*Ваня сидит за столом, читая книжку про OpenCV, изучая новый раздел - обработку видео. Он выглядит уставшим.*

Ваня (бормочет): Источником видеоданных может быть видеокамера или видеофайл…

Нейросеточка: Ваня, с тобой все хорошо? Ты уже сколько времени сидишь за книжкой?

Ваня: Да я только начал читать, но уже совсем нет сил, хочется спать.

Нейросеточка: Похоже, тебе нужно немного расслабиться. У Гигачата есть функция медитации. Давай попробуем вернуть твою энергию!

Рассказчик: Для этого в главном меню Гигачата выберите раздел «Полезные AI-функции» и нажмите на «Устройте себе сеанс медитации». Сначала вы можете ознакомиться с гайдом, а потом создать медитацию.

**Опишите, что сейчас чувствуете:** усталость, упадок сил

**Опишите желаемый результат медитации:** вернуть энергию и снова стать продуктивным

**Куда вы отправитесь медитировать:** в горы, горные реки

Затем выберите «Сгенерировать текст» и нажмите «Продолжить». Выберите звуковое сопровождение и немного подождите – медитация будет готова.

**Задание:** Создайте медитацию с помощью Гигачата, которая учтет ваши пожелания.

Ваня: Ух ты, это очень классно, так расслабляет и снимает усталость! Потрясающе себя чувствую!

Нейросеточка: Отлично, рада, что ты взбодрился! Теперь давай начнем изучение темы обработки видео в OpenCV. Может ты что-то конкретное хочешь узнать?

Ваня: Да, я прочитал, что можно использовать OpenCV для отображения видеоданных с web-камеры. Там используется какая-то детекция, но дальше я запутался.

Нейросеточка: Давай разбираться по порядку. Детекция в компьютерном зрении - это обнаружение объектов определенного типа на изображении или видео, например, лиц, жестов, движения. Для поиска и выделения объектов применяются различные специализированные алгоритмы и методы. Один из таких методов – каскады Хаара. *Классификатор Хаара — это метод, который используется для обнаружения объектов на изображениях.* Он был назван в честь венгерского математика Альфреда Хаара.

Ваня: Я ничего про него не слышал… А расскажи, как происходит обучение классификатора Хаара на видео.

Нейросеточка: Классификатор Хаара обучается на большом наборе положительных и отрицательных изображений. Положительные изображения содержат нужный объект, например, лицо, а отрицательные его не содержат. Во время обучения классификатор применяет несколько простых проверок — классификаторов последовательно, чтобы быстро и точно распознать объект. После обучения классификатор используется для анализа каждого кадра видео.

Ваня: Ну да, точно! Видео - это всего лишь набор кадров, являющихся изображениями. Теперь понятно, как это работает, не терпится применить метод самому!

Нейросеточка: Отлично, но начнем с анализа изображения и подсчета количества лиц на фото. Для этого создадим новую папку «Detection» и файл с расширением .py под названием «Count\_detection» внутри этой папки.

Рассказчик: Выполняем те же шаги, что делали в предыдущем занятии. Подробное описание этапов работы и картинку можно найти в материалах к занятию.

1. Создать новую папку “Detection”.
2. В VSC в левом верхнем углу нажать кнопку «Файл», выбрать «Открыть папку », затем выбрать на компьютере созданную папку «Detection» и нажать “Выбор папки”.
3. Создать в папке “Count\_detection” новый файл с расширением ***.py****,* а также добавить эту папку все необходимые для урока изображения.



Нейросеточка: Открываем visual studio code. Для начала импортируем модуль OpenCV. Затем в переменной img загружаем изображение и изменяем его в оттенки серого. Далее в переменной face\_cascade запишем каскад Хаара, необходимый для детекции лиц.

import cv2

img = cv2.imread('image.png')

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

Ваня: А что за конструкция в кавычках? Раньше не встречал такое расширение.

Нейросеточка: Это название файла, который содержит предобученный каскад Хаара для обнаружения лиц в изображениях и видео. Файлы с расширением .xml используются OpenCV для загрузки различных классификаторов, таких как детектор лиц, глаз, улыбок.

Продолжим. Создаем переменную “faces”, в которой будут храниться координаты обнаруженных лиц на изображении. Затем подсчитываем количество этих найденных лиц и выводим результат.

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

num\_people = len(faces)

print("Number of people: ", num\_people)

Рассказчик: В консоли вы увидите надпись “Number of people: 3”, которая отображает количество обнаруженных лиц на изображении.

**Задание:** Выберите фото с семьей или с друзьями и используйте предложенный код для подсчета количества лиц.

Ваня: Получилось, но только странно! На фотографии 4 человека, а обнаружилось только 3… Может, это связано с тем, что девочка и ее мама сидят в одинаковой красной одежде?

Нейросеточка: Возможно, алгоритм детекции лиц иногда путает людей, которые близко расположены или имеют похожий цвет одежды. Для улучшения точности можно настроить параметры алгоритма или использовать более сложные модели обучения.

Ваня: Понял, спасибо! Готов работать с видео!

Нейросеточка: Отлично! Давай создадим новый файл с расширением .py «Detection» и начнем с похожих строк кода, как в предыдущем примере.

import cv2

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + \

'haarcascade\_frontalface\_default.xml')

Ваня, теперь выбираем источник видеоданных. Если ты все так же хочешь использовать веб-камеру, мы сможем протестировать детекцию лиц прямо на тебе! Если перехотел, то используем видеофайл, который заранее тебе подготовила.

Ваня: Хочу попробовать все таки через web-камеру!

# раскомментируйте следующую строчку для чтения видео с веб-камеры

cap = cv2.VideoCapture(0)

# раскомментируйте следующую строчку для чтения видео из файла

#cap = cv2.VideoCapture('video.mp4')

Рассказчик: Помните, что одновременно можно использовать только одну из строчек, начинающихся с “cap = ..”

Ваня: Тогда, первый вариант беру к себе в код! А мне вот интересно, почему при вызове функции для использования web-камеры стоит в скобках 0?

Нейросеточка: Отличный вопрос! Давай отправим этот фрагмент Гигачату и спросим его.

**Промпт:** Почему в функции cv2.VideoCapture() используется аргумент 0 для захвата видео с веб-камеры?  **Результат:** Аргумент 0 в функции cv2.VideoCapture() используется для указания источника видео. В данном случае, когда мы хотим получить видео с веб-камеры, аргумент 0 указывает на то, что источником является первая доступная веб-камера. Если у вас есть несколько веб-камер, вы можете использовать аргументы 1, 2 и т.д. для выбора конкретной камеры.

**Задание:** При помощи Гигачата узнайте, какие форматы видео поддерживает OpenCV при использовании функции cv2.VideoCapture().

Ваня: Понятно, ну у меня одна камера, поэтому оставлю 0.

Нейросеточка: Хорошо, что разобрались! Давай теперь попробуем написать цикл, который будет считывать по очереди каждый кадр из видеопотока и анализировать его.

while True:

# считываем кадр

\_, img = cap.read()

# конвертируем в черно-белый формат

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# обнаруживаем лица

faces = face\_cascade.detectMultiScale(gray, 1.1, 4)

Ваня: Нейросеточка, а ты случайно не ошиблась в третье строке?

Нейросеточка: Нет, я не ошиблась. Метод cap.read() возвращает два значения: логическое True/False, которое указывает, успешно ли считался кадр, и сам кадр. Нам нужен только кадр, поэтому мы используем \_ для игнорирования первого значения. Если бы мы написали просто img = cap.read(), это бы не сработало, так как метод возвращает два значения, а не одно. Переменные обычно обозначаются именем \_, когда мы их не используем.

Продолжим писать внутри цикла. Следующая часть потребует особого внимания, потому что в этом цикле мы рисуем bounding box для лица. *Bounding box — это ограничивающий прямоугольник, который обводит обнаруженный объект на изображении.*

# обводим лица

for (x, y, w, h) in faces:

cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

Ваня: О, а я знаю, каким цветом будет этот прямоугольник! Он будет синим в цветовой модели BGR! ! Еще предполагаю, что x, y, w и h — это координаты, как в математике: h — это высота, правильно?

Нейросеточка: Верно, Ваня! x и y — это координаты верхнего левого угла прямоугольника. w — это ширина, а h — высота. Координаты нижнего правого угла вычисляются как x + w, y + h. Функция cv2.rectangle() рисует прямоугольник, для этого ей нужно само изображение, координаты верхнего левого угла, нижнего правого, цвет обводки и толщина линии. Эти параметры можно настроить по собственному желанию, но пока давай оставим все как есть.   
А теперь осталось показать изображение, то есть видео, в отдельном окне.

Рассказчик: Это окно будет открыто, пока вы не нажмете клавишу ESC на клавиатуре. Каждые 30 миллисекунд проверяется, была ли нажата клавиша с кодом 27, то есть ESC и если да, то цикл прерывается.

# показываем изображение

cv2.imshow('img', img)

# ждём 30 миллисекунд для смены следующего кадра и проверяем нажатие клавиши ESC

k = cv2.waitKey(30) & 0xff

# если код клавиши равен 27 (код ESC) то останавливаем цикл.

if k==27:

break

Нейросеточка: И наконец, после цикла дописываем команду cap.release(), останавливающую приём видеопотока.  
 # перестаём принимать видео

cap.release()

Рассказчик: Подробную инструкцию и весь код вы найдете в материалах к занятию.

**Задание:** Выполните детекцию лица в видеопотоке, используя любое видео. Подсказка: Нужно раскомментировать вторую строчку “cap”, а первую закомментировать.   
Ваня: Классно получилось! Теперь я могу анализировать различные видео с помощью детекции лиц!

**Сегодня на занятии вы:**

* Познакомились с медитацией от Гигачата.
* Узнали, что такое детекция компьютерного зрения.
* Познакомились с классификатором Хаара для обнаружения лиц.
* Приобрели навык подсчета количества лиц на фотографии с помощью классификатора Хаара.
* Научились обрабатывать видеопотоки с веб-камеры и видеофайла с использованием библиотеки OpenCV.

Увидимся на следующем занятии! До скорой встречи!