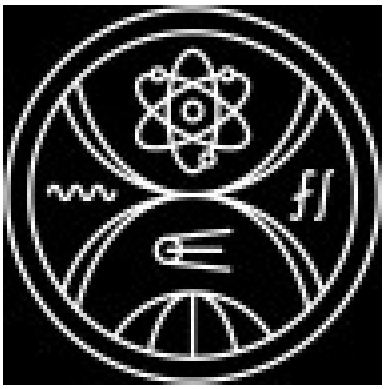
Report



Projekt

Vytvorenie modelu
ktorý dokáže rozoznávať recyklačné symboly pre plasty

Študent: Damián Gorčák

Predmet: Strojové učenie

Január 2023

Contents

1. Úvod

Cieľom tejto práce bolo vytvoriť model , ktorý by dokázal rozoznať jednotlivé symboly (čísla) označujúce typ plastu ,z akého je daný obal vytvorený.

2. Typy obalov

V tejto sekcií by som rád v krátkosti opísal ako vyzerajú jednotlivé symboly. Pre plasty existuje celkovo 7 symbolov a to sú:

Symbol 1: PETG alebo PETE
tento typ plastu sa často používa pre výrobu plastových fľiaš a môže sa reczklovať a vytvárať nové fľaše, koberce alebo nábytok.
Symbol 2: HDPE
Fľaše pre mlieko, olej. Dá sa ľahko reczklovať pre výrobu pier, obalov pre olej .. .
Symbol 3: PVC alebo Vinyl
Obaly na ovocie, bublikové folie. Veľmi zriedkavo sa recykluje
Symbol 4: LDPE
nakupovacie tašky atd.
Symbol 5: PP
plastové hračky, nábytok, autá. Dá sa recyklovať
Symbol 6: Styrén alebo PS
Polystyrén. Dá sa recyklovať pre napríklad výrobu obalov na vajcia.
Symbol 7: Ostatné

Figure 1: Všetky symboly označujúce plasty

3. Použité Technológie

3.1 Konvolučné neurónové siete (CNN)

CNN je hlboká neurónová sieť ktorá je určená pre spracovanie štruktúrovaných polí dát ako obrázky. Najviac sú používané pri klasifikácií obrázkov. Slovo konvolúcia je v CNN poukazuje na matematickú funkciu konvolúcie. Jednoducho povedané 2 obrázky reprezentované ako matice sú spolu vynásobené a výsledok je použitý na extrakciu vlastností. Konvolučná sieť sa skladá z týchto vrstiev:

Konvolučná vrstva
Je to prvá vrstva a používa sa na extrakciu vlastností z vstupných obrázkov. Extrakcia sa deje pomocou konvolúcie, ktorá funguje ako prechádzanie filtra obrázkom. Majme obrázok vo forme matice a filter, taktiež vo forme nejakej matice výsledok je potom skalárny súčin týchto dvoch matíc. Výsledok udáva informácie o vlastnostiach obrázka ako rohy, hrany... . Celkový výsledok sa nazýva mapa vlastností (feature map). Tieto výsledky sa používajú v dalších vrstvách.

poolingová vrstva
Zvyčajne nasleduje za CNN. Jej hlavným cieľom je zredukovať veľkosť mapy vlastností (feature map). Pre zredukovanie matematických operácii teda času aj ceny výpočtu. Zjednodušene sumarizuje jednotlivé vlastností. Existuje tzv max pooling a average pooling. Max pooling berie najväčšie prvky z matice vlasností, kdežto average pooling berie priemerné hodnoty. Zvyčajne je táto vrstva ako priechodník medzi CNN a plne prepojenov vrstvou.

Plne prepojená vrstva
Pozostáva z váh a zaujatostí (bias) s neuronmi ktoré spájajú dve vrstvy. Táto vrstva sa zvyčajne ukladá pred poslednú vrstvu. Význam tejto vrstvy spočíva v tom, že práve v týchto vrstvách prebieha klasifikácia. Všetky vrstvy sú pospájané preto, pretože majú lepšie výsledky ako iba samotne pospájané vrstvy.

Dropout
Pri plne prepojených vrstvách často dochádza k overfitingu (model funguje dobre na trénovacích dátach ale nie dobre na validačných). Dropout napomáha sa vynúť tomuto problému tak, že vynecháva niektoré z neurónov. Tento prístup nielenže napomáha k prevencií voči overfittingu ale aj znižuje výkon modelu (menej neurónov).

Aktivačné funkcie
Jedna z najdôležitejších častí. Aktivačné funkcie sa Používajú pre učenie sa vzťahov medzi premennými v sieti. Jednoducho povedané rozhoduje či danú informáciu posunie dopredu alebo ktorú opačne posunie dozadu. Existuje viacero takýchto funkcií medzi najviac používané patria ReLu, Softmax alebo tanH.

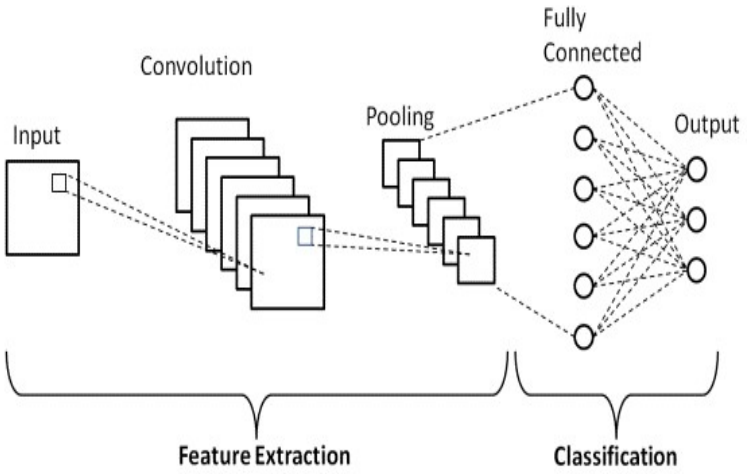


Figure 2: Ukážka poskladania vrstiev

3.2 Transfer Learning

Je technika, ktorá využíva vedomosti už iného vytrénovaného modelu, a aplikuje ich na iný, ale v nejakom zmysle podobný problém. Vzužíva sa keď chceme vytrénovať model, ale nemáme dostatočné množstvo dát. Vtedy využije iný model trénovaný na veľkom mnnožstve dát a náš model na ňom dotrénujeme.

3.3 Zväčšenie dát

Používa sa pri nedostatku trénovacích dát. Funguje tak, že umelo vytvoríme dáta z už existujúcich dát. Pri existujúcich dátach sa robia malé zmeny. Napr. pri obrázkoch sa dáta zvačšujú, otáčajú, posúvajú... Táto technika dokáže vysokou mierov ovplivniť výsledný model.

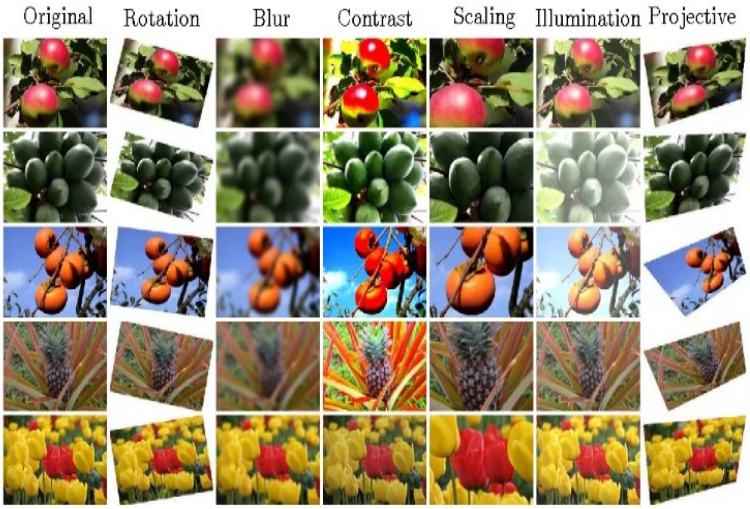


Figure 3: Ukážka umelých dát

4. Dátová sada

Dátová sada ktorú som použil na trénovanie modelu sa skladá z 2 datasetov. Prvý z nich je použitý zo stránky kaggle [1]. Obsahuje niečo cez 600 dátových bodov obsahujúcich znaky označujúce druh plastov. Dáta sú rozdelené do priečinkov, pričov v každom priečinku sú obrázky znakov označujúce čislo, ktoré je v názve zložky. Priečinkov je teda dokopy 7.

Druhá dátová sada bola použitá Bolońskej univerzity[2]. Dáta sú uložené v 2 zložkách s názvami Positive a Negative. Tieto zložky oddelujú obrázky ktoré obsahujú znak a ktoré nie. V zložke positive sa príznak čísla znaku získa z názvu. Tento dataset obsahuje okrem znakov pre plasty aj znaky označujúve iné materialy napr. papier - tie samozrejme bolo treba odfiltrovať.

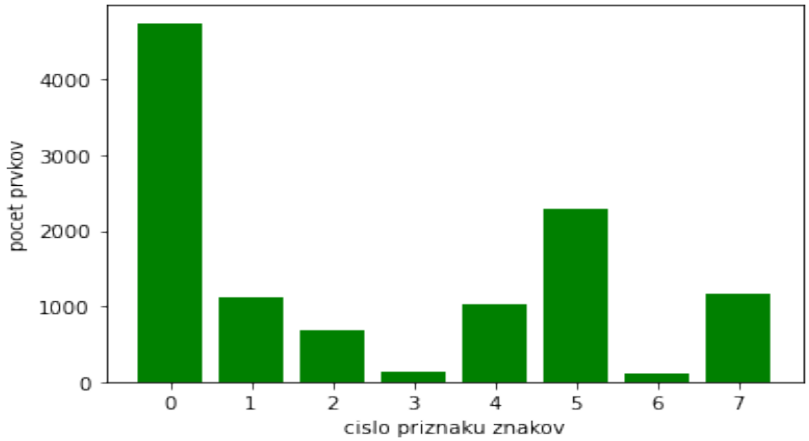


Figure 4: histogram rozloženia dát v jednotlivých príznakoch.

```
dataset_positive = 'Dataset_v1.2/Positive'
dataset_negative = 'Dataset_v1.2/Negative'

dataset_seven = 'seven_plastics'

num_of_datas = {'1': 0, '2':0, '3':0, '4':0, '5':0, '6':0, '7':0, '8':0}

# this function creates directories needed for easier loading
def create_dir_for_dataset():
    current_directory = os.getcwd()
    final_directory = os.path.join(current_directory, r'datasets')

    if not os.path.exists(final_directory):
        os.makedirs(final_directory)

    for i in range(1,9):
        if not os.path.exists(os.path.join(final_directory, f'{i}') ):
            child_directory = os.path.join(final_directory, f'{i}')
            os.makedirs(child_directory)

def remove_str(str):
    nums = re.findall(r'\d+', str)
    if len(nums) == 0:
        return None
    return int(nums[0])

## move -> separate dataset to appropriate directories bolognese university
def move_v12():
    # splitting data for getting just 1-7 labelled and added them to final folder dataset
    # !! note number 8 is included in folder negative
    for data in os.listdir(dataset_positive):
        tmp = data.split('_', -1)
        num = remove_str(tmp[2])
        if num is not None and num < 8 and num > 0:
            shutil.copy(os.path.join(dataset_positive, data), os.path.join(f'datasets/{num}', data))

            num_of_datas[str(num)] +=1

    for data in os.listdir(dataset_negative):
        shutil.copy(os.path.join(dataset_negative, data), os.path.join(f'datasets/{8}', data))
        num_of_datas[str(8)] +=1

## move -> separate dataset to appropriate directories kaggle
def move_seven():
    for nums in  os.listdir(dataset_seven):
        actual_num  = nums.split('_')[0]
        actual_dir = os.path.join(dataset_seven, nums)
        for img in os.listdir(actual_dir):
            shutil.copy(os.path.join(actual_dir, img), os.path.join(f'datasets/{actual_num}', img))
            num_of_datas[actual_num] +=1
```

Figure 5: kód pre spojenie a zatriedenie vyššie spomenutých datasetov.

[1] https://www.kaggle.com/datasets/piaoya/plastic-recycling-codes

[2] https://miatbiolab.csr.unibo.it/the-recycling-symbols-dataset/?fbclid=IwAR1ZpDzvAxO8pebxJ4vWfuM8Kmsjgajs_$YuliGZew9PTu3Q0uc9ysffEsg$

5. Experimenty

Ako mi bolo doporučené pri project proposal, rozhodol som sa trénovať model pomocou prístupu transfer learning. Po preštudovaní danej problematiky som sa rozhodol že použijem už predtrénované modely z knižnice Tensorflow.Keras.

5.1 Načítanie dát

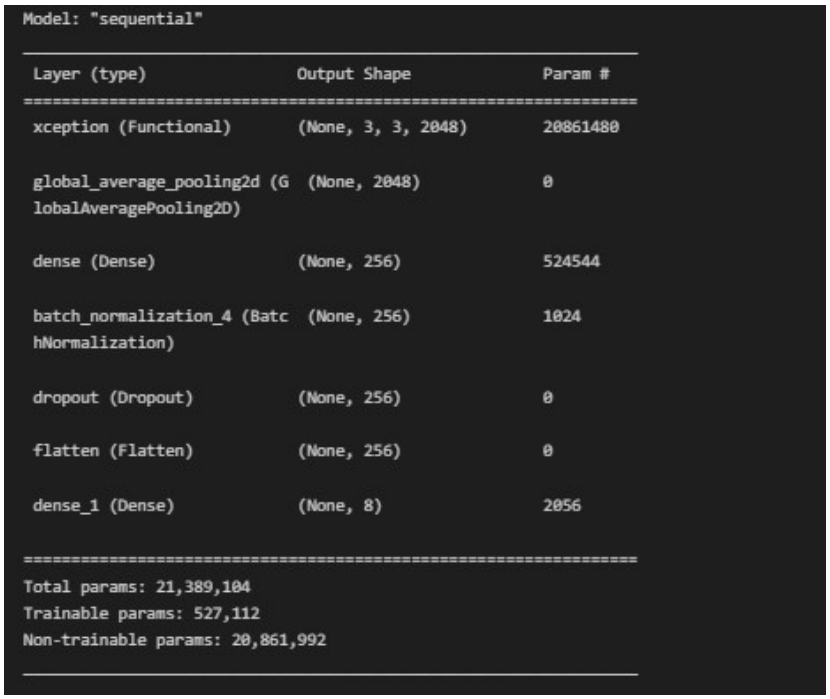
Dáta som načítaval dvoma spôsobmi a to vlastnoručné načítanie do numpy poľa a pomocou knižnice tensorflow.

vlastnoručné načítanie
Obrázky som prvoplánovo načítaval s rozmermi 256x256. Žiaľ pri neskoršom trénovaní modelu som zistil že takéto rozlíšenie bolo príliš veľké a dochádalo k chybe out of memory. Preto som musel zmeniť rozlíšenie na 128x128.
Pri týchto dátach som po preštudovaní si témi použil predtrénoavné modely Xception a MobileNetV2.

načítanie pomocou knižnice
Načítanie pomocou knižnice bolo veľmi spoľahlivé a účinné. Dáta sa načítali omnoho rýchlejšie ako predošlou spomenutou ako metódou. Taktiež som dokázal dáta načítavať s rozlíšením 256x256 - s tým súvisí jeden z experimentov kde som skúšal dáta trénovať na obrázkoch s rozmermy 128x128 a rozmermi 256x256. S dátami načítanými metódami som trénoval s predtrénovaným modelom MobileNetV2. Pre porovnanie pri trénovaní s tým istým modelom ale inak načítaními dátami a takým istým počtom opakovaní sa dáta vytrénovali 3x rýchlejšie.

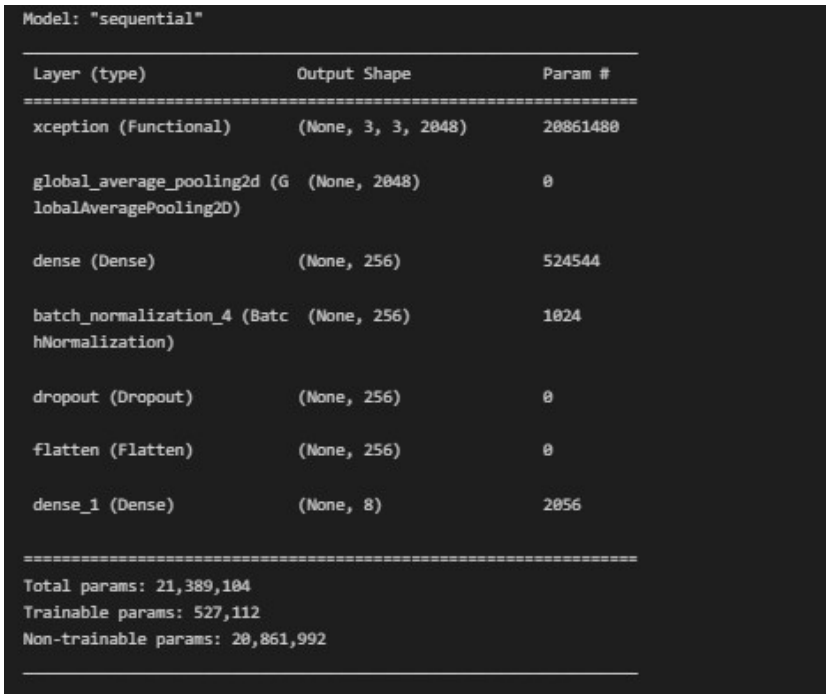
5.2 1. Experiment

Prvý experiment spočíval v porovnaní trénovania pomocou predtrénovaných modelov Xception a MobileNetV2. Pri tomto experimente som dáta načítaval vlastnoručne.
Ako je vidieť z obrázkov 8 a 9 pri trénovaní došlo k značnému overfittu. Pre porovnanie 11 a 12 pri predtrénovanom modely Xcentric nieje značne overfittovany ale je viac nestabilný. Pri porovnaní modelov na testovacích dátach 13 a 10 je model predtrénovaný z Mobilnet značne lepší a vie predikovať väčšie spektrum dát.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 3, 3, 2048)        20861480

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 256)               524544

 batch_normalization_4 (Batc  (None, 256)              1024
 hNormalization)

 dropout (Dropout)           (None, 256)               0

 flatten (Flatten)           (None, 256)               0

 dense_1 (Dense)             (None, 8)                 2056

=================================================================
Total params: 21,389,104
Trainable params: 527,112
Non-trainable params: 20,861,992
_________________________________________________________________
```

Figure 6: Sumarizácia modelu z predtrenovaného modelu Xception.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 3, 3, 2048)        20861480

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 256)               524544

 batch_normalization_4 (Batc  (None, 256)              1024
 hNormalization)

 dropout (Dropout)           (None, 256)               0

 flatten (Flatten)           (None, 256)               0

 dense_1 (Dense)             (None, 8)                 2056

=================================================================
Total params: 21,389,104
Trainable params: 527,112
Non-trainable params: 20,861,992
_________________________________________________________________
```

Figure 7: Sumarizácia modelu z predtrenovaného modelu Mobilnet.

Výsledky trénovaní modelov za 60 opakovaní pri rozdelení dát 0.8, 0.1, 0.1:

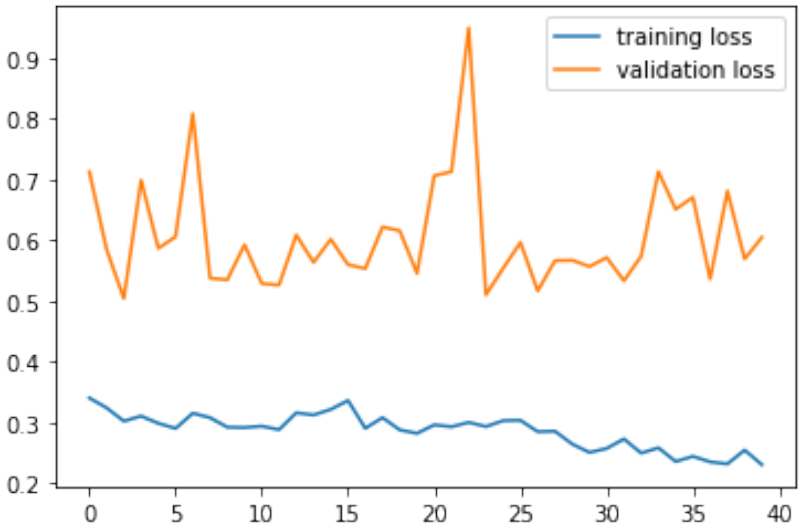


Figure 8: Zobrazenie trenovacej a validačnej presnosti z predtrenovaného modelu MobilnetV2.

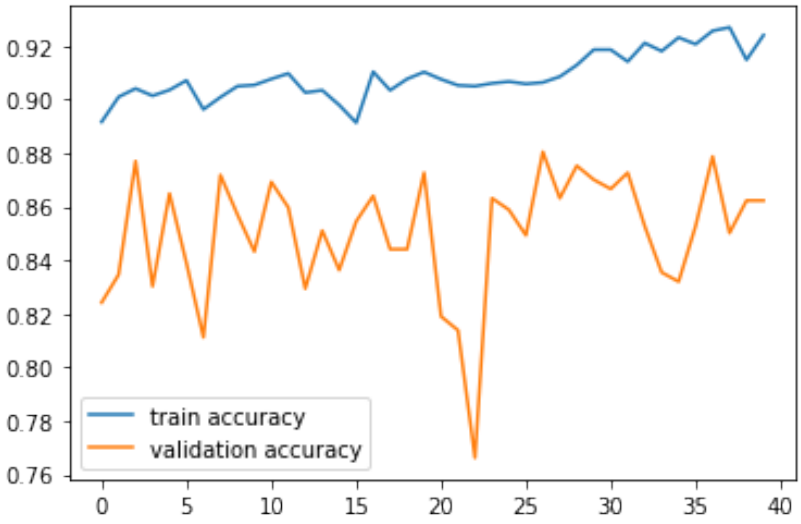


Figure 9: Zobrazenie trenovacej a validačnej chyby z predtrenovaného modelu MobilnetV2.

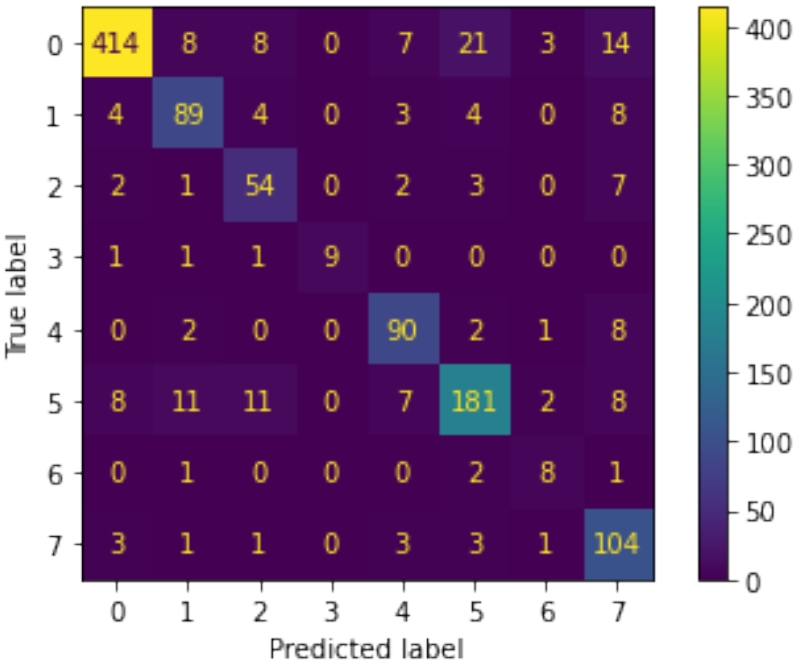


Figure 10: Zobrazenie testovacej presnosti MobilnetV2.

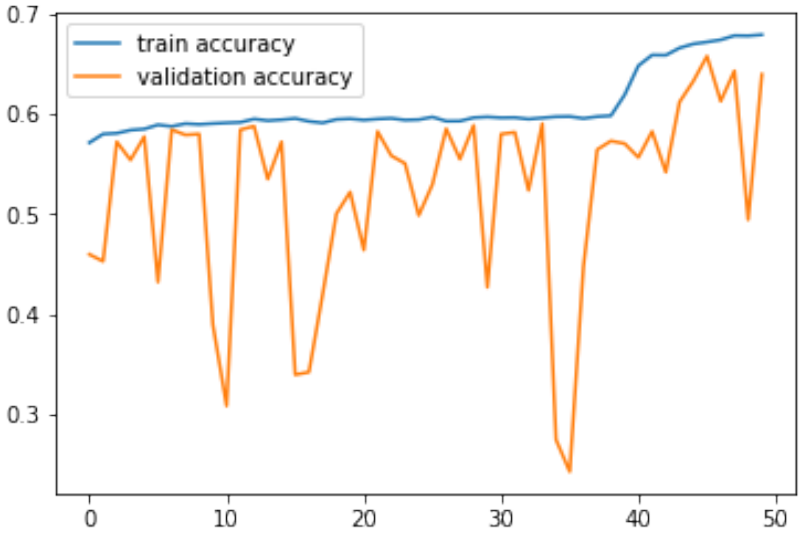


Figure 11: Zobrazenie trenovacej a validačnej presnosti z predtrenovaného modelu Xception.

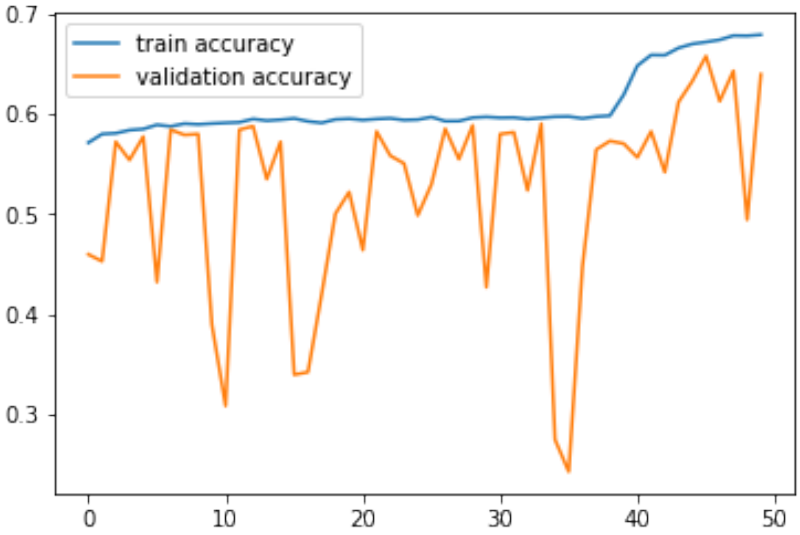


Figure 12: Zobrazenie trenovacej a validačnej chyby z predtrenovaného modelu Xceptio.

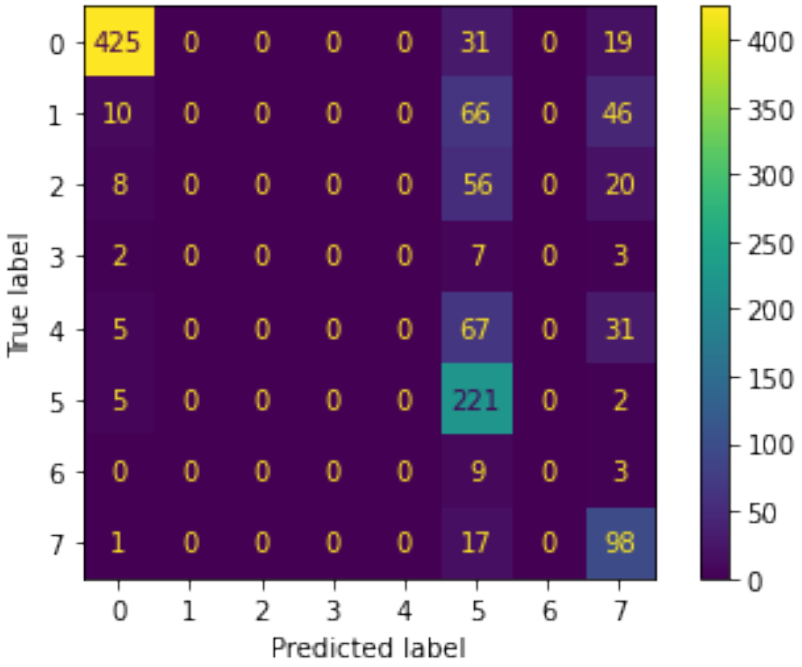


Figure 13: Zobrazenie testovacej chyby modelu Xceptiot.

5.3 2. Experiment

Po pokusoch a študovaní datasetu som zistil že existuje možnosť umelo vytvoriť dáta 3.3 a taktiež je možné dátanačítať pomocou knižnice Keras. Taktiež po Experimente 1 bolo zjavne že dáta overfittujú, čo bolo čiastočne sposobené nedostatkom dát, ale aj nedobrým nastavenim vstupných parametrov. Nasledujúci experiment bol vzkonávaný s 60 opakovaniami s pomerom trénovacích a testovacích dát 1:9. Dáta boli trénované taktiež na modeloch MobilNetV2 a Xception. výsledky:

```
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 256, 256, 3)]     0

sequential_1 (Sequential)    (None, 256, 256, 3)       0

tf.math.truediv_1 (TFOpLamb  (None, 256, 256, 3)       0
da)

tf.math.subtract_1 (TFOpLam  (None, 256, 256, 3)       0
bda)

xception (Functional)        (None, 8, 8, 2048)        20861480

global_average_pooling2d_1   (None, 2048)              0
(GlobalAveragePooling2D)

dropout_1 (Dropout)          (None, 2048)              0

dense_1 (Dense)              (None, 8)                 16392

=================================================================
Total params: 20,877,872
Trainable params: 16,392
Non-trainable params: 20,861,480
_________________________________________________________________
```

Figure 14: Zobrazenie sumarizácie poskladania vrstiev CNN ktoré boli použité.

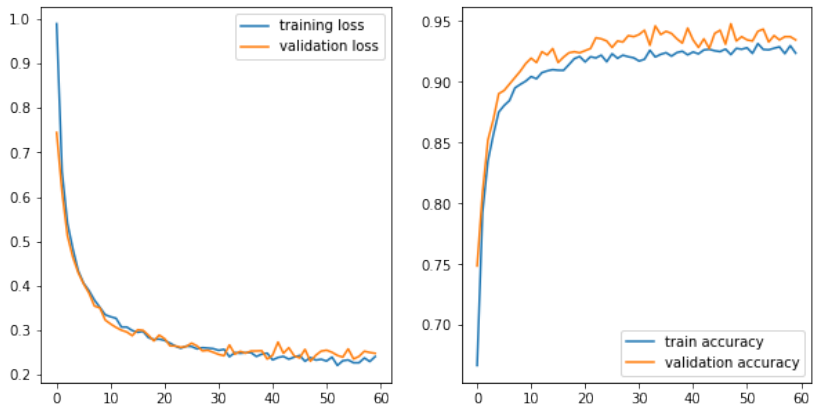


Figure 15: Validacna a trénovacia chyba,presnost modelu trenovaneho na Xception

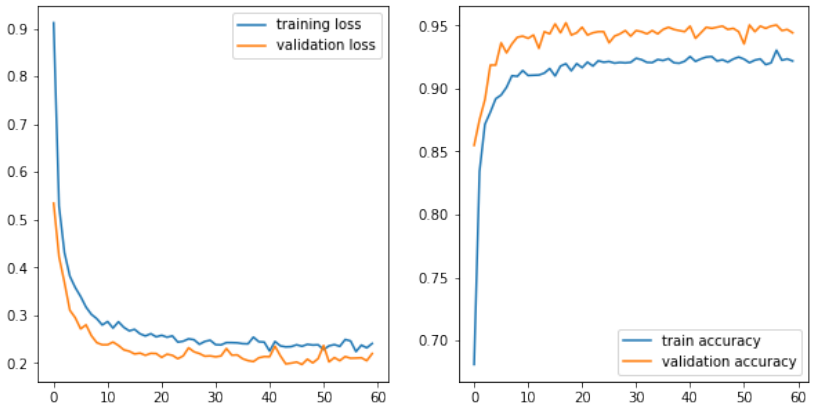


Figure 16: Validacna a trénovacia chyba,presnost modelu trenovaneho na Mobilnet

Na výsledkoch z dátovej sady trénovanej na Mobilnet možeme vidieť jemné znaky podtrénovania. Kdežto pri sade trénovanej na Xception nie. Môže to byť spôsobené tým, že model Xception bol trénovaný na robustnejšej CNN. Pri oboch modelch sa testovacia chyba nedostala pod hranicu 0.2.

6. Záver

Boli vykonané 2 experimenty prvý bol viac menej učiaci a hľadal som pri ńom veci ktoré by sa mohli odstrániť v ďalšom experimente. 2 experiment nadviazoval na prvý a boli v ńom zohľadnené poznaky prvého (správne nastavene vstupných parametrov do prvej vrsty, umele vytvorenie dát) Predišlo sa v ńom overfittovaniu avšak ani jeden model sa nedokázal dostať pod hranicu 0.2. Do budúcna by sa to dalo vyriešiť tým, že by sa natrénoval model ktorý by rozoznával ručne písané číslice (mnoho datasetov a veľkých). Tento model by sa následne použil ako základ pre dotrénovanie modelu pre predikciu dát z plastových symbolov. Python notebooky a data (nie vsetky kody vela experimetalnych som musel koli prehladnosti repozitara) odkaz na github repositar