.021021, 0.022022, 0.023023,

0.024024, 0.025025, 0.026026, 0.027027, 0.028028, 0.029029, 0.03003, 0.031031, 0.032032, 0.033033, 0.034034, 0.035035, 0.036036, 0.037037, 0.038038, 0.039039, 0.04004, 0.041041, 0.042042, 0.043043, 0.044044, 0.045045, 0.046046, 0.047047,

0.048048, 0.049049, 0.05005, 0.051051, 0.052052, 0.053053, 0.054054, 0.055055, 0.056056, 0.057057, 0.058058, 0.059059, 0.06006, 0.061061, 0.062062, 0.063063, 0.064064, 0.065065, 0.066066, 0.067067, 0.068068, 0.069069, 0.07007, 0.071071,

fitness: np.float64(0.7064625670440197)

keys: ['metrics/precision(B)', 'metrics/recall(B)', 'metrics/mAP50(B)', 'metrics/mAP50-95(B)']

maps: array([0.70646])

names: {0: 'cat'}

nt_per_class: array([235])

nt_per_image: array([232])

results_dict: {'metrics/precision(B)': 0.8803669327587443, 'metrics/recall(B)': 0.8454938582792981, 'metrics/mAP50(B)': 0.9414858826183359, 'metrics/mAP50-95(B)': 0.7064625670440197, 'fitness': 0.7064625670440197}

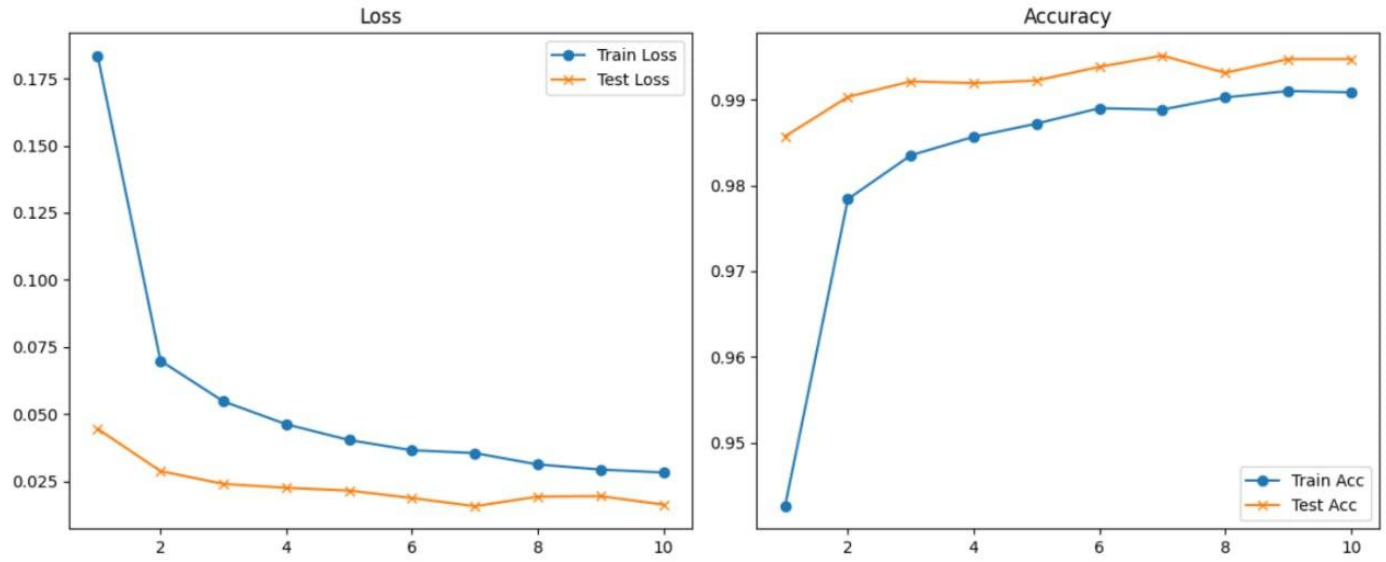
save_dir: WindowsPath('C:/Users/ASUS/PycharmProjects/oiis2/runs/detect/val')

speed: {'preprocess': 2.275566810818293, 'inference': 8.785608620984993, 'loss': 0.0003663795911867557, 'postprocess': 0.16235991367058636}

stats: {'tp': [], 'conf': [], 'pred_cls': [], 'target_cls': [], 'target_img': []}

task: 'detect'

`



Вывод: осуществил обучение нейросетевого детектора для решения задачи обнаружения заданных объектов.