# Универзитет у Београду
# Машински факултет

## *МАШИНСКО УЧЕЊЕ*

| Овера (потпис/датум): | Наставници: Проф. Др Младен Николић<br>Асистент: Невена Ћирић |
|---|---|
| | Група: |
| | РБ Презиме и име: Бр.инд. Број поена: |
| | 1. Ђорђе Кожул 4004-2021 |

# 1. Садржај

# 2. Увод

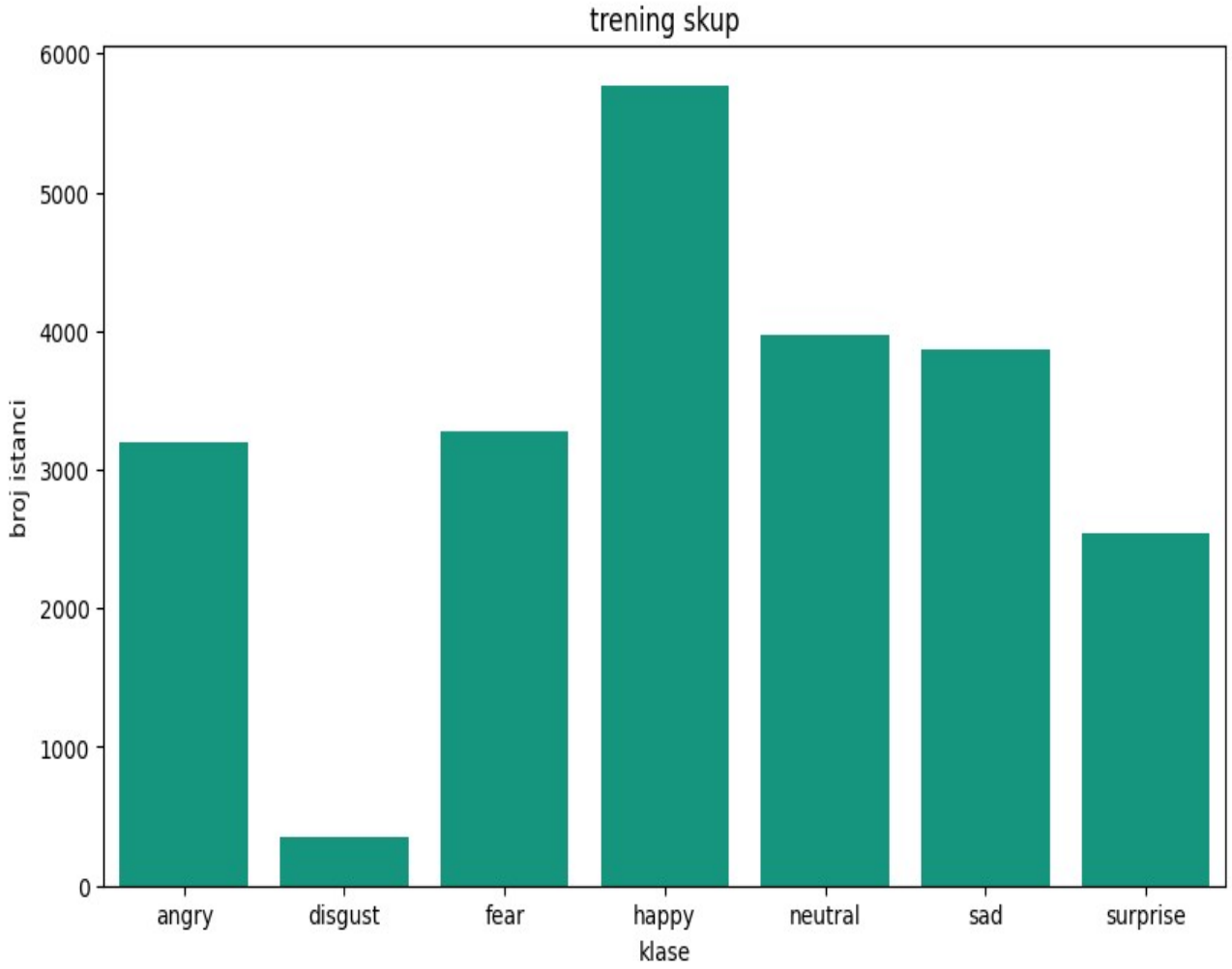
У оквиру пројекта разматрана је анализа израза лица (facial expression recognition) на скупу  FER 2013. Циљ пројекта је обучити моделе који ће у одређеној мери (колико им то квалитет података дозвољава)  успешно класфиковати различите изразе лица.

Скуп је састављен од 7 различитих израза лица. Поред тога скуп је небалансиран и у таквим ситуацијама може бити изазовно постоћи задовољавајућу тачност.

Поред класичних конволутивних мрежа биће коришћени и предренирани модел ResNet тј. различити његови модели.

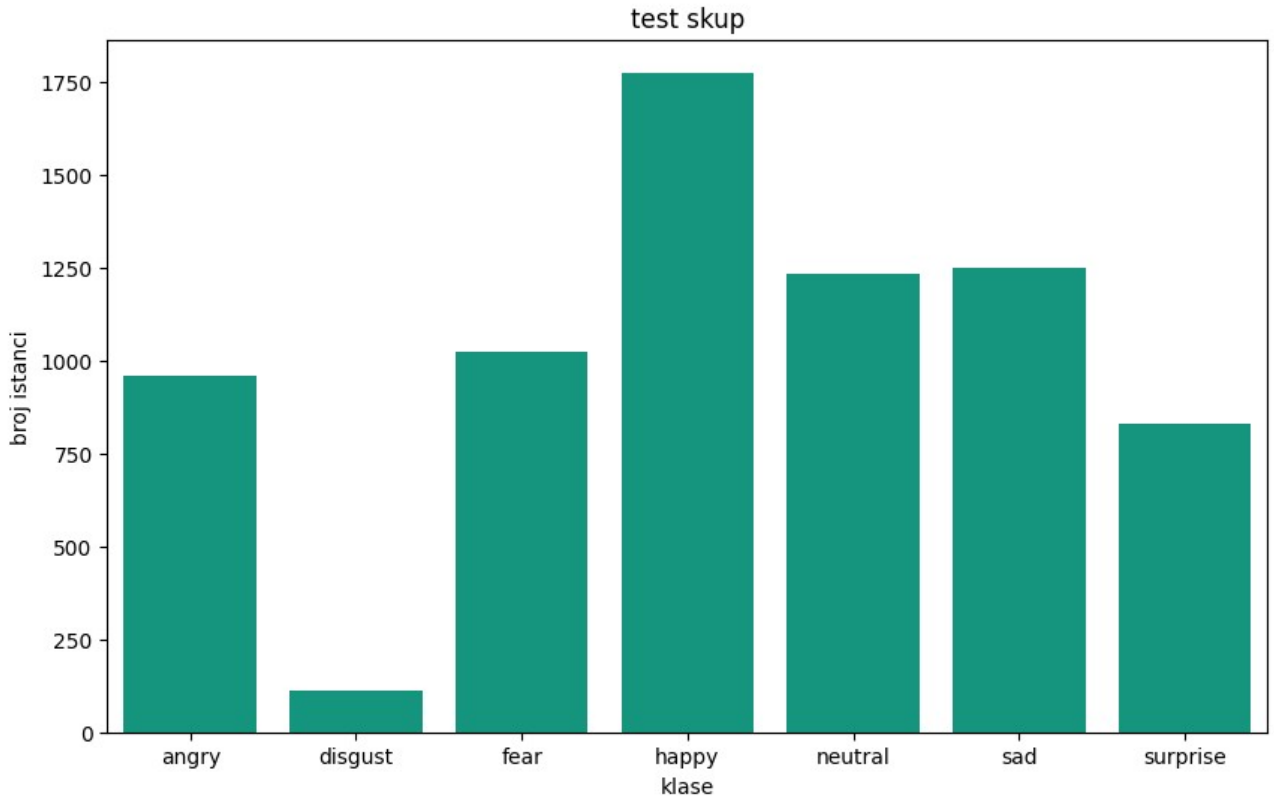
# 3. Предпроцесирање и анализа скупа података

Што се тиче препроцесирања неопходно је рећи да се анализа слика доста разликује од анализе табеларних података. За разлику од табеларних података код слика вероватно нема много смисла вршити класичне статистичке анализе као што су хистограм атрибута, корелацију средњу вредност итд. Овде су дате неке основне анализе као што је број истанци по класама.



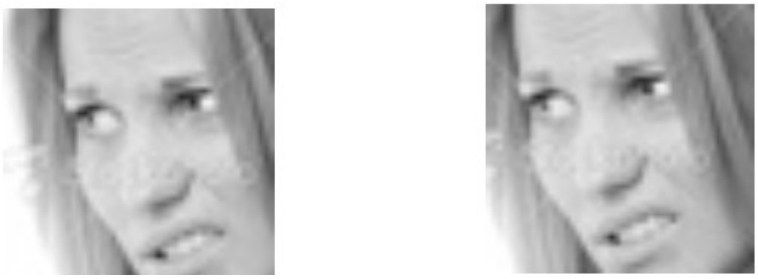

*Слика 1: број истанци по класама за тренинг*

Као што се може видети са слике, скуп података је неизбалансиран. Поред тога квалитет слика је веома лош, многе од слика су практично дупликати на некима малим додацима као што су шифтовање слике нпр. у десно. Један део слика има жиг неког сајта на коме се продаје та слика, у скупу се поред тога јављају слике без лица и скроз црне слике или црне слике са узвичником које штете тренингу и стварају шум.



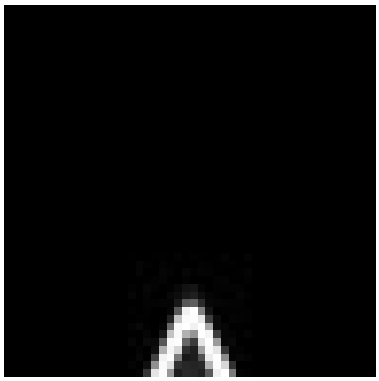

*Слика 2: број истанци по класама за тест скуп*

*Слична је ситуација и на тест скупу. Најмање је заступљена класа “Згађен”, највише “срећан“.*



*Слика 3 : силке из скупа disgust*

Као што се види овде две слике су практично исте само је над једном од њих примењена аугментација тј. нека од трансформација.



*Слика 4:  слика из скупа angry*

Ова слика је потпуно бесмислена у контексту препознавања емоција и можда би било најбоље да се овакве слике уклоне из скупа. То би могао бити проблем јер је потребно наћи начин да се овакве слике уклоне. Свакако да је број оваквих слика релативно мали и да поред нерелевантности за овај проблем ове слике су и одударајући подаци. Један од начина који би евентуално могао да успе, али је можда и ризичан јесте да се упросече вредности пикселе за све слике и се избаце оне које су препознати као одударајући подаци. Можда би могао да се нађе и Z-скор истанци и да се избаце истанце које не упадају у 3 стандардне девијације.
Ово можда не би дало никакве резултате, али то је била једна од идеја.

Поред свега наведеног треба напоменути да су неке од слика које се налазе у различитим класама изгледају више као истанце неке друге класе.
То је велики проблем и по мени суштински јер ће резултати који буду приказани у наставку бити незадовољавајући.

*Слика 5: Пример слике која изгледа као да је из скупа surprise али је из скупа fear*

```
train_generator=keras.preprocessing.image.ImageDataGenerator(
                                                rescale=1./255,
                                                horizontal_flip = True,
                                                vertical_flip= True,
                                                width_shift_range = [0.1,0.2],
                                                height_shift_range = [0.1,.2],
                                                validation_split=0.2
)
test_generator=keras.preprocessing.image.ImageDataGenerator(
                                                rescale=1./255
)
```

*Слика 6: Креирање објекта ImageDataGenerator и аугментација учитаних података*

Слике су учитане и извршена је њихова аугментација и одвојен је део од тренинг скупа за скуп за валидацију (20% скупа). Извршена је нормализација пиксела тако да они буду у распону од 0 до 1. Додато, као и остале трансформације слика као шифотвање по висини и ширини и флиповање.

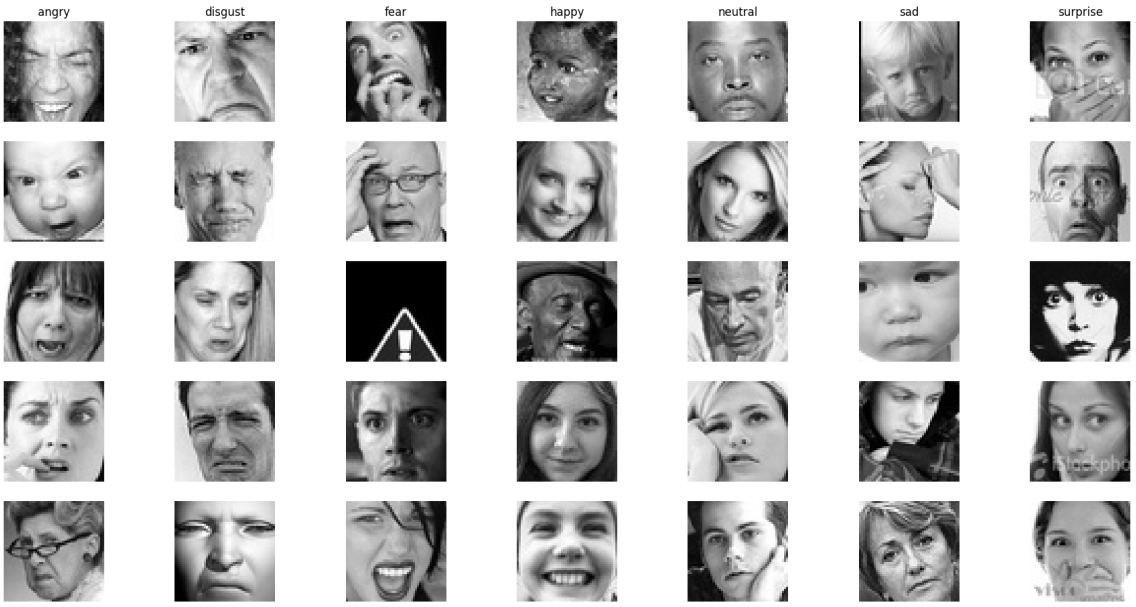
```
train_batch= train_generator.flow_from_directory(
    directory=train_dir,
    target_size=(48, 48),
    batch_size=64,
    color_mode="grayscale",
    class_mode="categorical",
    shuffle=True,
    subset="training",
    seed=11
)
```

```
validation_batch= train_generator.flow_from_directory(
    directory=train_dir,
    target_size=(48, 48),
    batch_size=64,
    color_mode="grayscale",
    class_mode="categorical",
    subset="validation",
    seed=11
)
```

```
# kreiranje objekta za batch za test
test_batches=train_gen.flow_from_directory(
    directory=direktorijum_test,
    target_size=(48,48),
    shuffle=True,
    classes=klase,
    color_mode='grayscale',
    batch_size=32,
    class_mode='categorical'
)
```

*Слика 7: Задавање путање до скупа података и дефинисање појединачног batcha*

Код који учитава податке за тренинг и валидацију учитава податке из тренинг скупа који се дели на тренинг и валидациони скуп. Док се тест скуп учитава из засебног директоријума. Задати су основни параметри као што је величина једног batcha, величина слике, класе које се учитавају мод за боје коме се каже да су скује црно беле и мод класификације за који се задаје да су циљне променљиве задате као бројеви.

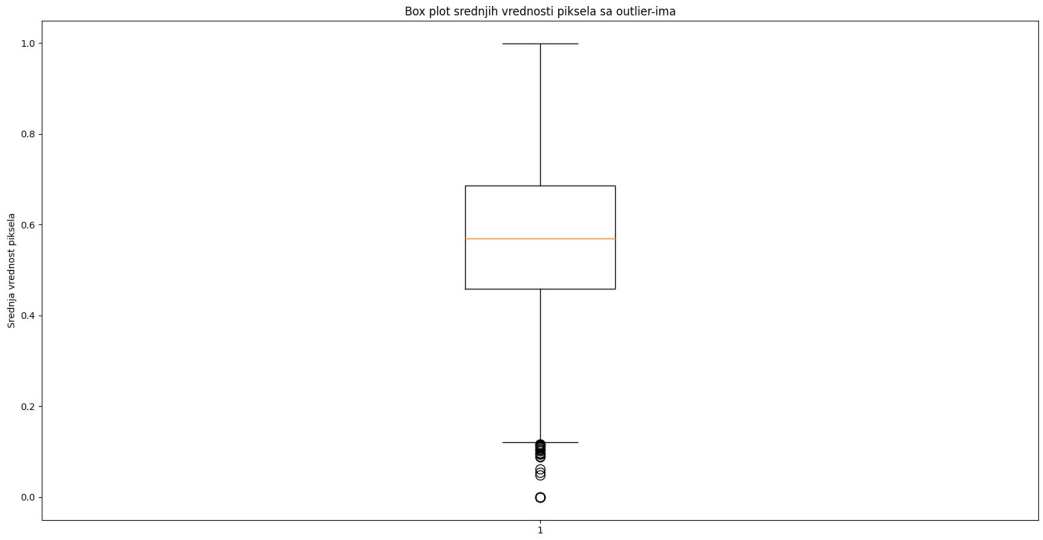



*Слика 8: Приказ 5 слика по класи*

```
# class_mode='sparse' da bi kod ispod radio zbog [klasa = klase[int(label)]]
# vrsimo iteraciju kroz svaki batch dok ne prikazemo slike iz svake klase po 5 puta
while any(broj < slike_po_klasi for broj in prikazane_slike_po_klasi.values()):
    images, labels = next(train_batches) #uzimanje iz slika i labela iz trening skupa
    #iteriranje korz slike
    # iteriranje kroz slike i labele iz batcha
    for img, label in zip(images, labels):
        klasa_idx = np.argmax(label)  # index klase
        klasa = klase[klasa_idx]  # ime klase preko njenog indexa(te klase)
        if prikazane_slike_po_klasi[klasa] < slike_po_klasi:

            red = prikazane_slike_po_klasi[klasa]
            kolona = klase.index(klasa)
            # plotovanje
            axes[red, kolona].imshow(img.reshape(48, 48), cmap='gray')
            axes[red, kolona].axis('off')
            if red == 0:
                axes[red, kolona].set_title(klasa)

            prikazane_slike_po_klasi[klasa] += 1
```

*Слика 9: Код који приказује слике по класи*

```
plt.tight_layout()
plt.show()
```



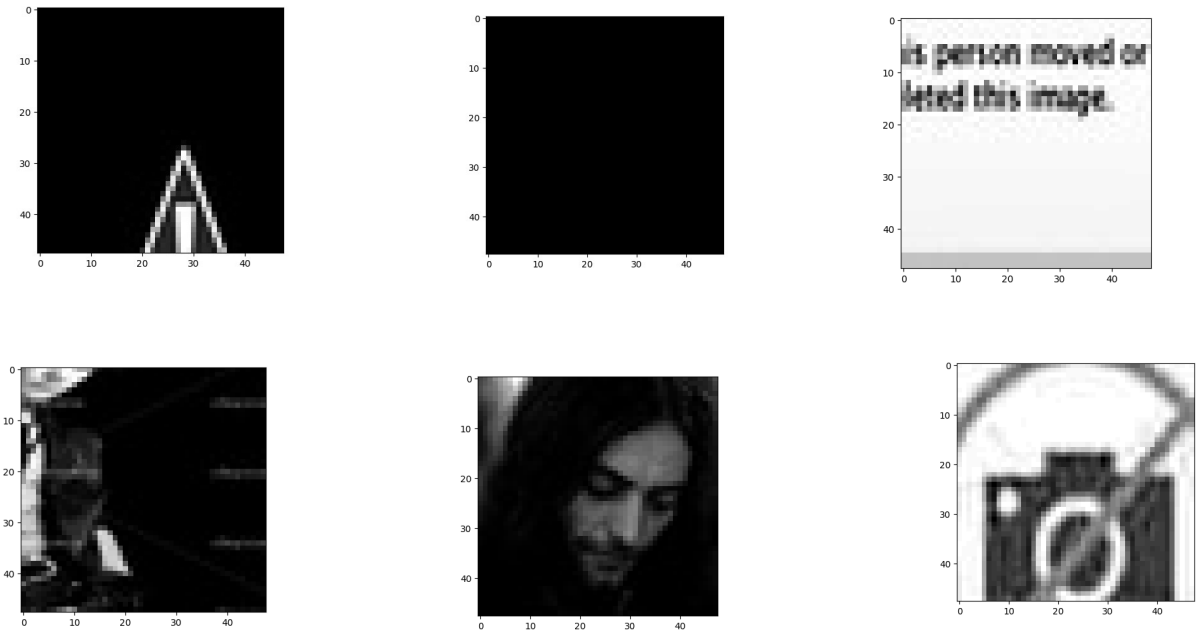

*Слика 10: приказ одударајућих података помоћу Boxplota*

Приказани одударајући подаци приказани који су добијени из Boxplota на раније поменути начин. Код:

```
# boxplot za srednje vrednosti piksela po slici
plot_vrednosti = dict(marker='o', color='olive', markersize=10)
bp=plt.boxplot(sr_vrednost_pixela, flierprops=plot_vrednosti)
outliers = bp['fliers'][0].get_data()[1]
plt.title('Boxplot srednjih vrednosti piksela sa outlierima')
plt.ylabel('srednja vrednost piksela')
plt.show()
```

*Слика 11: код за Boxplot*

Што се овог рада тиче ради чишћења података и уклањања нерелевантних слика користио сам средњу вредност пиксела слике. Свака слика која се није налазила у опсегу од 0.1 до 0.9 после нормализације пиксела сматрана је одударајућим податком и уклоњена је из скупа. Ово су неке од слика које су уклоњене



*Слика 12: неке од избачених слика*

Као што се може видети да нису све слике нерелевантне, али број слика које су заиста релевантне, а избаче су из скупа је занемарљив. Тако да је направљена мала жртва скупа података да би се избациле остале штетне слике.

*Слика 13: код који је омогућио аутоматско уклањање слика*

```python
import os
from keras.preprocessing.image import img_to_array, load_img
import matplotlib.pyplot as plt

# path do direktoriuma
direktorijum_train = 'C:\\Users\\Djole\\Desktop\\slike\\train'
direktorijum_test = 'C:\\Users\\Djole\\Desktop\\slike\\test'

# imena klasa
klase = ['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise']

# uklanjanje slika na osnovu srednje vrednosti pixela slike
def ukloni_slike_na_osnovu_proseka(direktorijum):
    for klasa in klase:
        putanja_klase = os.path.join(direktorijum, klasa)
        for img_name in os.listdir(putanja_klase):
            putanja_slike = os.path.join(putanja_klase, img_name)
            img = img_to_array(load_img(putanja_slike, color_mode='grayscale'))
            img=img/255.
            sr_vr=img.mean()
            if sr_vr <= 0.1 or sr_vr >= 0.9:
                plt.imshow(img.squeeze(), cmap='gray')
                plt.show()
                print(sr_vr)
                os.remove(putanja_slike)

# poziv funkcije za uklanjanje slike nad trening i test skupom
ukloni_slike_na_osnovu_proseka(direktorijum_train)
ukloni_slike_na_osnovu_proseka(direktorijum_test)
```

После уклањања слика број истанци по класи изгледа овако:

| Класа | Бр. истанци после чишћења |
|---|---|
| angry | 3185 |
| disgust | 348 |
| fear | 3274 |
| happy | 5767 |
| neutral | 3965 |
| sad | 3861 |
| surprise | 2532 |

*Табела 1 бр. истанци тренинг после чишћења скупа*

Овако је скуп изгледао пре чишћења:

| Класа | Бр. истанци пре чишћења |
|---|---|
| angry | 3196 |
| disgust | 349 |
| fear | 3278 |
| happy | 5772 |
| neutral | 3972 |
| sad | 3864 |
| **surprise** | **2537** |

*Табела 2 бр. истанци тренинг скупа пре чишћења скупа*

# 4. Класификација

## 4.1 Класификација помоћу конволутивних мрежа

После анализе и препроцесирања података на ред долази и класификација.

Учитавањем из скупа података задато је да је мод класе sparse. Што би значило да су циљне променљиве кодиране као целобројне вредности (0,1,2,3) што је било погодно за препроцесирање. Што се тиче класификације оне су учитане у категоричком моду тј. циљне променљиве су кодиране као вектори дужине броја класа и који имају 1 на индексу коју класу представљају, а на осталим местима имају 0. У наставку су коришћени скупови података без уклањања одударајућих података.

Касније у раду ће те слике бити уклоњене да би се побољшали резултати тренинга.

### Први модел

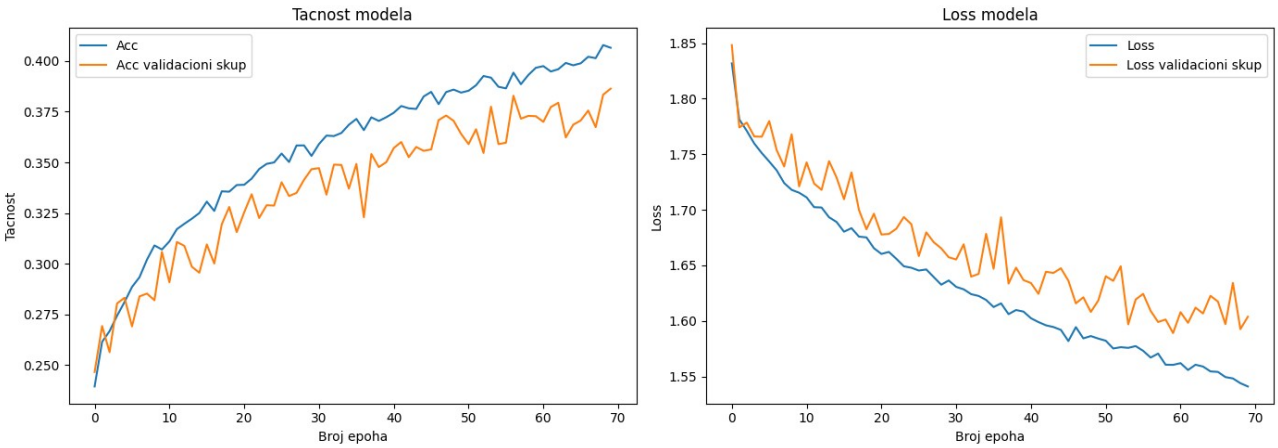
У првом покушају коришћена је конволутивна мрежа са следећом архитектуром. После које долази потпуно повезана мрежа као класификатор.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 48, 48, 32) | 320 |
| conv2d_1 (Conv2D) | (None, 48, 48, 64) | 18,496 |
| batch_normalization (BatchNormalization) | (None, 48, 48, 64) | 256 |
| max_pooling2d (MaxPooling2D) | (None, 24, 24, 64) | 0 |
| dropout (Dropout) | (None, 24, 24, 64) | 0 |
| flatten (Flatten) | (None, 36864) | 0 |
| dense_1 (Dense) | (None, 7) | 258,055 |

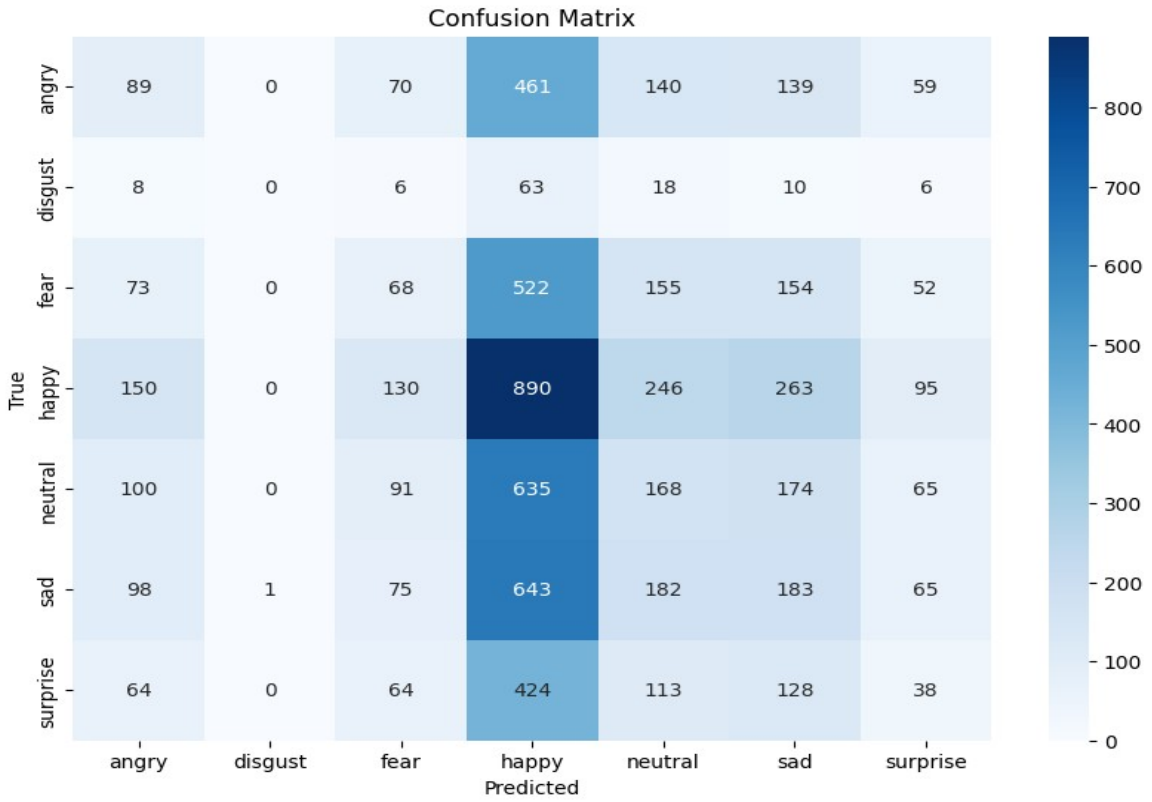
**Total params:** 277,127 (1.06 MB)
**Trainable params:** 276,999 (1.06 MB)
**Non-trainable params:** 128 (512.00 B)

*Слика 14: Архитектура прве мреже*

резултати тренинга мреже су дати у наставку.




*Слика 15: тачност модела и loss модела*




*Слика 16: Матрица конфузије*

| Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|
| 0.2001 | 0.2001 | 0.2001 | 0.2001 |

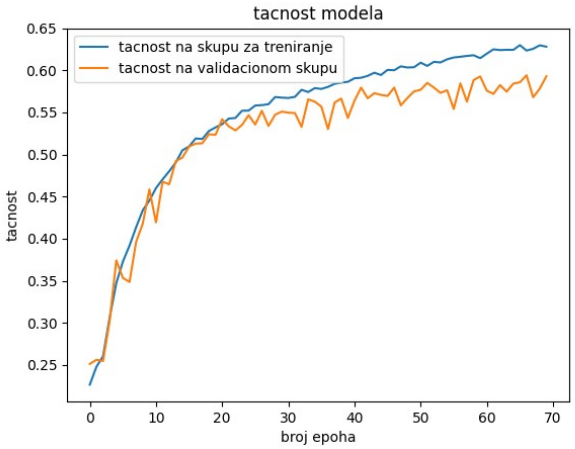
*Табела бр. 3 Модел 1*

Очигледно да ова архитектура мреже није успела да постигне значајну тачност и да смањи губитак. Могуће је да је то резултат плитке архитектуре мреже. То се односи и на конволутивне слојеве и на потпуно повезани слој. За разлику од првог модела за други модел је коришћена доста дубља мрежа.

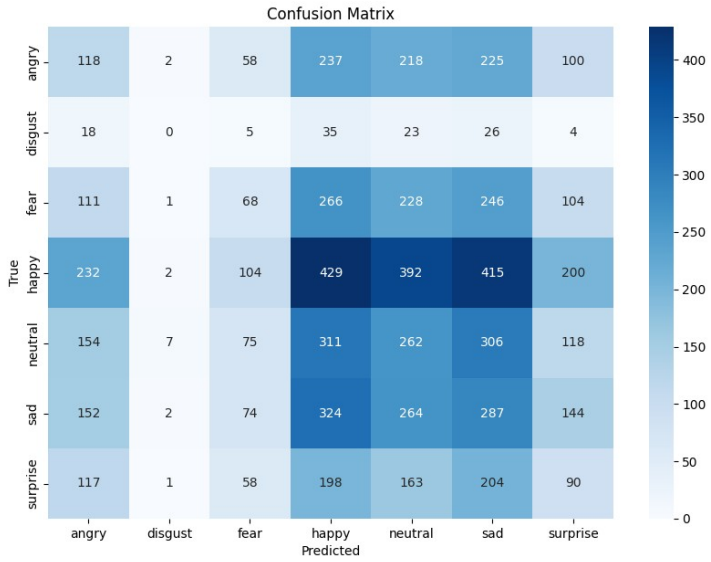
## Други модел

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 96, 96, 32)        320

conv2d_1 (Conv2D)            (None, 96, 96, 64)        18496

batch_normalization (Batch   (None, 96, 96, 64)        256
Normalization)

max_pooling2d (MaxPooling2   (None, 48, 48, 64)        0
D)

dropout (Dropout)            (None, 48, 48, 64)        0

conv2d_2 (Conv2D)            (None, 48, 48, 128)       73856

conv2d_3 (Conv2D)            (None, 48, 48, 256)       295168

batch_normalization_1 (Bat   (None, 48, 48, 256)       1024
chNormalization)

max_pooling2d_1 (MaxPoolin   (None, 24, 24, 256)       0
g2D)

dropout_1 (Dropout)          (None, 24, 24, 256)       0

conv2d_4 (Conv2D)            (None, 24, 24, 512)       1180160

conv2d_5 (Conv2D)            (None, 24, 24, 256)       1179904

batch_normalization_2 (Bat   (None, 24, 24, 256)       1024
chNormalization)

max_pooling2d_2 (MaxPoolin   (None, 12, 12, 256)       0
g2D)

dropout_2 (Dropout)          (None, 12, 12, 256)       0

conv2d_6 (Conv2D)            (None, 12, 12, 128)       295040

flatten (Flatten)            (None, 18432)             0

dense (Dense)                (None, 64)                1179712

dense_1 (Dense)              (None, 128)               8320

dropout_3 (Dropout)          (None, 128)               0

dense_2 (Dense)              (None, 256)               33024

dense_3 (Dense)              (None, 128)               32896

dropout_4 (Dropout)          (None, 128)               0

dense_4 (Dense)              (None, 7)                 903

=================================================================
Total params: 4300103 (16.40 MB)
Trainable params: 4298951 (16.40 MB)
Non-trainable params: 1152 (4.50 KB)
```

Слика 17: Други модел




*Слика 18: Тачност модела*




*Слика 19: Матрица конфузије*

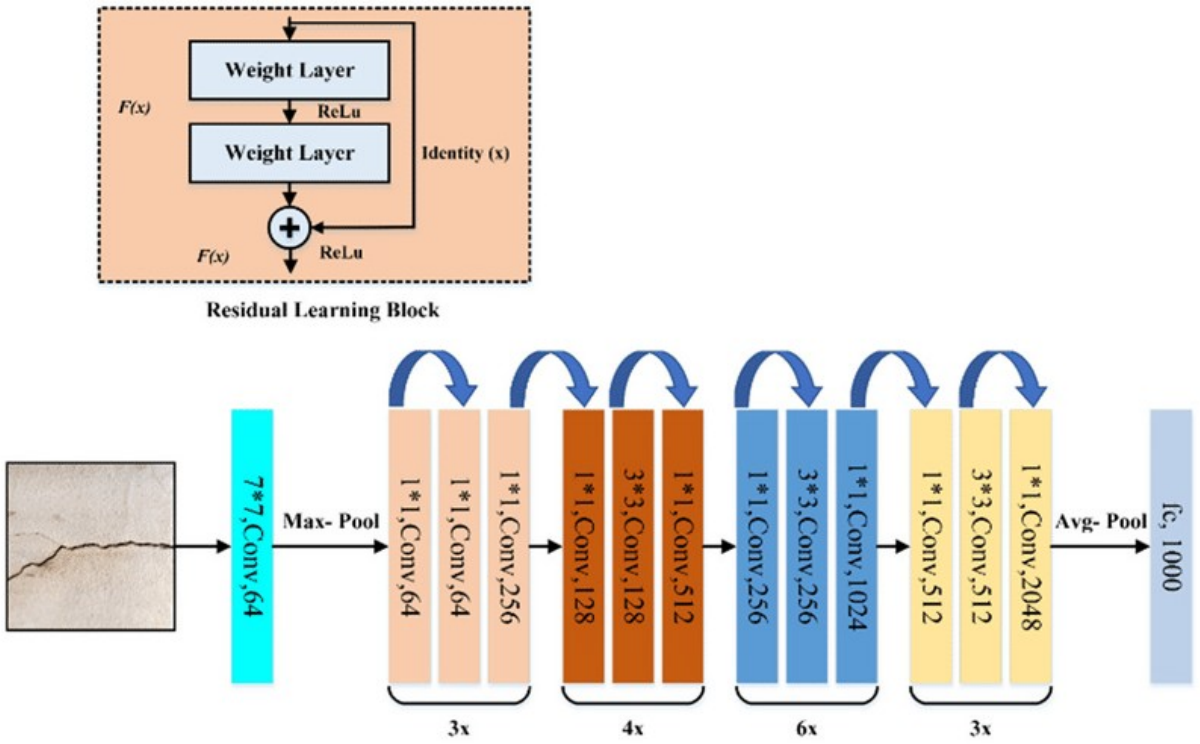
Резултати који су добијени на тестирању:

| Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|
| 0.1747 | 0.1747 | 0.1747 | 0.1747 |

*Табела 4 резултати добијени на тест скупу*

## 4.2 Класификација помоћу предтренираних модела ResNet

Због релативно мале количине података и лошег квалитета скупа могуће је применити неке од предтренираних модела као што су ResNet, MoblieNet, VGG, Inception итд. У пројекту  су коришћене само варијанте ResNet модела као што су ResNet50, ResNet101, ResNet152 итд. Главна предност оваквих модела је што су тренирани на великом скупу података на релативно дубоким мрежама што чини овакве моделе моћним алатом када су скупови података лошег квалитета. Модел је састављен од блокова који се зову резидуални блокови. У мрежи се јављају такозване „*skip*“ конекције које које служе да спрече проблем исчезавајућег градијента. Оне прескачу одређен број слојева  и спајају се са излазом из тих слојева.




*Слика 20: Приказ модела ResNet*

будући да се овај модел не тренира и да је класификатор модела укоњен и на његово место постављен класификатор који је направљен специфично за овај проблем који ће бити трениран. Архитектура класификатора који је додатак ResNet50 моделу је следећи:

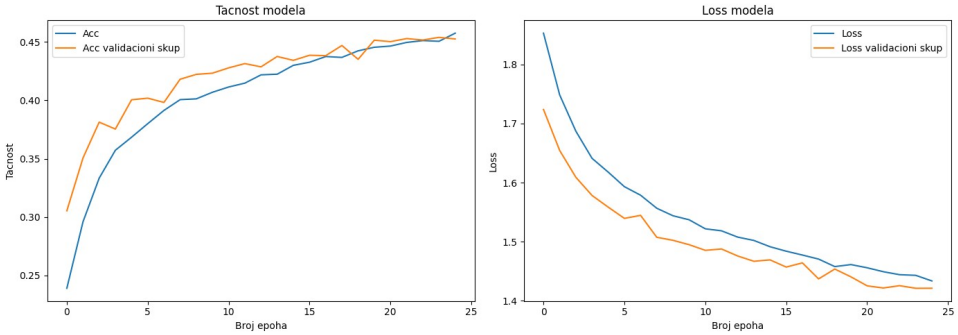
## ResNet50

| Layer (type) | Output Shape | Param # |
|---|---|---|
| resnet50 (Functional) | (None, 7, 7, 2048) | 23,587,712 |
| global_average_pooling2d_1 (GlobalAveragePooling2D) | (None, 2048) | 0 |
| flatten_1 (Flatten) | (None, 2048) | 0 |
| dense_4 (Dense) | (None, 256) | 524,544 |
| dense_5 (Dense) | (None, 256) | 65,792 |
| dropout_2 (Dropout) | (None, 256) | 0 |
| dense_6 (Dense) | (None, 128) | 32,896 |
| dropout_3 (Dropout) | (None, 128) | 0 |
| dense_7 (Dense) | (None, 7) | 903 |

**Total params:** 24,211,847 (92.36 MB)
**Trainable params:** 624,135 (2.38 MB)
**Non-trainable params:** 23,587,712 (89.98 MB)

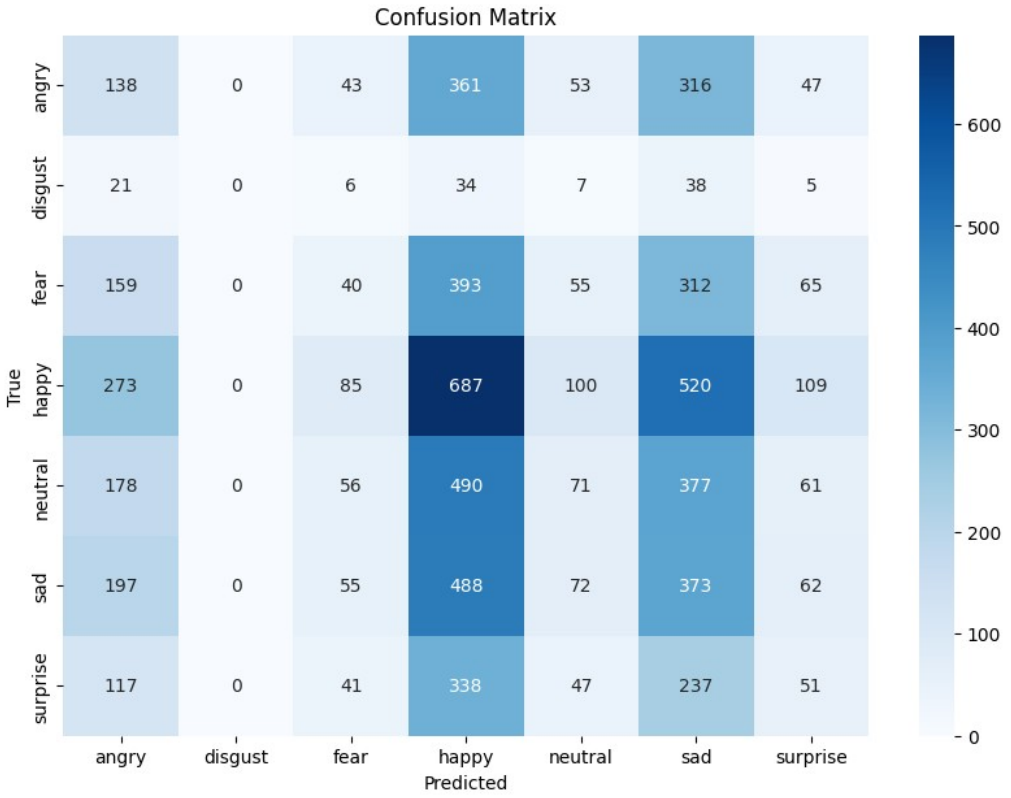
*Слика 21: ResNet50 модел са новим класификатором*

Модел је парцијално трениран у неколико наврата, а резултати су следећи.




*Слика22: Accuray и Loss*

Матрица конфузије




*Слика23: ResNet50 матрица конфузије за тест скуп.*

Остали резултати

| Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|
| 0.1895 | 0.1895 | 0.1895 | 0.1895 |

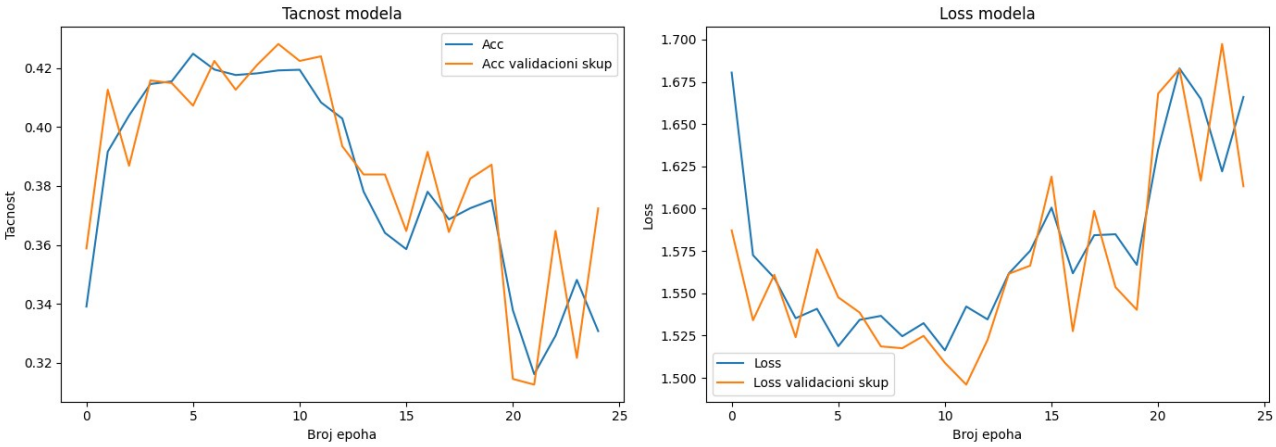
*Табела5: ResNet50 резултат*

## Resnet101

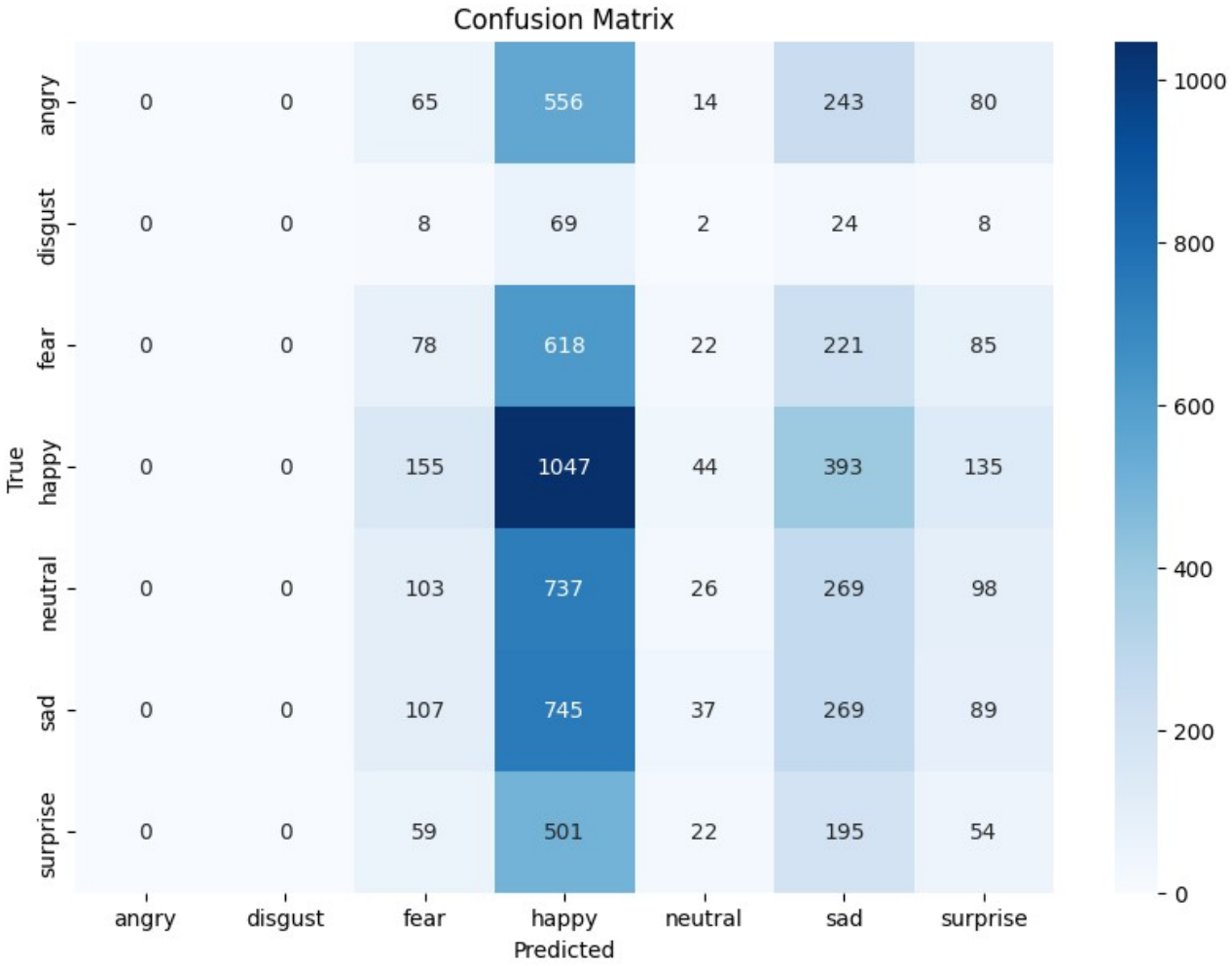
| Layer (type) | Output Shape | Param # |
|---|---|---|
| resnet101 (Functional) | (None, 7, 7, 2048) | 42,658,176 |
| global_average_pooling2d (GlobalAveragePooling2D) | (None, 2048) | 0 |
| flatten (Flatten) | (None, 2048) | 0 |
| dense (Dense) | (None, 64) | 131,136 |
| dense_1 (Dense) | (None, 128) | 8,320 |
| dropout (Dropout) | (None, 128) | 0 |
| dense_2 (Dense) | (None, 256) | 33,024 |
| dense_3 (Dense) | (None, 128) | 32,896 |
| dropout_1 (Dropout) | (None, 128) | 0 |
| dense_4 (Dense) | (None, 7) | 903 |

**Total params:** 42,864,455 (163.51 MB)
**Trainable params:** 206,279 (805.78 KB)
**Non-trainable params:** 42,658,176 (162.73 MB)

*Слика 24: ResNet101 модел*




*Слика 25: Accuracy и Loss*




*Слика 26: Матрица конфузије за тест скуп ResNet101*

Претпоставка да су до резултата овог модела довели велики параметар учења и мали број епоха тренирања.

| Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|
| 0.2053 | 0.2053 | 0.2053 | 0.2053 |

*Табела 6: ResNet101*

## Model ResNet152

За разлику од осталих модела овај модел је трениран на целом тренинг скупу док је тест скуп коришћен као валидациони скуп. Такође тренинг и тест скуп су пречишћени тако да су слике које имају просечну вредност пиксела после нормализације већу од 0.9 или мању од 0.1 уклоњене.

ResNet152 модел трениран 30 епоха, са следећом архитектуром:

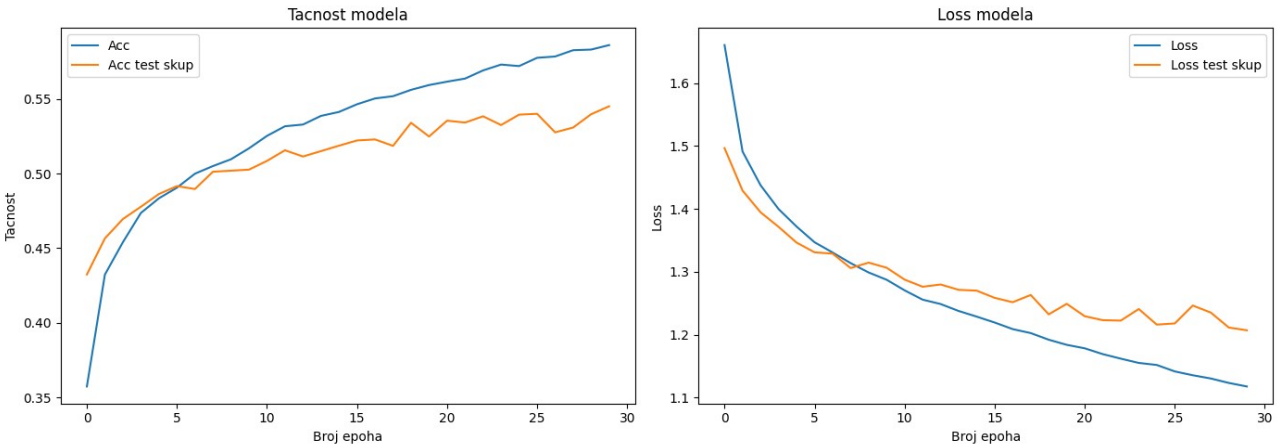
| Layer (type) | Output Shape | Param # |
|---|---|---|
| resnet152 (Functional) | (None, 7, 7, 2048) | 58,370,944 |
| global_average_pooling2d (GlobalAveragePooling2D) | (None, 2048) | 0 |
| flatten (Flatten) | (None, 2048) | 0 |
| dense (Dense) | (None, 1024) | 2,098,176 |
| dense_1 (Dense) | (None, 512) | 524,800 |
| dropout (Dropout) | (None, 512) | 0 |
| dense_2 (Dense) | (None, 7) | 3,591 |

**Total params:** 60,997,511 (232.69 MB)

**Trainable params:** 2,626,567 (10.02 MB)

**Non-trainable params:** 58,370,944 (222.67 MB)

*Слика 27: Модел ResNet152*




*Слика 28: Accuracy и Loss за модел ResNet 152*

# 5. Закључак

На основу добијених резултата, се види да ниједан од модела није постигао задовољавајућу тачност. На основу рада са универзитета Stanford може се видети да је навећа постигнута тачност око 75%. док други модел има тачност нешто мало мању од 60%. Предтренирани модели нису дали побољшање у односу на обичан модел. Могућ разлог је то што је можда требало висе епоха да би се модели истренирали, Односно њихови класификатори. Због времена које је потребно да се такви модели истренирају коришћено је знатно мање епоха него код обичних модела са конволутивним режама.
Закључак би био да и кад би смо имали довољан број епоха не би могли да се постигну савршени резултати због скупа који се користи. Скуп поред слика које су нерелевантне за класификацију, садржи велики број дупликата тј. аугментованих слика. Такође поред тога изрази лица у некој класи изгледају као да више припадају другој класи. Из тих разлога претпоставка је да се скуп морао да се модификује ручно и да осе поново означе класе слика.

# 6. Литература

**[1] Младен Николић, Анђелка зечевић** Машинско учење скрипта

**[2]** Stanford University - CS230 Deep Learning
Facial Expression Recognition with Deep Learning
Improving on the State of the Art and Applying to the Real World