Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования «Петрозаводский государственный университет»

Институт математики и информационных технологий
кафедра теории вероятностей и анализа данных

(подпись соискателя)

Чернышов Александр Сергеевич

Выпускная квалификационная работа бакалавра

Машинное обучение в задаче классификации эмоций

Направление 09.03.04 Программная инженерия

Научный руководитель:
к.т.н., доцент Н. В. Смирнов

(подпись руководителя)

Петрозаводск — 2022

Содержание

Введение

Цель выпускной работы: построение модели классификации эмоционального состояния человека на изображении.

Для достижения этой цели были выделены следующие задачи:

1. проанализировать существующие решения;
2. разработать и протестировать классификаторы на основе методов машинного обучения;
3. разработать и протестировать классификаторы на основе методов глубокого обучения;
4. разработать и протестировать каскады моделей;
5. выбрать классификатор, обеспечивающий максимум метрик классификации.

Распознавание эмоционального состояния человека на фотографиях и в видеопотоке является одной из актуальных задач. Человеческие эмоции выражаются мимикой, при этом каждой эмоции соответствует характерный набор положений антропометрических точек (ключевых точек на лице человека). Было обнаружено, что выражение основных эмоций на лицах людей не зависит от расы, культуры, пола или возраста человека [1].

На лице человека выделяются 7 основных эмоций: злость, страх, отвращение, удивление, счастье, грусть и нейтральная эмоция. Определение эмоциональной реакции человека может использоваться в различных интеллектуальных системах. Системы классификации эмоций установленные в автомобилях могут использоваться в распознавании усталости человека. Такие системы позволят избежать аварии, вызванные невнимательностью или плохим самочувствием водителя. Так же данные системы могут применяться в медицине и психологии для определения поддельных или подавленных эмоций.

1 Обзор публикаций, программных решений и наборов данных

1.1 Обзор публикаций по теме классификации эмоций

В представленной работе [2] автор предлагает использовать метод опорных векторов DLSVM — широко используемую альтернативу softmax для классификации. В качестве классификатора использовалась сверточная нейронная сеть. Автор показал, что DLSVM работает лучше, чем softmax, на черно-белых изображениях лиц из набора данных FER-2013 [3]. При этом переход с softmax на SVM прост и полезен для задач классификации. Точность для 7 эмоций на выбранном наборе данных составила 71,2%.

В данной статье [4] авторы решают задачу классификации эмоций на основе аудио-видео файлах из набора данных AFEW. Для анализа аудиодорожки авторы использовали как речевую спектрограмму, так и логарифмическую спектрограмму Mel. Для анализа эмоций на изображениях использовались различные предварительно обученные сверточные нейронные сети.

Для слияния результатов полученных из анализа аудиодорожек и изображений, авторы используют методы конкатенации и механизмы внимания для выделения важных эмоциональных признаков. Точность на обучающей выборке составила 65,5 % и 62,48 % на тестовом наборе.

В следующей публикации [5] авторы предлагает отказаться от использования паддинга и пулинга в связи с тем, что паддинг несет за собой лишнюю синтетическую информацию для изображений, а пулинг приводит к увеличению светлых областей на изображении. В качестве альтернативы авторы предлагают использовать три основных слоя Feature Arrangement (FA), De-Albino (DA) и Sharing Affinity (SA).

1. Слой Feature Arrangement является вспомогательным и при повреждении ключевой информации во время эрозии, помещает ее на края карты признаков.

2. Слой De-Albino сверточный слой, уменьшающий веса пикселей подвергающихся эрозии, тем самым, уменьшая их влияние на результат классификации.

3. Слой Sharing Affinity разделяет черты лица на две части и при этом сохраняет общие признаки для всех лиц из набора изображений, тем самым делая обучение наиболее

эффективным.

При тестировании модель показала следующие результаты:

- 90,42% на наборе данных RAF-DB;
- 65,2% на наборе данных AffectNet;
- 58,71% на наборе данных SFEW.

Авторы, в предложенной статье [6], описывают построение классификатора с двумя механизмами внимания:

1. Первый механизм для захвата признаков, посредством анализа зависимости выборочных областей изображения выражения лица, так что обновление параметров сверточных слоев при низкоуровневом изучении признаков упорядочено.
2. Второй механизм внимания визуального преобразователя использует последовательность визуальных семантических токенов (сгенерированных из признаков пирамиды блоков высокого сверточного слоя) для изучения глобального представления модели лица.

При тестировании классификатор показал 100% точности для 6 и 7 эмоций на наборах данных CK + без каких-либо дополнительных обучающих данных. На наборах данных FER+ и RAF-DB, точность 90,04% и 88,26% соотвественно.

В работе [7] авторы используют SPDNet для решения задач распознавания выражений лица. SPDNet, примененная к ковариации сверточных признаков, может более эффективно классифицировать выражения лица. Двухслойные сети способны лучше фиксировать искажения лицевых ориентиров. Точно так же ковариационная матрица, вычисленная из векторов признаков изображения, использовалась в качестве входных данных для SPDNet для задачи распознавания выражения лица на основе видео.

При тестировании модель показала следующие результаты:

- 87% на наборе данных RAF-DB;
- 58,4% на наборе данных SFEW;
- 39,78% на наборе данных AFEW;

Авторы, в предложенной статье [8], представляют схему Vision Transformer (ViT) + Squeeze and Excitation, которая (SE) оптимизирует обучение ViT с помощью блока внимания, называемого «Сжатие и возбуждение». Данный подход улучшает производительность ViT в задаче классификации эмоций на изображениях, и, кроме того, повышает надежность модели.

- Vision Transformer — для преобразования изображений в последовательность патчей используется линейную проекцию, а для классификации используются только вектор классов токенов. Изображения разбиваются на фрагменты и преобразуются в вектора, к полученным векторам присоединяется вектор классов.

- Squeeze and Excitation — к результату преобразования каждого фрагмента изображения добавляются дополнительные параметры, с возможностью обучения, для построения взаимосвязей между результатами свертки.

По результатам тестирования точность для CK+ составила 99,8%, для JAFFE 94,83%, для RAF-DB 87,22% и для SFEW 54,29%.

В работе [9] представлен классификатор эмоций человека DeXpression — сверточная нейронная сеть, которая требует небольших вычислительных усилий по сравнению с современными архитектурами искусственных нейронных сетей. Для его создания была введена новая составная структура FeatEx (feature-extraction). Модуль распознавания состоит из нескольких сверточных слоев разного размера, а также слоев Max Pooling и ReLU. Результаты 10-кратной кросс-валидации дают в среднем точность распознавания 99,6% для набора данных CKP и 98,36% для набора данных MMI.

Архитектура DeXpression сети состоит из четырех блоков:

- В первом блоке производится предварительная обработка, средствами свертки, уменьшения размерности и нормализации.

- Во втором и третьем блоке происходит выборка значимых признаков.

- Четвертый блок выполняет классификацию эмоций.

1.2 Обзор существующих решений

1.2.1 Multi-task EfficientNet-B2

Данный классификатор представляет из себя легковесную модель, позволяющую с высокой точностью распознавать выражения лица на изображениях и в видеопотоках. В отличие от существующих моделей, в данном решении, обеспечивается дополнительная устойчивость к нахождению и выравниванию лица.

При этом наблюдается не только высокая точность, но и отличное быстродействие и размер модели. Следовательно, модели, обученные предлагаемым подходом, могут быть использованы для классификации во встроенных системах, например, в мобильных интеллектуальных системах.

При разработке данного классификатора использовались легковесные сверточные сети MobileNet [10], EfficientNet [11], RexNet [12], с помощью которых была получена возможность детектировать лица, распознавать эмоции, определять пол, возраст и этнические принадлежности [13]. Метод детекции лиц обучался и тестировался на наборе данных VGGFace2, классификатор эмоций обучался и тестировался на фотографиях из набора данных AffectNet. Точность для 7 эмоций на выбранном наборе данных составила 66,34

1.2.2 EmoPy

EmoPy – совокупность методов для распознавания человеческих эмоций на языке Python. EmoPy содержит несколько моделей с открытым исходным кодом, показывающий точность 92% для трех эмоций и 68% для семи. Модели можно использовать как предобученными, так и повышать точность для конкретного датасета [14].

Цель проекта — сделать алгоритмы по распознанию эмоций человека [15] на изображениях бесплатными, открытыми, простыми в использовании и легко интегрируемые в различные приложения. При создании EmoPy рассматривалось несколько архитектур, однако наилучшие результаты были получены при использовании несколько сверточных нейроны сетей. Точность на которых и составляет 92% для трех эмоций и 68% для семи на наборах данных Microsoft FER2013 и Cohn-Kanade [16]. Однако при использовании новых изображений, классификатор работает хуже — ошибка достигает 37%.

1.2.3 FaceNet2ExpNet

FaceNet2ExpNet — двухэтапный алгоритм распознавания эмоций человека на изображениях [17].

Работа данного классификатора состоит из двух этапов.

- На первом этапе модель детектирует лица на изображениях сети с высокой точностью на основе уже настроенной сверточной нейронной сети.
- На втором этапе данные передаются в другую сеточную сеть, которая позволяет распознавать эмоции. Процесс обучения состоит из заморозки части весов сети и присоединении новых полносвязных слоев, для сохранения полезной информации о структуре лиц.

Для детекции ключевых точек используются алгоритмы Виолы-Джонса и InterFace. Полученный классификатор тестировался на различных наборах данных и показал следующие результаты: 98,6% для 6 эмоций, 96,8% для 8 эмоций (набор данных CK+); 87,71% для 6 эмоций (набор данных Oulu CAS); 88,9% для 7 эмоций (набор данных TFD); 55,15% для 7 эмоций (набор данных SFEW).

Предложенные решения используют ресурсоемкие вычисления. В работе предложено использование методов машинного обучения и полносвязных нейронных сетей — перцептронов, которые анализируют местоположение антропометрических точек на лице человека.

1.3 Наборы данных

Для работы было использовано несколько наборов данных: KDEF[18] и RaFD[19].

KDEF включает в себя 4900 фотографий 7 базовых классов эмоций. Данный датасет был собран Лундквистом Д., Фликтом А. и Оманом А. из Каролинского института, отдела клинической неврологии, секции психологии. Каждому изображению из набора присвоено уникальное названое состоящее из восьми символов, которое строится следующим образом:

1. первый символ отвечает за номер фотосессии, A — первая фотосессия, B — вторая фотосессия;
2. второй символ отвечает за пол: M - мужской, F - женский;
3. третий и четвертый символы, это идентификационный номер человека от 01 до 35;
4. пятый и шестой символы отвечают за эмоцию: NE – нейтральность, AN – злость, HA – счастье, DI – отвращение, SU – удивление, AF – страх, SA – грусть;

5. седьмой и восьмой символы отвечают за положение лица: FL – полный левый профиль, HL – половина левого профиля, S – фронтальное положение, HR – половина правого профиля, FR – полный правый профиль.

Для исследования из данного наборы выбраны только фронтальные лица каждого класса – 980 изображений. С помощью скрипта отобраны только те файлы, которые содержат символ S на седьмой позиции.

RaFD представляет собой набор изображений 67 классов, включая мужчин и женщин европеоидной расы, детей европеоидной расы, мальчиков и девочек и марокканских голландцев, демонстрирующих 8 эмоциональных выражений. RaFD создан по инициативе Института поведенческих наук Университета Радбауд в Неймегене, расположенного в Неймегене (Нидерланды), и может свободно использоваться для некоммерческих научных исследованиях в официально аккредитованном университете.

Название изображения строится следующим образом:

1. первые символы – название набора данных, после названия угол камеры относительно лица человека;

2. отделяются нижними подчеркиваниями номер человека, наименование его национальности и пол;

3. последние символы – наименование выраженной эмоции и направление взгляда;

В отличии от KDEF в RaDF присутствует эмоция «презрение», которая не будет рассмотрена в работе. Из данного набора было отобранно 1267 изображений с фронтальными лицами.

В общей сложности, для исследования было отобрано 2247 изображений. Для обучения модели использовано 2107 изображений, а остальные 140 – на тестирование (по 10 изображений каждого класса из каждого набора данных). Для этого были созданы две директории train и test, при этом все эмоции отсортированы по соответствующим им директориям.

2 Машинное обучение в задаче классификации эмоций

2.1 Предобработка данных

2.1.1 Определение и локализация лица

На первом этапе производится определение и локализация лица на изображении. Для этой задачи используется библиотека YOLOv5. YOLOv5 — это семейство архитектур и моделей обнаружения объектов, предварительно обученных на наборе данных COCO с открытым исходным кодом [20]. Предлагает методы искусственного интеллекта для компьютерного зрения.

Все найденные лица помещаются в прямоугольную область. Необходимо увеличить границы на 15 пикселей, связи с тем, что детектор незначительно обрезает найденные лица. После этого данные фрагменты вырезаются из исходных изображений для уменьшения времени обработки изображений в дальнейшем.

2.1.2 Антропометрические точки

На втором этапе на изображениях выделяются антропометрические точки лица. Для детекции антропометрических точек используется библиотека dlib [21]. Данная библиотека позволяет детектировать 68 антропометрических точек (рисунок 1), возвращая их координаты в качестве x, y.

Рис. 1: Результат работы dlib

2.1.3 Способы векторизации антропометрических точек

В качестве признаков для обучения и тестирования можно передавать координаты самих антропометрических точек, в таком случае входной вектор будет состоять из пар 68 координат x и 68 координат y для одного изображения. Работа классификатора с таким большим набором данных может быть слишком медленной.

Для уменьшения размерности входных данных было рассмотрено несколько способов их предобработки:

1. Использовать сумму длин векторов от левого верхнего угла лица до антропометической точки (рисунок 2), при этом эти расстояния масштабировать относительно размера изображения:

```
result = []
for point in face.landmarks:
    result.append((point[0] - face.zero_x) /face.width
    + (point[1] - face.zero_y) / face.height)
```

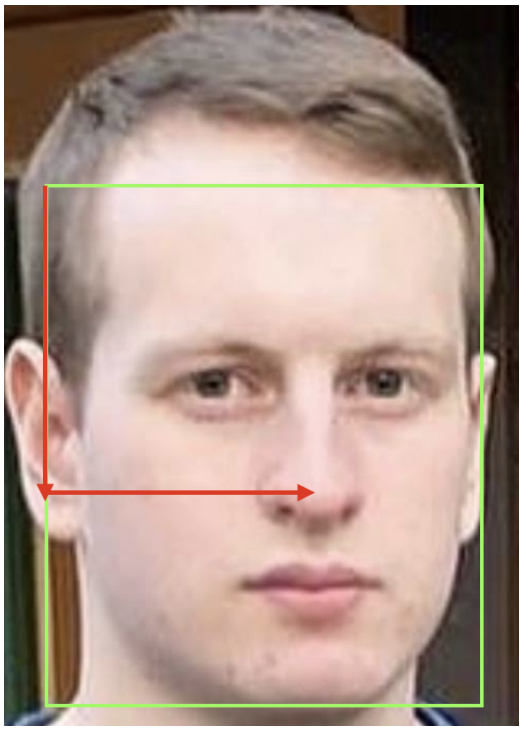

Рис. 2: Сумма векторов

2. Использовать длину вектора от левого верхнего угла лица до антропометической точки (рисунок 3), при этом это расстояние масштабировать относительно размера изображения:

```
result = []
for point in face.landmarks:
    x = (point[0] - face.zero_x) / face.width
    y = (point[1] - face.zero_y) / face.height
    result.append((x**2 + y**2)**(1/2)
```

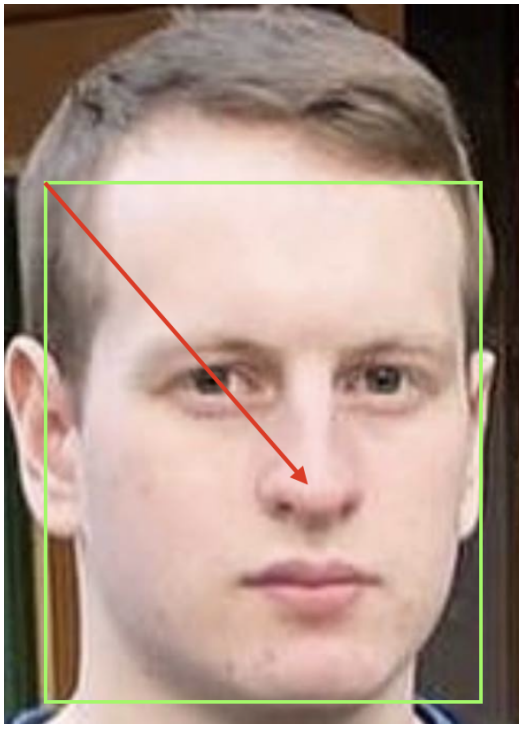

Рис. 3: Вектор до точки

3. Рассматривать расстояния между определенными точками. В качестве расстояний использовались 4 вертикали и 3 горизонтали (рисунок 4).

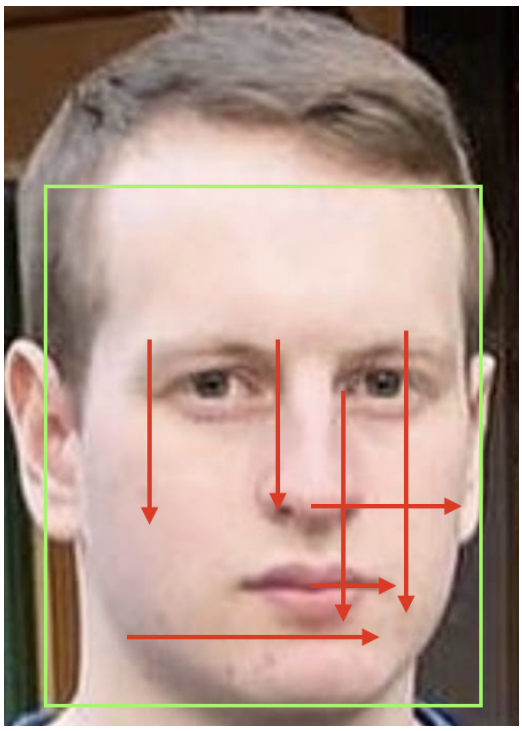

Рис. 4: Расстояния между выбранными точками

result – список расстояний, face.lendmarks – точки определенного лица, points – одна точка из face.lendmarks, points[0], points[1] – координаты точки x, y соответственно, face.width, face.height – размеры лица, face.zero_x, face.zero_y – координаты левого верхнего угла лица.

2.2 Применение известных методов машинного обучения

2.2.1 Оценка качества моделей

Для оценки качества и сравнения различных моделей будем использовать специальные метрики.

Пусть:

1. True Positive (TP) – количество элементов выборки, которые классификатор верно классифицировал как объекты целевого класса;

2. False Positive (FP) – количество элементов выборки, которые классификатор ложно классифицировал как объекты целевого класса;

3. True Negative (TN) – количество элементов выборки, которые классификатор верно классифицировал как объекты нецелевого класса;

4. False Negative (FN) – количество элементов выборки, которые классификатор ложно классифицировал как объекты нецелевого класса.

Таким образом, ошибки классификации бывают двух видов: False Negative (FN) и False Positive (FP).

Точность:

$$precision = \frac{TP}{TP + FP},$$

полнота:

$$recall = \frac{TP}{TP + FN},$$

метрика f1:

$$f1 = \frac{2 * precision * recall}{precision + recall}[22].$$

Критерием оценивания и выбора модели является максимизация значения метрики f1.

2.2.2 Дерево решений

В качестве первой рассматриваемой модели было принято решение использовать дерево решений, так как данный алгоритм является наиболее часто и широко используемым в задачах классификации.

Для каждого признака в наборе данных алгоритм дерева решений формирует узел проверки, где наиболее важным признаком помещается в центральный узел. Для оценки мы начинаем с корневого узла и продвигаемся вниз по дереву, следуя за узлом, который удовлетворяет нашему условию или решению. Процесс разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями — конечными узлами дерева.

Для решения задачи семиклассовой классификации применим готовый вариант модели решающего дерева DecisionTreeClassifier библиотеки scikit-learn[23].

Рассматриваемые параметры при обучении и тестировании:

- max_depth – максимальная глубина дерева;
- criterion – критерий расщепления;
- min_samples_leaf – ограничение на число объектов в листьях.

Перебор параметров для поиска наилучшей модели и максимизации метрики f1 будет проводиться с помощью метода GridSearchCV. Метод получает на вход модель, список параметров и название метрики, для которой будет проходить максимизация.

Результаты обучения и тестирования приведены в таблице 1.

Таблица 1

Результаты применения дерева решений

Представление данных	средняя точность	средняя полнота	среднее f1-score
координаты точек	0.54	0.52	0.52
сумма векторов	0.61	0.61	0.61
длины векторов	0.57	0.57	0.57
определенные расстояния	0.59	0.59	0.58

Наибольшее знаение метрики f1 было получено при использовании суммы векторов. При этом параметры приняли следующие значения: min_samples_leaf=10, criterion=entropy, max_depth=None.

2.2.3 Случайный лес

Следующая рассматриваемая модель — случайный лес. Случайный лес — это множество решающих деревьев, решение которых принимается голосованием по большинству. Все деревья строятся независимо по следующей схеме:

1. выбирается подвыборка обучающей выборки определенного размера — по ней строится дерево;
2. для построения каждого расщепления в дереве просматриваем максимальное количество случайных признаков;
3. выбираем наилучший признак и расщепление по нему, дерево строится до исчерпания выборки.

Решающий лес (RandomForestClassifier) так же предоставляется библиотекой scikit-learn [24].

Рассматриваемые параметры при обучении и тестировании:

- n_estimators – количество деревьев;
- max_depth – максимальная глубина деревьев;

- criterion – критерий расщепления;
- min_samples_leaf – ограничение на число объектов в листьях.

Результаты обучения и тестирования приведены в таблице 2.

Таблица 2

Результаты применения случайного леса

Представление данных	средняя точность	средняя полнота	среднее f1-score
координаты точек	0.71	0.70	0.70
сумма векторов	0.76	0.76	0.75
длины векторов	0.70	0.71	0.70
определенные расстояния	0.71	0.71	0.71

Наибольшее знаение метрики f1 было получено при использовании суммы векторов. Параметры приняли следующие значения: n_estimators=1000, min_samples_leaf =10, criterion=entropy , max_depth=16.

2.2.4 Бутстрэп-агрегирование

Бутстрэп-агрегирование или бэггинг — это алгоритм, предназначенный для улучшения стабильности и точности алгоритмов машинного обучения, используемых для задач классификации и регрессии. Бутстрэп-агрегирование будет использовано для сокращения чрезмерной дисперсии дерева решений.

Будем использовать BaggingClassifier [25], передавая в него DecisionTreeClassifier. Модели предоставляются библиотекой scikit-learn.

Рассматриваемые параметры при обучении и тестировании:

- n_estimators – количество базовых оценок;
- bootstrap – отбираются ли выборки с заменой;
- max_samples – количество выборок, которые нужно извлечь из X для обучения каждого базового оценщика.

Результаты обучения и тестирования приведены в таблице 3.

Таблица 3

Результаты применения бэггинга

Представление данных	средняя точность	средняя полнота	среднее f1-score
координаты точек	0.42	0.44	0.41
сумма векторов	0.56	0.55	0.52
длины векторов	0.46	0.49	0.45
определенные расстояния	0.47	0.49	0.47

Наибольшее знаение метрики f1 было получено при использовании суммы векторов. Параметры приняли следующие значения: bootstrap=True, max_samples=10, n_estimators=300.

2.2.5 Бустинг

Основной идеей бустинга является комбинирование слабых алгоритмов, которые строятся в ходе итеративного процесса, где на каждом шаге новая модель обучается с использованием данных об ошибках предыдущих.

GradientBoostingClassifier – готовый вариант модели бустинга библиотеки scikit-learn (Python 3.8) [26].

Рассматриваемые параметры при обучении и тестировании:

- n_estimators – количество базовых оценок;
- max_depth – максимальная глубина дерева;
- min_samples_split – минимальное число объектов, при котором выполняется расщепление;
- min_samples_leaf – ограничение на число объектов в листьях.

Результаты обучения и тестирования приведены в таблице 4.

Таблица 4

Результаты применения бустинга

Представление данных	средняя точность	средняя полнота	среднее f1-score
координаты точек	0.72	0.71	0.71
сумма векторов	0.80	0.80	0.80
длины векторов	0.75	0.76	0.75
определенные расстояния	0.68	0.66	0.66

Наибольшее знаение метрики f1 было получено при использовании суммы векторов. Параметры приняли следующие значения: max_depth=20, min_samples_split=5, n_estimators=100, min_samples_leaf=2.

2.2.6 Метод опорных векторов

Метод опорных векторов — один из популярных методов обучения, который применяется для решения задач классификации и регрессии. Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Алгоритм работает в предположении, что чем больше расстояние между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора.

Метод опорных векторов так же реализован в библиотеке scikit-learn (LinearSVC) [27].

Рассматриваемые параметры при обучении и тестировании:

- penalty — параметры регуляризации;
- dual — функция потерь;
- random_state — предустановка генератора псевдослучайных чисел;
- multi_class — отвечает за формулирование стратегий классификации в многоклассовой;
- max_iter — максимальное количество итераций по умолчанию.

Результаты обучения и тестирования приведены в таблице 5.

Таблица 5

Результаты применения метода опорных векторов

Представление данных	средняя точность	средняя полнота	среднее f1-score
координаты точек	0.84	0.84	0.83
сумма векторов	0.85	0.83	0.82
длины векторов	0.79	0.79	0.77
определенные расстояния	0.67	0.68	0.64

Наибольшее знаение метрики f1 было получено при использовании координат точек. Параметры приняли следующие значения: dual=True, loss=squared_hinge, max_iter=1500, multi_class=ovr, penalty=l2, random_state=20.

2.2.7 Распределенная структура повышения градиента LightGBM

LightGBM расширяет алгоритм градиентного бустинга, добавляя тип автоматического выбора объектов, а также фокусируясь на примерах бустинга с большими градиентами, что приводит к резкому ускорению обучения и улучшению прогнозных показателей.

LightGBM устанавливается как автономная библиотека, при этом в своей реализации использует стандарт scikit-learn [28].

Результаты обучения и тестирования приведены в таблице 6.

Таблица 6

Результаты применения LightGBM

Представление данных	средняя точность	средняя полнота	среднее f1-score
координаты точек	0.76	0.77	0.76
сумма векторов	0.83	0.82	0.81
длины векторов	0.74	0.72	0.73
определенные расстояния	0.67	0.68	0.66

Наибольшее знаение метрики f1 было получено при использовании суммы векторов. Параметры приняли следующие значения: boosting_type=gbdt, max_depth=-1.

2.2.8 CatBoost

CatBoost — это библиотека градиентного бустинга, использующая небрежные деревья решений, чтобы вырастить сбалансированное дерево [29]. Одни и те же признаки используются для создания левых и правых разделений на каждом уровне дерева.

По сравнению с классическими деревьями, небрежные деревья более эффективны при реализации на процессоре и просты в обучении.

Рассматриваемые параметры при обучении и тестировании:

- iterations — тип бустинга;
- depth — максимальная глубина;
- learning_rate — максимальная глубина;

Результаты обучения и тестирования приведены в таблице 7.

Таблица 7

Результаты применения CatBoost

Представление данных	средняя точность	средняя полнота	среднее f1-score
координаты точек	0.13	0.14	0.13
сумма векторов	0.24	0.22	0.23
длины векторов	0.36	0.36	0.36
определенные расстояния	0.14	0.11	0.12

Наибольшее знаение метрики f1 было получено при использовании суммы векторов. Параметры приняли следующие значения: depth=2, iterations=100, learning_rate=2.

2.2.9 XGBoost

XGBoost - это оптимизированная распределенная библиотека повышения градиента, разработанная для обеспечения высокой эффективности, гибкости и портативности. Он реализует алгоритмы машинного обучения в рамках платформы Gradient Boosting [30]. XGBoost так же в своей реализации использует стандарт scikit-learn.

Рассматриваемые параметры при обучении и тестировании:

- max_depth — максимальная глубина;

По результатам обучения и тестирования, были получены следующие значения метрик f1: Результаты обучения и тестирования приведены в таблице 8.

Таблица 8

Результаты применения XGBoost

Представление данных	средняя точность	средняя полнота	среднее f1-score
координаты точек	0.76	0.75	0.75
сумма векторов	0.85	0.83	0.82
длины векторов	0.73	0.74	0.73
определенные расстояния	0.65	0.64	0.64

Наибольшее знаение метрики f1 было получено при использовании суммы векторов. Параметры приняли следующие значения: max_depth=4.

2.3 Сравнение полученных результатов

В таблице 9 приведены результаты тестирования методов машинного обучения для каждого из признаков, в качестве метрики использовалось значение f1.

Таблица 9

Результаты применения машинного обучения

Классификатор	сумма векторов	длины векторов	координаты точек	определенные расстояния
Дерево решений	0.61	0.57	0.52	0.58
Случайный лес	0.75	0.70	0.70	0.71
Бэггинг	0.52	0.45	0.41	0.47
Бустинг	0.80	0.75	0.71	0.66
Метод опорных векторов	0.82	0.77	0.83	0.64
XGB	0.81	0.73	0.75	0.64
LGBM	0.81	0.73	0.76	0.66
CatBoost	0.23	0.36	0.13	0.12

Метод опорных векторов оказался наилучшим классификатором среди методов машинного обучения, при этом самый высокий результат был получен при использовании координат точек и составляет 0.83. При использовании в качестве признаков суммы векторов, значение метрики f1 отличается всего на 0.01 и равен 0.82. Средняя точность и средняя полнота для этих классификаторов приведены в таблице 10.

Таблица 10

Результаты применения метода опорных векторов

Представление данных/метрика	средняя точность	средняя полнота	среднее f1-score
координаты точек	0.83	0.83	0.83
сумма векторов	0.82	0.83	0.82

Хуже всего с задачей классификации справился метод CatBoost. Это связанно с тем, что данный классификатор требует более подробного изучения и подбора параметров.

3 Применение ансамблей классификаторов

3.1 Классификация голосованием

Идея VotingClassifier состоит в том, чтобы объединить концептуально разные классификаторы машинного обучения и использовать большинство голосов или средние прогнозируемые вероятности для прогнозирования меток классов. Классификация голосованием может быть полезна для набора одинаково хорошо работающих моделей, чтобы сбалансировать их отдельные недостатки [31].

Наилучшие результаты были показаны при использовании жесткого голосования, когда прогнозируемая метка класса для конкретной выборки представляет собой метку класса, которая представляет большинство меток классов, предсказанных каждым отдельным классификатором.

При этом классификация проводилась при использовании сумм расстояний. Не смотря на то, что при использовании координат x, y метод опорных векторов давал наибольшее значение метрики f1, другие классификаторы оказались слабыми, что уменьшит значение метрики.

Были выбраны 4 настроенных классификатора: LinearSVC, GradientBoostingClassifier, XGBClassifier, LGBMClassifier.

Матрицы метрик и ошибок приведены на рисунках 5 и 6.

	precision	recall	f1-score	support
0	0.76	0.80	0.78	20
1	0.95	0.95	0.95	20
2	0.75	0.75	0.75	20
3	0.75	0.90	0.82	20
4	0.90	0.95	0.93	20
5	0.90	0.90	0.90	20
6	0.86	0.60	0.71	20
accuracy			0.84	140
macro avg	0.84	0.84	0.83	140
weighted avg	0.84	0.84	0.83	140

Рис. 5: Матрица VotingClassifier

Рис. 6: Матрица ошибок VotingClassifier

При использовании VotingClassifier значение метрики f1 увеличилось до 0.83.

3.2 Стекинг

StackingClassifier — это метод объединения оценок для уменьшения их погрешностей. Прогнозы каждого отдельного базового классификатора объединяются и используются в качестве входных данных для окончательного классификатора, выходом является итоговый прогноз. В качестве окончательного классификатора по умолчанию используется логистическая регрессия [32].

В качестве модлей для StackingClassifier будут использоваться те же модели, что и при VotingClassifier. В качестве признаков – суммы векторов.

Матрицы метрик и ошибок приведены на рисунках 7 и 8.

	precision	recall	f1-score	support
0	0.78	0.70	0.74	20
1	0.90	0.95	0.93	20
2	0.75	0.75	0.75	20
3	0.82	0.90	0.86	20
4	0.95	0.95	0.95	20
5	0.86	0.90	0.88	20
6	0.83	0.75	0.79	20
accuracy			0.84	140
macro avg	0.84	0.84	0.84	140
weighted avg	0.84	0.84	0.84	140

Рис. 7: Матрица метрик StackingClassifier

Рис. 8: Матрица ошибок StackingClassifier

StackingClassifier показал еще более лучший результат по сравнению с VotingClassifier, значение метрики составило 0.84.

4 Применение глубокого обучения

4.1 Построение модели

Для решения поставленной задачи с помощью глубокого обучения было принято реализовать нейронную сеть – перцептрон. При построения такой модели будем использовать библиотеку PyTorch (Python 3.8) [33]. Для полноты эксперимента, необходимо реализовать три перцептрона с различными количествами скрытых слоев:

1. Mlp_rl – перцептрон с одним скрытым слоем, в котором 137 нейронов;
2. 2lp_rl – перцептрон с двумя скрытыми слоями, в которых 68 и 68 нейронов соответственно;
3. 3lp_rl – перцептрон с тремя скрытыми слоями, в которых по 68 нейронов.

В качестве функции активации выберем ReLU, которая менее требовательна к вычислительным ресурсам и работает как хороший аппроксиматор.

4.2 Классификация 7-ми эмоций

В качестве алгоритма оптимизации в обучении нейронных сетей выберем Adam, а функцию обратного распространения ошибки CrossEntropyLoss. Количество эпох – итера-

ций в процессе обучения 1000.

Результаты тестирования всех трех сетей представлены в таблице 11:

Таблица 11

Результаты применения глубокого обучения

Классификатор	сумма векторов	длины векторов	координаты точек	определенные расстояния
Однослойный перцептрон	0.84	0.68	0.79	0.70
Двухслойный перцептрон	0.86	0.74	0.83	0.62
Трехслойный перцептрон	0.80	0.70	0.79	0.69

По результатам тестирования, было установлено, что наибольшее значение метрики f1 было получено при использовании двухслойного перцептрона на суммах векторов.

4.3 Эксперименты с параметрами

Для того чтоб определить наилучшую модель, необходимо провести тесты с оптимизаторами или пересмотреть количество нейронов в скрытых слоях.

В связи с тем, что наилучшие значения метрик f1 были получены при использовании сумм векторов, то перебор параметров проводился только на этих признаках.

Результаты тестирования однослойного перцептрона представлены в таблице 12:

Таблица 12

Тестирование однослойного перцептрона

Оптимизатор	средняя точность	средняя полнота	среднее f1-score
Adam	0.85	0.84	0.84
Adamax	0.86	0.86	0.86
Adadelta	0.48	0.48	0.48
Adagrad	0.50	0.48	0.49
RMSprop	0.81	0.81	0.81
SGD	0.76	0.75	0.75

Результаты тестирования двуслойного перцептрона представлены в таблице 13:

Таблица 13

Тестирование двухслойного перцептрона

Оптимизатор	средняя точность	средняя полнота	среднее f1-score
Adam	0.86	0.86	0.86
Adamax	0.86	0.86	0.86
Adadelta	0.47	0.42	0.44
Adagrad	0.61	0.67	0.64
RMSprop	0.74	0.76	0.75
SGD	0.78	0.78	0.78

Результаты тестирования трехслойного перцептрона представлены в таблице 14:

Таблица 14

Тестирование трехслойного перцептрона

Оптимизатор	средняя точность	средняя полнота	среднее f1-score
Adam	0.81	0.80	0.80
Adamax	0.82	0.81	0.81
Adadelta	0.32	0.34	0.33
Adagrad	0.46	0.48	0.47
RMSprop	0.72	0.73	0.72
SGD	0.70	0.72	0.71

По результатам тестирования оптимизаторов, было получено, что наибольшее значение метрики f1 на однослойном перцептроне достигается при использовании оптимизатора adamax и составляет 0.86, как при использовании двухслойного перцептрона с оптимизатором Adam. Однако при использовании Adamax на двухслойном перцептроне значение метрики не увеличилось.

Для того чтоб повысить качество классификации модели рассмотрим различное количество нейронов в скрытых слоях у однослойного и двухслойного перцептронов.

Результаты тестирования однослойного и двухслойного перцептронов приведены в таблицах 15 и 16 соответственно.

Таблица 15

Тесты с количеством нейронов в однослойном перцептроне

Количество нейронов	средняя точность	средняя полнота	среднее f1-score
68	0.80	0.79	0.79
137	0.86	0.86	0.86
204	0.84	0.84	0.83
272	0.79	0.78	0.77
340	0.79	0.76	0.75
408	0.79	0.76	0.75

Таблица 16

Тесты с количеством нейронов в двухслойном перцептроне

Количество нейронов в первом слое	количество нейронов во втором слое	средняя точность	средняя полнота	среднее f1-score
136	68	0.86	0.86	0.86
204	136	0.80	0.74	0.73
272	204	0.80	0.80	0.80
340	272	0.82	0.80	0.80
408	340	0.80	0.71	0.72

По результатам экспериментов с количеством нейронов в скрытых слоях было получено, что первоначальная настройка архитектуры перцептрона давала наибольшее значение метрики f1. Увеличение количества нейронов уменьшало значение метрики f1.

Наибольшее значение метрики равное 0.86 было получено при использовании однослойного и двухслойного перцпетрона с оптимизатором Adamax, а так же при использовании двухслойного перцептрона с оптимизатором Adam. Однако, необходимо выбрать наилучший классификатор по максимизации метрики f1. Двухслойный перцептрон с оптимизатором Adamax показал значение метрики 0.8614, против однослойного перцпетрона с Adamax и двухслойного перцептрона с Adam, значение метрики которых было приближенно к 0.855. При этом если бы задача стояла в выборе наиболее быстрого алгоритма, то лучшим классификатором являлся бы однослойный перцептрон с оптимизатором Adamax.

	precision	recall	f1-score	support
Нейтральность	0.84	0.80	0.82	20
Злость	0.79	0.95	0.86	20
Грусть	0.88	0.70	0.78	20
Удивление	0.83	0.95	0.88	20
Счастье	0.95	1.00	0.98	20
Отвращение	0.90	0.95	0.93	20
Испуг	0.88	0.70	0.78	20
accuracy			0.86	140
macro avg	0.87	0.86	0.86	140
weighted avg	0.87	0.86	0.86	140

Рис. 9: Матрица метрик двухслойного перцептрона с оптимизатором adamax

Рис. 10: Матрица ошибок двухслойного перцептрона с оптимизатором adamax

Заключение

В ходе работы работы были выполнены все поставленные цели. Были подготовленны несколько наборов данных для обучения и тестирования. Были произведены эксперименты с различной предобработкой этих данных.

Были настроены и обучены методы машинного обучения различных библиотек. По итогам тестирования, наилучшие классификаторы участвовали в классификации голосованием и в стэкинге, что позволило улучшить результат.

Были разработаны методы глубокого обучения — 3 перцептрона с различным количеством скрытых слоев. Модели улучшались перебором оптимизаторов и количеством нейронов в скрытых слоях. Наибольшее значение метрики f1 составило 0.86.

Список литературы

1. В. Фризен Уоллес, Пол Экман. Узнай лжеца по выражению лица. СПб.: Питер, 2016. 158 с.

2. Y Tang. Deep learning using linear support vector machines. — Текст : электронный //arXiv; Computer Science; Machine Learning. 2 июля 2013. URL: https://arxiv.org/abs/2007.00992 (дата обращения: 11.01.2022).

3. FER-2013. — Текст : электронный // Kaggle : [сайт] – [США]. URL: https://www.kaggle.com/msambare/fer2013 (дата обращения: 17.12.2021).

4. H. Zhou. Exploring emotion features and fusion strategies for audio-video emotion recognition. — Текст : электронный //dl.acm.org : [сайт] – [США]. 14 октября 2019. URL: https://dl.acm.org/doi/10.1145/3340555.3355713 (дата обращения: 11.01.2022).

5. Shi Jiawei, Zhu Songhao, Liang Zhiwei. Learning to Amend Facial Expression Representation via De-albino and Affinity. — Текст : электронный //arXiv; Computer Science; Computer Vision and Pattern Recognition. 18 марта 2021. URL: https://arxiv.org/abs/2103.10189 (дата обращения: 11.01.2022).

6. Facial expression recognition with grid-wise attention and visual transformer. — Текст : электронный //ScienceDirect : [сайт] – [США] / Q. Huang, C. Huang, X. Wang [и др.]. Ноябрь 2021. URL: https://www.sciencedirect.com/science/article/pii/S0020025521008495 (дата обращения: 11.01.2022).

7. Covariance Pooling for Facial Expression Recognition. — Текст : электронный //arXiv / D. Acharya, Z. Huang, D. P. Paudel [и др.]. 13 мая 2018. URL: https://arxiv.org/abs/2103.10189 (дата обращения: 12.01.2022).

8. Learning Vision Transformer with Squeeze and Excitation for Facial Expression Recognition. — Текст : электронный //arXiv; Computer Vision and Pattern Recognition / Mouath Aouayeb, Wassim Hamidouche, Catherine Soladie [и др.]. 16 июля 2021. URL: https://arxiv.org/abs/2107.03107 (дата обращения: 12.01.2022).

9. DeXpression: Deep Convolutional Neural Network for Expression Recognition. — Текст : электронный //arXiv; Computer Vision and Pattern Recognition / Peter Burkert, Felix Trier, Muhammad Zeshan Afzal [и др.]. 17 августа 2016. URL: https://arxiv.org/abs/1509.05371 (дата обращения: 12.01.2022).

10. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. — Текст : электронный //arXiv; Computer Science; Computer Vision and Pattern Recognition / Andrew G. Howard, Menglong Zhu, Bo Chen [и др.]. 17 апреля 2017. URL: https://arxiv.org/abs/1704.04861 (дата обращения: 16.12.2021).

11. Tan Mingxing, Le Quoc V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. — Текст : электронный //arXiv; Computer Science; Machine Learning. 11 сентября 2020. URL: https://arxiv.org/abs/1905.11946 (дата обращения: 16.12.2021).

12. Rethinking Channel Dimensions for Efficient Model Design. — Текст : электронный //arXiv; Computer Science; Machine Learning / Dongyoon Han, Sangdoo Yun, Byeongho Heo [и др.]. 2 июля 2020. URL: https://arxiv.org/abs/2007.00992 (дата обращения: 16.12.2021).

13. A. Savchenko. Facial expression and attributes recognition based on multi-task learning of lightweight neural networks. — Текст : электронный //arXiv; Computer Science; Computer Vision and Pattern Recognition. 3 марта 2021. URL: https://arxiv.org/abs/2103.17107 (дата обращения: 16.12.2021).

14. Кравченко Ольга. Предобученные модели распознавания эмоций EmoPy выложили в открытый доступ. — Текст : электронный // Neurohive: [сайт] – [Россия]. 10 января 2019. URL: https://neurohive.io/ru/novosti/emopy-emotion-recognition/ (дата обращения: 16.12.2021).

15. Emotion recognition. — Текст : электронный // From Wikipedia, the free encyclopedia : [сайт] – [США]. URL: https://en.wikipedia.org/wiki/Emotion_recognition (дата обращения: 16.12.2021).

16. Cohn-Kanade AU-Coded Facial Expression Database. — Текст : электронный //Carnegie Mellon University : [сайт] – [США]. URL: https://www.ri.cmu.edu/project/cohn-kanade-au-coded-facial-expression-database/ (дата обращения: 17.12.2021).

17. Ding, Zhou S. K., Chellappa R. FaceNet2ExpNet: Regularizing a Deep Face Recognition Net for Expression Recognition. — Текст : электронный //ieeexplore. 29 июня 2017. URL: https://ieeexplore.ieee.org/document/7961731 (дата обращения: 16.12.2021).

18. E.Lundqvist, D., Flykt, A., Öhman, A. (1998). The Karolinska Directed Emotional Faces - KDEF, CD ROM from Department of Clinical Neuroscience, Psychology section, Karolinska Institutet, ISBN 91-630-7164-9.

19. Langner, O., Dotsch, R., Bijlstra, G., Wigboldus, D.H.J., Hawk, S.T., van Knippenberg, A. (2010). Presentation and validation of the Radboud Faces Database. Cognition Emotion, 24(8), 1377—1388. DOI: 10.1080/02699930903485076.

20. YOLOv5. — Текст : электронный // github : [сайт] – [США]. URL: https://github.com/ultralytics/yolov5 (дата обращения: 15.01.2022).

21. Библиотека dlib. — Текст : электронный //Заметки про роботов : [сайт] – [Россия]. 06.09.2016. URL: https://cv-blog.ru/?p=16 (дата обращения: 02.05.2021).

22. Метрики в задачах машинного обучения. — Текст : электронный //Хабр : [сайт] – [Россия]. 12 мая 2017. URL: https://habr.com/ru/company/ods/blog/328372/ (дата обращения: 01.05.2021).

23. DecisionTreeClassifier. — Текст : электронный //scikit-learn : [сайт] – [США]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (дата обращения: 10.10.2021).

24. RandomForestClassifier. — Текст : электронный //scikit-learn : [сайт] – [США]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (дата обращения: 12.10.2021).

25. BaggingClassifier. — Текст : электронный //scikit-learn : [сайт] – [США]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html (дата обращения: 14.10.2021).

26. GradientBoostingClassifier. — Текст : электронный //scikit-learn : [сайт] – [США]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html (дата обращения: 16.10.2021).

27. LinearSVC. — Текст : электронный //scikit-learn : [сайт] – [США]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html (дата обращения: 18.10.2021).

28. Welcome to LightGBM's documentation!. — Текст : электронный //LightGBM : [сайт] – [США]. URL: https://lightgbm.readthedocs.io/en/latest/ (дата обращения: 18.10.2021).

29. Быстрый градиентный бустинг с CatBoost. — Текст : электронный //Хабр : [сайт] – [Россия]. 11 ноября 2020. URL: https://habr.com/ru/company/otus/blog/527554/ (дата обращения: 18.10.2021).

30. XGBoost Documentation. — Текст : электронный //XGBoost : [сайт] – [США]. URL: https://xgboost.readthedocs.io/en/stable/ (дата обращения: 19.10.2021).

31. VotingClassifier. — Текст : электронный //scikit-learn : [сайт] – [США]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html (дата обращения: 10.05.2022).

32. StackingClassifier. — Текст : электронный //scikit-learn : [сайт] – [США]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html (дата обращения: 10.05.2022).

33. Туториал по PyTorch: от установки до готовой нейронной сети. — Текст : электронный //neurohive : [сайт] – [Россия]. 22 октября 2018. URL: https://neurohive.io/ru/tutorial/glubokoe-obuchenie-s-pytorch/ (дата обращения: 01.05.2021).