SLOVENSKÁ TECHNICKÁ UNIVERZITA

V BRATISLAVE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

“Strojové učenie a neurónové siete”

Zadanie 5

“Konvolučné siete”

2020/2021

Stanislav Marochok

Zadanie

I-SUNS: Zadanie č.5

KONVOLUČNÉ SIETE

Vo vybranom programovacom jazyku implementujte program, ktorý vedieť rozpoznávať artikle oblečenia.

Čas odovzdania je určený časom vloženia do AIS. Deadline pre získanie 8 bodov je pred vaším cvičením v 12. týždni semestra (**10/11.12.2020**). Keďže sa jedná o posledné zadanie, **nie je možné neskoršie odovzdanie.**

Dáta

K dispozícii máte obrázky, na ktorých je oblečenie a csv súbor, kde sú výstupné kategórie patriace ku obrázkom. Na vypracovanie tohoto zadania si môžete vybrať, ktorá z katégorií bude vašim výstupom (dobrou voľbou sú napr. stĺpce masterCategory, subCategory, gender). Dáta sú uložené na adrese tu.

Úlohy

1. **Načítajte obrázky a pripravte ich na spracovanie**. Nezabudnite ich normalizovať. Lepšie sa vám bude pracovať so štvorcovými obrázkami (výška je rovná šírke). Priraďte podľa csv súboru ku fotografiám kategórie (nevalidné záznamy alebo záznamy s neexistujúcimi obrázkami v csv súbore môžete vynechať). Vytvorte trénovaciu, validačnú a testovaciu množinu. Overte si správne načítanie obrázkov (spätným načítaním a zobrazením, výpismi z debuggera a pod). **2b**

2. **Natrénujte konvolučné siete na klasifikáciu oblečenia**. Konvolučná sieť by mala obsahovať minimálne konvolučné vrstvy, nelinearitu a plne-prepojené vrstvy. Pomôcť nám môže aj pooling vrstva, dropout, batchová normalizácia... V rámci zadania si môžete vybrať, či trénujete sieť od náhodných váh alebo pomocou transfer learning z predtrénovanej siete.

   - Natrénujte model na lepšiu ako náhodnú úspešnosť. **1b**
   - Zobrazte priebeh trénovania (úspešnosť/krit. funkcia od epochy) pre trénovaciu a validačnú množinu. **1b**
   - Nájdite dobrú hodnotu pre parameter rýchlosti učenia. **1b**

- Pokúste sa zlepšiť úspešnosť vašej siete rozumnými zmenami architektúry alebo nastaveniami trénovania. Dosiahnuté výsledky aj so skúmanými nastavenia prehľadne zobrazte v dokumentácii. Ak u vás nastane pretrénovanie, vyriešte ho. **2b**
- Nezabudnite vyhodnocovať a analyzovať vaše siete na testovacej množine (pomôžte si napr. confusion matrix). **1b**

Nepovinné úlohy

- Ukladajte si checkpointy z trénovania, ukážte, že viete modely ukladať a opäť načítavať a používať. **1b**
- Vizualizujte filtre z prvej vrstvy. **2b**
- Analyzujte siete pomocou nástroja Tensorboard. **1b**
- Otestuje vašu siet na množine vašich vlastných obrázkov (takých, ktoré neboli v dodanom datasete). Vyhodnoťte. **2b**

Uvod

Veľmi sa ospravedlňujem za všetky chyby a za ťažko čitatelnu dokumentaciu. Je mi veľmi ľúto, že ja nemôžem napísať lepšie. Ja tomu rozumiem, preto budem sa snažiť písať tak, aby bolo jasné čo som chcel povedať.

Použil som Python, a take knižnice, ako **tensorflow, sklearn**, **numpy**, **matplotlib**, **seaborn** a samozrejme **pandas**.
Celý môj kód, v súbore main.py, je uplne komentovany. Všetky svoje kroky som popisal a zdôvodnil.

Program som spravil tak, že viete veľmi jednoducho nastaviť podľa ktorej kategorie chcete klasifikovať obrazky! Stačí len zmeniť hodnotu premennej v hornej časti kódu:

```
# change this variable to a category you want to classify images
# possible values are "gender", "masterCategory", "usage", "season"
CLASSIFICATION_CATEGORY = "gender"
```

A to je všetko, cely program je flexibilny a nič viac netreba meniť. Ale musite potom vymazať úložene súbory (funkcia remove_saved_data) a pustiť program ešte raz:

```
if __name__ == '__main__':
    # uncomment next line if you want to remove saved data
    # # # # # # remove_saved_data()
    X, y = get_X_y()
    conv_net(X, y)
    test_conv_network_on_image('teniska.png')
```

Načítanie dát a trénovanie sieti bude trvať veľmi dlho... Ak nechcete tak dlho čakať stačí obmedziť počet načítaných obrazkov:

```
# ************************************************************
# This function reads images from DATADIR directory
# ************************************************************
def obtain_images(df):
    training_data = []

    def create_training_data():
        path = DATADIR
        for img in os.listdir(path):
            # uncomment this condition if you want to limit your data
            # if len(training_data) > 1500:
            #     break
            try:
                img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                img_id = os.path.splitext(img)[0]
```

Vážne, som snažil… Keďže mame posledne zadanie tak som ho chcel spraviť čo najlepšie.

Návod na použitie a kratky popis funkcij

Najprv musite zmeniť cestu ku datasetu obrazkov:

```
DATADIR = "d:\Study\Ing\\1 semester\student\I-SUNS\Zadanie 5\\files\data\images"
```

Tieto hodnoty kľudne môžete upraviť ak sa vam chce:

```
IMG_SIZE = 50
CONV_NETWORK_MODEL_SAVED_DIR = "conv_network_model_saved"
CONV_NETWORK_MODEL_NAME = "conv_network_model.h5"
CONV_NETWORK_FIT_HISTORY = "conv_network_git_history"
```

*** IMG_SIZE** - na takuto širku a vyšku bude roztiahnuty každy obrazok z datasetu obrazkov.

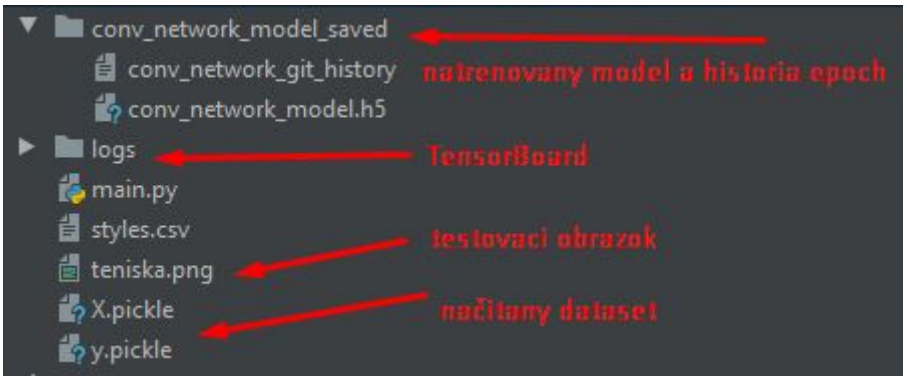
Ak sa vam chce môžete zmeniť hodnotu premennej **CLASSIFICATION_CATEGORY** - podľa tejto hodnoty sa budú klasifikovať obrazky.

```
# change this variable to a category you want to classify images
# possible values are "gender", "masterCategory", "usage", "season"
CLASSIFICATION_CATEGORY = "gender"
```

Program začína v dolnej časti kodu:

```
if __name__ == '__main__':
    # uncomment next line if you want to remove saved data
    # # # # # # remove_saved_data()
    X, y = get_X_y()
    conv_net(X, y)
    # test_conv_network_on_image('teniska.png')
```

Funkcia get_X_y nam vrati datasety X a y (vstupne a vystupne hodnoty). Ak sú potrebne dáta uložene - funkcia vrati tieto data, inak načita všetko z nuly. Uložene data vyzeraju tak:



Ďalej ide funkcia **conv_net**, tam sa robi zakladne testovanie, vyhodnotenie funkcie a vykresľovanie grafov a filtrov.

Ak chcete, odkomentujte funkciu **test_conv_network_on_image** a otestujte natrenovanu sieť na vašom osobnom obrazku. Funkcia funguje iba pre 1 obrázok za raz.

Ak chcete vymazať uložené dáta a načítať ich znovu, odkomentujte funkciu **remove_saved_data**, mam ju 5-krát zakomentovanu, aby som nahodou na vymazal dáta ak nechcem.

Funkcie majú isť presne v takom poradí v akom sú v skripte napisane.

Priprava dát

Za prve, priečinok s obrázkami oblečení som nechal mimo priečinku s projektom, spravil som to tak kvôli tomu, že nakoľko obrazkov je veľmi veľa indexácia súborov projektu v **PyCharm** prebiehala moc dlho, a ja som nechcel obťažovať projekt takým veľkým počtom súborov.

V riadku 6044 súboru styles.csv sa objavil nejaký zvyšný stlpec, preto mal som definovať natvrdo ktore stĺpce chcem načítať z tohto csv.

Chcel by som zistiť všetky unikátne hodnoty, ktoré sú v stĺpcoch tohto csv. Snáď toto mi pomôže potom.

Zistil som toto:

```
        gender    5  ['Men', 'Women', 'Boys', 'Girls', 'Unisex']
masterCategory    7  ['Apparel', 'Accessories', 'Footwear', 'Personal Care', 'Free Items', 'Sporting Goods', 'Home']
        season    4  ['Fall', 'Summer', 'Winter', 'Spring']
         usage    8  ['Casual', 'Ethnic', 'Formal', 'Sports', 'Smart Casual', 'Travel', 'Party', 'Home']
```

Teraz viem, koľko neurónov by mala obsahovať vystupna vrstva mojej sieti. Keď chcem klasifikovať obrazky podľa gender - tak dam 5 neurónov, **masterCategory** - 7 neuronov, season - 4, usage - 8. Ale v podstate mne to je jedno, lebo spravil som program tak že počet neurónov v poslednej vrstve sa meni v závislosti od počtu hodnôt výstupnej kategorie.

V tomto bode mal by som si zvoliť, čo budem používať ako cieľovú hodnotu. Myslim podľa čoho budem triediť oblečenie, podľa gender, alebo **masterCategory**, season, usage. Mám taký pocit, že najviac dávajú zmysel stlpce gender a **masterCategory**, lebo nie je tam až tak veľa možnosti, a oni obsahuju asi kľúčové rozdiely medzi oblečení.

Skusim natrenovať sieť s výstupom **gender**. Keďže obrázky v datasete majú rôznu šírku a výšku, každy obrazok som roztiahol na veľkosť **50x50** (**IMG_SIZE** = 50).

Načítanie obrazkov mi trvalo cca 7 minút, preto hneď po načítaní ja som uložil načítané pole obrázkov do súboru X.pickle. Potom načítanie z tohto súboru trvá iba 1 sekundy.

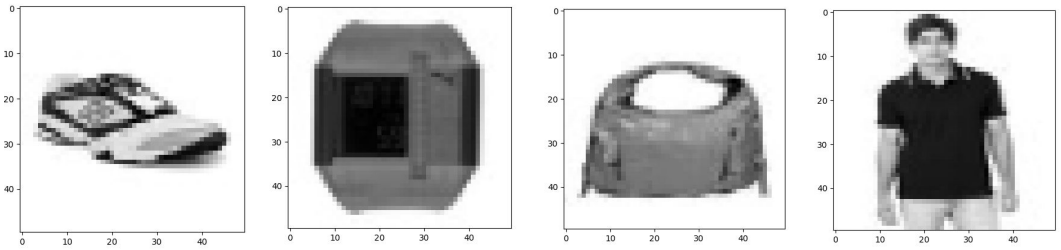
```
Could not load library cudnn_ops_infer64_8.dll. Error code 126
Please make sure cudnn_ops_infer64_8.dll is in your library path!

d:\Study\Ing\1 semester\student\I-SUNS\Zadanie 5\new_code_2>
```

Dostal som takuto chybu, dva dni som stratil na to aby ju vyriešiť. A hlavne keď som to testoval v PyCharme tak žiadnu chybu mi to nevypisovalo, program iba spadol a hotovo. Až keď som to rozbehal v prikazovom riadku tak ono mi aspoň nejakú chybu vypisalo.

Aby vyriešiť tento problém mal som reinštalovať **CUDA** a **cuDNN**. Dva dni som na to stratil…

Overenie načítania obrázkov

Aby sa presvedčiť, že som načital obrazky v spravnom formate, vykreslim na obrazovku nahodnz obrazok z poľa načitanych obrazkov. Na tento účel som vytvoril funkciu **show_image_from_array**, do ktorej pošlem prvok poľa načítanie obrazkov.



```python
# ************************************************************
# This function shows an image from array of obtained images
# This is just a check to control that we have written images to array
# in a correct form
# ************************************************************
def show_image_from_array(image):
    plt.imshow(image, cmap="gray")
    plt.show()
```

Trénovanie konvolučnej sieti

Hľadanie lepších parametrov

Keďže toto je dosť komplikovaná sieť a je tam veľa informacie a veľa faktorov, bohužial nemáme veľa priestoru na experimenty s hladanim najlepších parametrov, a musím už hneď setovať parametre, ktore budu dobre (respektive spraviť čo najmenej pokusov na hľadanie najlepších parametrov). Podľa mojich experimentov tieto parametre mi prišli ako dosť dobre, a používam ich pri trénovaní:

```
        batch_size=64,
        validation_split=0.2,
        epochs=10,
        verbose=1)
```

Spravil som taky model:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 48, 48, 256)       2560
_________________________________________________________________
activation (Activation)      (None, 48, 48, 256)       0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 22, 22, 256)       590080
_________________________________________________________________
activation_1 (Activation)    (None, 22, 22, 256)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 11, 11, 256)       0
_________________________________________________________________
flatten (Flatten)            (None, 30976)             0
_________________________________________________________________
dense (Dense)                (None, 64)                1982528
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 325
_________________________________________________________________
activation_2 (Activation)    (None, 5)                 0
=================================================================
Total params: 2,575,493
Trainable params: 2,575,493
Non-trainable params: 0
_________________________________________________________________
```

Je úžasný že? Ale ma nejak veľa trenovacich parametrov. Nevadi.V poslednej vrstve nastavil som taky počet neuronov, kolko mám unikatnych výstupných hodnôt. Popisoval som to hore. Keďže som sa rozhodol klasifikovať obrazky podľa parametru gender (gender má 5 unikatnych hodnôt), vystupna vrstva má 5 neuronov.

V poslednej vrstve aktivačná funkcia je **softmax**, to je dôležite lebo najprv som tam dal sigmoid, a kvôli tomu sieť prestala sa trénovať až v 2 epoche. Ale teraz je to v poriadku.

Po skončení trenovania bude vytvorený podadresar **conv_network_model_saved**, do ktoreho budu uložené súbory **conv_network_model.h5** (natrenovany model) a **conv_network_git_history** (historia trenovania tohto modelu).

Nastavil som 10 epoch trénovania, a pri načítaní obrazkov som ohraničil ich počet do 6000, preto že keď mám 44 000 obrazkov tak mi neuronova siet sa trénuje no veľmi veľmi dlho, ja

tolko čakať neviem, preto dam tam menej obrazkov ale aspoň si budem istý, že neuronka funguje.

Spravim predpovede pomocou funkcie **predict**(), potom použiem **argmax** na konvertaciu predpovedanych hodnôt na 0-1 (lebo oni sú v tvare desatinných čísel ako napriklad 0.1231521). A teraz už viem vypisať classification_report a confusion_matrix.

```
classification_report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
           1       0.00      0.00      0.00         7
           2       0.85      0.97      0.91      1312
           3       0.71      0.25      0.36       102
           4       0.89      0.71      0.79       548

    accuracy                           0.85      1981
   macro avg       0.49      0.39      0.41      1981
weighted avg       0.84      0.85      0.84      1981
```

Z tohto reportu je vidno že sieť spravne predpovedala hodnoty 2, 3 a 4, a vôbec nepredpovedala hodnoty 1 a 2. Podľa support vidím, že zaznamov kategorie 0 a 1 bolo iba 12 a 7 v testovacom datasete, čo je veľmi málo. Ale zase, načítal som iba 6000 obrazkov, inak moja sieť by učila sa moc dlho. A keď pozrieme na ďalšie kategórie - 2, 3, 4 - tak tam máme dosť načitanych hodnôt, a precision je 85%, 71% a 89% zodpovedne, čo je podľa mňa dosť dobrý výsledok (vzhľadom na to že doteraz mi žiadna sieť vôbec nefungovala).

```
confusion_matrix:
 [[   0    0   10    1    1]
 [   0    0    5    0    2]
 [   1    0 1278    5   28]
 [   0    0   58   25   19]
 [   0    0  157    4  387]]
```

Confusion matrix nam hovori o tom, že sieť spravne predpovedala 1278 prvkov kategorie 2, a spravila 10 + 5 + 58 + 157 = 230 chyb, takže sieť spravne predpovedala 25 prvkov kategorie 3 a spravila 1 + 5 + 4 = 10 chyby, a spravne predpovedala 387 prvkov kategorie 4 a spravila 1 + 2 + 28 + 19 = 50 chyb.

Som si istý že keby som mal viac času a lepši počitač, tak by som natrenoval sieť na ovela lepšie vysledky.

Pribeh trenovania vyzera tak nejak:

```
Epoch 1/10
75/75 [==============================] - 81s 1s/step - loss: 4.7927 - accuracy: 0.6542 - val_loss: 0.7937 - val_accuracy: 0.6844
Epoch 2/10
75/75 [==============================] - 76s 1s/step - loss: 0.7133 - accuracy: 0.7227 - val_loss: 0.5889 - val_accuracy: 0.7627
Epoch 3/10
75/75 [==============================] - 80s 1s/step - loss: 0.5662 - accuracy: 0.7710 - val_loss: 0.5424 - val_accuracy: 0.7777
Epoch 4/10
75/75 [==============================] - 83s 1s/step - loss: 0.5370 - accuracy: 0.7777 - val_loss: 0.4993 - val_accuracy: 0.8027
Epoch 5/10
75/75 [==============================] - 77s 1s/step - loss: 0.5032 - accuracy: 0.7967 - val_loss: 0.5201 - val_accuracy: 0.7835
Epoch 6/10
75/75 [==============================] - 77s 1s/step - loss: 0.4793 - accuracy: 0.8079 - val_loss: 0.4776 - val_accuracy: 0.8010
Epoch 7/10
75/75 [==============================] - 79s 1s/step - loss: 0.4668 - accuracy: 0.8119 - val_loss: 0.4583 - val_accuracy: 0.8135
Epoch 8/10
75/75 [==============================] - 87s 1s/step - loss: 0.4506 - accuracy: 0.8254 - val_loss: 0.4512 - val_accuracy: 0.8168
Epoch 9/10
75/75 [==============================] - 80s 1s/step - loss: 0.4419 - accuracy: 0.8310 - val_loss: 0.4280 - val_accuracy: 0.8301
Epoch 10/10
75/75 [==============================] - 77s 1s/step - loss: 0.4422 - accuracy: 0.8252 - val_loss: 0.4480 - val_accuracy: 0.8318
```

```
Test loss: 0.412958025932312
Test accuracy: 0.8399798274040222
```

Z tychto grafov historie trenovanie sieti je vidno, že hodnota training and validation accuracy čas od času klesá, a potom znovu rastie. Predpokladám že to je kvôli neidealnej štruktúre sieti. Skusim zmeniť štruktúru tak, aby úspešnosť sa zlepšila.

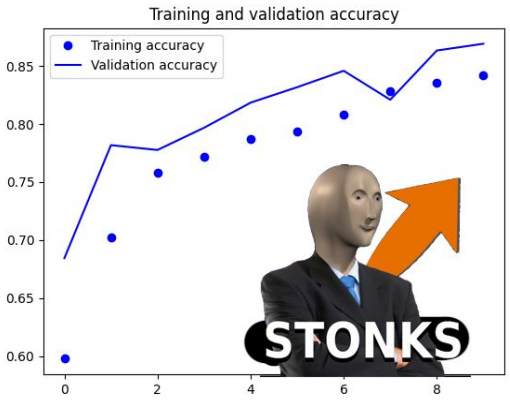
Zmena štruktury

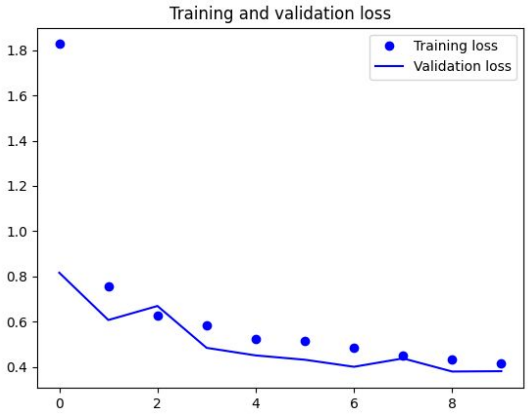
Zmenil som štrukturu sieti podľa tohto návodu https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python (mimochodom toto je veľmi zaujjimavy článok). Pridal som Dropout (lebo vždy treba pridavať Dropout) a LeakyRelu.

Teraz štruktúra mojej sieti vyzerá tak:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 50, 50, 32)        320
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 50, 50, 32)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 25, 25, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 25, 25, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 25, 25, 64)        18496
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 25, 25, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 128)       73856
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 13, 13, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 128)               802944
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 896,261
Trainable params: 896,261
Non-trainable params: 0
```

Mám oveľa menej trenovacich parametrov, a trénovanie mi prebieha oveľa rýchlejšie. Získal som aj lepšiu úspešnosť:



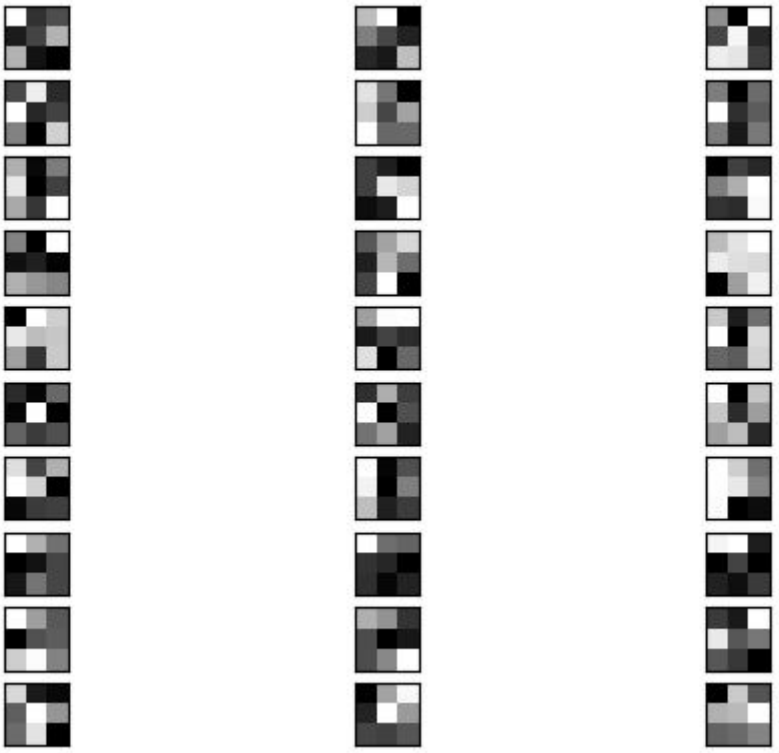



```
Test loss: 0.3700752854347229
Test accuracy: 0.8879354000091553
classification_report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
           1       0.00      0.00      0.00         7
           2       0.88      0.98      0.93      1312
           3       0.78      0.31      0.45       102
           4       0.91      0.81      0.86       548

    accuracy                           0.89      1981
   macro avg       0.51      0.42      0.45      1981
weighted avg       0.88      0.89      0.87      1981

confusion_matrix:
 [[   0    0   12    0    0]
 [   0    0    6    1    0]
 [   0    0 1285    3   24]
 [   0    0   51   32   19]
 [   0    0  101    5  442]]
```

Máme lepšie výsledky, kategória 2 už ma o 3 percent lepšiu úspešnosti, kategória 3 o 7 percent lepšiu uspešnosť a kategória 4 sa zlepšila o 2 percenta.

Vizualizacia filtrov

Keďže mám takú štruktúru modelu, a chcem zobraziť filtre 1 vrstvy (filtre sú v podstate vahy), tak musim vybrať vahy 4 vrstvy (lebo až na 4 vrstve sa mi začina kvazi 2 vrstva, v skutočnosti v modele keď pozriem vrstvy 2 a 3, tak oni nemaju žiadne vahy - jasne nemaju veď oni nie sú vrstvy...). Prve 6 filtrov, rozdelene na 3 kanaly, vyzerajú tak:



Ukladanie dát

Keďže máme tak veľa parametrov, a trénovanie sieti nam trva veľmi veľa času, musime ukladať dáta, aby nenačítavať ich od začiatku každý raz.

Ja som to spravil tak, že ukladám pole načitanych obrazkov (vstupne hodnoty trénovania sieti) do súboru “**X.pickle**”, ukladam aj vystupne hodnoty (zodpovedna každému obrázku hodnotu kategorie klasifikacie) do súboru “**y.pickle**”, to robim pomocou knižnici pickle.

Okrem toho ukladam aj model natrenovanej konvolučnej sieti do súboru “**conv_network_model_saved\conv_network_model.h5**” a historiu trenovania tejto sieti ukladam do súboru “**conv_network_model_saved\conv_network_git_history**”. Uložena historia mi umožnuje experimentovať s grafmi bez toho aby som pušťal trening sieti každy raz nanovo.

Na ukladanie používam tieto funkcie:

```python
# ********************************************************************************
# This function saves datasets X and y to files X.pickle and y.pickle
# With the help of library pickle
# ********************************************************************************
def save_dataset(X, y):
    pickle_out = open("X.pickle", "wb")
    pickle.dump(X, pickle_out)
    pickle_out.close()

    pickle_out = open("y.pickle", "wb")
    pickle.dump(y, pickle_out)
    pickle_out.close()


# ********************************************************************************
# This function saves datasets X and y to files X.pickle and y.pickle
# With the help of library pickle
# ********************************************************************************
def save_model(model, history):
    # create sub  folder if not exists
    if not os.path.exists(CONV_NETWORK_MODEL_SAVED_DIR):
        os.makedirs(CONV_NETWORK_MODEL_SAVED_DIR)

    # save model
    model.save(os.path.join(CONV_NETWORK_MODEL_SAVED_DIR, CONV_NETWORK_MODEL_NAME))

    # save fit history as json dictionary
    with open(os.path.join(CONV_NETWORK_MODEL_SAVED_DIR, CONV_NETWORK_FIT_HISTORY), mode='w') as f:
        json.dump(history, f)
```

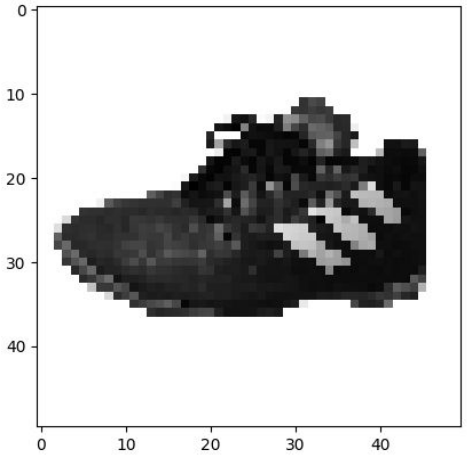
A vďaka tomu že ja načitavam už pripravené data, to načítanie trva maximalne par sekund, čo mi veľmi uľahčuje život.

Testovanie sieti na svojich datach

Aby otestovať sieť na svojich datach vytvoril som funkciu **test_conv_network_on_image**, do ktorej pošlem cestu k obrázku. Ak obrázok neexistuje - vypišem chybu a zatvorim funkciu.

Pripravil som fotku svojej tenisky a trošku som ju opravil v Photoshop:

Načitam obrazok presne takym istym spôsobom ako som načitaval obrazky z datasetu. A pre istotu vykreslim tento obrazok v takom tvare, ako to berie neuronka:



Ďalej sa začína magia. Ja zoberiem model natrenovanej neuronovej sieti pomocou funkcii "**get_convolutional_network_model**", a spravim predpovede pomocou **predict()**. Ale ja nechcem zobraziť iba hodnotu, ktorú mi ukáže neuronka - ja chcem ju transformovať na čitateľnú hodnotu, ako to bolo v datasete styles.csv.

Na tento účel ja načitam celý dataset ešte raz pomocou **get_dataframe(False)** - False tam davam aby hodnoty dataframu sa nekonvertovali na čísla pomocou **LabelEncoder()**, lebo ja ich chcem konvertovať sam a uložiť si objekt LabelEncoder-u, ktory spravil to kódovanie. Preto že keď mam objekt LabelEncoder(), ktory je natrenovany, tak ho môžem použiť na to, aby konvertovať čiselne hodnoty späť na textove. A tym pádom dostanem taky vypis:

```
  0 Boys        [1. 0. 0. 0. 0.]
  1 Girls       [0. 1. 0. 0. 0.]
  2 Men         [0. 0. 1. 0. 0.]
  3 Unisex      [0. 0. 0. 1. 0.]
  4 Women       [0. 0. 0. 0. 1.]
Predicted category of image teniska.png for category gender is Men
```

To znamená že sieť vrátila hodnotu v tvare [0, 0, 1, 0, 0], čo som pomocou argmax konvertoval na hodnotu 2, čo som pomocou natrenovaneho na gender **LabelEncoderu** konvertoval na hodnotu Men, a toto je naozaj panska teniska! Bože jaku šikovnu mam neuronovu sieť… Teším sa :)

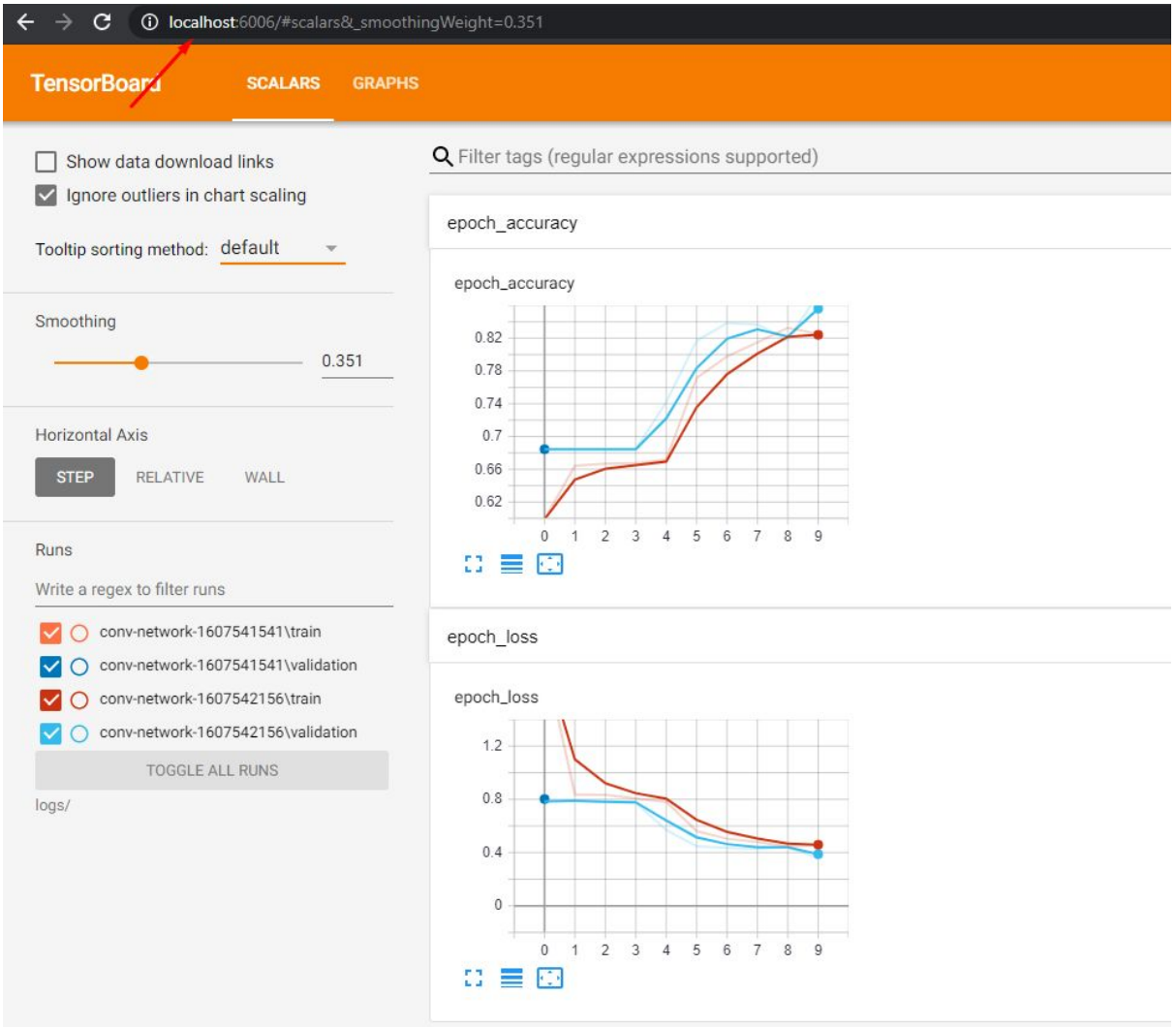
Tensorboard

Pridal som do funkcie fit() pri trénovaní sieti, parameter **callbacks=[TensorBoard(log_dir='logs/{}'.format("conv-network-{}".format(int(time.time()))))]**, ktory mi bude logovať pribeh trénovania.

Keď trenovanie sa skončilo, spustim TensorBoard takto:

```
D:\Study\Ing\1 semester\student\I-SUNS\Zadanie 5\new_code_2>python -m tensorboard.main --logdir=logs/
2020-12-09 20:27:21.093226: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic
library cudart64_101.dll
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.2.2 at http://localhost:6006/ (Press CTRL+C to quit)
```

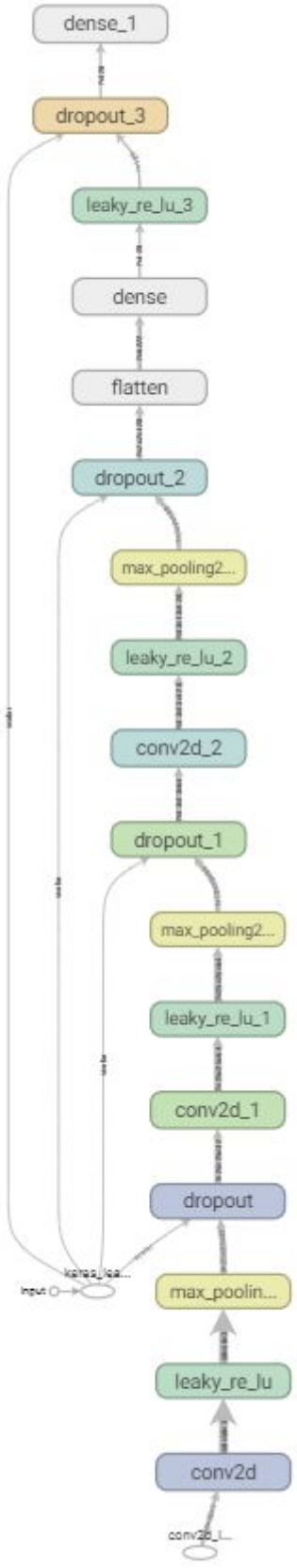
Na stranke localhost:6006 tym padom sa mi naštartoval server, a vidim tam nejake grafy:



To sú tie isté grafy čo som vykresloval vo funkcii **conv_net**, len vyzeraju inak. Tu sa zobrazuje pribeh trenovania sieti, a vykresľuje sa graf úspešnosti trenovacej množiny oproti validačnej. V jednom grafe sú dáta **accuracy**, v druhom **loss**.

Keď pozriem čo všetko na tejto webovej aplikácie mám, tak zistím že sú tu take pekne veci, ako Main Graph:

Main Graph


dense_1
dropout_3
leaky_re_lu_3
dense
flatten
dropout_2
max_pooling2...
leaky_re_lu_2
conv2d_2
dropout_1
max_pooling2...
leaky_re_lu_1
conv2d_1
dropout
max_poolin...
leaky_re_lu
conv2d

Flexibilita programu

Vyhoda mojho kodu spočíva v tom, že tomu, kto bude tento kod používať naozaj stačí zmeniť hodnotu len jednej jedinej premennej na začiatku kodu (pisal som o tom v úvode), aby nastaviť klasifikaciu obrazkov podľa inej kategorie. Samozrejme bude treba vymazať uložené dáta, tym padom program začne vytvárať nové dáta a trénovať siet s inou kategóriou výstupných hodnôt.

Okrem toho kod je napisany veľmi pekne a cela funkcionalita je rozdelená do príslušných funkcij, a každá funkcia je odkomentovana.

Suhrn

Spravil som konvolučnu neuronovu sieť, ktorá vie rozpoznať obrazky a klasifikovať ich podľa rôznych kategórii. Naučil som sa robiť s obrázkami, naučil som sa riešiť technické problémy ktore sa tykaju TensorFlow, naučil som sa robiť s TensorBoard, a nakoniec napisal som dobrý flexibilny a lahko kontrolovany kód.