Klasifikácia vtáčích druhov cez Konvolučnú neurónovú sieť

www.kaggle.com

Obsah

Zdrojový kód	ann/softmax-cnn at main · viktorFIIT/ann (github.com)

Dátová množina vtáčích druhov

Obrázky na úlohu klasifikácie sú prebraté z databázy Kaggle dostupnej na webovom odkaze BIRDS 400 - SPECIES IMAGE CLASSIFICATION | Kaggle a obsahujú 60 388 RGB (3-kanálových) obrázkov. Celý dataset obsahuje obrázky 400 druhov vtákov. Obrázky vtákov sa nachádzajú vo formáte JPG a obrázky pre každý druh sú organizované v samostatnom priečinku. Celkovo pozostáva z:

- 58 388 obrázkov v trénovacej množine - 79 druhov vtákov.
- 2000 obrázkov vo validačnej množine - 400 druhov vtákov.
- 2000 obrázkov v testovacej množine (5 JPG obrázkov pre každý druh) - 400 druhov vtákov.

Dáta získané z verejnej databázy sú už pri ich získaní organizované do trénovacej, validačnej a testovacej množiny. Na každom obrázku je len 1 vták. Z výsledkov rozdelenia dát autormi vyplýva že sa rozhodli takmer 97 % obrázkov z celej množiny obrázkov použiť na trénovanie modelu a len 3,31 % dostupných obrázkov použiť na validáciu a testovanie modelu. Takáto situácia je však výnimkou ako pravidlom, pretože rozdelenie dát v tomto (alebo obdobnom pomere 98 %, 1 % a 1%) je skôr bežné pre množiny dát s miliónmi vzoriek.

Obrázky sú v RGB troj-kanálovom farebnom modeli s výškou 224 pixelov a šírkou 224 pixelov. Vo všetkých obrázkoch vták o ktorého klasifikáciu máme záujem zaberá 50 % priestoru obrázku. Ten istý druh vtáku a jeho obrázok z trénovacej množiny je tiež možné nájsť vo validačnej a testovacej množine. Z toho vyplýva:

- že na obrázkoch vtákov ktoré sa nachádzajú aj v trénovacej aj vo validačnej množine bude klasifikujúca neurónová sieť pravdepodobne úspešnejšia ako by mala byť.
- na ostatných obrázkoch ktoré neurónová sieť pri trénovaní nevidela je zvýšená pravdepodobnosť že bude pri klasifikácii týchto obrázkov z validačnej množiny neúspešná.

Vzhľadom na to že v trénovacej množine je prevaha vtákov samcov ako samičiek (85 % samcov) a že samce vtákov sú farebnejšie ako samičky jedná sa o množinu dát z nevyváženým množstvom tried do ktorých sa neskôr posnažíme obrázky klasifikovať[1]. Bežnou technikou pri obmedzenej dostupnosti záznamov z ďalších tried aplikovanou na množinou dát je vytvorenie syntetických dát z už existujúcich ich transformáciou. V prípade transformácie časti existujúcich obrázkov v trénovacej množine do jednokanálovej podoby (grayscale) by modelu farebnosť obrázkov chýbala (absencia hrán by konvolučnej neurónovej sieti neumožnila správne zovšeobecniť informácie o tele vtákov).

[1] nevyváženosť tried sa pokúsime vyriešiť tiež manipulovaním hyper-parametra *stride*. Pre vysvetlenie pozri 6.3. Padding and Stride — Dive into Deep Learning 0.17.5 documentation (d2l.ai)

Ďalšou (aj keď teoretickou) implikáciou z našich úvah je že klasifikujúca neurónová sieť môže byť menej úspešnejšia pri klasifikácii tých obrázkov z validačnej množiny ktoré sa nenachádzajú v trénovacej množine[2]. V trénovacej množine očakávame čo najrovnomernejšie zastúpenie obrázkov všetkých možných druhov vtákov ktoré môžu byť pri reálnom použití modelu na klasifikáciu vložené. Je teda jasné že ak by sme analyzovali rozloženie hodnôt konkrétnych pixelov v obrázkoch zo všetkých 3 množín nemali by rovnaké rozloženie. Najčastejšou situáciou ktorá nastáva je že rozloženie trénovacej množiny je iné ako rozloženie množiny určenej na validáciu. Takáto situácia nám zároveň logicky dovoľuje porovnávať výkon modelov hlbokého učenia na validačnej množine. Jednou z prvých podmienok ktorú na náš očakávaný klasifikačný model kladieme je že naučený klasifikačný model musí dosahovať nízku hodnotu chyby klasifikácie podľa stanovenej metriky už pri jeho trénovaní. Výkon viacerých modelov porovnáme podsunutím validačnej množiny. Modely ktoré dostanú na klasifikáciu obrázky vtákov z validačnej množiny zároveň budeme optimalizovať menením hodnôt hyperparametrov (napr. rýchlosti učenia pri Nesterovom momente), teda tých parametrov umelej neurónovej siete ktoré sa sieť pri trénovaní sama neučí.

Problém klasifikácie vtáčích druhov

Konvolučné neurónové siete predstavujú len jednu z mnohých architektúr ktoré existujú[3]. Logicky však nastáva otázka prečo ich použiť práve na klasifikáciu obrázkov vtáčích druhov. Historicky základy konvolučných neurónových sietí položil Frank Rosenblatt vo svojej práci opisujúcej biológiou inšpirovanú implementáciu viac-vrstvového plne-prepojeného perceptrónu "*A probabilistic model for information storage and organization in the brain*". V nej opisoval spôsob akým možno takýto perceptrón naučiť identifikovať obrázok z trénovacej množiny. Použitie logického perceptrónu navrhnutého Frankom Rosenblattom by bolo však na náš problém klasifikácie nevhodné, pretože:

- logický perceptrón by ignoroval globálnu informáciu o tvaroch vtákov ktorú potrebujeme získať a zovšeobecniť.
- takýto logický perceptrón by len obtiažne dokázal identifikovať vtákov na obrázkoch na ktorých by sa nachádzalo mnoho našej úlohe nesúvisiacich objektov.
- logický perceptrón by tiež nedokázal identifikovať tých istých vtákov nachádzajúcich sa len v rôznych častiach obrázku.
- perceptrón by od nás predom vyžadoval určenie vhodných príznakov ktoré odlišujú vtákov. My však chceme určenie týchto príznakov naučiť konvolučnú sieť

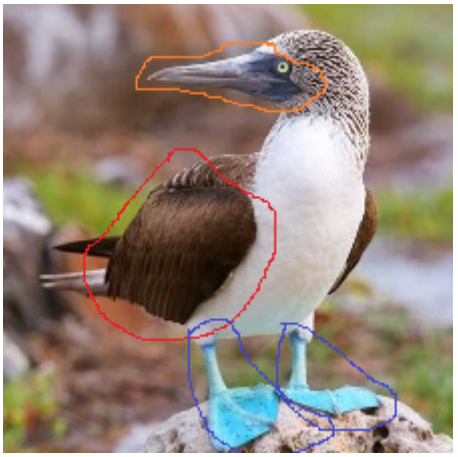
Na vyriešenie problému získania globálnej informácie o tvare vtákov z obrázkov nadviažeme na myšlienku Hubel a Wiesela, pričom podmienkou ktoré na funkciu konvolučnej neurónovej siete kladieme je, že:

---

[2] Takýto druh vtáku neexistuje, ale vo validačnej množine môže byť iný obrázok vtáku z toho istého druhu

[3] https://www.asimovinstitute.org/neural-network-zoo

- vrstvy umelej neurónovej siete ktoré budú krok za krokom vytvárať abstraktnejší pohľad na tvar vtákov (napr. identifikujúc hrany krídel) budeme nazývať konvolučné vrstvy. Tieto vrstvy vytvárajú topografickú mapu.
- skupiny perceptrónov budú zodpovedné za identifikáciu čiastkovej informácie: úzke nohy, zobák, oči na odvrátených stranách hlavy.
- skupiny perceptrónov ktoré budú identifikovať inú časť rovnakej časti tela vtáka (napr. ľavú a pravú nohu) budú spojené a skupiny perceptrónov prepojené
- nižšie vrstvy umelej neurónovej siete budú získavať informáciu o samostatných častiach tela vtáka. Stredné vrstvy začnú vytvárať informáciu o zakrivení častí jeho tela (nohy vo vertikálnej polohe). Vyššie vrstvy doplnia informáciu o prepojení častí tela vtáka, ktoré identifikovali jej stredné vrstvy (oranžová + červená + modrá časť blízko seba).



Druh vtáku: Abbotts Booby žijúceho na ostrovoch vo východnej časti Indického oceánu

Konvolučná neurónová sieť však bude len extraktorom informácií pre klasifikátor predstavovaný viacvrstvovým perceptrónom. Konvolučná vrstva sa bude učiť extrahovať informácie o vzhľade vtákov učením sa hodnôt konvolučných jadier. Farebnosť obrázkov, resp. ich zobrazenie vo viacerých kanáloch akým je napr. trojkanálový farebný model RGB je dôležitým faktorom pretože pri jednom kanáli (“grayscale” obrázkoch, inverzných transformáciách farebných obrázkov) by sieť nedokázala nachádzať črty tela vtákov tak dobre ako pri farebných.

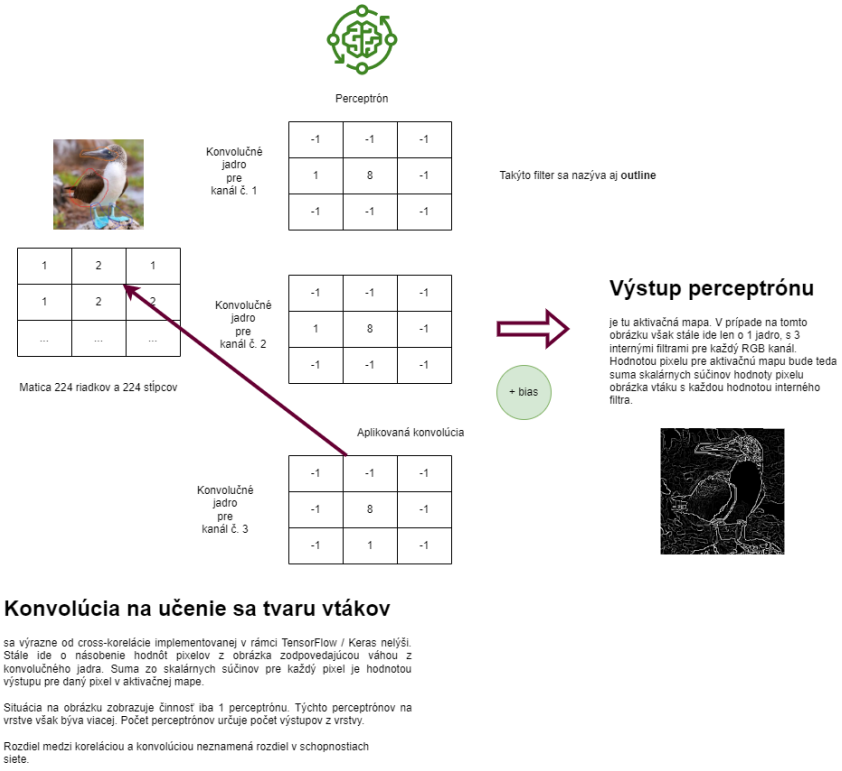
Vstupom pre extraktor informácií o tvare tela vtákov bude pri učení množina 58 388 obrázkov, resp. trojíc matíc o 224 riadkoch a 224 stĺpcoch pre každý kanál farebného modelu RGB. Poskytnutie vstupu extraktoru vo forme matíc je dôležité pre zachovanie priestorovej informácie - hodnôt príznakov pre jednotlivé časti tela.

Odpoveďou na našu predom položenú otázku, prečo je použitie konvolučnej neurónovej siete výhodnejšie oproti použitiu viacvrstvového perceptrónu je teda “*preto, lebo konvolučná neurónová sieť sa nemusí naučiť tvar všetkých druhov vtákov nachádzajúcich sa na všetkých možných miestach na obrázku ale na identifikáciu druhu sa stačí naučiť len abstraktné príznaky správne identifikujúce druh vtáku*”. Konvolučné neurónové siete sú zároveň efektívnejšou implementáciou klasifikujúceho modelu a to aj pre ich konvolučné

vrstvy, ktoré vyžadujú menej parametrov[4] (v porovnaní s plne prepojenými vrstvami) a tiež preto lebo ich výpočty možno ľahko paralelizovať na grafickej karte.

Návrh architektúry konvolučnej neurónovej siete

Predtým než začneme učiť našu sieť klasifikovať vtáčie druhy je potrebné ju naučiť vtáčie druhy na obrázkoch identifikovať. Na vykonanie tejto úlohy sa musí naučiť váhy pre filtre - extraktory informácií (inak volané aj konvolučné jadrá). V našom prípade budeme konvolučnú sieť učiť na 2-dimenzionálnych obrázkoch, no konvolučné siete môžu byť použité aj na prácu s 3-dimenzionálnym obrazom. Rozmer konvolučných jadier pre našu sieť určuje počet kanálov vo farebnom modeli. Konvolučné jadrá, resp. filtre budú mať teda rozmer 3x3. Predpokladáme že na to aby bola sieť schopná klasifikovať vtáky musí mať na to výpočtovú silu, čo znamená viacej ako 1 perceptrón. Vzhľadom na to že sieť bude pracovať s obrázkami vo farebnom modeli RGB bude každý perceptrón tvorený trojicou konvolučných jadier, resp. filtrov. Každý perceptrón vyprodukuje aktivačnú mapu, ktorej hodnoty pixelov nazývame aktiváciami. Aktivácie sú výstupy procesu nazývaného konvolúcia (resp. cross-korelácia) ktorý znázorňuje nasledujúci obrázok. Súčasťou experimentov bude tiež výmena inicializátorov váh, započítavanie a nepoužívanie chýb / bias-ov ku každej lineárnej funkcii *z* perceptrónov:



[4] 6. Convolutional Neural Networks — Dive into Deep Learning 0.17.5 documentation (d2l.ai)

Ďalším dôvodom pre ktorý chceme použiť filtre v konvolučnej neurónovej sieti je ich schopnosť naučiť sieť extrahovať príznaky ktoré odlišujú rôzne druhy vtákov od seba. Tieto filtre na prvých aktivačných mapách nie sú ešte schopné zachytiť veľa informácie, no po prechode vstupu viacerými konvolučnými vrstvami sú tieto filtre efektívnejšie práve vďaka "vyčisteniu" aktivačných máp od často nepotrebných informácií.

Experiment č. 1, č. 2 - Softmax regresia - Klasifikácia druhov vtákov

Zdrojový kód k experimentu	ann/1-experiment.ipynb at main · viktorFIIT/ann (github.com)
Výsledok experimentu	preučený model na trénovacej množine, slabá schopnosť klasifikácie, slabá schopnosť učenia

Parametre ktorým riadim proces učenia môžeme rozdeliť do dvoch skupín. Na tie ktoré určujú architektúru siete a váhy filtrov konvolučných vrstiev, a na doplňujúce ktoré určujú napr. počiatočné hodnoty váh[5] alebo započítanie chýb (bias) do modelu. Prvá konvolučná vrstva s filtrami o rozmere 3x3[6] (podľa počtu kanálov v RGB farebnom modeli) obsahuje 32 blokov (perceptrónov) kde každý blok obsahuje trojicu konvolučných filtrov (pre každý kanál). Výber väčších konvolučných filtrov znásobuje potrebné výpočtové zdroje. Zmenšenie rozmeru konvolučných filtrov zase ovplyvňuje množstvo vizuálnej informácie ktorú filter - detektor tela vtáku vo vstupe resp. aktivačnej mape vidí.

```
# lowest convolutional layer for identification of the edges of birds
model.add(Conv2D(32, (3, 3), padding='same', input_shape=SHAPE))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Obrázok č. 1: Prvá konvolučná vrstva na zachytenie lokálnych príznakov

Pridanie vrstvy pre MaxPooling zvyšuje citlivosť neurónov na skrytých vrstvách na vizuálne príznaky tela vtákov a zároveň znižuje závislosť siete od priestorovej informácie tak že sa bude vedieť rozhodnúť pre daný druh vtáku bez ohľadu na umiestnenie vtáka na obrázku. MaxPooling znižuje závislosť konvolučných vrstiev od priestorovej informácie, tak že prenecháva posledné rozhodnutie na viacvrstvový perceptrón. Vzhľadom na to že riešime problém viactriednej klasifikácie sa nám ako najčastejšia možnosť ponúka použiť ako aktivačnú funkciu *Sigmoid*, *Softmax, ReLU* alebo Leaky ReLU:

- pri výbere aktivačnej funkcie *Sigmoid* pre všetky vrstvy treba počítať s jej nevýhodou že saturuje gradient. To znamená že pri veľkých hodnotách gradientu sa gradient výrazne nemení a učenie preto stagnuje.

[5] Vyskúšané incializátory váh sú:
[6] 6.4. Multiple Input and Multiple Output Channels — Dive into Deep Learning 0.17.5 documentation (d2l.ai)

- pri výbere aktivačných funkcií typu *Rectified Linear Unit tre*ba počítať s tým, že niektoré neuróny môžu byť inhibované. Túto nevýhodu rieši jej náhrada Leaky ReLU.
- *softmax* aktivácia sa používa na viac triednu klasifikáciu na výstupnej vrstve pričom produkuje vierohodnosť / pravdepodobnosť príslušnosti klasifikovaného objektu do jednej z tried.

Výstupom z prvej konvolučnej vrstvy bude aktivačná mapa s 32 kanálmi. Filtre nie sú zoradené paralelne ale sekvenčne. Paralelne zoradené filtre a teda preferovanie širších než hlbších sietí by sa aj podľa autora[7] vyplatilo skôr pri klasifikácii obrázkov na ktorých by bol vták umiestnený len v ich časti, pričom by sme museli brať do úvahy aj informáciu o ďalších elementoch na obrázku. Toto nám však pri orezaných obrázkoch a vycentrovaných vtákoch na obrázkoch odpadá. Parameter *padding* je pre túto konvolučnú vrstvu a aj všetky ostatné nastavený na hodnotu "*same*"[8] čo zároveň pri prihliadnutí na veľkosť filtra (3x3 pixely) znamená že konvolučná vrstva nastaví všetky pixely v prvých stĺpcoch zľava aj sprava aktivačnej mapy a zároveň v prvých riadkoch zhora aj dolu aktivačných máp na hodnotu 0. Veľkosť aktivačných máp sa pri tomto nastavení parametra konvolúcií nemení. Vzhľadom na to že pri tomto experimente ešte s veľmi hlbokou neurónovou sieťou nepracujem je dôvodné predpokladať, že aktuálne nastavenie nespôsobí výraznú akumuláciu chyby klasifikácie pri učení (predpokladáme že sieť aj v produkcii dostane vycentrovaný a orezaný obrázok s vtákom). Vzhľadom na to že sa vtáci v trénovacej, validačnej a testovacej množine nachádzajú vždy takmer v strede obrázka predpokladáme že dané nastavenie siete nespôsobí výraznú nepresnosť klasifikácie vtákov (hoci bude sieť nepresná na okrajoch obrázka).

```
tf.keras.backend.clear_session()

IMAGE = load_img(os.getcwd() + "\\testing\\ABBOTTS BABBLER\\1.jpg")
IMAGEDATA = img_to_array(IMAGE)
SHAPE = IMAGEDATA.shape

model = tf.keras.models.Sequential()

# lowest convolutional layer for identification of the edges of birds
model.add(Conv2D(32, (3, 3), padding='same', input_shape=SHAPE))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# convolutional layer to learn and store mid-level features of the bird species
model.add(Conv2D(32, (3, 3), padding="same"))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# highest convolutional layer to store complex information about the look of birds
model.add(Conv2D(32, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# and finally mlp
model.add(Flatten())
model.add(Dense(32))
model.add(Activation('tanh'))
model.add(Dense(Train_Category_count))
model.add(Activation('softmax'))
```

Obrázok č. 2: Architektúra siete z 1. experimentu

Prechodom konvolučnými vrstvami sa zároveň zvýšil počet kanálov v aktivačných mapách (čo je však bežnou praktikou 6.4. Multiple Input and Multiple Output Channels — Dive into

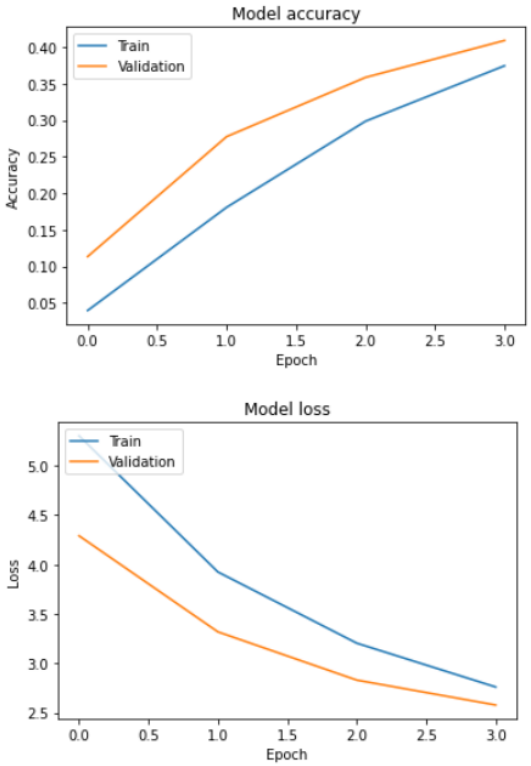
[7] A Simple Guide to the Versions of the Inception Network | by Bharath Raj | Towards Data Science
[8] Hodnota "same" sa uvádza pri práci s rámcom TensorFlow

Deep Learning 0.17.5 documentation (d2l.ai)). Fakt že sa nám počet kanálov prechodom cez konvolučné vrstvy zvyšuje nám zaručuje zachytávanie informácie o hranách - lokálne informácie z viacerých kanálov sa spájajú na vytvorenie globálnej informácie o hrane. Tréningový dataset má nerovnomerné zastúpenie samičiek a samcov vtákov (samci ktorých je 85 % sú farebnejší), no cieľom klasifikácie je klasifikovať druhy a nie samcov a samičiek. Vzhľadom na tento fakt sa však oplatí nastaviť ďalší parameter *stride* a to pre dva dôvody: **a)** na umelé zníženie vplyvu počtu farebnejších samcov k samičkám[9] a **b)** na zefektívnenie náročného výpočtu. Na zníženie vplyvu nadmerného počtu farebnejších samcov vtákov je taktiež dôvodné pristúpiť k augmentácii trénovacej množiny a to transformáciou časti existujúcich obrázkov z trénovacej množiny na vytvorenie tých nových syntetických (súčasťou experimentu č. 4)[10]. S aktuálnym nastavením siete tak ako je znázornený na obrázku č. 2 počas experimentu č. 1 nedosahujeme želaný výsledok. Po takmer hodine aj 6 minútach trénovania modelu sa chyba na validačnej množine (variance) výrazne nemení, no čo je možno ešte horšie podiel úspešných klasifikácií voči všetkým stagnuje na nízkej hodnote. Na vyriešenie aj tohto problému vieme stopnúť trénovanie modelu ak sa po 5 epochách nedostaneme aspoň na úroveň úspešnosti klasifikácie 0.5, pričom za pokrok pri klasifikácii budeme považovať zvýšenie hodnoty tejto metriky o jednotku (na implementáciu vieme použiť komponent rámca Keras: EarlyStopping (keras.io)). Schopnosť klasifikácie obrázkov vtákov model v súčasnom nastavení zvládal lepšie na (plnej, nezmenšenej) trénovacej množine ako na validačnej. Model bol súčasne v čase experimentu č. 1 preučený na trénovacej množine, pretože chyba jeho klasifikácií bola menšia na trénovacej množine ako na validačných dátach a to takmer počas všetkých epóch - prechodov nad danou množinou. Najväčším problémom modelu vytvoreného počas 1. experimentu bola jeho veľmi slabá schopnosť učiť sa.

```
Epoch 1/4
913/913 [==============================] - 1083s 1s/step - loss: 5.3023 - accuracy: 0.0397 - val_loss: 4.2912 - val_accuracy:
0.1135
Epoch 2/4
913/913 [==============================] - 1084s 1s/step - loss: 3.9250 - accuracy: 0.1808 - val_loss: 3.3192 - val_accuracy:
0.2775
Epoch 3/4
913/913 [==============================] - 906s 993ms/step - loss: 3.2048 - accuracy: 0.2986 - val_loss: 2.8310 - val_accuracy:
0.3585
Epoch 4/4
913/913 [==============================] - 946s 1s/step - loss: 2.7614 - accuracy: 0.3744 - val_loss: 2.5789 - val_accuracy: 0.
4090
```

[9] 6.3. Padding and Stride — Dive into Deep Learning 0.17.5 documentation (d2l.ai)

[10] Cieľom je získanie rovnomernejšie rozdeleného datasetu technikou “downsampling”, teda umelým znížením vplyvu záznamov z majoritnej triedy



Obrázok č. 3: Výstupy z učenia modelu z 1. experimentu hovoria o jeho neschopnosti klasifikovať

Vzhľadom na veľkosť datasetu a obmedzený prístup k optimalizácii na GPU (viď Poznámky k implementácii) vidíme, že sa nám neoplatí v trénovaní ani pokračovať nakoľko by sme pri takejto rýchlosti učenia čakali príliš dlho na akceptovateľnú hodnotu metriky *úspešnosť klasifikácie*. Slabú schopnosť učenia môžeme teda vidieť na nízkych hodnotách metriky "*accuracy*", ktorá je podielom úspešných klasifikácií voči všetkým spraveným. Je taktiež dôležité poznamenať že aktuálny problém klasifikácie je hlbší ako len v nesprávnom použití aktivačných funkcií (v experimente 1 je použitá ReLU, v experimente č. 2 je použitý hyperbolický tangens). Je dôsledkom toho že sieť sa v nastavení v akom bola počas 2. experimentu nebola schopná sa dobre naučiť klasifikovať obrázky. Tento nedostatok bez ohľadu na mierne úpravy modelu v experimente č. 2 stále neviedol k želanému zlepšeniu úspešnosti klasifikácie:

- a to ani po tom, čo sme v experimente č. 2 zmenšili dávku z pôvodných 64 obrázkov na 32. Trénovací dataset je príliš rozsiahly na dávkovú optimalizáciu učenia. Stochastická optimalizácia po 1 vzorke je zase neželaná pretože by mohla viesť ku neželaným veľkým výkyvom pri učení modelu[11]. Vybrali sme preto štandardný postup učenia modelu mini-dávkovou optimalizáciou[12]. Zníženie veľkosti dávky malo za cieľ znížiť generalizačnú chybu modelu.
- na zlepšenie a zrýchlenie učenia sme taktiež za konvolučné vrstvy pridali vrstvu na normalizáciu výstupu - vstupu pre každú nasledujúcu vrstvu. Normalizácia dávok ešte neznamená že sa nám sieť preučí, čo nám vyhovuje. Proti preučeniu takto

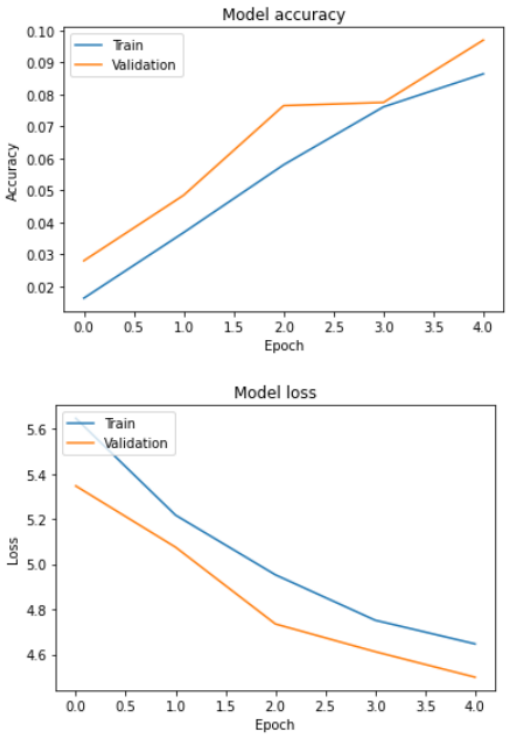
[11] Stochastická optimalizácia učenia by si vyžadovala dodatočnú úpravu hyper-parametrov, napr. rýchlosti učenia
[12] Mini-dávková optimalizácia tiež umožňuje online learning a ako technika učenia je presná

bojujeme úpravou distribúcii výstupov z konvolučných vrstiev čím sme nasledujúcim vrstvám pomohli zovšeobecniť informáciu ktorú sa mala sieť naučiť.

Architektúra siete v experimente č. 2 pozostávala z 3 konvolučných vrstiev, operáciách max pooling po aktiváciách na uľahčenie učenia tela farebných vtákov na tmavšom pozadí. Pridaná normalizácia na úpravu distribúcie vstupov pre nasledujúce vrstvy (bojujeme proti preučeniu na trénovacích dátach) nemala vo výsledku želaný efekt:

```
model = tf.keras.models.Sequential()

# lowest convolutional layer for identification of the edges of birds
# input shape is provided, so no deferred initialization https://d2l.ai/chapter_deep-learning-computation/deferred-init.html
model.add(Conv2D(32, (3, 3), padding='same', input_shape=SHAPE))
model.add(Activation('tanh'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

# convolutional layer to learn and store mid-level features of the bird species
model.add(Conv2D(32, (3, 3), padding="same"))
model.add(Activation('tanh'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

# highest convolutional layer to store complex information about the look of birds
model.add(Conv2D(32, (3, 3), padding='same'))
model.add(Activation('tanh'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

# and finally MLP
model.add(Flatten())
model.add(Dense(32))
model.add(Activation('tanh'))
model.add(Dense(Train_Category_count))
model.add(Activation('softmax'))

model.compile(optimizer = "adam",
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])
```

Obrázok č. 4: Architektúra siete z 2. experimentu



Obrázok č. 5: Model z 2. experimentu je veľmi slabý

```
Epoch 1/5
1825/1825 [==============================] - 1111s 609ms/step - loss: 5.6467 - accuracy: 0.0163 - val_loss: 5.3468 - val_accura
cy: 0.0280
Epoch 2/5
1825/1825 [==============================] - 1116s 611ms/step - loss: 5.2173 - accuracy: 0.0368 - val_loss: 5.0752 - val_accura
cy: 0.0485
Epoch 3/5
1825/1825 [==============================] - 1216s 666ms/step - loss: 4.9528 - accuracy: 0.0580 - val_loss: 4.7350 - val_accura
cy: 0.0765
Epoch 4/5
1825/1825 [==============================] - 1235s 677ms/step - loss: 4.7514 - accuracy: 0.0761 - val_loss: 4.6120 - val_accura
cy: 0.0775
Epoch 5/5
1825/1825 [==============================] - 1180s 647ms/step - loss: 4.6465 - accuracy: 0.0864 - val_loss: 4.4989 - val_accura
cy: 0.0970
Epoch 5: early stopping
```

Obrázok č. 5: Model z 2. experimentu má pri prvých 5 epochách veľmi slabú úspešnosť, no môže vyžadovať dlhšie učenie

Experiment č. 3 - Softmax regresia - Klasifikácia druhov vtákov

Zdrojový kód k experimentu	
Cieľ experimentu	zlepšiť správnosť klasifikácie (*accuracy*) modelu z experimentu č. 1 a č. 2, zlepšenie schopnosti učiť sa
Výsledok experimentu	ponúka doposiaľ najúspešnejšiu klasifikáciu s úspešnosťou 83,83 %
Časová náročnosť trénovania na celej trénovacej množine	viac ako 10 hodín (>37 909 sekúnd trvali len 3 prvé epochy bez výpomoci GPU)

Sieť z experimentu č. 2 dosahovala veľmi nízku správnosť klasifikácie, čo znamená že sa vzhľad vtákov potrebný na rozhodnutie o jeho zaradení do triedy nevedela dobre naučiť. Preto bolo cieľom experimentu pristúpiť k nasledujúcemu posilneniu modelu (použitá aktivačná funkcia zostala ReLU[13]):

- na zvýšenie výpočtovej sily bol rozšírený počet filtrov na prvej konvolučnej vrstve na dvojnásobok, z 32 na 64 konvolučných jadier.
- na zvýšenie výpočtovej sily modelu boli pridané dve nové konvolučné vrstvy, taktiež so 64 filtrami.
- na zlepšenie a zrýchlenie učenia používam techniku normalizácie výstupov z konvolučných vrstiev (batch normalization). Zároveň touto technikou pomáham proti preučeniu modelu, pretože týmto spôsobom upravujem distribúciu výstupov z predchádzajúcej vrstvy čím ďalšej pomáham generalizovať.
- ako preventívnu techniku voči preučeniu aplikujem techniku Dropout na dočasné vynulovanie vplyvu niektorých neurónov. Tu ide skôr o techniku regularizácie na docielenie lepšej schopnosti modelu naučiť sa vzhľad vtákov (a zovšeobecniť informácie) aplikovanú voči preučeniu modelu na tých vrstvách na ktorých sú oproti

[13] Alebo aj Rectified Linear Unit, Upravená lineárna jednotka

iným neúmerne zvýšené hodnoty váh. Takýmito vrstvami v modeli sú napr. 2. a 4. konvolučná vrstva.

- na zníženie citlivosti konvolučných vrstiev používam samostatnú vrstvu pre operáciu Maximum pooling namiesto Average pooling. Operácia Average pooling by rozostrela obrázky vtákov a preto by pre konvolučné vrstvy bolo obtiažnejšie identifikovať hrany farebných pásikov ktoré vtáky na krídlach majú. Dôvodom na použitie týchto operácií[14] je znižovanie závislosti detekcie tvarov tiel vtákov na ich konkrétnom umiestnení na obrázku. Operácia Maxpooling zase konvolučnej vrstve uľahčí identifikáciu farebných tiel vtákov na tmavom pozadí, tak ako je napr. na obrázku nižšie. (6.5. Pooling — Dive into Deep Learning 0.17.5 documentation (d2l.ai))
- ako optimalizačný algoritmus bol použitý Nesterov momentum, ktorého jednou z výhod je oprava chybného kroku pri zostupe podľa gradientu. Nevýhodou SGD je že sa blíži k optimálnym hodnotám parametrov pomalšie (v porovnaní s napr. SGD s momentom) ale zároveň ručí že sa k ním raz dostane[15].

Obrázok č. 5: Operácia Max pooling by mala uľahčiť klasifikovať farebného vtáka na tmavom pozadí

Skúmaná sieť v 3. experimente má 5 konvolučných vrstiev, každá so 64 konvolučnými jadrami. Použitá aktivačná funkcia je Rectified Linear Unit.

```
model = tf.keras.models.Sequential()

# lowest convolutional layer for identification of the edges of birds
# input shape is provided, so no deferred initialization https://d2l.ai/chapter_deep-learning-computation/deferred-init.html
model.add(Conv2D(64, (3, 3), padding='same', input_shape=SHAPE))
model.add(Activation('relu'))
model.add(BatchNormalization())

# convolutional layer to learn and store mid-level features of the bird species
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.35))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(BatchNormalization())

# highest convolutional layer to store complex information about the look of birds
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.35))

model.add(Conv2D(64, (3, 3), padding='same')) #54x54
model.add(Activation('relu'))
model.add(BatchNormalization())

# and finally mlp
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Dense(Train_Category_count))
model.add(Activation('softmax'))
```

Obrázok č. 6: Architektúra siete v 3. experimente

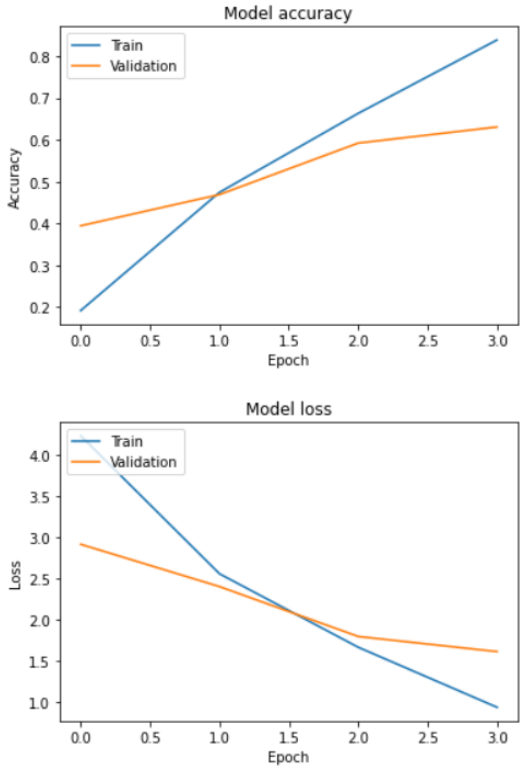
[14] najčastejšie používané sú Maximum pooling, Minimum pooling a Average pooling.
[15] O globálnom minime má zmysel hovoriť skôr pri minimalizácii nákladovej funkcie, pri učení však je často nedosiahnuteľné

Po viac ako 10 hodinách učenia siete a (len!) 4 epochách sa dostávame na akceptovateľnú hodnotu metriky *úspešnosť klasifikácie*, čo znamená že sme schopní správne klasifikovať viac ako 83 % druhov vtákov. Metrika *loss* je chyba klasifikácie na trénovacej množine a metrika val_loss je chyba klasifikácie na validačnej množine. Chyba na trénovacej množine by nemala byť výrazne menšia ako na validačnej aby nedošlo k preučeniu modelu. To sa nám aj podarilo dosiahnuť.

```
Epoch 1/4
1825/1825 [==============================] - 8988s 5s/step - loss: 4.2376 - accuracy: 0.1919 - val_loss: 2.9205 - val_accuracy:
0.3945 - lr: 0.0010
Epoch 2/4
1825/1825 [==============================] - 8630s 5s/step - loss: 2.5619 - accuracy: 0.4742 - val_loss: 2.4064 - val_accuracy:
0.4690 - lr: 0.0010
Epoch 3/4
1825/1825 [==============================] - 20291s 11s/step - loss: 1.6691 - accuracy: 0.6630 - val_loss: 1.8002 - val_accurac
y: 0.5920 - lr: 0.0010
Epoch 4/4
1825/1825 [==============================] - 8941s 5s/step - loss: 0.9398 - accuracy: 0.8383 - val_loss: 1.6167 - val_accuracy:
0.6305 - lr: 0.0010
Model: "sequential"
```

Obrázok č. 7: Výsledky učenia modelu pri spracovaní 3. experimentu sú akceptovateľné. Ide o najlepší model

Sieť sa začína preúčať na trénovacej množine cca. od 2. epochy. V čase medzi 1. a 2. epochou dosahuje rovnakú úspešnosť klasifikácie na trénovacej a validačnej množine. Úspešnosť modelu na testovacích dátach je 65,29 % správnosti predikcií. Úspešnosť klasifikácie je takmer po celý čas lepšia na trénovacej množine a model treba oslabiť.



Obrázok č. 8: Chyba na trénovacej množine sa strieda s chybou na validačnej. Nejde o výrazné preučenie

Experiment č. 4 - Softmax regresia - Klasifikácia druhov vtákov

Zdrojový kód k experimentu	
Cieľ experimentu	zovšeobecniť naučenú informáciu modelom z experimentu č. 2
Výsledok experimentu	
Časová náročnosť trénovania na celej trénovacej množine	

Cieľom tohto experimentu bolo mierne oslabiť a zovšeobecniť informáciu naučenú modelom z 1. experimentu. Na tento účel boli vybrané nasledovné regularizačné techniky:

- normalizácia výstupov z konvolučných vrstiev - vždy po aplikácii max poolingu na predprípravu vstupu pre ďalšiu vrstvu vo vyššej kvalite, čo robí úpravou distribúcie dát, na tie s nulovou strednou hodnotou a jednotkovým rozptylom.
- dočasné vynulovanie niektorých neurónov cez techniku dropout.
- použitie najefektívnejšej optimalizačného algoritmu Adam: [Adam: A Method for Stochastic Optimization (arxiv.org)]

Poznámky k implementácii

Konvolučné neurónové siete opísané v tomto dokumente boli spúšťané na zariadení s týmito vlastnosťami:

Zariadenie	HP Pavilion Gaming Laptop 15
CPU	AMD Ryzen 5 5600H so 6 jadrami
Rámec pre CNN	TensorFlow 2.8.0, Keras 2.8.0
GPU	NVIDIA GeForce RTX 3050 Ti Laptop GPU, architektúra 8.6
Inštalovaný NVIDIA ovládač GPU	GeForce Game Ready Driver pre NVIDIA GeForce RTX 3050 Ti a MS Windows 11, verzia 512.15
Podpora pre TensorFlow GPU	nie, TensorFlow podporuje CUDA 11.2, ktorá nepodporuje Windows 11 https://www.tensorflow.org/install/gpu#software_requirements
CPU Optimalizácia	cez oneAPI Deep Neural Network Library a inštrukčný set AVX AVX2