Национальный иследовательский ядерный университет
“Московский инженерно-физический институт”
Предуниверситарий НИЯУ МИФИ Университеский
лицей 1511

# *Психоэмоциональный тренажер*

## *Psychoemotional trainer*

**Подготовили**: *Пустовит Владислав, Коняхина Изабелла*

# Проблема и актуальность

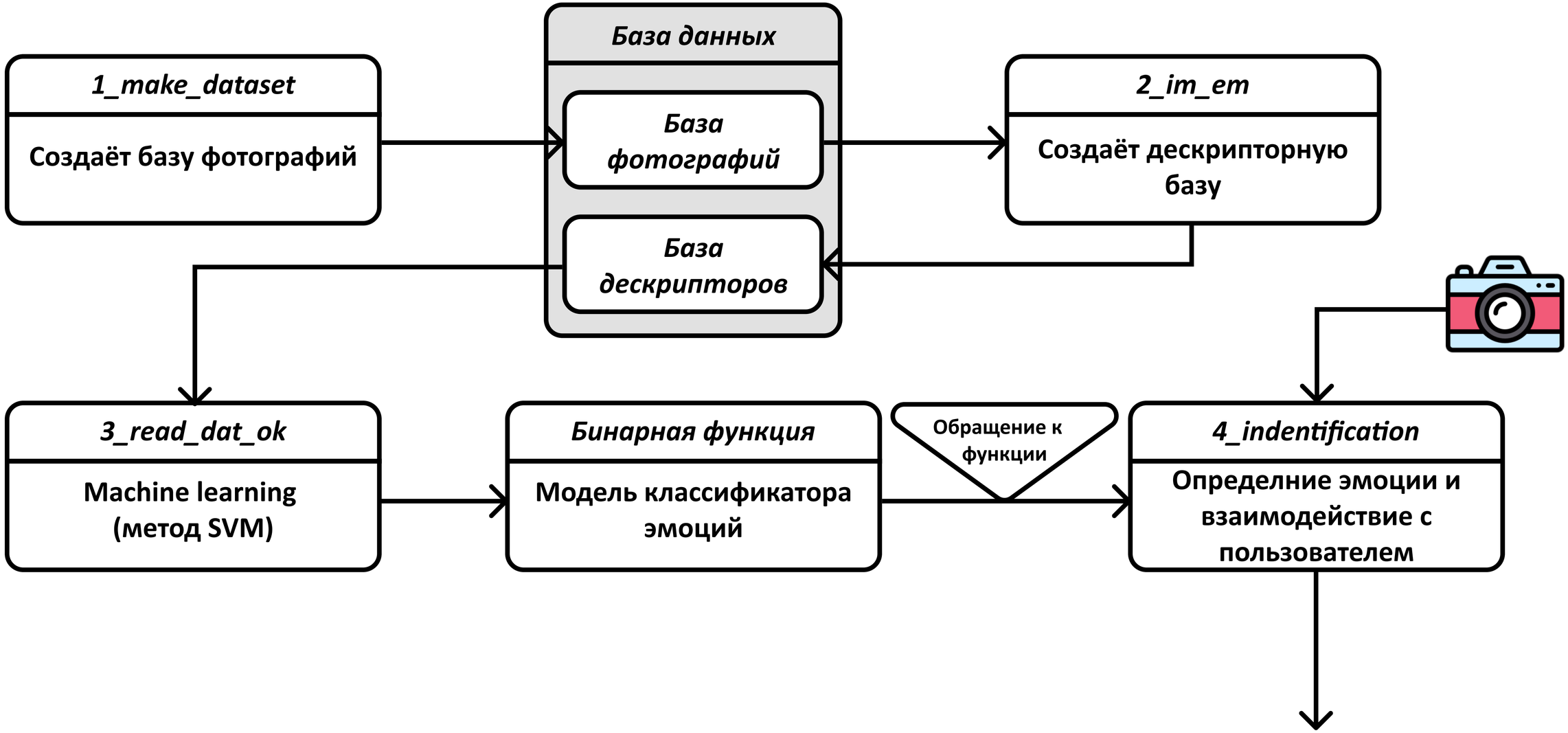
В настоящее время появляется все больше людей с нарушением мимики.

Единственными способами реабилитации является ЛФК или дорогостоящие процедуры в клинике, что не всегда является доступным.

# Цель

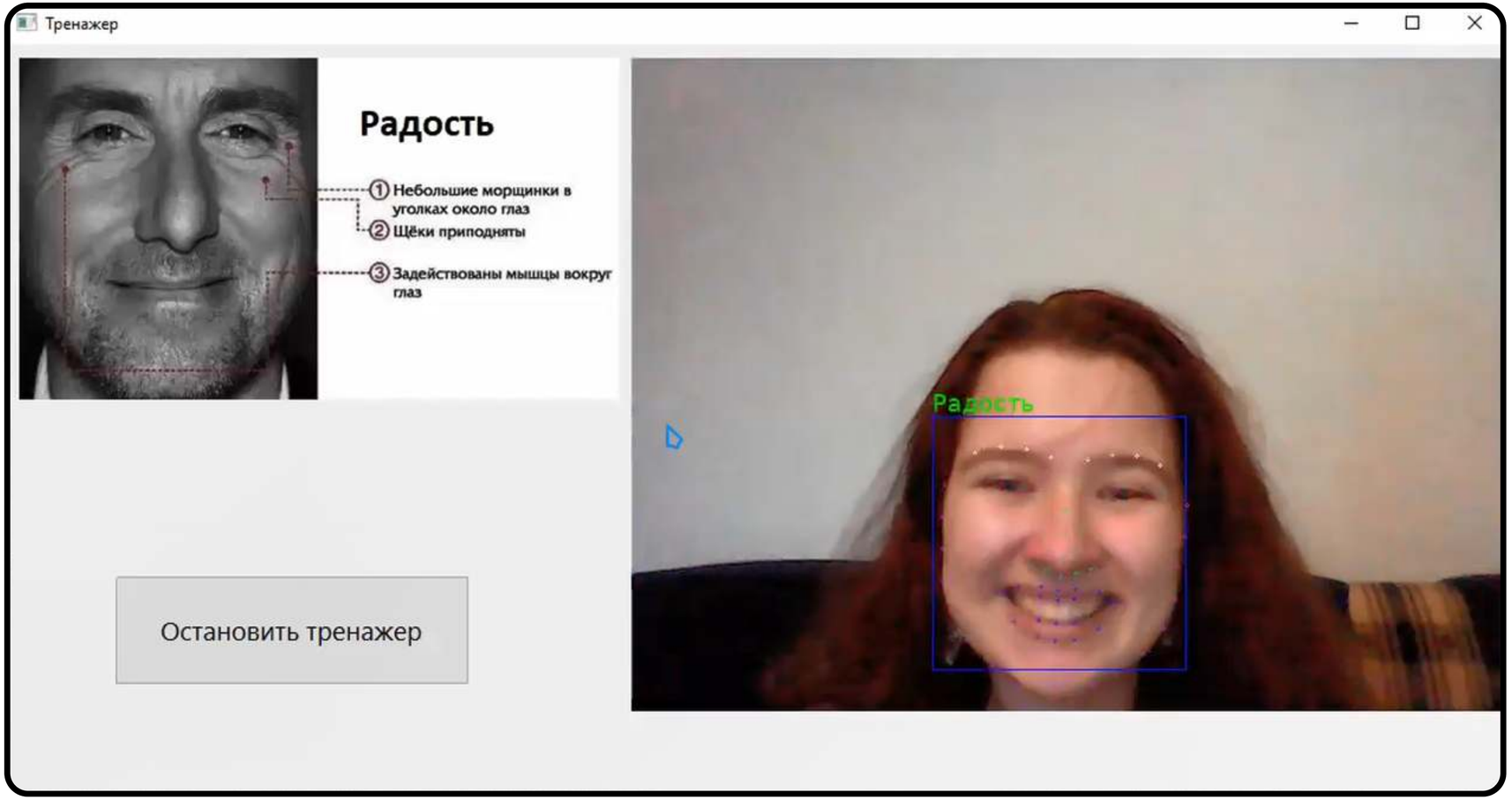
Создать доступный АПК для реабилитации мимических мышц

# Тренажёр. Особенности



# Тренажёр. Интерфейс

Интерфейс сделан с помощь библиотеки QT5 для Python3 PySide6



# Тренажёр. Примеры кода

```
while(True):
    ret, img = cam.read()
    # Переводим изображение в оттенки серого
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Выделяем лицо в прямоугольную область
    faces = face_detector.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        count += 1
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
        # Сохраняем захваченное изображение в папку dataset
        cv2.imwrite(directory.format(face_id, int(time.time() * 1e9)), gray[y:(y + h), x:(x + w)])
        cv2.imshow('image', img)
    if cv2.waitKey(70) & 0xff == 27:
        # Для завершения работы программы нажмите «ESC»
        break
    elif count >= 2:
        # Остановка программы после создания заданного количества кадров
        break
    print(count)
```

***1_make_dataset***

# Тренажёр. Примеры кода

```python
# Функция, рассчитывающая два вида дискрипторов лица
def get_landmarks(image):
    # Функция выделяет лицо в прямоугольник
    detections = detector(image, 1)
    # Цикл по всем найденым на изображении лицам
    for k, d in enumerate(detections):
        # Рисуем лицевые ориентиры с помощью класса предиктора
        # (Возврат координат точек на лице)
        shape = predictor(image, d)
        # Получаем дискрипторы лица
        face_descriptor = facerec.compute_face_descriptor(image, shape)
        # Список для размещения координат точек лица по оси X
        xlist = []
        # Список для размещения координат точек лица по оси Y
        ylist = []
        # Сохраняем координаты X и Y в двух списках
        for i in range(0, 68):
            # Список для X
            xlist.append(float(shape.part(i).x))
            # Список для Y
            ylist.append(float(shape.part(i).y))
            # Берем 30 точку на носу как центральную на лице
            meannp = np.asarray((shape.part(30).x, shape.part(30).y))
            # Создаем список для записи дискрипторов
            landmarks_vectorised = []
            # Расчитываем дискрипторы 1
            for w, z in zip(xlist, ylist):
                # Создаем масив из координат векторов расстояний
                coornp = np.asarray((z, w))
                # Рассчитываем расстояние от центральной точки до данной
                dist = np.linalg.norm(coornp - meannp)
                # Добавляем в список дискрипторы1
                landmarks_vectorised.append(dist)
                # Масштабируем параметры изображения
            landmarks_vectorised[:] = landmarks_vectorised[:] / landmarks_vectorised[27]
    # Добавляем значения переменных, если нет выделенных лиц на экране
    if len(detections) == 0:
        xlist = 0
        ylist = 0
        meannp = np.asarray((0, 0))
        landmarks_vectorised = 0
        face_descriptor = 0
    # Возвращаем дискрипторы1, количество выделенных лиц
    return xlist, ylist, meannp, landmarks_vectorised, face_descriptor
```

***2_im_em***

# Тренажёр. Примеры кода

```python
# Запуск цикла, пробегающегося по списку file_names,
# доставая и его индекс i, и то, что в нем находится
for i, file_name in enumerate(file_names):
    # открываем каждый из этих файлов с разрешением на чтение
    f = open(dis_path + file_name, 'r')
    # Считываем параметры в формате CVS
    reader = csv.reader(f)
    # Пробегаемся по считанным данным
    for row in reader:
        # Добавляем в список считанные данные
        dat.append(row)
        # Добавляем в список номера всех эмоций
        emotion_number.append(i + 1)
    # Закрываем файлы
    f.close
```

```python
# Индекс в списке, соответствующий 80-ти процентам
# (0.8 - доля от общей выборки на обучаение)
pp = int(len(dat) * 0.8)
# Смешивает полученные данные одинаковым способ,
# сохраняя соответствие номера эмоций и их параметров
dat, emotion_number = shuffle(dat, emotion_number, random_state=0)
# Обучающая выборка
learning_dat, learning_emotion_number = dat[:pp], emotion_number[:pp]
# Тестирующая выборка
test_dat, test_emotion_number = dat[pp+1:], emotion_number[pp+1:]
# Задаем параметры для метода обучения Support Vector Machine
clf = svm.SVC(C=2.0, gamma=2.0, kernel='poly', degree=3)
# Запускаем метод обучения Support Vector Machine
clf.fit(learning_dat, learning_emotion_number)
# Доля правильно определенных эмоций на фотографиях из базы
succesfull_prediction = clf.score(test_dat, test_emotion_number)
# Выводим полученную долю в консоль
print (succesfull_prediction)
# Сохроняем полученные данные в файл svm_dat
f = open(dat_path + 'svm_dat.dat', 'wb')
# Запись на диск
pickle.dump(clf, f)
f.close
```

***3_read_dat_ok***

# Тренажёр. Примеры кода

```python
def rectangle_face(self):
    detection = self.detector(self.image, 1)
    if len(detection) == 0:
        self.main_window.ui.label_2.setText('Нет никого в кадре')
        self.em_n = 0
    for k, d in enumerate(detection):
        if self.i % 3 == 0:
            self.shape = self.predictor(self.image, d)
            face_description = self.face_rec.compute_face_descriptor(self.image, self.shape)
            dat = [face_description]
            self.em_n = self.clf.predict(dat)
        utils.rectangle_face(self.image, detection, self.shape)
        img_pil = Image.fromarray(self.image)
        draw = ImageDraw.Draw(img_pil)
        if self.em_n == self.e:
            self.emotion_result += 1
            draw.text((detection[0].left(), detection[0].top() - 20), self.emotions[int(self.em_n) - 1],
                      font=self.font, fill=(0, 255, 0))
        else:
            draw.text((detection[0].left(), detection[0].top() - 20), self.emotions[int(self.em_n) - 1],
                      font=self.font, fill=(255, 0, 0))
        self.image = np.array(img_pil)
    cv2.waitKey(10)
```

```python
if self.i == self.nn:
    self.emotion_result = self.emotion_result / self.nn
    if self.emotion_result >= self.P_em:
        self.main_window.ui.label_2.setText(
            "Молодец:  " + "{0:.2f}".format(
                self.emotion_result * 100) + '%')
        self.emotion(self.e, 'stop')
    else:
        self.main_window.ui.label_2.setText(
            "Попробуй ещё:  " + "{0:.2f}".format(
                self.emotion_result * 100) + '%')
    self.i = 0
else:
    self.i += 1
```

***4_identification***

# Эксперимент

| | *Радость* | *Удив.* | *Грусть* | *Злость* | *Отвращ.* | *Презрен.* | *Страх* |
|---|---|---|---|---|---|---|---|
| *Сестра без нарешений* | *0.985* | *0.893* | *0.896* | *0.936* | *0.925* | *0.847* | *0.948* |
| *Сестра с ДЦП (до АПК)* | *0.881* | *0.836* | *0.639* | *0.763* | *0.625* | *0.385* | *0.575* |
| *Сестра с ДЦП (после АПК)* | *0.911* | *0.915* | *0.846* | *0.894* | *0.796* | *0.837* | *0.692* |

# Итог

Мы создали АПК, которым может воспользоваться почти любой нуждающийся в нём человек.

К сожалению, наш апаратно-программный комплекс не может помочь людям с акенитическим мутизмом.