Универзитет у Крагујевцу
Факултет инжењерских наука

Основи дубоког учења

Пројекат:
Препознавање насмејаних лица на сликама

Студент:
Михаило Јаћимовић 636/2018

Предметни професори:
Проф. Др. Владимир Миловановић

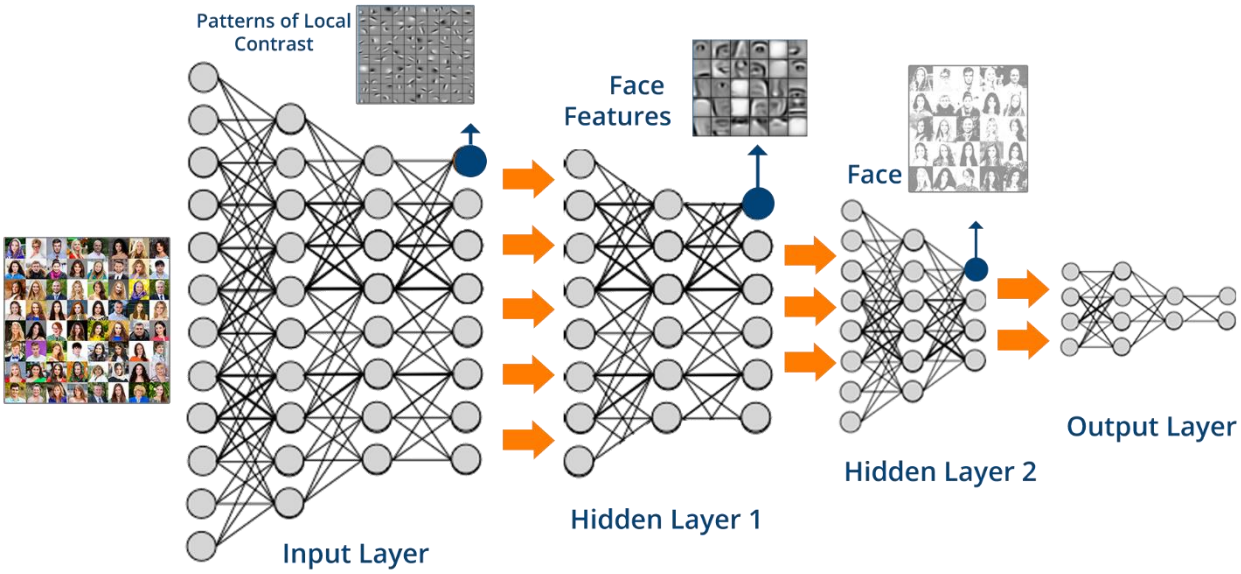
Крагујевац, Јун 2022.

Садржај

1. Поставка задатка

1.1. Циљ пројектног задатка

Циљ пројектног задатка је да се направи програм који ће да препознаје насмејана лица на сликама. Потребно је сакупити улазни скуп података који ће представљати слике на којима су људи насмејани. Затим је потребно креирати модел који ће се тренирати на улазном скупу података и сачувати тако истрениран модел.
Након тренирања наша мрежа ће на новим сликама моћи да детектује да ли се особа на слици смеје или се не смеје.



Слика 1 – Изглед једне CNN мреже

1.2. Опис података

Наш улазни скуп података представљају слике на којима особа може да се:

1. смеје
2. не смеје
3. не налази на слици

За израду овог пројектног задатка сакупљено је 300 слика, што не представља велики скуп података. Над датим скупом је извршена прерада слика(data augmentation) како бисмо добили више слике над којима бисмо могли да тренирамо мрежу.
Након извршене обраде слика, скуп смо повећали са 300 на 6000 слика.
Када смо истренирали мрежу, она би требало да тачно предвиђа да ли се особа смеје на сликама које до тада није видела.

Слика 2 – Део слика из улазног скупа података

2. Израда модела

За потребе овог пројекта коришћен је програмски језик Python. У делу 2.1. описане су библиотеке које су коришћене у реализацији датог задатка, у делу 2.2. описано је креирање модела. У одељку 2.3. описана је дискусија о добијеним резултатима.

2.1. Python библиотеке

За решавање датог задатка коришћене су следеће python библиотеке:

1. matplotlib – користи се за визуализацију података
2. numpy – користи се за рад са матрицама и низовима
3. tensorflow – библиотека за дубоко учење
4. os – библиотека која омогућава лакши приступ фајловима и фолдерима
5. albumentations – користи се за прерађивање и увећање скупа података
6. cv2 – библиотека за манипулацију сликама
7. keras – интерфејс за tensorflow библиотеку

```
import tensorflow as tf
import cv2
import json
import numpy as np
import os
from matplotlib import pyplot as plt
import albumentations as alb
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, Dense, GlobalMaxPooling2D
from tensorflow.keras.applications import VGG16, ResNet50V2
from tensorflow.keras.models import load_model
```

Слика 3 – Листа свих коришћених библиотека

2.2. Креирање модела

Скуп почетних података са којим располажемо има 300 слика, то не предтавља велики број улазних података на којем може да се истренира мрежа која би радила добро, тако да је пре креирања модела потребно увећати улазни број података, то радимо на два начина:

1. прикупљањем додатних слика
2. коришћењем *data augmentation* метода

У овом пројекту је коришћена друга метода.

```
transformation = alb.Compose([alb.RandomCrop(450, 450),
                        alb.HorizontalFlip(p=0.5),
                        alb.RandomBrightnessContrast(p=0.3),
                        alb.RandomGamma(p=0.3),
                        alb.RGBShift(p=0.3),
                        alb.VerticalFlip(p=0.5)],
                        bbox_params=alb.BboxParams(format='albumentations',
                                                   label_fields=['class_names'])
                       )
```

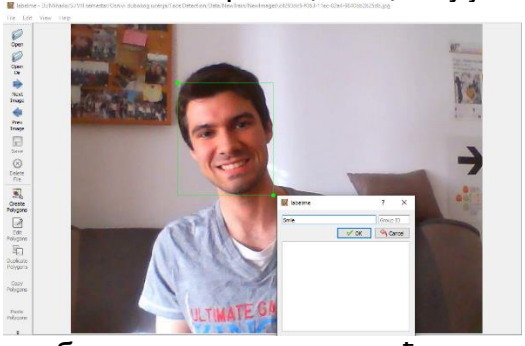
Слика 4 – Изглед дефинисања правила за обраду слика

Од сваке слике креирано је 20 нових слика. Неке су ротиране хоризонтално, неке су ротиране вертикално, неке су одсечене, некима је контраст боја промењен,...
На тај начин повећан је скуп улазних података.
Величину слике стандардизоване су на димензије (120, 120) и нормализоване су вредност пиксела на вредности између 0 и 1 како би процес учења био убрзан.

```
train_images = tf.data.Dataset.list_files('Aug_data\\NewTrain\\NewImages\\*.jpg'
train_images = train_images.map(load_image)
train_images = train_images.map(lambda x: tf.image.resize(x, (120,120)))
train_images = train_images.map(lambda x: x/255)
test_images = tf.data.Dataset.list_files('Aug_data\\NewTest\\NewImages\\*.jpg',
test_images = test_images.map(load_image)
test_images = test_images.map(lambda x: tf.image.resize(x, (120,120)))
test_images = test_images.map(lambda x: x/255)
val_images = tf.data.Dataset.list_files('Aug_data\\NewVal\\NewImages\\*.jpg', sh
val_images = val_images.map(load_image)
val_images = val_images.map(lambda x: tf.image.resize(x, (120,120)))
val_images = val_images.map(lambda x: x/255)
```

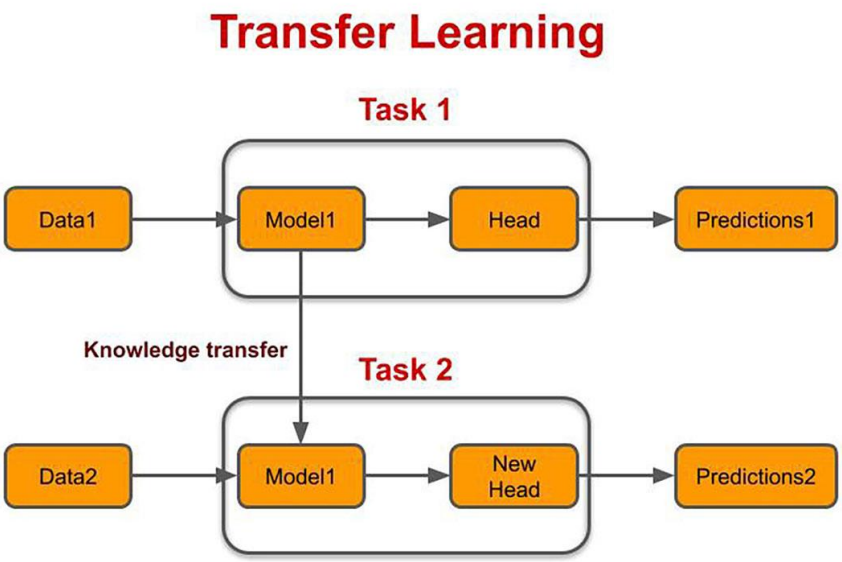
Слика 5 – Припремање података за учење

Програм ће поред детекције да ли се особа на слици смеје или не и да ограничи правоугаоником лице на којем је препознао осмех. Зато је потребно да на свакој слици на којој се особа смеје ручно означити и границе лица. Ту је коришћен програм labelme.

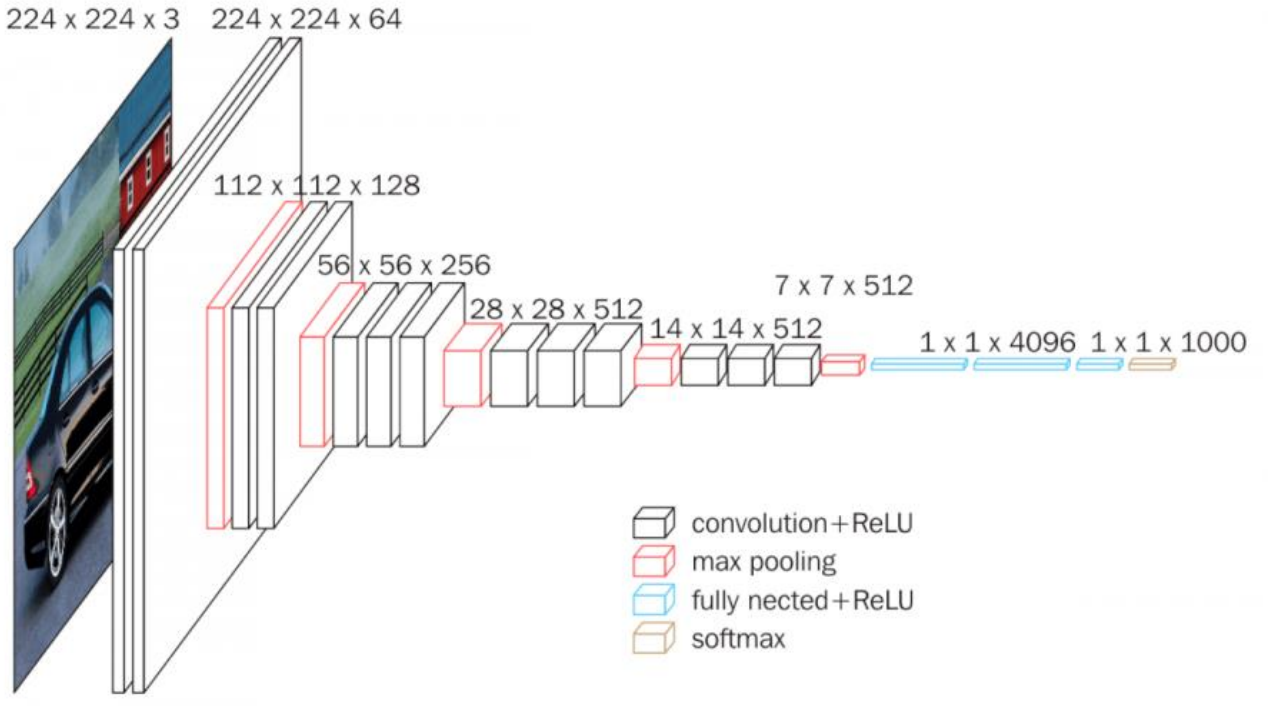


Слика 6 – Приказ лабелирања слика коришћењем програма labelme

За креирање модела коришћена је метода *transfer learning.* Keras раслопаже са великим бројем већ истренираних мрежа. Суштина *transfer learning-a* је у томе да се неколико последњих слојева истренираног модела уклоне са датог модела и да се додају нови слојеви на којим ће се тренирати мрежа. Модел је трениран на неколико различитих истренираних мрежа(VGG16, VGG19 и ResNet50) и најбоље резултате је постигла мрежа VGG16.



Слика 7 – Transfer learning



Слика 8 – Изглед VGG16 мреже

Пошто ће модел који се тренира имати две функције губитака потребно је креирати модел специфичан за дати случај(редефинисањем неких фунција општег модела).
Прави се класа која ће наследити главни модел и у себи изменити одређене фунције. Први измена јесте постављање да ће модел имати две фунције губитака, а не једну. И то бинарну класификацију за препознавање да ли се особа смеје или не и фунцију губитака локализације која ће приказивати где се налази лице које се смеје(мењамо фунцију *compile()* класе модел).
Када желимо да променимо фунционалност *fit()* фунције, мењамо *train_step()* функцију класе модел. Пошто имамо две фунције губитака, потребно је кориговати укупну фунцију губитака као збир ове две функције губитака. Затим рачунамо градијент и ажурирамо вредности коефицијената.
Када желимо да променимо фунционалност *evaluate()* фунције, мењамо *test_step()* функцију класе модел. У овом кораку само рачунамо појединачне функције губитака и затим рачунамо укупне губитке.
За оптимизацију смо користили *adam* алгоритам.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

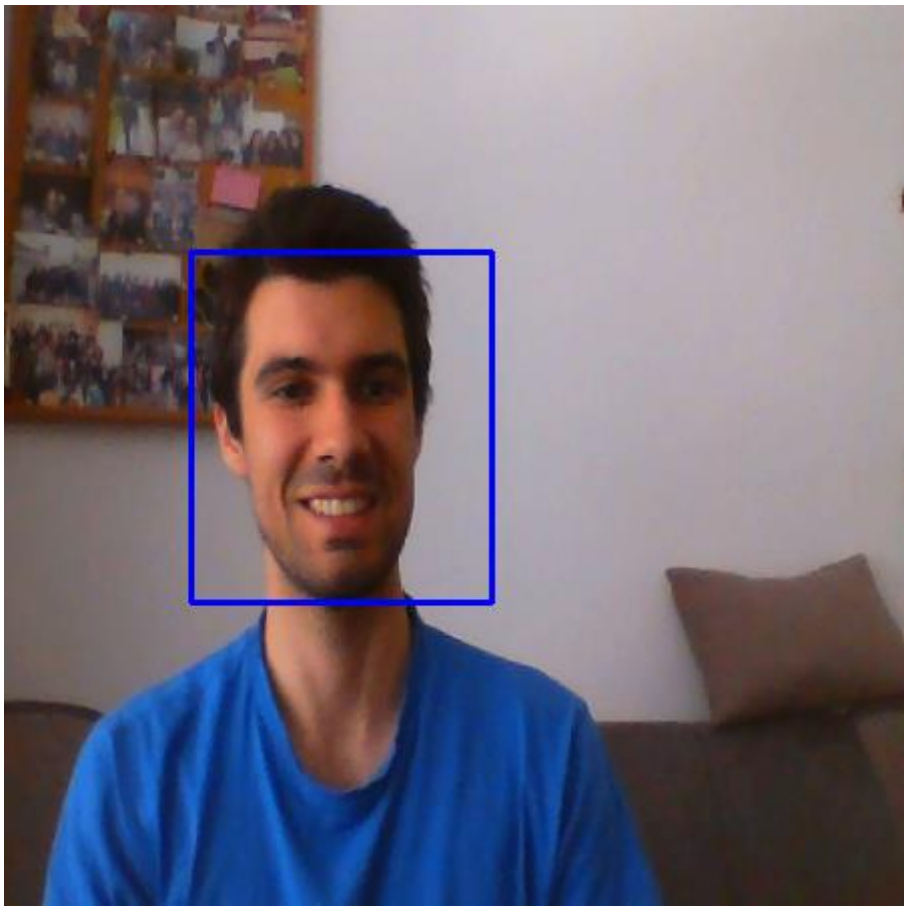
Слика 9 – Бинарна функција губитака и функција губитака локализације

2.3. Дискусија резултата

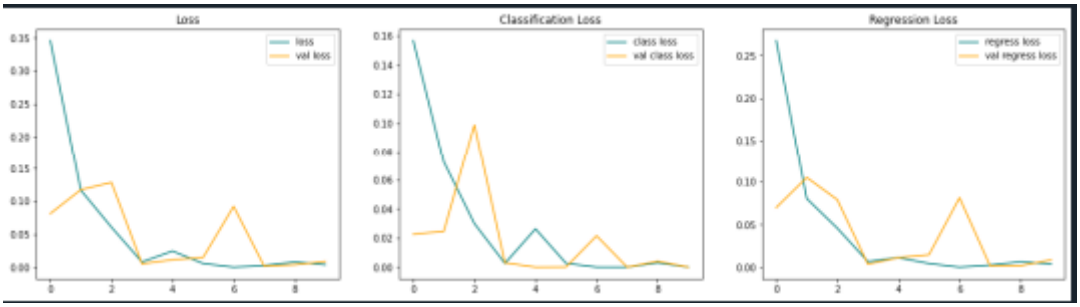
За дати пројекат истренирано је неколико модела.
Сваки модел је био изграђен из два дела. Први део чини истренирана мрежа, чији су параметри залеђени, док други део представља комбинацију различитог броја слојева. Најбоље резултате су постизали модели који су у себи садржали део VGG16 мреже.

```
def build_model():
    input_layer = Input(shape=(120,120,3))

    vgg = VGG16(include_top=False)(input_layer)

    # Classification Model
    f1 = GlobalMaxPooling2D()(vgg)
    class1 = Dense(2048, activation='relu')(f1)
    class2 = Dense(1, activation='sigmoid')(class1)

    # Bounding box model
    f2 = GlobalMaxPooling2D()(vgg)
    regress1 = Dense(2048, activation='relu')(f2)
    regress2 = Dense(4, activation='sigmoid')(regress1)

    smiletracker = Model(inputs=input_layer, outputs=[class2, regress2])
    return smiletracker
```

Слика 10 – Изглед једног од модела



Слика 12 – Тестирање модела на новим подацима



Слика 13 – График укупне функције губитака, функције губитака бинарне класификације и функције губитака локализације модела VGG16

3. Закључак

Циљ пројекта је био реализација мреже за детекцију осмеха на сликама. Да би се дошло до мреже која ће давати добре резултате било је потребно направити улазни скуп података, извршити одређену манипулацију над њим, направити одговарајућу мрежу.
Методе за побољшање модела:

1. већи скуп улазних података
2. разноврснији скуп података
3. тестирање дубљих мрежа

4. Литература

1. Stuart Russell, Peter Norvig – Artificial Intelligence, a modern approach
2. https://keras.io/
3. https://docs.python.org/3/
4. Coursera – Object localization with TensorFlow - project