



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

**Домашнее задание**  
**по дисциплине «Методы машинного обучения»**

Выполнила: Андрианов А.А.

Группа: ИУ5-23М

Проверил: Гапанюк Ю.Е.

Москва, 2022 г.

# ОГЛАВЛЕНИЕ

<b>1. ЗАДАНИЕ.....</b>	<b>3</b>
<b>2. ПОСТАНОВКА ЗАДАЧИ .....</b>	<b>5</b>
<b>3. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ.....</b>	<b>9</b>
<b>3.1. Тест адаптации распознавания речевых эмоций (SERAB).....</b>	<b>9</b>
3.1.1. Задачи и наборы данных .....	9
3.1.2. Конвейер оценки.....	11
3.1.3. Базовые подходы .....	12
3.1.4. Результаты оценки.....	15
<b>4. ПРАКТИЧЕСКАЯ ЧАСТЬ.....</b>	<b>18</b>
<b>4.1. Обзор страницы в GitHub .....</b>	<b>18</b>
4.1.1. Описание статьи .....	19
4.1.2. Демо-пример .....	19
4.1.3. Настройка среды.....	19
4.1.4. Оценка (предварительно обученной модели) с помощью SERAB.....	20
4.1.5. Обучение модели «à la BYOL-A» .....	20
4.1.6. SERAB датасеты .....	21
4.1.7. Цитирование.....	21
<b>4.2. Демонстрационный пример работы модели .....</b>	<b>22</b>
<b>5. ВЫВОДЫ .....</b>	<b>29</b>
<b>6. СПИСОК ИСТОЧНИКОВ.....</b>	<b>30</b>

# 1. ЗАДАНИЕ

Домашнее задание по дисциплине направлено на анализ современных методов машинного обучения и их применение для решения практических задач. Домашнее задание включает три основных этапа:

1. выбор задачи;
2. теоретический этап;
3. практический этап.

Этап выбора задачи предполагает анализ ресурса paperswithcode [1]. Данный ресурс включает описание нескольких тысяч современных задач в области машинного обучения. Каждое описание задачи содержит ссылки на наиболее современные и актуальные научные статьи, предназначенные для решения задачи (список статей регулярно обновляется авторами ресурса). Каждое описание статьи содержит ссылку на репозиторий с открытым исходным кодом, реализующим представленные в статье эксперименты. На этапе выбора задачи обучающийся выбирает одну из задач машинного обучения, описание которой содержит ссылки на статьи и репозитории с исходным кодом.

Теоретический этап включает проработку как минимум двух статей, относящихся к выбранной задаче. Результаты проработки обучающийся излагает в теоретической части отчета по домашнему заданию, которая может включать:

- описание общих подходов к решению задачи;
- конкретные топологии нейронных сетей, нейросетевых ансамблей или других моделей машинного обучения, предназначенных для решения задачи;
- математическое описание, алгоритмы функционирования, особенности обучения используемых для решения задачи нейронных сетей, нейросетевых ансамблей или других моделей машинного обучения;
- описание наборов данных, используемых для обучения моделей;
- оценка качества решения задачи, описание метрик качества и их значений;
- предложения обучающегося по улучшению качества решения задачи.

Практический этап включает повторение экспериментов авторов статей на основе представленных авторами репозитория с исходным кодом и возможное улучшение обучающимися полученных результатов. Результаты проработки обучающийся излагает в практической части отчета по домашнему заданию, которая может включать:

- исходные коды программ, представленные авторами статей, результаты документирования программ обучающимися с использованием диаграмм UML, путем визуализации топологий нейронных сетей и другими способами;
- результаты выполнения программ, вычисление значений для описанных в статьях метрик качества, выводы обучающегося о воспроизводимости экспериментов авторов статей и соответствии практических экспериментов теоретическим материалам статей;
- предложения обучающегося по возможным улучшениям решения задачи, результаты практических экспериментов (исходные коды, документация) по возможному улучшению решения задачи.

Отчет по домашнему заданию должен содержать:

1. Титульный лист.
2. Постановку выбранной задачи машинного обучения, соответствующую этапу выбора задачи.
3. Теоретическую часть отчета.
4. Практическую часть отчета.
5. Выводы обучающегося по результатам выполнения теоретической и практической частей.
6. Список использованных источников.

## 2. ПОСТАНОВКА ЗАДАЧИ

В результате анализа содержимого ресурса «Papers with code» была выбрана область обработки речи (Speech). В данной области было решено изучить решения задачи распознавания эмоций (Emotion Recognition). Данная задача предполагает решение нескольких подзадач (см. Рисунок 1). Среди них для рассмотрения в данном домашнем задании была выбрана подзадача распознавания речевых эмоций (Speech Emotion Recognition) [2].

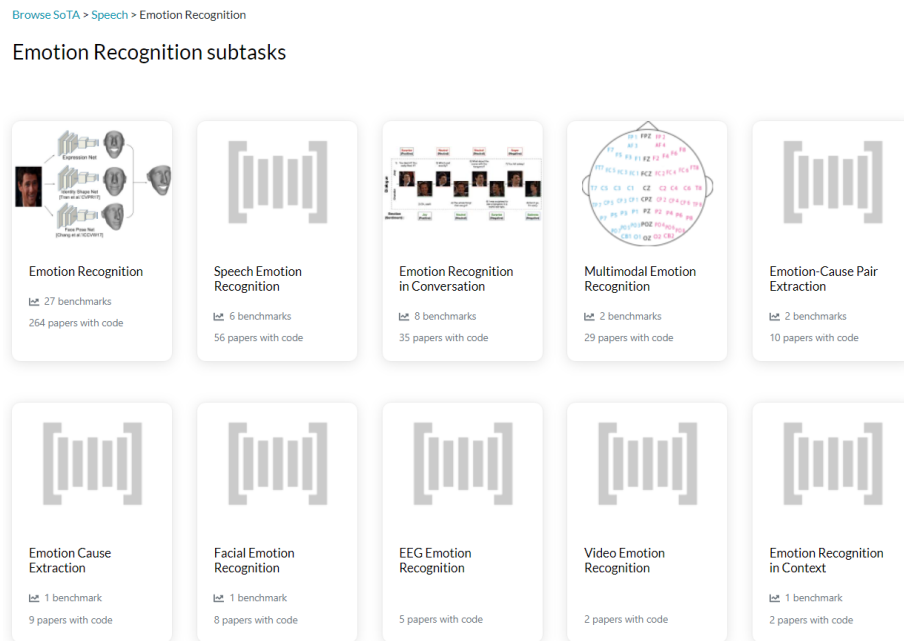


Рисунок 1 - Подзадачи «Распознавания речи»

Распознавание речевых эмоций (SER) – это система, которая может идентифицировать эмоции различных аудиосэмплов. [303] Эта задача похожа на анализ настроений текста. Обе эти задачи также имеют некоторые общие приложения, поскольку они отличаются только модальностью данных – текст и аудио. Как и анализ настроений текста, распознавание речевых эмоций можно использовать для поиска эмоционального диапазона или сентиментальной ценности в различных аудиозаписях, таких как собеседования, звонки агента вызывающего абонента, потоковое видео и песни. Более того, даже системы музыкальных рекомендаций или классификации могут группировать песни на основе их настроения и рекомендовать определенные плейлисты пользователю. Можно с уверенностью предположить, что сложные алгоритмы Spotify и YouTube также имеют компонент SER, который помогает в музыкальных рекомендациях.

С точки зрения машинного обучения распознавание речевых эмоций — это проблема классификации, когда входной образец (аудио) должен быть классифицирован на несколько predetermined эмоций.

Звуковые файлы отличаются от текстовых по своей структуре, поэтому из всей области распознавания эмоций выделяем отдельную подзадачу распознавания речевых эмоций.

Рисунок 2 демонстрирует страницу, посвященную данной задаче на ресурсе Papers With Code. После формального описания задачи (для данной задачи описание отсутствует) следует таблица метрик сравнения качества работы методов, предназначенных для решения поставленной задачи (Benchmarks).

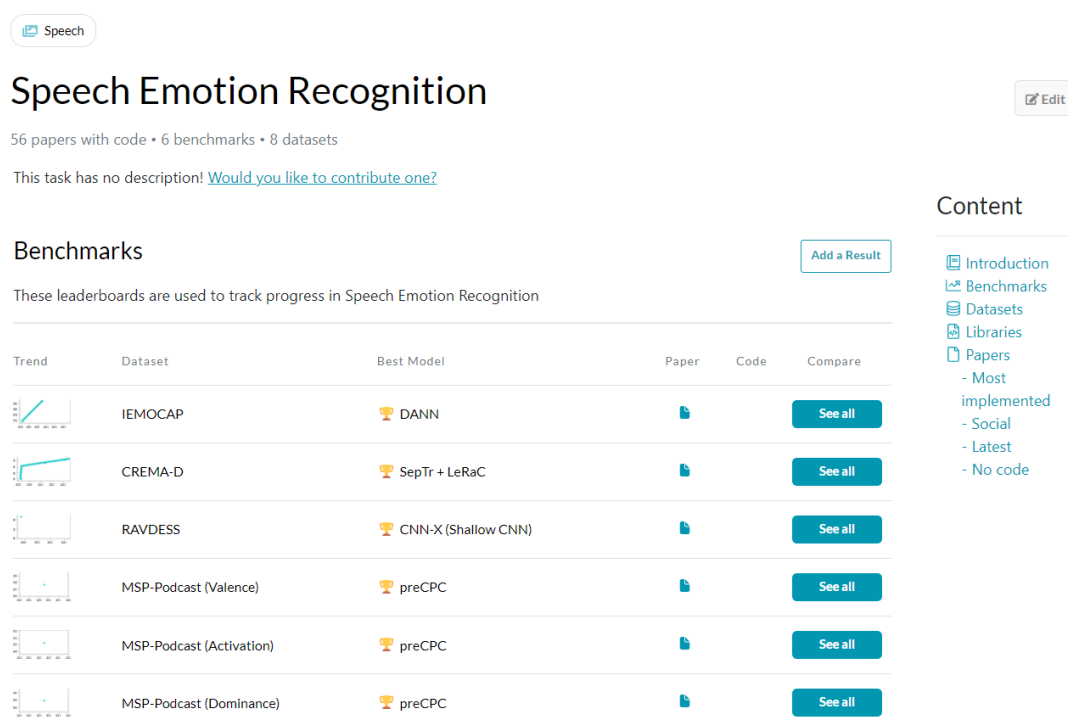


Рисунок 2 - Страница, посвященная классификации документов

Таблица состоит из колонок:

- Trend — тенденция качества решения задачи распознавания на соответствующем датасете;
- Dataset — набор данных, на котором производится оценка качества работы алгоритма распознавания;
- Best Model — метод (или модель машинного обучения), показавший на данный момент лучшие результаты в решении задачи распознавания на соответствующем датасете;
- Paper — индикатора наличия статьи, описывающей соответствующий метод;
- Code — индикатор наличия кода, реализующего соответствующий метод (модель);
- Compare — ссылка для перехода на новую страницу со сравнением всех моделей и датасетов.

Далее ниже на той же странице (см. Рисунок 3) можно найти следующие разделы:

- раздел, посвященный библиотекам (Libraries), содержащим проверенные временем модели, предназначенные для решения задачи распознавания эмоций;
- раздел Datasets, содержащий все датасеты, которые можно использовать для решения поставленной задачи;
- раздел Most Implemented Papers, содержащий статьи, отсортированные по применимости сопутствующим им кода и/или теоретической информации.

Последний раздел также можно отсортировать по недавним статьям, выпущенным по данной теме.

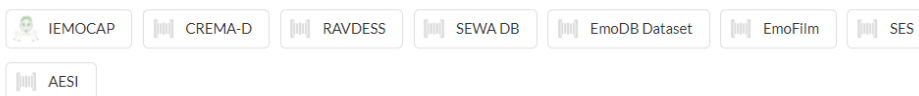
Следует также отметить, что данный раздел содержит не только статьи, описывающие методы, применяемые исключительно для распознавания речевых эмоций, но и для распознавания эмоций на видео, и для других задач. Подробнее это можно посмотреть на странице, посвященной конкретной статье. Например, наиболее применяемая статья называется «Continuous control with deep reinforcement learning» [4], и описывает алгоритм, не зависящий от модели, основанный на детерминированном градиенте политики, который может работать в непрерывных пространствах действий. Используя этот алгоритм обучения, сетевую архитектуру и гиперпараметры, можно надежно решить более 20 имитируемых физических задач.

### Libraries ⓘ

Use these libraries to find Speech Emotion Recognition models and implementations

 aris-ai/Audio-and-text-based-emotio... 2 papers 51 ★


### Datasets




### Most implemented papers

Most implemented Social Latest No code

Search for a paper, author or keyword



#### Continuous control with deep reinforcement learning

 DLR-RM/stable-baselines3 •  PyTorch • 9 Sep 2015

We adapt the ideas underlying the success of Deep Q-Learning to the continuous action domain.

141

 Paper


 Code

Рисунок 3 - Продолжение страницы, посвященной распознаванию эмоций

В рамках данного домашнего задания было интересно найти статью, описывающую решение некой специфичной для данной области задачи. В итоге выбор остановился на статье «SERAB: Speech Emotion Recognition Adaptation Benchmark» [5], которая рассматривает определение показателей адаптации к распознаванию речевых эмоций.

Рисунок 4 демонстрирует страницу, посвященную данной статье. На данной странице можно:

- прочитать Аннотацию (Abstract) к статье;
- получить текст статьи в формате PDF;
- открыть код, реализующий методы, описанные в данной статье;
- посмотреть задачи, решение которых предлагают авторы статьи;
- посмотреть наборы данных, на которых обучалась или тестировалась модель;
- посмотреть результаты решения задачи (необязательно указываются на данном сайте авторами статей)
- посмотреть методы, используемые авторами статьи для решения данной задачи.

## SERAB: A multi-lingual benchmark for speech emotion recognition

7 Oct 2021 · Neil Scheidwasser-Crow, Mikolaj Kegler, Pierre Beckmann, Milos Cernak · [Edit social preview](#)

Recent developments in speech emotion recognition (SER) often leverage deep neural networks (DNNs). Comparing and benchmarking different DNN models can often be tedious due to the use of different datasets and evaluation protocols. To facilitate the process, here, we present the Speech Emotion Recognition Adaptation Benchmark (SERAB), a framework for evaluating the performance and generalization capacity of different approaches for utterance-level SER. The benchmark is composed of nine datasets for SER in six languages. Since the datasets have different sizes and numbers of emotional classes, the proposed setup is particularly suitable for estimating the generalization capacity of pre-trained DNN-based feature extractors. We used the proposed framework to evaluate a selection of standard hand-crafted feature sets and state-of-the-art DNN representations. The results highlight that using only a subset of the data included in SERAB can result in biased evaluation, while compliance with the proposed protocol can circumvent this issue.

[PDF](#)

[Abstract](#)

### Code

[Edit](#)

[neclow/serab](#) [official](#)

★ 16

[TensorFlow](#)

[Quickstart in Colab](#)

[gasserelbanna/serab-byols](#)

★ 6

[PyTorch](#)

### Tasks

[Edit](#)

[Emotion Recognition](#)

[Speech Emotion Recognition](#)

### Datasets

[Edit](#)

[IEMOCAP](#)

[AudioSet](#)

### Results from the Paper

[Edit](#)

[Submit results from this paper](#) to get state-of-the-art GitHub badges and help the community compare results to other papers.

### Methods

[Edit](#)

No methods listed for this paper. Add [relevant methods here](#)

Рисунок 4 - Страница, посвященная статье



### **3. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

В рамках домашнего задания рассматриваются материалы статьи «SERAB: Многоязычный эталон для распознавания речевых эмоций» (SERAB: A Multilingual Benchmark for Speech Emotion Recognition).

Последние разработки в области распознавания речевых эмоций (SER) часто используют глубокие нейронные сети (DNN). Сравнение и сравнительный анализ различных моделей DNN часто могут быть утомительными из-за использования различных наборов данных и протоколов оценки. Чтобы облегчить этот процесс, авторы статьи представляют тест адаптации распознавания речевых эмоций (SERAB), основу для оценки производительности и способности к обобщению различных подходов к SER на уровне высказывания. Эталонный показатель состоит из девяти наборов данных для SER на шести языках. Поскольку наборы данных имеют разные размеры и количество эмоциональных классов, предлагаемая настройка особенно подходит для оптимизации способности к обобщению предварительно обученных экстракторов признаков fea на основе DNN. Использовалась предложенная структура для оценки набора стандартных наборов функций, созданных вручную, и современных представлений DNN. Результаты подчеркивают, что использование только подмножества данных, включенных в SERAB, может привести к предвзятой оценке, в то время как соблюдение предлагаемого протокола может обойти эту проблему.

Авторы выкладывают наборы данных, код и обученные модели в общем доступе.

#### **3.1. Тест адаптации распознавания речевых эмоций (SERAB)**

##### **3.1.1. Задачи и наборы данных**

Краткое описание задач, используемых в SERAB, представлено на Рисунок 5. Тест включает девять задач по классификации речевых эмоций на шести языках: четыре на английском (CREMA-D, IEMOCAP, RAVDESS & SAVEE) и по одному на французском (CaFE), немецком (EmoDB), греческом (AESDD), итальянском (EMOVO) и персидском (ShEMO). В каждом наборе данных образцы речи имеют три атрибута: аудиоданные (т. е. необработанная форма сигнала в монофоническом формате), идентификатор говорящего и метку эмоций (например, злость, радость, грусть). Наборы данных различаются по размеру (т. е. количеству высказываний), количеству выступающих, распределению по классам и количеству классов. В то время как гнев, счастье и печаль встречаются во всех наборах данных, отвращение, страх, нейтральные

эмоции, удивление, спокойствие и скука присутствуют по крайней мере в одном наборе данных. С другой стороны, все наборы данных имеют примерно одинаковую среднюю продолжительность высказывания (от 2,5 до 4,5 секунд).

Dataset	Code	Access	Language	Classes	Utterances	Speakers	Average duration (s)	Total duration (h)
AESDD [18]	AES	Open	Greek	5	604	6	4.2	0.7
CaFE [19]	CAF	Open	French	7	864	12	4.5	1.1
CREMA-D [13]	CRE	Open	English	6	7,442	91	2.5	5.3
EmoDB [20]	EMB	Open	German	7	535	10	2.8	0.4
EMOVO [21]	EMV	Open	Italian	7	588	6	3.1	0.5
IEM4 [17]	IEM	Restricted	English	4	5,531	10	3.4	7.0
RAVDESS [22]	RAV	Open	English	8	1,440	24	3.7	1.5
SAVEE [14]	SAV	Restricted	English	7	480	4	3.8	0.5
ShEMO [23]	SHE	Open	Persian	6	3,000	87	4.0	3.3

Рисунок 5. Задачи и наборы данных SERAB. IEM 4: 4 класс IEMOCAP.

Тест был разработан таким образом, чтобы сбалансировать популярность набора данных, языковое разнообразие и открытый доступ. В распознавании речевых эмоций EmoDB, IEMOCAP и RAVDESS являются одними из наиболее широко используемых наборов данных. Для смягчения серьезного дисбаланса классов в исходном наборе данных было использовано подмножество IEMOCAP из 4 классов (IEM4). Для других задач использовались все образцы и классы из исходных наборов данных (см. Рисунок 5). Как уже было показано в NOSS, CREMA-D и SAVEE были включены в SERAB. Для завершения теста CaFE (французский) и EMOVO (итальянский) были выбраны наборы данных на итальянском языке, в то время как AESDD (греческий) и ShEMO (персидский) представляли эллинскую и индоиранскую ветви индоевропейской семьи. В целом, тест в основном включает написанную по сценарию и озвученную речь, за исключением IEM4, RAVDESS и ShEMO, которые также содержат спонтанные высказывания.

Каждый набор данных был разделен на обучающие, валидационные и тестовые наборы для соответственно обучения, оптимизации и оценки классификаторов речевых эмоций, специфичных для конкретной задачи. За исключением CREMA-D, каждый набор данных был разделен на 60% обучающих, 20% валидационных и 20% тестовых наборов. Для CREMA-D авторы статьи придерживались деления на 70/10/20% (обучение / проверка / тестирование), которое было применено в NOSS. Каждый раздел данных был независимым от динамики речи, т. е. наборы ораторов, включенные в каждую часть, были взаимно непересекающимися. Поскольку наборы данных SERAB различаются по размеру, фиксированное деление данных позволяет оценить, как различные методы справляются с различными объемами данных, специфичных для конкретной задачи.

### 3.1.2. Конвейер оценки

Конвейер оценки SERAB используется для оценки представлений эмоций, основанных на речи, полученных с использованием различных экстракторов признаков (см. Рисунок 6). В частности, рабочий процесс включает в себя обработку входных высказываний с помощью предварительно обученного / необучаемого средства извлечения признаков и использование полученных вложений вместе с классификатором для конкретной задачи для прогнозирования речевых эмоций. При использовании простых классификаторов точность классификации отражает полезность извлеченных признаков для задач SER на уровне высказывания.

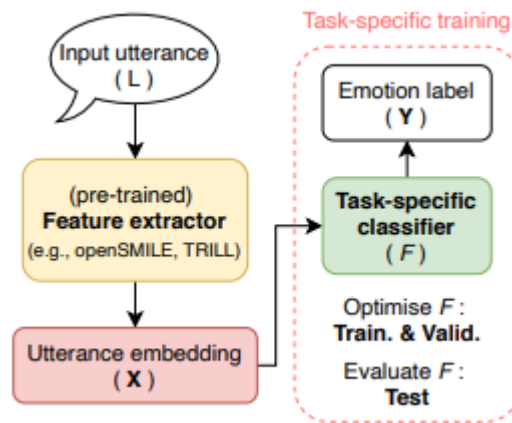


Рисунок 6. Конвейер оценки SERAB.

Важно отметить, что высказывания, включенные в SERAB, различаются по продолжительности. Решение о том, как интегрировать информацию во все высказывания, является важным выбором дизайна, который может повлиять на производительность извлечения функций. Таким образом, решение остается за авторами feature extractor. Большинство современных подходов, как правило, извлекают информацию из кадров фиксированной длины и усредняют выходные данные по кадрам. Однако другие подходы могут использовать временные зависимости во входном высказывании. В результате требуется, чтобы метод возвращал один набор признаков для каждого входного высказывания с различной продолжительностью.

Чтобы оценить производительность метода, все входные высказывания обрабатываются с помощью экстрактора объектов для получения их вложений (X). Полученные встраивания затем используются в качестве функций для базовых классификаторов машинного обучения (F), обученных предсказывать метки эмоций (Y). Чтобы обеспечить тщательную оценку, рассмотрены несколько различных классификаторов: логистическая регрессия (LR), машина опорных векторов (SVM), линейный и квадратичный дискриминантный анализ (LDA/QDA) и случайные леса (RF).

Гиперпараметры классификатора были оптимизированы с помощью поиска по сетке с использованием обучающей и валидационной частей данных. Наиболее эффективный классификатор из процедуры поиска по сетке был оценен на выделенном тестовом наборе. Все процедуры оптимизации и оценки классификаторов были реализованы с использованием `scikit-learn`.

В качестве показателя производительности использовалась точность классификации тестовых наборов в каждой задаче. Результирующая точность по девяти задачам SERAB была усреднена для количественной оценки эталонной производительности метода. В дополнение к невзвешенной средней точности (UM) в задачах SERAB также было вычислено средневзвешенное значение, полученное из размера тестового набора (WM), а также среднее геометрическое значение (GM).

### 3.1.3. Базовые подходы

**openSMILE** — это разработанный вручную набор акустических функций, основанный на функционалах низкоуровневых контуров дескрипторов. Несмотря на то, что openSMILE напрямую не управляется данными, он способен превосходить средства извлечения объектов на основе DNN, например, в задачах с небольшим количеством данных, специфичных для конкретной задачи. В данной статье самая последняя реализация openSMILE использовалась для извлечения функций из каждого высказывания в задачах SERAB. Впоследствии, для каждой задачи, классификатор речевых эмоций был оптимизирован с использованием обучающей и валидационной частей данных и оценен с использованием отложенного набора тестов.

**VGGish** — один из первых экстракторов функций для аудио на основе DNN, вдохновленный сверточным [306] DNN VGG-16 (CNN). Предварительно обученные веса моделей были изучены с помощью контролируемой классификации звуковых событий из набора данных Youtube-8M ( $\geq 350\,000$  часов видео, 3000 занятий). В модели используются входные окна фиксированного размера. Чтобы справиться с аудиоклипами переменной длины, каждое входное высказывание было разделено на неперекрывающиеся кадры длиной 960 мс. Спектрограмма логарифмической амплитуды ( $N = 64$  ячейки частоты mel) была вычислена на основе кратковременного преобразования Фурье с окнами по 25 мс с шагом 10 мс для каждого кадра. Полученные кадры затем передавались в предварительно обученную модель для извлечения признаков. После обработки  $M$  кадров полученные  $M$  вложений были усреднены для получения одного набора признаков на каждое высказывание. Оставшаяся оценка проводилась в соответствии с протоколом, описанным в Конвейере оценки.

**YAMNet** является еще одним широко используемым средством извлечения функций на основе DNN. Этот подход использует MobileNetv1, эффективную архитектуру CNN, оптимизированную для мобильных устройств. Авторы статьи использовали веса модели, предварительно обученной с помощью контролируемой классификации событий из AudioSet ( $\approx 5800$  часов аудио, 521 класс). Поскольку модель работает с использованием окон фиксированного размера, входные высказывания обрабатывались аналогично VGGish.

**TRILL** — В то время как VGGish и YAMNet были обучены работе с различными источниками звука (речь, музыка, звуки окружающей среды и т.д.), TRILL был специально разработан как средство извлечения несемантических речевых признаков. Модель DNN приняла архитектуру ResNetish и была предварительно обучена самоконтролем с использованием образцов речи из AudioSet, что составляет примерно 50% от всего набора данных ( $\approx 2800$  часов аудио). Используемая здесь "предварительно обученная модель" была получена в результате оптимизации потерь триплетов, которая направлена на минимизацию расстояния в пространстве вложения между привязкой и положительной выборкой (т. е. из того же клипа) при максимизации расстояния между той же привязкой и отрицательной выборкой (т. е. из другого клипа). В контексте аудио соседние по времени аудиосегменты будут находиться ближе в пространстве представления, и наоборот. Опять же, модель работает с кадрами фиксированного размера, поэтому входные высказывания обрабатывались аналогично VGGish и YAMNet. Авторы статьи использовали встраивание из первого слоя свертки глубиной 512 (слой 19), который лучше всего работал на NOSS.

**BYOL-A** — В качестве альтернативы установкам контрастного обучения, таким как TRILL, BYOL-A предлагает использовать ваш собственный латентный (BYOL) для обучения звуковому представлению, вдохновленный успехом BYOL для самоконтролируемой классификации изображений. Предварительно обученный на всей аудиосети, этот подход достиг самых современных результатов в различных задачах классификации звука, даже превзойдя TRILL в задачах обработки речи. Вместо оценки временной близости двух разных аудиосегментов BYOL-A полагается на сравнение двух дополненных версий одного сэмпла. Более конкретно, каждая версия соответственно передается в онлайн-овую сеть и целевую сеть. В то время как оба состоят из кодера и блока проецирования, интерактивная сеть включает в себя уровень прогнозирования, который нацелен на прогнозирование проецируемого представления второго расширенного представления. Таким образом, BYOL (и BYOL-A) изучает представление, отрицая случайные дополнения данных, чтобы получить необходимую

информацию о входных данных. Что касается BYOL-A, предварительно обученные веса для моделей разных размеров были выпущены авторами и использованы в этой работе. Поскольку модель принимает входные данные переменной длины, она возвращает одно вложение для каждого входного высказывания. Полученные вложения используются для обучения и оценки классификаторов SER.

**BYOL-S** — В то время как BYOL-A может достигать самых современных результатов в ряде задач классификации звука, его общее звуковое представление может быть неоптимальным для обработки речи и особенно для паралингвистических задач. Таким образом, мы переобучили BYOL-A, используя только речевые образцы аудиосети, что привело к специфичному для речи BYOL-S (S, обозначающему речь). Архитектура модели, процедура предварительной подготовки и использование остались такими же, как и в оригинальной версии.

**BYOL-S/CvT** — В этой модели предлагается расширение BYOL-S с представлением трансформатора. Более конкретно, авторы статьи заменили блоки свертки в BYOL-S на сверточный трансформатор" (CvT). CvT заметно расширяет самоконтроль с помощью глубокой свертки для проецирования запросов, ключей и вложений значений. Между модулями attention добавляются традиционные слои свертки для декомпозиции входных данных, как в большинстве CNN. Следовательно, CvT сочетает в себе качества CNN (например, инвариантность трансляции) и трансформаторов (например, захват зависимостей на большие расстояния и способность к обобщению). Здесь каждая ступень вариатора включала только один уровень самоконтроля, чтобы обеспечить справедливое сравнение с BYOL-S как с точки зрения архитектуры модели, так и с точки зрения количества параметров. Проводился эксперимент с тремя различными конфигурациями модели. Чтобы исследовать влияние размера модели, количество фильтров в ступенях вариатора было изменено, чтобы уменьшить количество параметров (см. Рисунок 7). аналогично BYOL-A. Кроме того, модель была протестирована с тремя различными размерами встраивания: 256, 512 и 2048. Последний использовал *среднюю + максимальную* временную агрегацию в последнем слое вместо глобального среднего объединения. Как BYOL-S, предварительная подготовка и применение к задачам SERAB были аналогичны BYOL-A.

На Рисунок 7 продемонстрирована оценка базовых подходов на SERAB. Размер встраивания относится к размерности набора признаков на уровне высказывания (X) (X), используемого для классификации эмоциональных меток (Y).

Model	Parameters (M)	Embedding size
openSMILE [4]	-	6,373
VGGish [11]	62.0	128
YAMNet [16]	4.2	1,024
TRILL (layer 19) [6]	9.0	12,288
BYOL-A [7]	0.6 / 1.6 / 5.3	512 / 1,024 / 2,048
<b>Proposed:</b>		
BYOL-S	0.6 / 1.6 / 5.3	512 / 1,024 / 2,048
BYOL-S/CvT, <i>small</i>	1.6	256
<i>CvT stages: 64/128/256</i>		
BYOL-S/CvT, <i>large</i>	5.0	512 / 2,048
<i>CvT stages: 64/256/512</i>		

Рисунок 7. Базовые подходы, оцененные на SERAB.

На Рисунок 8 показана точность тестирования (%) для различных последующих задач в SERAB, обозначенных их кодом. UM: невзвешенное среднее, WM: средневзвешенное (по количеству высказываний в тестовом наборе), GM: среднее геометрическое. Модели сортируются по их единой системе обмена сообщениями для всех задач. Наиболее эффективные подходы для каждой задачи и показателя выделены жирным шрифтом.

Model	AES	CAF	CRE	EMB	EMV	IEM	RAV	SAV	SHE	UM	WM	GM
YAMNet	53.6	48.1	53.9	60.7	35.7	56.1	52.3	54.2	81.7	55.1	55.8	54.0
VGGish	46.4	50.0	55.5	73.8	36.2	60.1	53.0	53.3	83.6	56.9	57.7	55.4
TRILL, layer 19	66.7	68.5	73.3	81.0	36.7	57.7	73.7	76.7	86.8	69.0	68.3	67.3
openSMILE	70.0	70.4	72.8	<b>90.5</b>	37.2	62.1	71.3	72.5	84.9	70.2	69.3	68.4
BYOL-A, 512	71.5	73.1	70.2	84.5	39.3	62.5	74.7	76.7	90.1	72.7	69.3	69.6
BYOL-A, 2048	72.0	75.5	73.7	88.1	38.3	62.8	<b>77.7</b>	78.3	89.0	72.8	71.2	71.0
BYOL-A, 1024	75.4	74.1	71.3	88.1	44.4	62.1	76.0	80.8	89.5	73.5	70.5	72.2
<b>Proposed:</b>												
BYOL-S/CvT, 256	72.9	71.8	72.9	85.7	47.4	64.8	76.0	75.8	89.0	72.9	71.5	71.9
BYOL-S, 512	74.9	<b>76.4</b>	74.4	86.9	34.2	63.3	77.3	79.2	90.6	73.0	71.7	70.8
BYOL-S, 1024	75.4	72.7	75.3	84.5	39.3	63.8	74.0	<b>82.5</b>	90.9	73.2	72.1	71.5
BYOL-S/CvT, 512	71.0	75.5	74.0	88.1	46.9	65.0	76.3	80.0	87.9	73.9	72.1	72.8
BYOL-S/CvT, 2048	75.8	71.3	<b>76.9</b>	84.5	<b>48.5</b>	<b>65.1</b>	76.3	76.7	<b>93.0</b>	74.2	<b>73.6</b>	73.2
BYOL-S, 2048	<b>77.3</b>	74.5	<b>76.9</b>	88.1	44.4	64.8	76.7	81.7	91.1	<b>75.1</b>	<b>73.6</b>	<b>73.7</b>

Рисунок 8. Точность тестирования (%) для различных последующих задач в SERAB.

### 3.1.4. Результаты оценки

На Рисунок 7 представлены конфигурации различных базовых подходов, описанных в разделе 3.1. Для моделей, подобных BYOL (BYOL-A, -S, -S/CvT), исследовались различные размеры моделей и различные размеры встраивания выходных данных, подаваемых в классификаторы для конкретных задач, предсказывающие метки эмоций. Контрольные показатели для всех базовых методов представлены на Рисунок 8. Большой BYOL-S с размером встраивания 2048 оказался лучшей моделью по всем рассмотренным показателям производительности и обеспечивает наилучшую индивидуальную точность в двух из девяти задач SERAB. Важно отметить, что ранги моделей в целом были одинаковыми по всем показателям в масштабах всего бенчмарка.

Таким образом, было принято решение отсортировать производительность модели по UM в соответствии с предыдущими бенчмарками для систем компьютерного зрения. Несмотря на несколько более низкие баллы по всему тесту, самый большой BYOL-S/CvT оставался конкурентоспособным, показав лучшие результаты в четырех из девяти задач. Более того, увеличение размера встраивания и общего размера модели, как правило, последовательно улучшает производительность предлагаемых подходов.

В целом, все модели, основанные на BYOL, даже с небольшими размерами, получили значительно более высокие оценки (до 5% абсолютной разницы в UM), чем TRILL, VGGish, YAMNet и openSMILE. Эта значительная разница в производительности, скорее всего, происходит из-за совершенно разных стратегий предварительной подготовки. Другой причиной превосходства моделей, производных от BYOL, может быть тот факт, что они предназначены для обработки входных данных переменной длины, а не кадров фиксированной длины, как это делают openSMILE, VGGish, YAMNet и TRILL. Это, в свою очередь, предполагает, что агрегирование временного контекста может улучшить производительность SER на уровне высказывания.

Модели BYOL-S работали стабильно лучше, чем оригинальные подходы BYOL-A. Это указывает на то, что специализация задачи предварительной подготовки путем сосредоточения внимания только на отрывках речи привела к более подходящим вложениям для SER. При таком предварительном обучении, ориентированном на речь, модель развила лучшую способность к представлению речи, включая независимые от языка паралингвистические сигналы, такие как эмоции, основанные на речи.

С другой стороны, обогащение BYOL-S механизмами самоконтроля с помощью вариатора не привело к заметному увеличению производительности, которое мы ожидали, но модель была легче с параметрами на 0,3 М меньше, чем у BYOL-S. Небольшая разница в производительности может быть обусловлена минимальными индуктивными смещениями, подразумеваемыми в трансформаторных моделях, в отличие от CNNS. Хотя это выгодно при обучении больших моделей на больших наборах данных, такие смещения становятся критическими в небольших сетях и небольших наборах данных. Таким образом, увеличение размера набора данных перед обучением может помочь развить способность трансформаторов к обобщению и, таким образом, улучшить общую производительность BYOL-S/CvT.

В то время как SERAB позволяет сравнивать различные модели по широкому кругу задач, что более важно, он предоставляет оптимизированную платформу для сравнения различных подходов. В частности, некоторые задачи демонстрируют



значительные различия между моделями (например, EMOVO, SAVEE, EmoDB), так что в целом подходы с более низкой производительностью могут казаться лучше, чем они есть на самом деле. Некоторые из этих различий могут быть специфичны для конкретного набора данных, что приводит к еще большему смещению, преодолеть которое непросто. Включение нескольких задач на разных языках обеспечивает надежные оценки производительности, как показала оценка базовых подходов, представленная в статье.

## 4. ПРАКТИЧЕСКАЯ ЧАСТЬ

Чтобы облегчить использование SERAB, авторы статьи выложили в общий (свободный) доступ все материалы, относящиеся к опубликованной статье, фреймворк, включая инструкции по настройке, оценочные конвейеры, наборы данных и примеры и результаты [Error! Reference source not found.].

### 4.1. Обзор страницы в GitHub

Рисунок 9 демонстрирует страницу на сайте GitHub, посвященную данной статье.

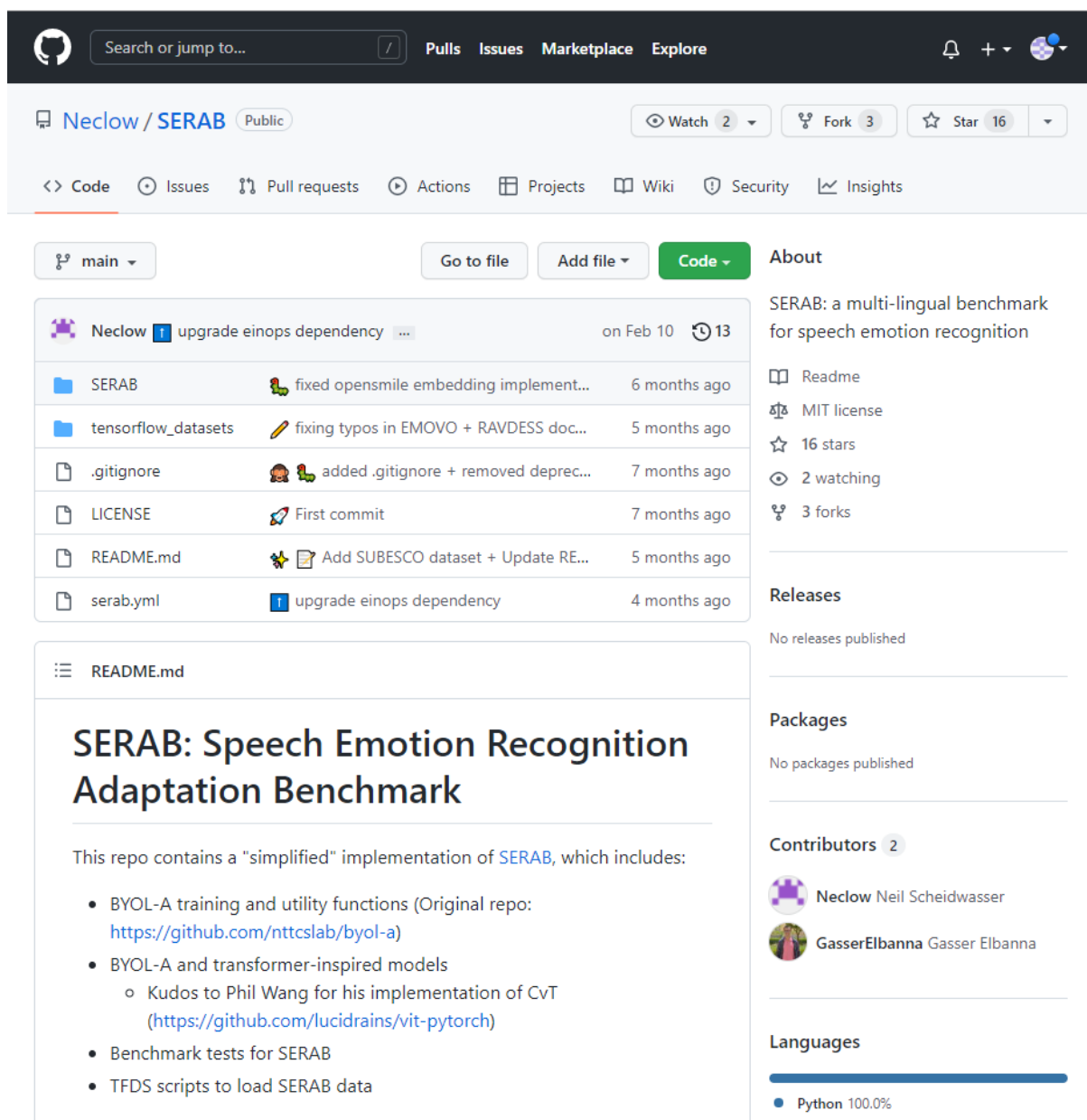


Рисунок 9 - Страница статьи в GitHub

На данной странице авторы дают краткое описание содержимого репозитория, а также описывают порядок работы с ним в нескольких разделах страницы.

#### 4.1.1. Описание статьи

В первом разделе помимо описания авторы также указывают ссылки на сторонние источники, связанные со статьей. Текст статьи можно найти на сайте [arxiv.org](https://arxiv.org) [8**Error! Reference source not found.**].

Данный репозиторий содержит "упрощенную" реализацию SERAB, которая включает в себя:


- BYOL-A обучающие и служебные функции (оригинальный репозиторий [9])
- BYOL-A модели и модели в стиле трансформеров
  - Фил Вэнг внедрил CvT. Реализацию можно посмотреть в его репозитории [10]
- Контрольные тесты для SERAB
- Скрипты TFDS для загрузки данных SERAB

BYOL-S был одним из самых сильных участников конкурса HEARCHIPS 2021 Challenge! Результаты представлены в таблице лидеров: [11].

#### 4.1.2. Демо-пример

В следующем разделе авторы оставили ссылку на демонстрационный пример работы модели на платформе Google Colaboratory [12][13]. (см. Рисунок 10).

### Demo

- A quick demo detailing the SERAB evaluation procedure on a Colab notebook is available  [Open in Colab](#)

*Рисунок 10 - Описание демонстрационного примера*

#### 4.1.3. Настройка среды

В следующем разделе авторы описывают, как настроить среду для более углубленного тестирования приведенного программного кода.

Библиотеки для воспроизведения среды подробно описаны в `serab.yml`.

Чтобы запустить среду, необходимо запустить:

```
conda env create -f serab.yml
```

Чтобы запустить внешние исходные файлы из патчей, необходимо скопировать следующие команды после клонирования репозитория:

```
cd SERAB/  
curl -O https://raw.githubusercontent.com/nttcs/serab/byol-a/f2451c366d02be031a31967f494afdf3485a85ff/config.yaml  
patch --ignore-whitespace < config.diff
```

```

curl -O https://raw.githubusercontent.com/nttcslab/byol-
a/f2451c366d02be031a31967f494afdf3485a85ff/train.py
patch < train.diff
cd byol_a/
curl -O https://raw.githubusercontent.com/nttcslab/byol-
a/f2451c366d02be031a31967f494afdf3485a85ff/byol_a/augmentations.py
patch < augmentations.diff
curl -O https://raw.githubusercontent.com/nttcslab/byol-
a/f2451c366d02be031a31967f494afdf3485a85ff/byol_a/common.py
patch < common.diff
curl -O https://raw.githubusercontent.com/nttcslab/byol-
a/f2451c366d02be031a31967f494afdf3485a85ff/byol_a/dataset.py
patch < dataset.diff
curl -O https://raw.githubusercontent.com/nttcslab/byol-
a/f2451c366d02be031a31967f494afdf3485a85ff/byol_a/models.py
mv models.py models/audio_ntt.py

```

#### 4.1.4. Оценка (предварительно обученной модели) с помощью SERAB

В данной упрощенной версии можно использовать только модели PyTorch.

Перед запуском необходимо убедиться, что конфигурационный файл `config.yaml` правильно настроен для вашей модели.

Чтобы запустить уже существующую модель, требуется выполнить:

```
python clf_benchmark.py --model_name {MODEL_NAME} --dataset_name
{DATASET_NAME}
```

По умолчанию выполняется оптимизация гиперпараметров классификатора на основе поиска по сетке. Чтобы запустить существующую модель с классификаторами «по умолчанию», необходимо добавить `model_selection -none` ключ:

```
python clf_benchmark.py --model_name {MODEL_NAME} --dataset_name
{DATASET_NAME} --model_selection none
```

Для запуска модели на всех наборах данных SERAB можно использовать DVC.

Для этого необходимо внести соответствующие изменения в `dvc.yaml` и запустить его:

```
dvc repro
```

#### 4.1.5. Обучение модели «à la BYOL-A»

Модели могут быть предварительно обучены на подвыборке звукового набора, содержащего только речь.

Возможно, потребуется внести изменения в `train.py` и `config.yaml` перед началом обучения.

Чтобы обучить модель, необходимо запустить:

```
python train.py {MODEL_NAME} # or dvc repro
```

Поскольку время обучения обычно длительное (10-20 часов в зависимости от модели), авторы рекомендуют использовать `tmux` [14] для подключения и отключения терминалов от данного сеанса.

#### 4.1.6. SERAB датасеты

Далее следует раздел, посвященный датасетам SERAB и их источникам.

В то время как CREMA-D и SAVE уже интегрированы в TFDS, другие наборы данных были добавлены в качестве наборов данных tensorflow.

Код для загрузки этих наборов данных можно найти в `tensorflow_datasets` [15].

Ниже приведены шаги по загрузке и загрузке наборов данных SERAB:

1. В папке `tensorflow_datasets` создайте папки загрузка/руководство (`download/manual`)
2. Загрузите сжатые наборы данных (.zip-файлы) в раздел `tensorflow_datasets/download/manual/`

Далее представлены ссылки на наборы данных SERAB:

- AESDD: <http://m3c.web.auth.gr/research/aesdd-speech-emotion-recognition/>
- CaFE: <https://zenodo.org/record/1478765>
- EmoDB: <http://emodb.bilderbar.info/download/>
- EMOVO: <http://voice.fub.it/activities/corpora/emovo/index.html>
- IEM4 (restricted access): <https://sail.usc.edu/iemocap/>
- RAVDESS: <https://smarticlaboratory.org/ravdess/>
- SAVEE (restricted access): <http://kahlan.eps.surrey.ac.uk/savee/Download.html>
- ShEMO: <https://github.com/mansourehk/ShEMO>
- SUBESCO: <https://zenodo.org/record/4526477#.YcyUeGjMJPY>

3. Создайте каждый набор данных с помощью TFDS CLI:

```
cd tensorflow_datasets/{DATASET_NAME}
tfds build # Download and prepare the dataset to ~/tensorflow_datasets/
```

После выполнения перечисленных выше действий датасеты готовы к использованию.

#### 4.1.7. Цитирование

Также авторы указали информацию о статье для использования при цитировании.

```
@article{scheidwasser2021serab,
  title={SERAB: A multi-lingual benchmark for speech emotion recognition},
  author={Scheidwasser-Clow, Neil and Kegler, Mikolaj and Beckmann, Pierre
and Cernak, Milos},
  journal={arXiv preprint arXiv:2110.03414},
  year={2021}
```

```
}
```

## 4.2. Демонстрационный пример работы модели

Авторами статьи был предоставлен .irupb-файл, в котором можно оценить работу представленной предварительно обученной модели по сравнению с работой SERAB на практике. Скрипт показывает основные шаги, стоящие за сценарием. В начале файла следуют служебные строки кода, необходимые для клонирования репозитория. (см. Рисунок 11).

```
[ ] !git clone https://github.com/Neclow/SERAB.git

Cloning into 'SERAB'...
remote: Enumerating objects: 79, done.
remote: Counting objects: 100% (79/79), done.
remote: Compressing objects: 100% (56/56), done.
remote: Total 79 (delta 20), reused 76 (delta 20), pack-reused 0
Unpacking objects: 100% (79/79), done.

[ ] !git clone https://github.com/Neclow/SERAB.git
!cd SERAB/SERAB/ && curl -O https://raw.githubusercontent.com/nttcs/serab/byol-a/f2451c366d02be031a31967f494afdf3485a85ff/config.yaml && patch --ignore-whitespace < config.diff
!cd SERAB/SERAB/ && curl -O https://raw.githubusercontent.com/nttcs/serab/byol-a/f2451c366d02be031a31967f494afdf3485a85ff/train.py && patch < train.diff
!cd SERAB/SERAB/byol_a/ && curl -O https://raw.githubusercontent.com/nttcs/serab/byol-a/f2451c366d02be031a31967f494afdf3485a85ff/byol_a/augmentations.py && patch < augmentations.diff
!cd SERAB/SERAB/byol_a/ && curl -O https://raw.githubusercontent.com/nttcs/serab/byol-a/f2451c366d02be031a31967f494afdf3485a85ff/byol_a/common.py && patch < common.diff
!cd SERAB/SERAB/byol_a/ && curl -O https://raw.githubusercontent.com/nttcs/serab/byol-a/f2451c366d02be031a31967f494afdf3485a85ff/byol_a/dataset.py && patch < dataset.diff
!cd SERAB/SERAB/byol_a/ && curl -O https://raw.githubusercontent.com/nttcs/serab/byol-a/f2451c366d02be031a31967f494afdf3485a85ff/byol_a/models.py && mv models.py models/audio_ntt.py
```

Рисунок 11. Служебные строки для клонирования репозитория.

Далее представлены служебные строки кода для установки всех библиотек, предназначенных для работы с моделями и для вывода данных (см. Рисунок 12).

### ▼ Install libraries

```
!apt-get install libsox-fmt-all libsox-dev sox > /dev/null
!pip install -q opensmile

!pip install -q pytorch-lightning==1.3.7
!pip install -q torch==1.9.0 torchvision==0.10.0 torchaudio==0.9.0 torchtext==0.10.0
!pip install -q einops==0.4.0
!pip install -q pydub==0.25.1
!pip install -q tensorflow_datasets==4.3.0
!pip install -q librosa==0.8.1
```

7.6 MB 3.7 MB/s

Рисунок 12 - Служебные фрагменты кода для установки библиотек

Далее следует фрагмент кода для загрузки импортов и конфигурационных переменных. В SERAB\_PATH указывается путь к SERAB (см. Рисунок 13).

```
SERAB_PATH = 'SERAB/SERAB/'

import sys

sys.path.append(SERAB_PATH)
```

Рисунок 13. Путь к SERAB.

Далее представлены служебные строки кода для импорта всех библиотек, предназначенных для работы с моделями и для вывода данных (см. Рисунок 14).

```
[ ] import collections
import os

import librosa
import numpy as np
import pandas as pd
import tensorflow_datasets as tfds
import tensorflow_hub as hub
import torch

from pytorch_lightning.utilities.seed import seed_everything
from sklearn.model_selection import GridSearchCV, PredefinedSplit
from sklearn.metrics import recall_score
from torchaudio.transforms import MelSpectrogram

from byol_a.common import load_yaml_config
from byol_a.augmentations import PrecomputedNorm
from clf_benchmark import dat_from_split, get_sklearn_models
from settings import CLF_STATS_DICT, RANDOM_SEED, REQUIRED_SAMPLE_RATE
from utils import compute_norm_stats, generate_embeddings, load_model, save_results, speaker_normalization
```

Рисунок 14. Импорт библиотек.

Для начала работы необходимо определить тип устройства и настроить конфигурационный файл (см. Рисунок 15).

```
seed_everything(RANDOM_SEED)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
cfg = load_yaml_config(f'{SERAB_PATH}/config.yaml')
to_melspec = MelSpectrogram(
    sample_rate=cfg.sample_rate,
    n_fft=cfg.n_fft,
    win_length=cfg.win_length,
    hop_length=cfg.hop_length,
    n_mels=cfg.n_mels,
    f_min=cfg.f_min,
    f_max=cfg.f_max,
)
results = {}

Global seed set to 42
```

Рисунок 15. Определение глобального начального набора.

Далее следует фрагмент кода, с которым можно взаимодействовать для загрузки данных.

Если нужно протестировать модель на наборе данных, отличном от CREMA-D, нужно будет загрузить и создать набор данных (см. Рисунок 18).

Пример того, как работать с emodb, показан ниже. Необходимо раскомментировать строки ниже (UNCOMMENT CODE HERE), чтобы создать набор данных emodb. Создаются подпапки downloads/manual в разделе tensorflow\_datasets и вставляется туда сжатый набор данных emodb (см. Рисунок 16).

```

# Create tensorflow_datasets folders
!mkdir -p ../root/tensorflow_datasets/
!mkdir -p ../root/tensorflow_datasets/downloads/
!mkdir -p ../root/tensorflow_datasets/downloads/manual

## UNCOMMENT CODE HERE
## Copy compressed dataset files into download/manual
# !cp -n drive/MyDrive/SERAB/tensorflow_datasets/downloads/manual/*.zip ../root/tensorflow_datasets/downloads/manual/

## Copy the emoDB TFDS scripts and build the emoDB dataset
# !cp -r -n SERAB/tensorflow_datasets/emoDB ../root/tensorflow_datasets/emoDB
# !cd ../root/tensorflow_datasets/emoDB && tfds build emoDB

```

Рисунок 16. Создание папок и подгрузка наборов данных.

Для демонстрации такого расклада использовался CREMA-D как уже предварительно встроенный в TFDS (см. Рисунок 17).

```
dataset_name = 'crema_d'
```

Рисунок 17. Использование CREMA-D.

```

model_selection = 'predefined'

SingleSplit = collections.namedtuple('SingleSplit', ['audio', 'labels', 'speaker_id'])
Data = collections.namedtuple('Data', ['train', 'validation', 'test'])
all_data = Data(
    train=SingleSplit(*dat_from_split(dataset_name, 'train')),
    validation=SingleSplit(*dat_from_split(dataset_name, 'validation')),
    test=SingleSplit(*dat_from_split(dataset_name, 'test'))
)

orig_sr = tfds.builder(dataset_name).info.features['audio'].sample_rate
num_classes = len(np.unique(all_data.train.labels))

```

Downloading and preparing dataset 579.25 MiB (download: 579.25 MiB, generated: 1.65 GiB, total: 2.21 GiB) to /root/tensorflow\_datasets/crema\_d/1.0.0...  
 DI Completed... 100% 1/1 [00:00<00:00, 1.27 url/s]  
 DI Size... 100% 1/1 [00:00<00:00, 1.37 MiB/s]  
 DI Completed... 100% 1/1 [00:00<00:00, 1.60 url/s]  
 DI Size... 0/0 [00:00<?, ? MiB/s]  
 DI Completed... 100% 7438/7438 [03:16<00:00, 39.69 url/s]  
 DI Size... 0/0 [03:16<?, ? MiB/s]  
 Generating splits... 100% 3/3 [02:15<00:00, 37.88s/ splits]  
 Generating train examples... 100% 5138/5144 [01:25<00:00, 65.17 examples/s]  
 Shuffling crema\_d-train.tfrecord... 100% 5134/5144 [00:08<00:00, 154.42 examples/s]  
 Generating validation examples... 100% 736/738 [00:12<00:00, 58.82 examples/s]  
 Shuffling crema\_d-validation.tfrecord... 99% 729/738 [00:00<00:00, 1650.73 examples/s]  
 Generating test examples... 100% 1552/1556 [00:26<00:00, 46.55 examples/s]  
 Shuffling crema\_d-test.tfrecord... 100% 1556/1556 [00:00<00:00, 1914.41 examples/s]  
 Dataset crema\_d downloaded and prepared to /root/tensorflow\_datasets/crema\_d/1.0.0. Subsequent calls will reuse this data.  
 Finished train  
 Finished validation  
 Finished test

Рисунок 18. Загрузка и подготовка наборов данных.

Для подтверждения успешности загрузки и подготовки наборов данных выводится статистика загрузки данных (см. Рисунок 19).

```

[ ] # Load data statistics
try:
    stats = CLF_STATS_DICT[dataset_name]
except KeyError:
    print(f'Did not find mean/std stats for {dataset_name}.')
    stats = compute_norm_stats(dataset_name, all_data.train.audio, orig_sr, to_melspec)

    CLF_STATS_DICT[dataset_name] = stats

    print(CLF_STATS_DICT)
    normalizer = PrecomputedNorm(stats)

```

Рисунок 19. Статистика загрузки данных.



Далее представлен блок кода для загрузки модели. Для рассматриваемой демонстрации использовался "BYOL-S", переобученный BYOL-A на речевых образцах аудиосетей с 512 функциями (см. Рисунок 20).

```

▶ model_name = 'default'
  ckpt_folder = f'{SERAB_PATH}/checkpoints/'

  model, weight_file = load_model(model_name, cfg, device, ckpt_folder)

  print(weight_file)
  SERAB/SERAB/checkpoints/default1024_BYOLAs64x96-2107292000-e100-bs256-lr0003-rs42.pth

```

*Рисунок 20. Загрузка модели по умолчанию.*

Далее необходимо сгенерировать эмбединги для тренировочной, валидационной и тестовой выборок (см. Рисунок 21).

```

▶ # Generate embeddings
  embeddings = Data(
    train=generate_embeddings(
      model,
      model_name,
      all_data.train.audio,
      'train',
      orig_sr,
      to_melspec,
      normalizer,
      device
    ),
    validation=generate_embeddings(
      model,
      model_name,
      all_data.validation.audio,
      'validation',
      orig_sr,
      to_melspec,
      normalizer,
      device
    ),
    test=generate_embeddings(
      model,
      model_name,
      all_data.test.audio,
      'test',
      orig_sr,
      to_melspec,
      normalizer,
      device
    )
  )

```

```

  Generating embeddings for train: 0%|          | 0/5144 [00:00<?, ?it/s]usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:718: UserWarning: Nam
    return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)
  Generating embeddings for train: 100%|#####| 5144/5144 [00:25<00:00, 201.22it/s]
  Finished train
  Generating embeddings for validation: 100%|#####| 738/738 [00:03<00:00, 199.17it/s]
  Finished validation
  Generating embeddings for test: 100%|#####| 1556/1556 [00:07<00:00, 206.45it/s]Finished test

```

*Рисунок 21. Генерация эмбединга.*

Когда эмбединги сгенерированы, необходимо вывести средние значения, стандартные отклонения и форму эмбедингов (см. Рисунок 22). Также нужно загрузить классификаторы.

```
[ ] print(embeddings.train.mean(), embeddings.train.std())
    print(embeddings.validation.mean(), embeddings.validation.std())
    print(embeddings.test.mean(), embeddings.test.std())
    print(embeddings.train.shape)
    print(embeddings.validation.shape)
    print(embeddings.test.shape)

# Load classifiers
log_list, estimator_list, param_list = get_sklearn_models()

4.344952 4.3818913
4.3764834 4.4149346
4.289814 4.3754816
(5144, 1024)
(738, 1024)
(1556, 1024)
```

*Рисунок 22. Вывод данных эмбедингов.*

Далее нормализуются данные, полученные от спикеров (см. Рисунок 23).

```
[ ] # Speaker normalization
    # Can also try with normal standardization (StandardScaler), should not change the results too much
    normalized_train = speaker_normalization(embeddings.train, all_data.train.speaker_id)
    normalized_validation = speaker_normalization(embeddings.validation, all_data.validation.speaker_id)
    normalized_test = speaker_normalization(embeddings.test, all_data.test.speaker_id)

# Aggregate labels and speaker IDs
normalized_train = np.append(normalized_train, normalized_validation, axis=0)
labels_train = np.append(all_data.train.labels, all_data.validation.labels, axis=0)
speaker_id_train = np.append(all_data.train.speaker_id, all_data.validation.speaker_id, axis=0)
```

*Рисунок 23. Нормализация данных спикера.*

Выводятся шаги применения модели (см. Рисунок 24) и результаты работы программы (см. Рисунок 25).

```

for i, (estimator_name, estimator, param_grid) in enumerate(zip(log_list, estimator_list, param_list)):
    print(f'Step {i+1}/{len(estimator_list)}: {estimator_name}...')
    if model_selection == 'predefined':
        split_indices = np.repeat([-1, 0], [embeddings.train.shape[0], embeddings.validation.shape[0]])
        split = PredefinedSplit(split_indices)
        clf = GridSearchCV(
            estimator,
            param_grid,
            cv=split,
            n_jobs=-1,
            refit=True,
            verbose=0
        )
    else:
        clf = estimator

    clf.fit(normalized_train, labels_train)
    test_acc = clf.score(normalized_test, all_data.test.labels)
    test_uar = recall_score(all_data.test.labels, clf.predict(normalized_test), average='macro')

    results[estimator_name] = {
        'test_acc': test_acc,
        'test_uar': test_uar
    }

    print('Done')

results_df = pd.DataFrame(results).apply(lambda x: round(x * 100, 1))

results_df = results_df[results_df.loc['test_acc'].idxmax()].to_frame()
print(results_df)
results_df = results_df.rename(columns={results_df.columns[0]: dataset_name})

```

```

Step 1/5: LDA...
Done
Step 2/5: LR...
Done
Step 3/5: QDA...
/usr/local/lib/python3.7/dist-packages/sklearn/discriminant_analysis.py:691: UserWarning: Variables are collinear
  warnings.warn("Variables are collinear")
Done
Step 4/5: RF...
Done
Step 5/5: SVC...
Done
      SVC
test_acc  75.3
test_uar  75.4

```

Рисунок 24. Вывод шагов обучения и применение модели.

```
[ ] results_df
```

	crema_d
test_acc	75.3
test_uar	75.4

Рисунок 25. Вывод результатов.

Результаты выводятся и сохраняются в отдельный файл (см. Рисунок 26).

```

[ ] filename = os.path.splitext(os.path.basename(weight_file))[0] if weight_file else model_name
    results_folder = f'{SERAB_PATH}/clf_results/'
    save_results(filename + '.csv', results_df, results_folder)

```

File default1024\_BYOLAS64x96-2107292000-e100-bs256-lr0003-rs42.csv does not exist yet. Creating a new results file.

Рисунок 26. Создание нового файла с результатами.

Считывание и вывод содержимого файла с результатами продемонстрировано на Рисунок 27.

```
[ ] pd.read_csv(f'{SERAB_PATH}/clf_results/default1024_BYOLAs64x96-2107292000-e100-bs256-lr0003-rs42.csv', index_col=0)
```

	crema_d
test_acc	75.3
test_uar	75.4

*Рисунок 27. Считывание файла с результатами.*

## 5. ВЫВОДЫ

В рамках домашнего задания был выполнен обзор теоретических и практических материалов, связанных со статьей «SERAB: A Multi-lingual Benchmark for Speech Emotion Recognition».

В статье рассматривался SERAB, многоязычный тест для распознавания речевых эмоций. С быстрым появлением основанных на DNN репрезентаций речи и эмоций, основанных на речи, бенчмарк предоставляет универсальную платформу для сравнения различных методов. Благодаря включению разнообразных задач, охватывающих разные языки, размеры наборов данных и эмоциональные категории, SERAB производит надежные оценки производительности и способности к обобщению. В статье использовался SERAB для оценки ряда недавних исходных показателей. Среди протестированных фреймворков подходы, основанные на BYOL, показали превосходную производительность по всем рассмотренным показателям. Предварительное обучение моделей BYOL-A только на речевых образцах AudioSet (BYOL-S) привело к повышению точности почти на 3% по сравнению с оригинальным методом. Представленные результаты оценки могут быть использованы в качестве исходных данных для разработки новых подходов, таких как методы, основанные на CvT, рассмотренные здесь. В будущем авторы статьи планируют сосредоточить работу на включении большего количества наборов данных на еще большем количестве языков в SERAB, а также на расширении круга задач, включив в него проблемы регрессии, такие как оценка валентности или возбуждения. Чтобы облегчить использование SERAB, фреймворк, включая инструкции по настройке, оценочные конвейеры и примеры, находится в свободном доступе в Интернете".

В практической части был выполнен обзор содержимого репозитория авторов статьи на GitHub, а также было выполнено тестирование моделей в демо-примере. В результате тестирования можно подтвердить, что модели действительно были обучены распознавать речевые эмоции.

## 6. СПИСОК ИСТОЧНИКОВ

1. Browse State-of-the-Art. – Текст. Изображение: электронные // Papers With Code : [сайт]. – URL: <https://paperswithcode.com/sota> (дата обращения: 25.05.2022).
2. Speech Emotion Recognition. – Текст. Изображение: электронные // Papers With Code : [сайт]. – URL: <https://paperswithcode.com/task/speech-emotion-recognition> (дата обращения: 25.05.2022).
3. Speech Emotion Recognition Project using Machine Learning. – Текст. Изображение: электронные // ProjectPro : [сайт]. – URL: [Speech Emotion Recognition Project using Machine Learning \(projectpro.io\)](https://projectpro.io/Speech-Emotion-Recognition-Project-using-Machine-Learning) (дата обращения: 25.05.2022).
4. Continuous control with deep reinforcement learning. – Текст. Изображение: электронные // Papers With Code : [сайт]. – URL: <https://paperswithcode.com/paper/continuous-control-with-deep-reinforcement> (дата обращения: 25.05.2022).
5. SERAB: A multi-lingual benchmark for speech emotion recognition. – Текст. Изображение: электронные // Papers With Code : [сайт]. – URL: <https://paperswithcode.com/paper/serab-a-multi-lingual-benchmark-for-speech> (дата обращения: 25.05.2022).
6. Распознавание эмоций речи с помощью сверточной нейронной сети. – Текст. Изображение: электронные // machinelearningmastery : [сайт]. – URL: <https://www.machinelearningmastery.ru/speech-emotion-recognition-with-convolution-neural-network-1e6bb7130ce3/?> (дата обращения: 25.05.2022).
7. SERAB. – Текст. Изображение: электронные // GitHub : [сайт]. – URL: <https://github.com/Neclow/SERAB> (дата обращения: 25.05.2022).
8. SERAB: A multi-lingual benchmark for speech emotion recognition. – Текст. Изображение: электронные // arXiv : [сайт]. – URL: <https://arxiv.org/abs/2110.03414> (дата обращения: 25.05.2022).
9. byol-a. – Текст. Изображение: электронные // GitHub : [сайт]. – URL: <https://github.com/nttcs/ab/byol-a> (дата обращения: 25.05.2022).
10. vit-pytorch. – Текст. Изображение: электронные // GitHub : [сайт]. – URL: <https://github.com/lucidrains/vit-pytorch> (дата обращения: 25.05.2022).

11. HEAR Leaderboard. – Текст. Изображение: электронные // HEAR Benchmark : [сайт]. – URL: <https://hearbenchmark.com/hear-leaderboard.html> (дата обращения: 25.05.2022).
12. demo. – Текст. Изображение: электронные // Colaboratory : [сайт]. – URL: <https://colab.research.google.com/drive/1EiHujFVt9Hb0VbI0b5RaMfYOYaHq9NrQ?usp=sharing> (дата обращения: 25.05.2022).
13. Serab-BYOL-S Package. – Текст. Изображение: электронные // Colaboratory: [сайт]. – URL: <https://colab.research.google.com/drive/1tvL-rAY6uYPGLrCdSFJoaFG1mkOZkud?usp=sharing> (дата обращения: 25.05.2022).
14. tmux. – Текст. Изображение: электронные // GitHub : [сайт]. – URL: <https://github.com/tmux/tmux> (дата обращения: 25.05.2022).
15. Writing custom datasets. – Текст. Изображение: электронные // TensorFlow : [сайт]. – URL: [https://www.tensorflow.org/datasets/add\\_dataset](https://www.tensorflow.org/datasets/add_dataset) (дата обращения: 25.05.2022).