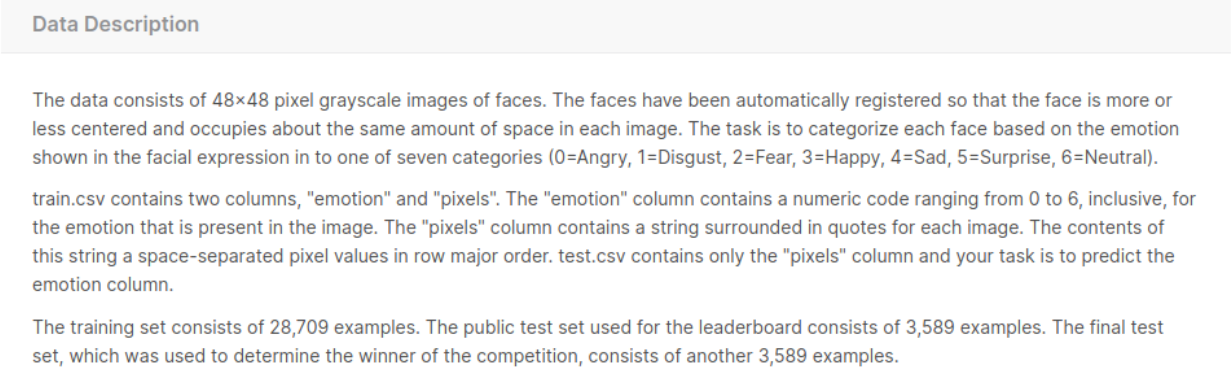
Структура

Основой тестового задания является решение некоторой целевой задачи (например задачи классификации/регрессии/сегментации).
Разработка проекта может состоять из двух частей:
1) подготовка инструментов решения целевой задачи. Как правило, это обучение искусственной нейронной сети, включающее следующие этапы: подготовка/валидация данных для обучения, выбор и обоснование подходящего инструмента (архитектура модели искусственной нейронной сети), обучение модели (baseline), корректировка данных и модели для увеличения точности (аугментация данных для обучения, подбор параметров модели, использование пред обученных весов и т.п.). В зависимости от целевой задачи могут использоваться альтернативные подходы (алгоритмы классического computer vision, boosting алгоритмы и т.п.);
2) реализация веб-приложения (или десктопного приложения) для демонстрации работы полученных инструментов решения целевой задачи.

Целевые задачи

Классификация эмоций

Цель: подготовить модели для классификации эмоций лица, оценить качество их работы, предложить варианты улучшения качества работы.
Данные: https://drive.google.com/file/d/1b_IdRm6pp0WmsofaG3dLGPrF08vRseu2/view?usp=sharing



Data Description

The data consists of 48×48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. test.csv contains only the "pixels" column and your task is to predict the emotion column.

The training set consists of 28,709 examples. The public test set used for the leaderboard consists of 3,589 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples.

Задачи:
1. Исследовать текущие актуальные модели по данной задаче.
2. Распарсить данные в удобный для обучения вид. Данные разбить на трэин и тест. Данные предварительно аугментировать керас/пайторч генератором или вручную. (выбор обосновать)
3. Обучить классификатор с бэкбон архитектурами типа ResNet, EfficientNet, Inception, GoogleNet, etc. (на выбор, выбор обосновать)
4. Снять метрики точности на тестовых данных. Построить матрицу ошибок (confusion matrix)
5. Сравнить результаты с текущими актуальными моделями.
6. Предложить свои варианты улучшения метрик точности результатов.

Примечания:
1. При отсутствии необходимых мощностей gpu. Использовать средства Google Colab, либо AWS cloud
2. Рекомендуемый фреймворк для использования Keras+Tensorflow2.0.

Приложение для демонстрации

Общие требования:
1) язык интерфейса - English;
2) логирование в файл и консоль (с помощью модуля logging);
3) настроить авторизацию и регистрацию пользователей;
4) наличие формы для заполнения исходных данных (например загрузка изображения и ввод некоторой дополнительной информации);
5) возможность сохранить полученные результаты на диск (например в форматах pdf и json);
6) если целевая задача связана с компьютерным зрением, в приложении также необходимо визуализировать результат на исходном изображении. Полученное изображение отобразить пользователю;
7) данные о зарегистрированных пользователях и результаты проведенных вычислений необходимо сохранять в базу данных;
8) для каждого пользователя хранить количество проведенных вычислений.
9) выделить отдельно сервис, который занимается обработкой, запускать его в виде Docker контейнера.
10) обмен данными между бэком и сервисом обработки может быть по http или через брокер сообщений (например, Kafka).
11) бонусом будет использование multiprocessing в сервисе обработки.

Основной вариант использования:
1) пользователь открывает приложение;
2) пользователь проходит авторизацию/регистрацию;
3) после авторизации происходит переход на главную страницу (главное меню). Здесь пользователь может перейти либо в свой аккаунт (там отображается информация о пользователе с возможностью редактирования), либо на страницу с формой для проведения вычислений.
4) пользователь переходит на страницу с формой, загружает исходные данные и запускает вычисление (щелчком по соответствующей кнопке);
5) после завершения вычислений пользователю отображается результат;
6) пользователь сохраняет полученные результаты в виде файла (щелчком по соответствующей кнопке).
Бэк может быть реализован на выбор:
Django web-application
1) рекомендуется использовать встроенную систему авторизации и аутентификации;
2) необходимо продемонстрировать работу с DTL;
3) можно выбрать любую БД, которую поддерживает Django ORM.
Flask web-application
1) рекомендуется использовать Flask-Login;
2) взаимодействовать с базой данных можно либо с помощью Flask-SQLAlchemy, либо используя другие ORM (например peewee).