**Документация за AI проект „Разпознаване на емоции от лицеви мимики“**

София Желева и Зорница Ламбова

**Принос**: Зорница Ламбова – модел на обучение; София Желева – интерфейс

**Използването на модели и технологии и обосновка на използването им**

Програмата е написана на програмата Visual Studio Code. Dataset за обучение на модела е взет от интернационално състезания проведено в 2013 по Разпознаване на емоции от снимки в Kaggle(https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge). Средствата използвани за програмата са:

* **Numpy** – библиотека за математически функции, генератори на случайни числа и др.
* **Pandas** – библиотека за функция за анализ и манипулация на данни с отворен код като предлага структури от данни и операции за манипулиране на числови таблици и времеви серии.
* **Tensorflow** + **keras** – библиотека с колекция на работа за съставяне на модели за обучение, съдържа инструменти и няколко под библиотеки за обработка на модели и дайтасетове.
* **Sklearn** – инструменти за анализ на данни за предсказване
* **IPython** – seems to be a support for the program python
* **Time** - библиотека свързана с времето; често използвана за отсичане на време
* **Os** – свържва се с външни файлове
* **PIL** – библиотека за работа с изображения; използва се заедно с Tkinter
* **Tkinter** – библиотека, която позволява създаването на Графичен потребителски интерфейс (GUI)
* **OpenCV** – библиотека за open source компютърно зрение и машинно обучение; използва се най – често за изграждане на инфраструктурата на приложенията за компютърно зрение
* **PySimpleGUI** – библиотека, която позволява да се създават графични потребителски интерфейси (GUIs), които са прости за изграждане и лесно разбираеми
* **Classifiers** – алгоритми, които решават дали дадена снимка е позитивна или негативна за дадена категория (Пр. дали съдържа лице или не); използваме ги за засичане на лица в изображения и в реално време

**Данни за проведените експерименти**

Първоначално проведохме множество експерименти с два вида код – един написан на Visual Studio Code, друг на Google colab, защото използва няколко функции, които само там се срещат. Нашия модел беше по-кратък само 83 реда, но всяко обучение на модела отнемаше 25 минути и точността му при тестовите данни не надвишава 0.5, при тестване на модела това се виждаше много по-явно(- 41s 45ms/step - loss: 0.0773 - accuracy: 0.9877 - val\_loss: 2.1991 - val\_accuracy: 0.5325). Промяна на епохите или броя на изображение на епоха не промениха значително точността на модела, затова трябваше да се добавят нови слоеве към нивата на модела, което увеличи точността с една стотица. Накрая сменихме кода за обучение на модела и използвахме предварително натрениран модел с 83% точност в периода на трениране и 79% точност върху тестовите данни.

Chart, bar chart

Description automatically generatedПреди да започнем с изграждането на новия модел се занимавахме с анализиране на данните и подготвянето им за използване в самия модел. След изпълнение на анализа открихме няколко проблема. Първо, от седемте първоначални класа в два от тях (Неутралност и Страх) изображенията не са с добро качество и биха могли да объркат модела. Второ, за Отврат почти липсват данни, както можете да видите в приложената схема:

След получаването на тези резултати взехме решението да работим само върху класовете „Гняв“, „Радост“, „Тъга“ и „Изненада“. Затова подготвихме данните като премахнахме ненужните класове и разделихме тези които ще използваме в директори, за да са по – лесно достъпни за модела. Тези действия бяха изпълнение в Google colab, след което пуснахме да се тренира намерения от нас готов модел, но тъй като му отнемаше твърде много време(90 минути на епоха), използвахме предварително натренирания модел.

В нашия модел са използвани 3 нива всеки със по слой Conv2D, MaxPool2D, BatchNormalization, Dropout и след тях един Flatten слой и два Dense слоя. Слоя Conv2D създава конволюционно ядро, което се навива с входния слой, за да произведе тензор на изходите. Слоя MaxPool2D намалява броя на изображения на входа по неговите височина и ширина, като взема максималната стойност за входен прозорец (с размер, определен от pool\_size) за всеки канал на входа, Прозорецът се измества с крачки по всяко измерение. BatchNormalization прилага трансформация, която поддържа средната продукция близо до 0 и стандартното отклонение близо до 1. Dropout е слой, който задава произволно входните единици на 0 с честота на скорост на всяка стъпка по време на обучението, което помага за предотвратяване на overfitting. Входовете, които не са зададени на 0, се увеличават с 1, така че сумата за всички входове да остане непроменена. Flatten е функция, която превръща входните данни в едномерен масив вместо двумерен, докато го прави размерът на данните не се променя. Накрая се ползва два слоя Dense за да се уплътни модела за да може да се изведат получените данни.

**Данни за финалното обучение – точност**

Данни за финалното обучение на нашия модел са 69% в периода на трениране и 58% върху тестове данни, ако се махне Dropout слоя в всяко ниво данните са 98% в периода на трениране и 53% върху тестовите данни. Докато предварително натренирания модел, който е интегриран в интерфейса е с 83% точност върху тренировъчните данни и 79% точност върху тестовите.

Graphical user interface, application

Description automatically generated**Данни за архитектурата на крайното приложение**

Целта на крайното приложение е да разпознава емоциите от данни приложени от потребителя и дори в реално време да разпознава емоциите през камерата на устройството.

Самият интерфейс е изграден чрез библиотекaта Tkinter в допълнение с библиотеките Pillow, OpenCV и PySimpleGUI. Основата на интерфейса заедно с бутоните и етикетите са изградени чрез функциите на Tkinter. За разпознаването от потребителски данни се използват библиотеките Pillow и OpenCV за оразмеряване на изображенията и засичане на лицата в тях чрез classifiers (https://github.com/opencv/opencv/tree/4.x/data/haarcascades). Докато чрез PySimpleGUI и отново OpenCv се извършва разпознаването в реално време.

A person taking a selfie

Description automatically generated

Това беше първоначалната идея за потребителския интерфейс. Обаче поради срещнатите затруднения със самия модел, достигнахме и до трудности в неговата интеграция. Затова се наложи да опростим първоначалната идея до разпознаване в реално време на емоции чрез снимки направени от камерата на устройството. За целта използвахме библиотеките OpenCv и Tensorflow, съответно за изграждане на интерфейса и свързване на модела с него.

**Предимства и недостатъци на реализирания проект**

* **Предимства** – би подпомогнало в обучение на малки деца по – лесно да разпознават и разграничават различните емоции в реално време и от изображения;
* **Недостатъци** – поради недостатъчните данни в определени категории липсват няколко от основните емоции; тренирането на крайния модел отнема прекалено много време; постигане на по – голяма точност при изготвения от нас модел

**Бъдещи подобрения и цели**

За бъдещето планираме:

* да добавим повече категории за разграничаване на различните емоции;
* да оптимизираме модела;
* да постигнем по – голяма точност;
* да интегрираме успешно модела в желания от нас интерфейс;
* да добавим съобщения, които да помогнат на потребителите да се справят с емоциите, които изпитват в момента;