perplexity

у меня есть python функция

Пример использования:

result = face_recognition_dct(

'test_face.jpg',

['reference1.jpg', 'reference2.jpg'],

block_size=8,

num_coefficients=15,

threshold=0.85

)

print("Тот же человек" if result else "Другой человек")

напиши программу, которая будет проверять эту функцию. у меня есть директория с изображениями 1_1.jpg, 2_1.jpg, ... 10_1.jpg, 11_2.jpg, где первая цифра для тебя не имеет значения и может быть любой (просканируй директорию чтобы правильно открывать файлы), а вторая цифра (то есть после _) означает номер человека

итого имеем 10 фото для каждого человека

первый тест: для каждого человека: используй 8 его фото как эталоны, а 2 как тестовые изображения. результат - процент правильных распознаваний

второй тест: для каждого человека используй 8 его фото как эталоны, а затем по одному фото каждого другого человека. результат - процент правильных отклонений

```
import os
import glob
from collections import defaultdict
from typing import List, Tuple
import random

def test_face_recognition(directory: str, face_recognition_func):
```

```
    """
    Тестирование функции распознавания лиц

    Args:
        directory: путь к директории с изображениями
        face_recognition_func: функция распознавания лиц
    """

    # Сканирование директории и группировка по ID человека
    image_files = glob.glob(os.path.join(directory, '*_*.jpg'))
    people_images = defaultdict(list)

    for img_path in image_files:
        filename = os.path.basename(img_path)
        # Извлекаем ID человека (цифра после _)
        person_id = filename.split('_')[1].split('.')[0]
        people_images[person_id].append(img_path)

    print(f"Найдено {len(people_images)} человек")
    for person_id, images in people_images.items():
        print(f"Человек {person_id}: {len(images)} фото")

    # Тест 1: распознавание своих фото
    print("\n" + "="*60)
    print("ТЕСТ 1: Распознавание фото того же человека")
    print("="*60)

    correct_recognitions = 0
    total_tests = 0

    for person_id, images in people_images.items():
        if len(images) < 10:
            print(f"Недостаточно фото для человека {person_id}")
            continue

        # Случайное разделение на эталоны и тестовые
        shuffled = images.copy()
        random.shuffle(shuffled)
        reference_images = shuffled[:8]
        test_images = shuffled[8:10]

        for test_img in test_images:
            result = face_recognition_func(
                test_img,
                reference_images,
                block_size=8,
                num_coefficients=15,
                threshold=0.85
            )

            if result:
                correct_recognitions += 1

            total_tests += 1
            status = "✓" if result else "✗"
            print(f"{status} Человек {person_id}: {os.path.basename(test_img)} -> {'Распо
```

```
accuracy_1 = (correct_recognitions / total_tests * 100) if total_tests > 0 else 0
print(f"\nРезультат теста 1: {correct_recognitions}/{total_tests} ({accuracy_1:.1f}%)

# Тест 2: отклонение чужих фото
print("\n" + "="*60)
print("ТЕСТ 2: Отклонение фото других людей")
print("="*60)

correct_rejections = 0
total_tests = 0

person_ids = list(people_images.keys())

for person_id in person_ids:
    images = people_images[person_id]
    if len(images) < 10:
        continue

    # 8 эталонных фото текущего человека
    shuffled = images.copy()
    random.shuffle(shuffled)
    reference_images = shuffled[:8]

    # Тестируем по одному фото каждого другого человека
    for other_person_id in person_ids:
        if other_person_id == person_id:
            continue

        other_images = people_images[other_person_id]
        if len(other_images) == 0:
            continue

        # Берем случайное фото другого человека
        test_img = random.choice(other_images)

        result = face_recognition_func(
            test_img,
            reference_images,
            block_size=8,
            num_coefficients=15,
            threshold=0.85
        )

        if not result:  # Правильно отклонено
            correct_rejections += 1

        total_tests += 1
        status = "✓" if not result else "✗"
        print(f"{status} Эталон: человек {person_id}, Тест: человек {other_person_id}

accuracy_2 = (correct_rejections / total_tests * 100) if total_tests > 0 else 0
print(f"\nРезультат теста 2: {correct_rejections}/{total_tests} ({accuracy_2:.1f}%)")

# Итоговая статистика
print("\n" + "="*60)
```

```
    print("ИТОГОВАЯ СТАТИСТИКА")
    print("="*60)
    print(f"Тест 1 (распознавание своих): {accuracy_1:.1f}%")
    print(f"Тест 2 (отклонение чужих): {accuracy_2:.1f}%")
    print(f"Общая точность: {((correct_recognitions + correct_rejections) / (total_tests


# Использование:
if __name__ == "__main__":
    from your_module import face_recognition_dct  # замените на реальный импорт

    directory = "path/to/your/images"  # укажите путь к директории
    test_face_recognition(directory, face_recognition_dct)
```

Программа выполняет следующее:

Сканирование директории

Автоматически находит все файлы вида `*_*.jpg` и группирует их по ID человека (цифра после underscore).

Тест 1: Распознавание своих фото

Для каждого человека случайным образом выбирает 8 эталонных фото и 2 тестовых. Проверяет, распознаёт ли функция тестовые фото как того же человека.

Тест 2: Отклонение чужих фото

Для каждого человека использует 8 его фото как эталоны, затем тестирует по одному случайному фото каждого другого человека. Проверяет, правильно ли функция отклоняет чужие лица.

Вывод результатов

Показывает детальные результаты каждого теста с визуальными индикаторами (✓/✗) и итоговые проценты точности.

Замените `from your_module import face_recognition_dct` на реальный импорт вашей функции и укажите правильный путь к директории с изображениями.

я хочу чтобы все параметры задавались в одном месте

```
import os
import glob
from collections import defaultdict
from typing import List, Tuple
import random

# ========================================
```

```
# НАСТРОЙКИ - задайте все параметры здесь
# ========================================
DIRECTORY = "path/to/your/images"  # путь к директории с изображениями
NUM_REFERENCE = 8  # количество эталонных фото
NUM_TEST = 2  # количество тестовых фото
BLOCK_SIZE = 8  # размер блока для DCT
NUM_COEFFICIENTS = 15  # количество коэффициентов
THRESHOLD = 0.85  # порог распознавания
RANDOM_SEED = 42  # seed для воспроизводимости результатов (None для случайности)
# ========================================


def test_face_recognition(directory: str, face_recognition_func):
    """
    Тестирование функции распознавания лиц

    Args:
        directory: путь к директории с изображениями
        face_recognition_func: функция распознавания лиц
    """

    if RANDOM_SEED is not None:
        random.seed(RANDOM_SEED)

    # Сканирование директории и группировка по ID человека
    image_files = glob.glob(os.path.join(directory, '*_*.jpg'))
    people_images = defaultdict(list)

    for img_path in image_files:
        filename = os.path.basename(img_path)
        # Извлекаем ID человека (цифра после _)
        person_id = filename.split('_')[1].split('.')[0]
        people_images[person_id].append(img_path)

    print(f"Найдено {len(people_images)} человек")
    print(f"Параметры: эталоны={NUM_REFERENCE}, тесты={NUM_TEST}, block_size={BLOCK_SIZE}
    print()
    for person_id, images in people_images.items():
        print(f"Человек {person_id}: {len(images)} фото")

    # Тест 1: распознавание своих фото
    print("\n" + "="*60)
    print("ТЕСТ 1: Распознавание фото того же человека")
    print("="*60)

    correct_recognitions = 0
    total_tests = 0

    for person_id, images in people_images.items():
        required_photos = NUM_REFERENCE + NUM_TEST
        if len(images) < required_photos:
            print(f"Недостаточно фото для человека {person_id} (требуется {required_photo
            continue

        # Случайное разделение на эталоны и тестовые
        shuffled = images.copy()
```

```
        random.shuffle(shuffled)
        reference_images = shuffled[:NUM_REFERENCE]
        test_images = shuffled[NUM_REFERENCE:NUM_REFERENCE + NUM_TEST]

        for test_img in test_images:
            result = face_recognition_func(
                test_img,
                reference_images,
                block_size=BLOCK_SIZE,
                num_coefficients=NUM_COEFFICIENTS,
                threshold=THRESHOLD
            )

            if result:
                correct_recognitions += 1

            total_tests += 1
            status = "✓" if result else "✗"
            print(f"{status} Человек {person_id}: {os.path.basename(test_img)} -> {'Распо

    accuracy_1 = (correct_recognitions / total_tests * 100) if total_tests > 0 else 0
    print(f"\nРезультат теста 1: {correct_recognitions}/{total_tests} ({accuracy_1:.1f}%)

    # Тест 2: отклонение чужих фото
    print("\n" + "="*60)
    print("ТЕСТ 2: Отклонение фото других людей")
    print("="*60)

    correct_rejections = 0
    total_tests_2 = 0

    person_ids = list(people_images.keys())

    for person_id in person_ids:
        images = people_images[person_id]
        if len(images) < NUM_REFERENCE:
            continue

        # эталонных фото текущего человека
        shuffled = images.copy()
        random.shuffle(shuffled)
        reference_images = shuffled[:NUM_REFERENCE]

        # Тестируем по одному фото каждого другого человека
        for other_person_id in person_ids:
            if other_person_id == person_id:
                continue

            other_images = people_images[other_person_id]
            if len(other_images) == 0:
                continue

            # Берем случайное фото другого человека
            test_img = random.choice(other_images)

            result = face_recognition_func(
```

```
                test_img,
                reference_images,
                block_size=BLOCK_SIZE,
                num_coefficients=NUM_COEFFICIENTS,
                threshold=THRESHOLD
            )

            if not result:  # Правильно отклонено
                correct_rejections += 1

            total_tests_2 += 1
            status = "✓" if not result else "✗"
            print(f"{status} Эталон: человек {person_id}, Тест: человек {other_person_id}

    accuracy_2 = (correct_rejections / total_tests_2 * 100) if total_tests_2 > 0 else 0
    print(f"\nРезультат теста 2: {correct_rejections}/{total_tests_2} ({accuracy_2:.1f}%)

    # Итоговая статистика
    print("\n" + "="*60)
    print("ИТОГОВАЯ СТАТИСТИКА")
    print("="*60)
    print(f"Тест 1 (распознавание своих): {accuracy_1:.1f}%")
    print(f"Тест 2 (отклонение чужих): {accuracy_2:.1f}%")
    total_correct = correct_recognitions + correct_rejections
    total_all = total_tests + total_tests_2
    overall_accuracy = (total_correct / total_all * 100) if total_all > 0 else 0
    print(f"Общая точность: {total_correct}/{total_all} ({overall_accuracy:.1f}%)")


# Использование:
if __name__ == "__main__":
    from your_module import face_recognition_dct  # замените на реальный импорт

    test_face_recognition(DIRECTORY, face_recognition_dct)
```

Теперь все параметры находятся в верхней части файла в секции **НАСТРОЙКИ**. Просто измените нужные значения в одном месте, и они будут использоваться во всей программе.

я хочу логирование только при ошибках и пиши результаты сравнения (число)

```
import os
import glob
from collections import defaultdict
from typing import List, Tuple
import random

# ==========================================
# НАСТРОЙКИ - задайте все параметры здесь
# ==========================================
DIRECTORY = "path/to/your/images"  # путь к директории с изображениями
NUM_REFERENCE = 8  # количество эталонных фото
```

```
NUM_TEST = 2  # количество тестовых фото
BLOCK_SIZE = 8  # размер блока для DCT
NUM_COEFFICIENTS = 15  # количество коэффициентов
THRESHOLD = 0.85  # порог распознавания
RANDOM_SEED = 42  # seed для воспроизводимости результатов (None для случайности)
# ==========================================


def test_face_recognition(directory: str, face_recognition_func):
    """
    Тестирование функции распознавания лиц

    Args:
        directory: путь к директории с изображениями
        face_recognition_func: функция распознавания лиц
    """

    if RANDOM_SEED is not None:
        random.seed(RANDOM_SEED)

    # Сканирование директории и группировка по ID человека
    image_files = glob.glob(os.path.join(directory, '*_*.jpg'))
    people_images = defaultdict(list)

    for img_path in image_files:
        filename = os.path.basename(img_path)
        # Извлекаем ID человека (цифра после _)
        person_id = filename.split('_')[1].split('.')[0]
        people_images[person_id].append(img_path)

    print(f"Найдено {len(people_images)} человек")
    print(f"Параметры: эталоны={NUM_REFERENCE}, тесты={NUM_TEST}, block_size={BLOCK_SIZE}

    # Тест 1: распознавание своих фото
    print("="*60)
    print("ТЕСТ 1: Распознавание фото того же человека")
    print("="*60)

    correct_recognitions = 0
    total_tests = 0

    for person_id, images in people_images.items():
        required_photos = NUM_REFERENCE + NUM_TEST
        if len(images) < required_photos:
            print(f"ОШИБКА: Недостаточно фото для человека {person_id} (требуется {requi
            continue

        # Случайное разделение на эталоны и тестовые
        shuffled = images.copy()
        random.shuffle(shuffled)
        reference_images = shuffled[:NUM_REFERENCE]
        test_images = shuffled[NUM_REFERENCE:NUM_REFERENCE + NUM_TEST]

        for test_img in test_images:
            result, similarity = face_recognition_func(
                test_img,
```

```
                reference_images,
                block_size=BLOCK_SIZE,
                num_coefficients=NUM_COEFFICIENTS,
                threshold=THRESHOLD
            )

            if result:
                correct_recognitions += 1
            else:
                # Логируем только ошибки
                print(f"✗ Человек {person_id}: {os.path.basename(test_img)} -> НЕ распоз

            total_tests += 1

    accuracy_1 = (correct_recognitions / total_tests * 100) if total_tests > 0 else 0
    print(f"\nРезультат теста 1: {correct_recognitions}/{total_tests} ({accuracy_1:.1f}%)

    # Тест 2: отклонение чужих фото
    print("="*60)
    print("ТЕСТ 2: Отклонение фото других людей")
    print("="*60)

    correct_rejections = 0
    total_tests_2 = 0

    person_ids = list(people_images.keys())

    for person_id in person_ids:
        images = people_images[person_id]
        if len(images) < NUM_REFERENCE:
            continue

        # эталонных фото текущего человека
        shuffled = images.copy()
        random.shuffle(shuffled)
        reference_images = shuffled[:NUM_REFERENCE]

        # Тестируем по одному фото каждого другого человека
        for other_person_id in person_ids:
            if other_person_id == person_id:
                continue

            other_images = people_images[other_person_id]
            if len(other_images) == 0:
                continue

            # Берем случайное фото другого человека
            test_img = random.choice(other_images)

            result, similarity = face_recognition_func(
                test_img,
                reference_images,
                block_size=BLOCK_SIZE,
                num_coefficients=NUM_COEFFICIENTS,
                threshold=THRESHOLD
            )
```

```
            if not result:  # Правильно отклонено
                correct_rejections += 1
            else:
                # Логируем только ошибки
                print(f"✗ Эталон: человек {person_id}, Тест: человек {other_person_id} -

            total_tests_2 += 1

    accuracy_2 = (correct_rejections / total_tests_2 * 100) if total_tests_2 > 0 else 0
    print(f"\nРезультат теста 2: {correct_rejections}/{total_tests_2} ({accuracy_2:.1f}%)

    # Итоговая статистика
    print("="*60)
    print("ИТОГОВАЯ СТАТИСТИКА")
    print("="*60)
    print(f"Тест 1 (распознавание своих): {accuracy_1:.1f}%")
    print(f"Тест 2 (отклонение чужих): {accuracy_2:.1f}%")
    total_correct = correct_recognitions + correct_rejections
    total_all = total_tests + total_tests_2
    overall_accuracy = (total_correct / total_all * 100) if total_all > 0 else 0
    print(f"Общая точность: {total_correct}/{total_all} ({overall_accuracy:.1f}%)")


# Использование:
if __name__ == "__main__":
    from your_module import face_recognition_dct  # замените на реальный импорт

    test_face_recognition(DIRECTORY, face_recognition_dct)
```

Теперь программа:

- Выводит только ошибки (когда свой человек не распознан или чужой ошибочно распознан)
- Показывает числовое значение сходства для каждой ошибки
- Сохраняет чистый вывод с итоговой статистикой

Важно: функция `face_recognition_dct` должна возвращать кортеж `(result, similarity)`, где `result` — булево значение, а `similarity` — числовое значение сходства.