Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Použití konvoluční neuronové sítě pro klasifikaci snímků ptačího peří

Kateřina Zárybnická

© 2021/2022 ČZU v Praze

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Kateřina Zárybnická

Informatika

Název práce

Použití konvoluční neuronové sítě pro klasifikaci snímků ptačího peří

Název anglicky

Use of convolutional neural network for bird feathers classification

---

Cíle práce

Bakalářská práce je tematicky zaměřena na umělé neuronové sítě a jejich použití pro klasifikaci. Hlavním cílem práce je vytvoření a optimalizace neuronové sítě, která bude schopna klasifikovat snímky ptačího peří.
Dílčí cíle práce jsou:
- vymezit pojmy strojové učení a umělá neuronová síť,
- rozdělit data pro proces učení sítě, testování a jejich validaci,
- trénovat model a vymezit jeho rychlost učení, vhodnou velikost dávek a počet epoch.

Metodika

Teoretická část bakalářské práce se bude zakládat na studiu a analýze odborných informačních zdrojů, budou zde vymezeny základní pojmy. Praktická část se bude skládat z hledání vhodného modelu umělé neuronové sítě a jeho tréninku, na základě analýzy výsledků bude síť optimalizována. Výstupem praktické části bude naučená neuronová síť schopná klasifikace daných snímků. Syntézou teoretických poznatků a výsledků praktické části budou formulovány závěry bakalářské práce.

Doporučený rozsah práce

40 až 50 stran.

Klíčová slova

neuronová síť, strojové učení, knihovny keras, klasifikace, ptačí peří

Doporučené zdroje informací

AGGARWAL, Charu C., [2018]. Neural networks and deep learning: a textbook. Cham: Springer. ISBN 978-3-319-94462-3.

CHOLLET, François, 2019. Deep learning v jazyku Python: knihovny Keras, Tensorflow. Praha: Grada Publishing. Knihovna programátora (Grada). ISBN 978-80-247-3100-1.

MALÝ, Marek, 2007. Vícevrstvé dopředné neuronové sítě: úvod do teorie a aplikací. Ústí nad Labem: Univerzita J.E. Purkyně, Přírodovědecká fakulta. ISBN 978-80-7044-915-8.

MARČEK, Dušan, 2016. Supervizované a nesupervizované učení z dat: statistický a soft přístup. Ostrava: Vysoká škola báňská – Technická univerzita, Ekonomická fakulta. ISBN 978-80-248-3884-7.

TUČKOVÁ, Jana, Marek BÁRTŮ a Petr ZETOCHA, 2009. Aplikace umělých neuronových sítí při zpracování signálů. V Praze: České vysoké učení technické. ISBN 978-80-01-04400-1.

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

Ing. Michal Stočes, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 17. 8. 2021

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 5. 10. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 28. 02. 2022

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Použití konvoluční neuronové sítě pro klasifikaci snímků ptačího peří" jsem vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autorka uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušila autorská práva třetích osob.

V Praze dne 14.3.2022 _____

Poděkování

Ráda bych touto cestou poděkovala Ing. Michalu Stočesovi, Ph. D. za pomoc, cenné rady a přínosné připomínky během vedení této bakalářské práce.

Použití konvoluční neuronové sítě pro klasifikaci snímků ptačího peří

Abstrakt

Bakalářská práce je tematicky zaměřena na umělé neuronové sítě, z velké části se věnuje konvolučním neuronovým sítím a jejich schopnosti správně rozpoznat a zařadit fotografie ptačích per. Práce je rozdělena na teoretickou a praktickou část. V teoretické části dojde k vymezení pojmů strojové učení, umělá neuronová síť, velký prostor je věnován konvolučním neuronovým sítím. Krátce se také teorie věnuje ptačímu peří. Dále jsou zde charakterizovány použité knihovny Keras, TensorFlow a další technologie. Druhá část práce, tedy ta praktická, se zabývá trénováním konvoluční neuronové sítě a hledáním optimálního řešení, které bude schopno klasifikace snímků s minimálně 90% přesností na testovací množině dat. Bude zde provedeno několik experimentů, na základě analýzy výsledku každého z nich budou nastaveny vstupní parametry pro další experiment. První experiment bude mít parametry zvoleny náhodně na základě studia odborné literatury. Na konci práce jsou zhodnoceny výsledky, je analyzován finální model umělé neuronové sítě a je zde stanoven závěr.

Klíčová slova: neuronová síť, strojové učení, konvoluční neuronová síť, Keras, TensorFlow, klasifikace, peří

Use od convolutional neural network for bird feathers classification

Abstract

The bachelor thesis is focused on artificial neural networks, deals with convolutional neural networks and their ability to correctly recognize and classify photos of bird feathers. The theoretical part defines the concepts of machine learning and artificial neural network. Big part of this work is devoted to convolutional neural networks. The theory also briefly deals with bird feathers. The Keras and TensorFlow libraries and other technologies are described here. The second part of the work deals with the training of convolutional neural netrork and search for an optimal solution that will be able to classify images with 90 % or higher accuracy for the test data. Several experiments will be performed and inputs for every following experiment will be set based on the analysed results of the previous ones. The first experiments will have inputs chosen randomly, based on recommendations in the literature. In final part of this work the results are described, the final model of neural network is analyzed here and the conclusion is drawn.

Keywords: neural network, machine learning, convolutional neural network, Keras, TensorFlow, classification, feather

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Bakalářská práce je věnována konvolučním neuronovým sítím, tvorbě modelu umělé neuronové sítě, jeho učení a následně jsou výsledky i průběh učení analyzovány. Z nich jsou vyvozeny důsledky pro další trénink sítě, který pokračuje až do nalezení optimálního řešení. Cíl stanovený pro tuto práci je vytvoření takového modelu, který bude schopen klasifikace na minimálně 90 % přesnosti na testovací množině dat.

Strojové učení je oblast, která v posledních letech zažívá obrovský vzestup i přes to, že jeho základy byly položeny již v druhé polovině dvacátého století. Téměř každý se s ním denně setkává. Současné mobilní telefony jsou odemykány rozpoznáním tváře či otisku prstu, automobily jedoucí bez řidiče již také nejsou jen budoucností. Pro rozpoznání neznámé rostliny už málokdo sáhne po knize, nyní je snazší jen rostlinu vyfotit do aplikace v mobilním telefonu a správná odpověď je známa během několika vteřin. To vše by nebylo možné bez strojového učení.

Rozpoznávání objektů mají obvykle na starost umělé neuronové sítě. Taková síť je matematickým modelem biologického neuronu, má i podobnou funkci. Neuronové sítě jsou součástí počítačového vidění.

Tato práce zkoumá konvoluční neuronové sítě, což jsou sítě využívané primárně pro klasifikaci obrazových dat. V průběhu několika experimentů budou zkoumány vlivy změn hyperparametrů na proces učení až do nalezení optimálního řešení. Budou rozebrány možné problémy během učení, jako je nedoučení nebo přeučení sítě. Učení bude probíhat v prostředí Colab s využitím jazyka Python a knihoven Keras a TensorFlow. Práce je strukturována do teoretické a praktické části. V části teoretické dojde k vymezení základních pojmů souvisejících se strojovým učením a s neuronovými sítěmi. V praktické části dojde k tvorbě modelu neuronové sítě, jsou zde popsány jednotlivé experimenty. Výsledkem bude analýza finálního modelu, tedy takového modelu, který bude schopen splnit stanovený cíl. Na základě teoretických i praktických poznatků je stanoven závěr této práce.

2 Cíl práce a metodika

2.1 Cíl práce

Bakalářská práce je tematicky zaměřena na umělé neuronové sítě a jejich použití pro klasifikaci. Hlavním cílem práce je vytvoření a optimalizace neuronové sítě, která bude schopna klasifikovat snímky ptačího peří, a to s přesností minimálně 90 % na testovací množině dat.

Dílčí cíle práce jsou:

- vymezit pojmy strojové učení a umělá neuronová síť,
- rozdělit data pro proces učení sítě, testování a jejich validaci,
- trénovat model a vymezit jeho rychlost učení, vhodnou velikost dávek a počet epoch.

2.2 Metodika

Teoretická část bakalářské práce se bude zakládat na studiu a analýze odborných informačních zdrojů, budou zde vymezeny základní pojmy. Praktická část se bude skládat z hledání vhodného modelu umělé neuronové sítě a jeho tréninku, na základě analýzy výsledků bude síť optimalizována. Výstupem praktické části bude naučená neuronová síť schopná klasifikace daných snímků. Syntézou teoretických poznatků a výsledků praktické části budou formulovány závěry bakalářské práce.

3 Teoretická východiska

V následujících kapitolách jsou vymezeny pojmy související se strojovým učením, neuronovými sítěmi a také je zde představena základní teorie o ptačím peří. Poslední kapitola teoretické části práce je věnována použitým technologiím nezbytných pro trénink modelu umělé neuronové sítě.

3.1 Strojové učení

Strojové učení spadá do oblasti umělé inteligence. Je to vědní disciplína zabývající se algoritmy, které způsobí takovou změnu vnitřního stavu systému, že se zefektivní schopnost přizpůsobení se změnám okolního prostředí. Zjednodušeně lze říci, že strojové učení je o přizpůsobení se stroje novým situacím, tedy o jeho schopnosti učit se. Učením se rozumí automatické zlepšování se na základě zkušeností.

Samotný pojem strojové učení poprvé použil Arthur Lee Samuel z IBM. Přišel s ním v roce 1959, když pracoval v oblasti počítačových her. *(Hook, & Norman, 2002)* Strojové učení se prolíná a vzájemně doplňuje s dalšími vědními disciplínami, například s matematikou i statistikou, lze ho dosadit do celé řady odvětví, jako je bankovnictví, zdravotnictví, doprava, služby zákazníkům, marketing či zemědělství. Strojové učení má mnoho možností uplatnění, mezi jeho hlavní přínosy patří schopnost automatizovat a urychlit rozhodování a zkrátit dobu k získání prospěchu. Strojový učící se systém není ani tak naprogramován, jako je spíše natrénován. *(Chollet, 2019)*

Princip fungování strojového učení lze zjednodušeně shrnout do čtyř bodů. Prvním je shromažďování a příprava dat, kdy algoritmus identifikuje zdroje a na základě sestavených dat vytvoří struktury. Data se rozdělí do dvou segmentů – tréninkového a testovacího. Dalším bodem je trénování modelu. S využitím tréninkové sady dat dochází k vyladění na maximální rychlost a přesnost zpracování. Třetím bodem je ověření modelu, k tomu dochází za pomoci testovací sady, která vyhodnocuje efektivitu algoritmu. Poslední bod lze nazvat interpretace výsledků. Ty jsou vyhodnoceny ve chvíli, kdy je algoritmus dostatečně odladěn. *(Koďousková, 2021)*

3.1.1 Základní algoritmy strojového učení

Algoritmy strojového učení lze podle způsobu učení rozdělit do několika kategorií. Strojové učení, podobně jako klasická pedagogika, pro získání znalostí využívá rozličné metody, jako je učení se zapamatováním, učení se instrukcí, učení se z analogie, učení se na základě vysvětlení, učení se z příkladů, nebo učení se pozorováním a objevováním. Tyto metody mohou být analogické ke skupinové, individuální, prezenční nebo distanční výuce. *(Koďousková, 2021)*

Učení s učitelem, anglicky nazýváno supervised learning, je takový způsob učení, kdy učitel plní funkci průvodce, který učí algoritmus, jaké závěry má učinit. Při učení s učitelem je algoritmus trénován datovou sadou, která je již označena a má předdefinovaný výstup. Tento přístup využívá bohaté datové sady, které ukazují varianty objektu tak, aby je následně stroj mohl sám rozeznat a provést s nimi danou akci. Mezi příklady strojového učení s učitelem patří algoritmy, jako je lineární a logistická regrese, klasifikace s více třídami a podpůrné vektory. *(Koďousková, 2021)*

Učení bez učitele, anglicky nazýváno unsupervised learning, využívá nezávislejší přístup, ve kterém se počítač učí rozpoznat složité procesy a vzorce bez toho, aby mu člověk poskytoval bližší trvalé vedení. Strojové učení bez učitele zahrnuje školení založené na neoznačených datech nebo na konkrétním, definovaném výstupu. (Olej, & Hájek, 2010) Tento způsob strojového učení tak hledá podobnosti a spojitosti mezi jednotlivými objekty a přiřazuje jim vlastní označení. Mezi příklady algoritmů strojového učení bez učitele patří shluková analýza s k-průměry, analýza hlavních a nezávislých komponentů a pravidla přidružení.

Kombinované učení je takové učení, kdy se vyskytují oba výše zmíněné způsoby učení – tedy učení s učitelem i bez něj. V praxi to znamená, že některá ze vstupních dat mohou postrádat označení. V literatuře se také lze setkat s anglickým názvem semisupervised learning.

Zpětnovazebné učení neboli reinforcement learning, či učení posilováním, je takový způsob učení, kdy počítačový program nahrazuje lidskou obsluhu a pomáhá určit výsledek na základě smyčky zpětné vazby. Algoritmus hledá optimální řešení, ke kterému získává

hodnocení úspěšnosti procesu. Postupy, které algoritmus schválí, jsou pak využívány i při řešení dalších podobných problémů. (Koďousková, 2021) Tento druh učení je vhodný pro situace, kdy je zpětná vazba o dobrých či špatných rozhodnutích k dispozici se zpožděním. Příkladem pro použití mohou být hry, kdy je o výsledku rozhodnuto až na konci.

3.1.2 Základní druhy úloh

Strojové učení je postaveno na analýze dat. Z toho vyplývá, že čím víc dat má počítač k dispozici, tím efektivnější bude jejich vyhodnocení. Stejně tak se účinnost zvyšuje s opakováním jednotlivých algoritmů, kdy jsou data postupně optimalizována. V praxi strojové učení zpracovává data čtyřmi různými způsoby – klasifikací, regresí, shlukováním nebo asociací. Při klasifikaci jsou data rozpoznány a rozděleny do příslušných tříd. Regrese data analyzuje a výstupu jsou přiřazeny hodnoty na základě vstupu. Při shlukování dochází k seskupování dat do skupin na základě podobnosti, nicméně obsah je neznámý. Asociace utváří mezi daty závislost a poté je skládá do ucelených konceptů. (Koďousková, 2021)

3.2 Podoblasti strojového učení

Strojové učení zahrnuje vytvoření modelu, který je trénován na tréninkových datech a poté může model zpracovat doplňková data, na základě kterých vytváří predikce. Mezi používané modely patří rozhodovací stromy, algoritmus k-nejbližších sousedů, podpůrné vektory, lineární diskriminační analýza, kvadratická diskriminační analýza, Bayersovské sítě a nebo právě neuronové sítě, které budou dále rozebrány v následující kapitole této práce.

Jednou z podoblastí strojového učení, který dále rozšiřuje jeho možnosti, je také hluboké učení, neboli deep learning. Jedná se o specifickou oblast, která ke své činnosti využívá především umělé neuronové sítě. V knize Deep learning, jejímž autorem je Francoise Chollet se lze dočíst následující definici:

„Hluboké učení je nový přístup k učení se reprezentací z dat, který klade důraz na učení se následujících vrstev stále smysluplnějších reprezentací. Hloubka není odkazem

na jakýkoli druh hlubšího porozumění dosaženého tímto přístupem; spíše to znamená myšlenku postupných vrstev reprezentací. „ (Chollet, 2019)
Příkladem využití hlubokého učení v praxi mohou být překlady textů nebo automatické odpovědi na email.

3.3 Neuronová síť

Neuronová síť je řada algoritmů, které se snaží rozpoznat základní vztahy v souboru dat prostřednictvím procesu, který imituje způsob fungování lidského mozku, a to hned ve dvou aspektech – znalosti jsou získávány během učení neuronové sítě a synapse jsou využívány pro ukládání znalostí. *(Olej, & Hájek, 2010)*

Neuronové sítě se mohou přizpůsobit měnícímu se vstupu, takže síť generuje nejlepší možný výsledek bez toho, aniž by bylo nutné přepracovat výstupní kritéria. Jak již bylo zmíněno v úvodu této práce, jsou inspirovány poznatky o biologických neuronech a neuronových sítích živých organismů a jejich schopnostmi získávat a reprezentovat závislosti v datech, učit se a naučené znalosti zevšeobecňovat na neznámých vstupech. *(Olej, & Hájek, 2010)* Od ostatních technik se liší především tzv. adaptační fází. *(Malý, 2007)* Využívají se pro klasifikaci, regresi nebo predikci časových řad.

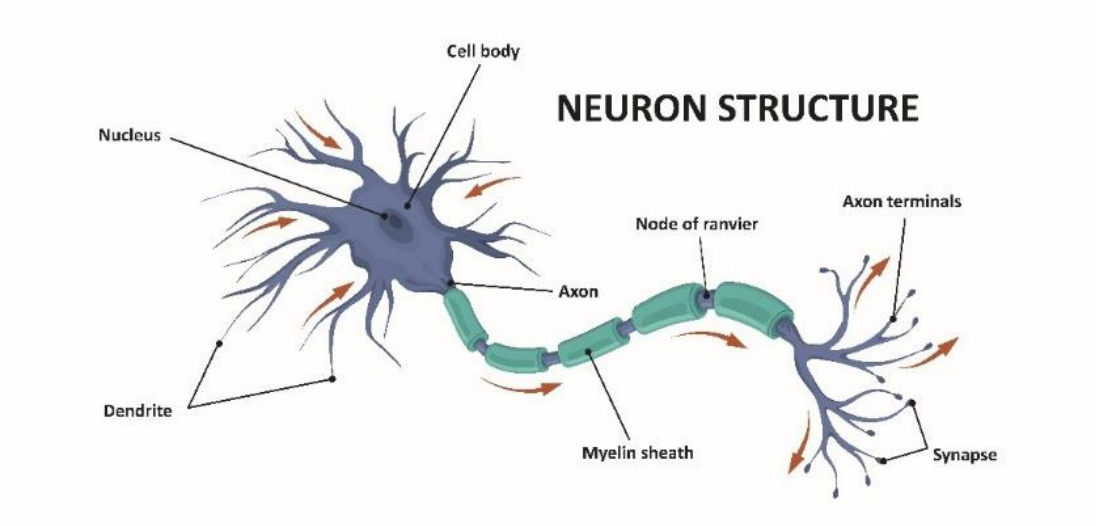
Síť je složena z elementárních výpočetních jednotek, které se nazývají neurony. Ty jsou uspořádány do vrstev. *(Malý, 2007)* Těchto vrstev mohou neuronové sítě mít několik, nebo se mohou skládat jen z jedné vrstvy. Dělí se na sítě jednovrstvé a vícevrstevné. Nejjednodušším modelem jednovrstevné neuronové sítě je perceptron, který se skládá pouze z jediného neuronu. Tento model byl vytvořen Frankem Rossenblattem v roce 1958. *(Rossenblatt, 1958)*

Rozlišují se dva základní druhy struktur neuronových sítí. Prvním typem je síť dopředná, v níž se signál šíří po synapsích jedním směrem – dopředu. Druhým typem je síť rekurentní, v níž existuje skupina neuronů zapojených v kruhu a má synapse orientované různými směry. V rekurentní neuronové síti může být neuron současně vstupní i výstupní, takový neuron je nazýván duálním neuronem. *(Malý, 2007)*

3.3.1 Biologický neuron

Inspirace ke vzniku neuronových sítí přišla z oblasti biologických systémů. Biologický neuron je základním stavebním prvkem nervové soustavy. Hlavní funkcí těchto nervových buněk je přenos, zpracování a uchování informací nutných pro realizaci životních funkcí organismu. Skládá se z těla (soma) a přenosových kanálů – vstupních (dendrity) a výstupních (axony). Zatímco dendritů, které přijímají podněty a dále vedou vzruchy do somatu buňky, má každá buňka několik, axon je pouze jeden. Axony jsou zakončeny výběžky (terminály), které se napojují pomocí synapse na vstupní přenosové kanály dalších neuronů, čímž se přenáší informace z jednoho neuronu na druhý. *(Čihák, 2011-2016)*

Obrázek 1: Struktura biologického neuronu



(„Neuron Structure", 2003-2022)

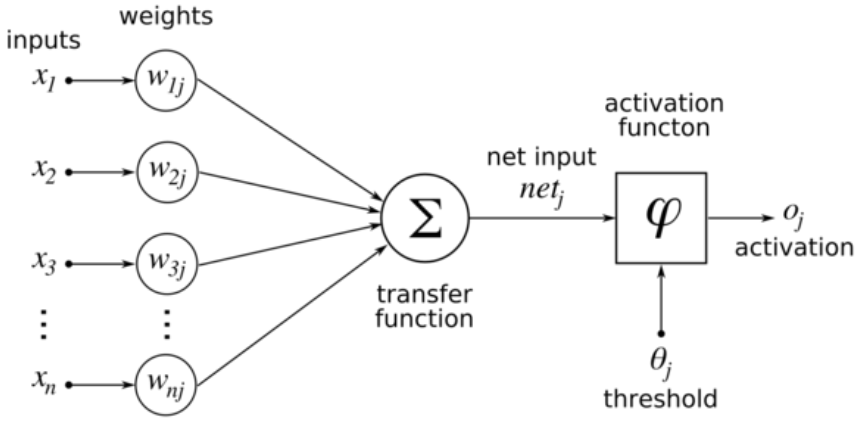
3.3.2 Matematický model neuronu

Matematický model neuronu je základní jednotkou matematického modelu neuronové sítě. Formální model, také označován jako základní model, nebo McCulloch Pittsův model, popsali neurovědec Warren McCulloch a logik v oblasti výpočetní neurovědy Walter Pitts v roce 1943. *(McCulloch, & Pitts, 1943)*

Tento model je analogií biologického neuronu, kdy vstupy modelují dendrity, váhy modelují synapse, výstup zas činnost axonu. Každý neuron obsahuje několik vstupů a pouze jediný výstup y. Vstupy neuronů představují buď výstupy jiných (presynaptických) neuronů a nebo podněty z okolí, tedy vstupní vektory. Po každém takovém vstupu přichází v daném

čase informace v podobě reálného čísla x = [x_1, x_2, …, x_n]. Každý spoj vedoucí do neuronu je spojen s dalším reálným číslem w_i, které udává synaptickou váhu tohoto spoje. Tato hodnota reprezentuje citlivost, s jakou příslušný vstup působí na výstup. Váhy mohou nabývat kladných i záporných hodnot, což je vyjádřeno jejich inhibičním charakterem. Každý neuron také obsahuje práh θ, jehož hodnota určuje kdy je neuron aktivní. Je-li hodnota vstupního signálu nižší než hodnota prahová, je na výstupu z neuronu signál odpovídající pasivnímu stavu neuronu. Neuron se stává aktivním, jakmile dojde k překročení prahové hodnoty a tento neuron roste až do maximální hodnoty, která je dána oborem hodnot příslušné aktivační funkce. Na aktuální stav neuronu je nahlíženo jako na dynamický systém, neboť se jeho vlastnosti mění v závislosti na čase. *(Olej, & Hájek, 2010)*

Obrázek 2: Struktura McCulloch-Pittsova modelu



(„Artificial Neuron Model", 2005)

3.3.3 Učení neuronových sítí

Stejně jako u strojového učení, i u neuronových sítí je učící proces rozdělen na učení s učitelem a učení bez učitele.

Pro učení, tedy trénink neuronové sítě je třeba mít dostatek reprezentativních dat uspořádaných do datových setů. Tyto data se rozdělí do třech kategorií na trénovací, validační a testovací množinu, kdy data tréninková a validační slouží pro samotné naučení sítě a s daty pro test se síť při učení vůbec nesetká, jsou použity až naučenou sítí pro ověření úspěšnosti učení. Důležitou součástí procesu učení jsou váhy, ty bývají na začátku nejčastěji

nastaveny na náhodná čísla a síť si je během učení sama neustále upravuje. Proces učení se snaží minimalizovat odchylku (chybu) mezi skutečným a požadovaným výstupem. Každá neuronová síť má jiný algoritmus učení, vesměs jsou to ale iterační procesy.

Pro řešení každé úlohy musí být navržena jedinečná neuronová síť. Tato síť musí být vhodně vybrána, důležitá je struktura sítě, tedy počet vstupů, výstupů, vrstev, skrytých neuronů, typ aktivačních funkcí a tak dále. Také je třeba zvolit vhodný trénovací algoritmus a vyhnout se typickým problémům při tréninku sítě, jako je nedoučení nebo přeučení sítě.

Nedoučení (underfitting) je stav sítě, kdy model není dostatečně schopný zachytit vnitřní strukturu vstupních dat. Tento problém je způsobený nevhodně zvolenou architekturou, nastavením parametrů a nebo nedostatečnou velikostí datové množiny.

Přeučení (overfitting) je stav, kdy síť není schopna generalizace vstupních dat, pouze si zapamatuje jejich detail. To způsobí, že ačkoliv na trénovací množině dat síť dosahuje uspokojivých výsledků, na množině validačních dat má síť výsledky špatné. K přeučení dochází ve chvíli, kdy síť obsahuje příliš velký počet neuronů, když model obsahuje velký počet vstupních parametrů a relativně málo pozorování. *(Uldrich, & Jurczyk, 2014)*

3.3.4 Typy neuronových sítí

Existuje celá řada neuronových sítí, které se liší architekturou a použitými stavebními prvky. Každý typ se hodí pro jinou třídu úloh.

Vícevrstevná perceptronová síť je nejrozšířenější a nejpoužívanější. Nazýváme ji také dopřednou sítí, její vrstvy se skládají z perceptronů. Hodí se jak pro klasifikaci, tak pro regresi. K jejím nevýhodám patří poměrně dlouhá doba učení.

Síť RBF (Radial Basis Function) je síť radiálních jednotek. Má pevný počet vrstev – vstupní, skrytou a výstupní. Obsahuje dva typy neuronů: radiální a perceptronového typu. Váhy v první vrstvě jsou nastavovány pevně na začátku učení, ve druhé vrstvě se postupuje podobně jako u vícevrstevné perceptronové sítě nebo přímo regresí. *(Aggarwal, 2018)*

Kohonenovy neuronové síťe (SOM – self organizing map) obsahují jedinou vrstvu radiálních neuronů, které mohou být uspořádány do tzv. mřížky. Jejich učení probíhá bez učitele. Neurony v této síti jsou formální, jsou bez prahové hodnoty a jejich výstup je ve dvou hodnotách – aktivní (1) nebo neaktivní (0). Princip učení této sítě spočívá v určení vzdálenosti mezi vstupními vzory a vektory souřadnic neuronů v kompetiční vrstvě. Z nich se pak vybere neuron s nejbližším umístěním a jeho poloha a také poloha okolních neuronů se upraví tak, aby byla blíže k danému vzoru. Cílem učení této sítě je přiblížení rozmístění vstupních dat. *(Olej, & Hájek, 2010)*

Hopfieldova síť byla navržena v roce 1982 jako model pro ukládání paměti. Tato síť je neorientovaná, pracuje s bipolárními (binárními) hodnotami vstupů i výstupů a má autoasociativní paměť. Používá se pro řešení optimalizačních problémů.
(Aggarwal, 2018)

Konvoluční neuronová síť, která je hlavním tématem této práce a dále se jí budeme věnovat v následující kapitole.

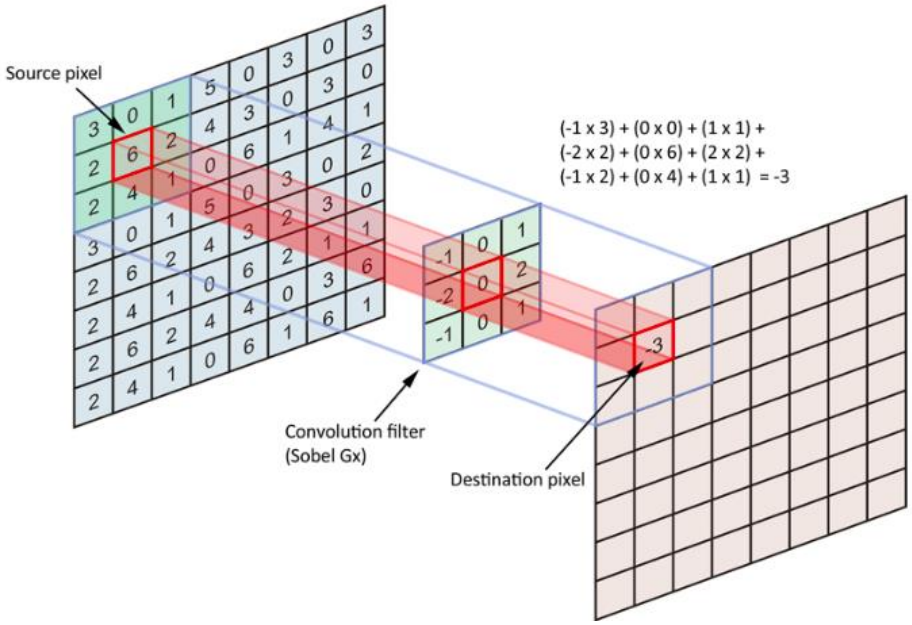
3.3.5 Konvoluční neuronová síť

Konvoluční neuronové sítě jsou biologií inspirované sítě používané v počítačovém vidění pro klasifikaci snímků a detekci objektů. Základní podnět pro konvoluční neuronové sítě byl získán z Hubelova a Wieselova chápání funkce kočičího zrakového centra v kortexu, v němž se zdá, že specifické části zrakového pole excitují konkrétní neurony. Hubel a Wiesel objevili v roce 1959 dva typy buněk ve zrakovém centru mozkové kůry. Nazývají je „simple cell“ a „complex cell“ (jednoduchá a komplexní buňka) a navrhují kaskádový model těchto dvou druhů buněk pro použití při rozpoznávání vzorů. *(Hubel, & Weisel, 1959)*

Konvoluční neuronové sítě se skládají z neuronů uspořádaných do vrstev. Tyto neurony jsou matematickým operátorem, který zpracovává dvě funkce – první funkcí je obrázek, který je zkoumán a druhou funkcí je filtr. *(Procházka, 2021)*

Uvažujeme, že vstupní proměnná x má LxL 2D zdrojových pixelů, které jsou filtrovány s HxH tabulkou vážených hodnot. Konvoluce vezme stejnou velikost hodnot ze zdrojového obrázku a poté vynásobí vážené hodnoty okna HxH filtrovanými hodnotami zdrojových pixelů. Filtrování pokračuje posunutím okna ve zdrojovém obrázku, celý proces je znázorněn na obrázku 3. *(Kimura & Yoshinaga & Sekijima & Azechi, & Baba, 2020)*

Obrázek 3: Konvoluční vrstva sítě s filtrem



(Kimura & Yoshinaga & Sekijima & Azechi, & Baba, 2020)

Konvoluce je tečkovou operací (dot-product operation) mezi mřížkově strukturovanou sadou vah a podobnými mřížkově strukturovanými vstupy čerpanými z různých prostorových lokalit ve vstupním objemu. Tento typ operace je užitečný pro data s vysokou úrovní prostorové lokality, jako jsou obrazová data. Aby mohla být síť nazvána konvoluční, musí používat konvoluci aspoň v jedné své vrstvě, nicméně většina konvolučních neuronových sítí používá tuto operaci hned v několika svých vrstvách. *(Aggarwal, 2018)*

Každá vrstva sítě je schopná rozpoznávat jinak složité obrazce. Spodní vrstva obvykle zachytí elementární prvky, jako jsou vodorovné nebo svislé okraje. Výstup z vrstvy předcházející se pak převádí na vstup vrstvy následující a ta již vyhledává složitější objekty,

jako jsou rohy a hrany. V hlubších vrstvách sítě jsou dále detekovány složitější prvky, jako celé objekty nebo tváře. *(Procházka, 2021)*

Při prvním tréninku každé konvoluční neuronové sítě jsou váhy nastaveny na náhodné hodnoty. Pokud je síť učena s učitelem, musí každý snímek v datovém korpusu obsahovat správně označení. Síť pak jednotlivé snímky zpracuje s původně náhodnými vahami a na základě nesrovnalosti svého výstupu se správným označením snímku své váhy upraví. *(Procházka, 2021)*

V architektuře konvolučních neuronových sítí je každá vrstva sítě trojrozměrná, má prostorový rozsah a hloubku odpovídající počtu prvků. Hloubka jedné vrstvy v konvoluční neuronové síti je odlišná od pojmu hloubka z hlediska počtu vrstev sítě. Ve vstupní vrstvě sítě tyto funkce odpovídají barevným RGB kanálům (tedy červená, zelená, modrá) a ve skrytých kanálech tyto funkce představují mapy skrytých prvků, které v obrázku kódují různé typy tvarů. Pokud je vstup v odstínech šedi, pak bude mít vstupní vrstva hloubku 1, ale další vrstvy budou opět trojrozměrné. Architektura obsahuje dva typy vrstev, které označujeme jako konvoluční a takzvanou subsampling (pooling) vrstvu, tedy vrstvu podvzorkovací. *(Aggarwal, 2018)*

Pro konvoluční vrstvu je definována konvoluční operace, při které filtr mapuje aktivace z jedné vrstvy do druhé. Konvoluční operace používají trojrozměrný filtr vah se stejnou hloubkou jako aktuální vrstva, ale s menším prostorovým rozsahem. Bodový součin mezi všemi váhami ve filtru a kteroukoliv volbou prostorové oblasti o stejné velikosti jako filtr ve vrstvě definuje hodnotu skrytého stavu v další vrstvě. Operace mezi filtrem a prostorovými oblastmi ve vrstvě se provádí na každé možné pozici, aby se definovala další vrstva, ve které si aktivace zachovají své prostorové vztahy z předchozí vrstvy. Spojení konvoluční vrstvy jsou velmi řídká, protože jakákoliv aktivace na v určité vrstvě je funkcí pouze malé prostorové oblasti v předchozí vrstvě. Je tedy možné prostorově vizualizovat jaké části obrazu ovlivňují jednotlivé části aktivací ve vrstvě. Jak již bylo zmíněno výše, prvky ve vrstvách nižší úrovně zachycují čáry nebo jiné primitivní tvary, zatímco prvky ve vrstvách vyšších zachycují složitější tvary, jako třeba smyčky, které se běžně vyskytují v mnoha číslicích. Toto je klasický příklad způsobu, jakým se sémantické poznatky o konkrétních datových doménách používají k návrhu chytrých architektur.

Kromě toho podvzorkovací vrstva jednoduše zprůměruje hodnoty v místních oblastech, zpravidla o velikosti 2x2, aby se prostorové stopy vrstev zkomprimovaly faktorem 2. *(Aggarwal, 2018)*

Konvoluční neuronové sítě byly historicky nejúspěšnější ze všech typů neuronových sítí. Jsou široce používány pro rozpoznávání obrazu, detekci nebo lokalizaci objektů a také pro zpracování textu. Výkon těchto sítí v problému klasifikace obrazu nedávno překonal lidský výkon – to je situace, kterou většina expertů na počítačové vidění před několika desetiletími považovala za nemožnou. *(He & Zhang & Ren, & Sun, 2015)*

Předučené konvoluční sítě jsou běžně dostupné z veřejných zdrojů, jako je například ImageNet a lze je použít v jiných aplikacích a pro jiné soubory dat. Tento fakt je možný díky natrénovaným vahám sítě.

Konvoluční neuronové sítě jsou navrženy tak, aby pracovaly s mřížkově strukturovanými vstupy, které mají silné prostorové závislosti v místních oblastech mřížky. Důležitou vlastností prostorových dat je, že vykazují určitou úroveň translační invariance, což není případ mnoha jiných typů dat s mřížkovou strukturou. Obrázek umístěný vlevo dole bude mít tedy stejnou interpretaci, jako stejný obrázek umístěný na jiné fotografii vpravo dole.

3.4 Ptačí peří

Peří je v říši živočichů jedinečným tělním pokryvem, patří mezi základní morfologickou charakteristiku třídy ptáků. Jedná se o epidermální strukturu, která v evoluci vznikla z plazí šupiny složitou přestavbou. Ptačí peří zastává několik funkcí, jako je ochrana před vlhkostí, umožnění letu a slouží jako tepelný izolant pro udržení tělesné teploty. *(Šťastný & Bejček, & Hudec, 1998)* U některých ptačích druhů má maskovací funkci díky mimezi, hraje také signalizační roli během námluv. Ideálním příkladem pro dvě poslední zmíněné funkce je samec páva korunkatého (Pavo Cristatus), jenž svá typická ocasní pera vějířovitě roztahuje a zvedá vzhůru jak pro výstrahu, tak při toku pro nalákání samice, viz. obrázek 4.

Zbarvení peří je dáno jednak přítomností pigmentů, jednak fyzikálně-optickými jevy na mikrostrukturách paprsků a větví a také kombinací obou možností. Nejčastější pigmenty jsou melaniny (černé, hnědé, tmavožluté) a lipochromy či karotenoidy (žluté, červené), vzácnější jsou porfyriny (zelené, růžové). Strukturní barvy jsou bílá a modrá.
(Gaisler, & Zima, 2007)

Obrázek 4: Páv korunkatý a jeho ocasní pera

(Close-up of a colourful male peacock with spread tail, 2022)

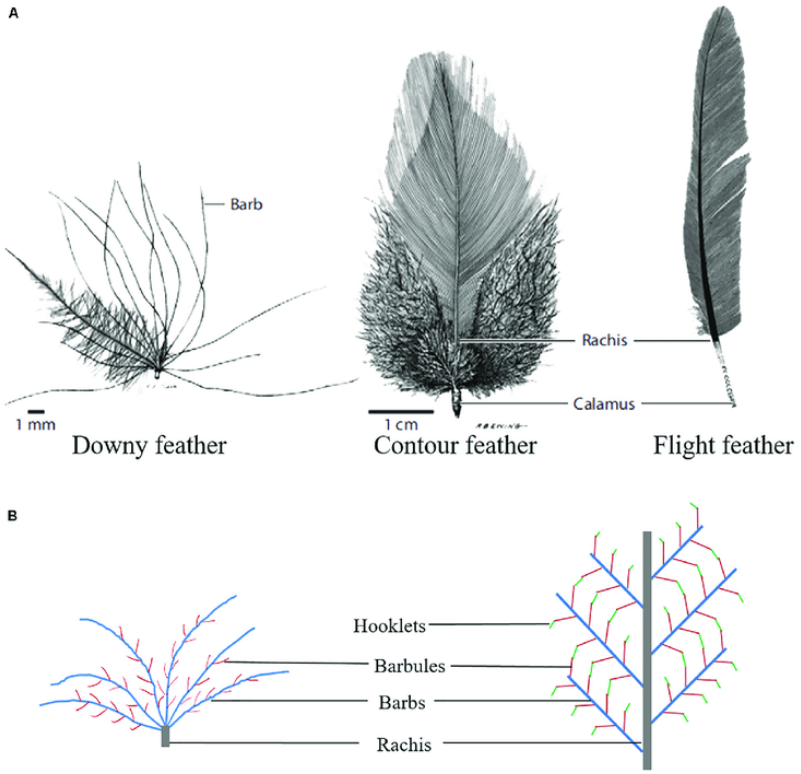
U mnoha druhů ptáků se samec od samice barevně odlišuje, rozdílné zbarvení mohou mít také mládě a dospělý jedinec téhož druhu. Zbarvení a celkový vzhled per se také liší v závislosti na ročním období, dalším faktorem může být také sociální postavení jedince.

Peří vyrůstá z kůže z tzv. péřového váčku v ostrém úhlu, konce per směřují k zadní části těla. Rozlišujeme dva základní typy per: obrysová a prachová. Obrysové pero se skládá z osy a na ní se upínajícího praporu. Vrchní část osy se nazývá osten, spodní, dutá část se nazývá brk. Opěrnou plochu pera tvoří prapor, skládá se z větví rostoucích po obou stranách ostnu a z nich rostoucích paprsků, které jsou vzájemně propojeny háčky – tím vzniká souvislá plocha praporu. Obrysová pera dělíme do třech skupin: pera krycí, která pokrývají hlavu, trup a nohy, známá jsou také jako krovky. Jsou hlavními nositeli zbarvení. Další skupinou jsou letky tvořící nosnou plochu křídel. Poslední skupinou jsou rýdovací pera narůstají na ocasu, ta umožňují ptákům kormidlovat během letu a také zastávají funkci stabilizátoru a brzdy při přistávání. Pro šplhavce jsou tato pera oporou

při pohybu po kmeni stromu. Pro některé druhy ptactva mají význam při pářících rituálech – například ocasní pera páva korunkatého, jak již bylo zmíněno výše.

Prachové pero je tvořeno tenkou osou s dlouhými větvemi, jejichž paprsky nejsou vzájemně propojeny, pero tedy netvoří soudržný povrch. Hlavní funkcí prachového peří je ochrana před ztrátou tělesného tepla. *(Gaisler, & Zima, 2007)*

Obrázek 5: Typy ptačích per



(Chen et al., 2020)

Pro určování ptáků má v ornitologii význam jak barva peří, tak i jejich tvar, proto je důležité věnovat pozornost oběma zmíněným znakům. Zvažuje se také symetričnost praporu, propojenost větví po celé délce a nezanedbatelný je také tvar vrcholu pera. Z tohoto důvodu byly do vstupního datového setu zařazeny fotografie zachycující jak krovky, tak letky. V několika případech byly zařazeny i fotografie per prachových.

3.5 Knihovny a další technologie pro tvorbu modelu

V této kapitole budou představeny knihovny Keras a TensorFlow a také zde budou zhodnoceny nezbytné technologie pro tvorbu modelu umělé neuronové sítě.

3.5.1 Datová množina

Datový korpus je nezbytný pro každý proces učení sítě, výběr vhodných vstupních dat je jeho klíčovou součástí. Jednotlivé druhy zvolených zástupců by měly být dostatečně variabilní, aby byla zvýšena pravděpodobnost, že bude síť dosahovat co nejlepších výsledků. Platí pravidlo, že pokud od sebe jednotlivé druhy nedokáže při důkladném prozkoumání odlišit laik, nedokáže to s největší pravděpodobností ani neuronová síť.

Dalším kritériem při výběru je dostupnost vstupních dat, neboť jich je potřeba velké množství, udává se alespoň 100 fotografií pro jednu klasifikační třídu. *(Mitsa, 2019)*
Je také třeba dbát na splnění licenčních podmínek fotografií a neporušit žádná autorská práva. Dalším kritériem je volba vhodného počtu tříd pro klasifikaci. Čím více tříd zvolíme, tím delší bude celý proces učení sítě. Pro účely této práce bylo zvoleno 12 zástupců (klasifikačních tříd), ačkoliv je dnes celkem známo asi 9 800 druhů ptáků. *(Legris, 2017)*
Datová množina použitá pro trénink modelu v této bakalářské práci bude dále rozebrána v praktické části.

3.5.2 Knihovny

Implementovat každý algoritmus od nuly je náročný úkol, který někdy může trvat týdny i měsíce, obzvlášť při práci s velikými datovými soubory. Pro usnadnění takového úkolu byly vytvořeny knihovny strojového učení. Knihovny jsou souborem pravidel a funkcí, které jsou psány v programovacích jazycích.

Mezi nejpoužívanější knihovny patří například NumPy, Matplotlib, Tensorflow nebo Keras. Všechny výše zmíněné knihovny jsou také použity při tvorbě modelu během psaní této práce.

Keras je modelová knihovna poskytující vysokorychlostní stavební bloky pro vývoj hlubokých učebních modelů. Nepracuje s nízkoúrovňovými operacemi, jako je manipulace s tenzory a derivace, místo toho se spoléhá na specializovanou a dobře optimalizovanou tenzorovou knihovnu, která tak slouží jako podpůrný motor. Místo volby tenzorové knihovny, na níž se bude vázat implementace, řeší problém modulárním způsobem, takže do něj lze bez problémů zapojit několik různých motorů. Keras poskytuje snadné

a rychlé prototypování, jedná se o rozhraní API pro neuronové sítě na vysoké úrovni, které běží na vrcholu TensorFlow. Keras nabízí velké množství již předučených modelů. („About Keras”, 2015)

Tensorflow je softwarová knihovna vyvinuta společností Google, má otevřený zdrojový kód pro strojové učení a umělou inteligenci. Použití má pro celou řadu úkolů, ale zejména se zaměřuje na trénink a odvození hlubokých neuronových sítí. Výchozím programovacím jazykem je Python. TensorFlow plní různé úkoly, včetně optimalizace modelu, grafického znázornění, pravděpodobnostního zdůvodnění nebo statické analýzy. („Why TensorFlow”, 2015)

NumPy je nejzákladnější knihovnou pro zpracování dat, běžně se používá pro vědecké práce v jazyce Python. Umožňuje uživateli zpracovat velké N-rozměrné pole se schopností provádět matematické operace, poskytuje infrastrukturu pro práci s vektory a maticemi. Je známá svou rychlostí běhu, paralelizací a vektorizací.
(„NumPy Introduction”, 2005)

Matplotlib je vykreslovací knihovna pro programovací jazyk Python. Pracuje s NumPy i s dalším interaktivním prostředím, lze ji přizpůsobit tak, aby vykreslila grafy, osy, obrázky nebo publikace. („Matplotlib”, 2021)

3.5.3 Prostředí pro trénink modelu

Poslední zde zmíněnou, ale také nezbytnou součástí procesu učení sítě je prostředí pro tvorbu a trénink modelu. Pro účely této práce bylo vybráno prostředí Google Colaboratory (zkráceně google Colab), které umožňuje psát i spouštět Python v prohlížeči počítače. Pro Colab není nutná konfigurace, umožní po omezenou dobu přístup k GPU a nahrání vstupních dat je možné přímo z Disku Google díky jejich vzájemné kompatibilitě. Záznam již hotových tréninků sítě navíc pak lze na Disk uložit.

Základní verze je bezplatná, placené verze pro a pro+ jsou pro Českou republiku momentálně nedostupné.

3.5.4 Optimizer

Optimizer je algoritmus, který vyhledává optimální parametry neuronové sítě. Dá se říci, že má za úkol zkoušet různé hodnoty parametrů a porovnávat hodnotu chybové funkce při takto nastavených parametrech. Na základě zvětšení či zmenšení chyby optimizer volí směr kterým dále upraví parametry. Různé algoritmy mohou volit různé strategie vedoucí k cíli různou rychlostí. Optimizeru existuje celá řada, z těch nejznámějších lze uvést například Optimizer Adam, Adadelta, SGD nebo Rmsprop. *(Gupta, 2021)*

3.5.5 Hyperparametry

Hyperparametr je parametr, který se nastavuje před zahájením procesu tréninku sítě. Hodnoty těchto parametrů se mohou měnit a tím lze přímo ovlivnit, jak dobře se model vytrénuje. Řadíme mezi ně například velikost dávky, rychlost učení, počet epoch, ale i momentum, regulační konstantu a nebo počet větví v rozhodovacích stromech. *(„Hyperparameter", 2017)*

Velikost dávky je parametr určující kolik dat bude zpracováno před tím, než dojde k úpravě vah. Obvyklé doporučené hodnoty pro menší datasety (tisíce až desetitisíce snímků) jsou 32, 64 nebo 128. *(Brownlee, 2019)*

Rychlost učení je parametr určující velikost kroku, o který se změní nastavení hodnoty jednotlivých vah. Pokud bude rychlost nastavena na příliš malou hodnotu, hrozí riziko, že síť při učení zůstane v lokálním minimu. Pokud bude rychlost příliš velká, může se stát, že algoritmus přeskočí lokální minimum a může dojít ke zvýšení chyb modelu. Existují obecné praktiky při nastavení rychlosti pro každý proces učení, ale je nutné toto nastavení ověřit experimentálně. *(Brownlee, 2020)*

Počet epoch je parametr určující kolikrát učení proběhne na celé tréninkové množině dat, než dojde k ukončení procesu. Při příliš malém počtu epoch hrozí, že síť nebude dostatečně naučena. Příliš velkému počtu epoch se dá předejít použitím funkce early stopping, kdy se celý proces zastaví, pokud po určitou dobu nedojde při učení k žádnému zlepšení. *(„Epoch", 2017)*

3.5.6 Architektura

Ve veřejně dostupných databázích lze nalézt nespočet připravených architektur, na kterých lze trénovat model neuronové sítě. Platí, že čím více vrstev architektura má, tím těžší je trénink sítě. Pro účely této práce byla zvolena architektura EfficientNetB1 pro svoji relativně malou velikost, proces učení. Mezi další hojně používané architektury lze zařadit např. MobileNet, která byla vyvinuta s důrazem na efektivitu a je často využívaná pro modely určené pro mobilní zařízení. Další známou architekturou je ResNet. *(Tumiel, 2016)*

4 Vlastní práce

Tato sekce je druhou částí bakalářské práce. Zabývá se uvedením znalostí získaných z odborné literatury do praxe. Cílem následujících kapitol je tvorba a trénink modelu konvoluční neuronové sítě. Vhodnost výběru hyperparametrů bude experimentálně ověřena a výsledky budou zhodnoceny.

4.1 Příprava prostředí

V této kapitole bude uvedeno jakým způsobem byla předzpracována vstupní data a co obnáší příprava prostředí pro trénování modelu neuronové sítě.

4.1.1 Předzpracování vstupních dat

V teoretické části práce byly popsány obecná pravidla pro datasety. Jako základ datového korpusu byl použit veřejně dostupný soubor fotografií FeathersV1 Dataset. *(Belko & Kuznetsov, & Dobratulin, 2020)* Pro účely této práce z něj bylo vybráno celkem 12 druhů ptáků, všichni zvolení jedinci jsou druhy běžně se vyskytující v České republice. Konkrétně husa velká (Anser Anser), výr velký (Bubo bubo), káně lesní (Buteo buteo), stehlík obecný (Carduelis carduelis), holub hřivnáč (Columba palumbus), havran polní (Corvus frugilegus), labuť velká (Cygnus olor), vlaštovka obecná (Hirundo rustica), zvonek zelený (Chloris chloris), sýkora koňadra (Parus major), čížek lesní (Spinus spinus) a sova pálená (Tyto alba), celkem 1442 snímků. Tento základ byl doplněn o dalších 1479 fotografií z internetové databáze featherbase *(Featherbase, 2015)*, celkem tedy datový set obsahuje 2921 fotografií ptačího peří ve formátu .jpg a .png v různém rozlišení. Na snímcích jsou vyobrazeny jak krovky, tak letky, v menší míře se zde vyskytují také pera prachová. Jednotliví zvolení zástupci mají dostatečné morfologické odlišnosti, tím se zvýší pravděpodobnost, že síť bude dosahovat při klasifikaci dobrých výsledků.

Dalším krokem předzpracování dat je rozdělení dat na tréninkovou, validační a testovací množinu. To je provedeno dle pravidla 70:15:15. *(Draelos, 2019)* Tedy 70 % dat je použito pro samotný trénink sítě, 15 % dat je použito pro validaci a 15 % dat slouží pro ověření úspěšnosti učení.

Jako ukázka morfologických odlišností jednotlivých kategorií ptáků v datasetu je níže zobrazen obrázek 6. Zleva doprava se jedná o tyto druhy: husa velká, výr, káně lesní, stehlík obecný, holub hřivnáč, havran polní, labuť velká, vlaštovka obecná, zvonek zelený, sýkora koňadra, čížek lesní a sova pálená.

Obrázek 6: Ukázka odlišnosti per

(Belko & Kuznetsov, & Dobratulin, 2020)

4.1.2 Příprava prostředí Colab

Google Colaboratory, zkráceně také Colab, je prostředí umožňující psát i spouštět příkazy v jazyce Python v prohlížeči počítače. Podrobněji byl představen v teoretické části práce v kapitole 2.4.3. Pro trénink sítě v této práci byl Colab zvolen hlavně pro svoji dostupnost, mezi další výhody také patří využití přístupu k GPU, i když jen po omezenou dobu. Další výhodou je nahrání vstupních dat za pomoci google Disku.

Základní příprava prostředí je pro všechny experimenty stejná, bude docházet pouze k úpravám hyperparametrů, jako je velikost dávky, počet epoch nebo rychlost učení. Prvním krokem je nahrání již dříve rozdělených vstupních dat z disku google do prostředí Colab. Dále je třeba všem fotografiím nastavit jednotnou velikost, konkrétně rozlišení 224x224 pixelů. Pro ověření správného nahrání dat a pro zobrazení přiřazených číselných označení je využit příkaz train_generator.class_indices, viz. obrázek níže.

Obrázek 7: Ukázka kódu - správné nahrání dat

```
train_generator.class_indices
```

```
Found 2040 images belonging to 12 classes.
Found 436 images belonging to 12 classes.
Found 445 images belonging to 12 classes.
{'anser_anser': 0,
 'bubo_bubo': 1,
 'buteo_buteo': 2,
 'carduelis_carduelis': 3,
 'chloris_chloris': 4,
 'columba_palumbus': 5,
 'corvus_frugilegus': 6,
 'cygnus_olor': 7,
 'hirundo_rustica': 8,
 'parus_major': 9,
 'spinus_spinus': 10,
 'tyto alba': 11}
```

zdroj: autor

Je také nahrána architektura modelu sítě, dále potřebné knihovny a optimizer. Popis těchto funkcí je uveden v teoretické části této práce, viz. kapitoly 2.4.2 a 2.4.4. Nezbytnou součástí každého tréninku je volba vstupních parametrů. Velikost dávky je náhodně zvolena na 32, počet epoch na 200, rychlost učení je pro první experiment ponechána na výchozí hodnotu pro optimizer Adadelta, tedy na hodnotu 1.

Bude proveden první experiment. Na základě analýzy jeho výsledku jsou dále přenastaveny parametry učení a probíhají další experimenty až do dosažení požadovaného cíle, tedy konvoluční neuronová síť se schopností klasifikace s minimálně 90% správností na testovací množině dat. Jednotlivé experimenty budou shrnuty v následujících podkapitolách.

4.2 Průběh experimentů

Jak bylo již zmíněno v předchozí kapitole, pro první experiment byly zvoleny počáteční hodnoty takto: velikost dávky = 32, počet epoch = 200, rychlost učení = 1. Ačkoliv počet epoch je dostatečný, nejlepší výsledek modelu dosahuje 24,02 % úspěšnosti při klasifikaci dat na tréninkové množině, což nesplňuje stanovený cíl, proto nebude výsledek ověřován na testovací množině dat.

Druhý experiment byl spuštěn s následujícími parametry: velikost dávky = 16, počet epoch = 50, rychlost učení = 1. Velikost dávky se tedy snížila z předchozí hodnoty 32 na 16. Změna velikosti opačným směrem, tedy její zvýšení by nebylo vhodné, vzhledem k malé obsáhlosti datasetu používaného v této práci. Počet epoch byl snížen kvůli zkrácení doby potřebné k natrénování modelu. Rychlost učení zatím zůstává beze změny. Výsledek tohoto experimentu ukázal naopak zhoršení výsledků oproti experimentu prvnímu. Nejlepší model dosáhl hodnot 14,56 % na tréninkové množině dat.

Pro třetí experiment byla změněna rychlost učení sítě. Aby se zkrátila doba potřebná pro trénink sítě je přistoupeno ke snížení počtu epoch na 25. Protože v prvním i druhém experimentu byla použita konstantní rychlost učení po celou dobu experimentu a výsledky nebyly dostačující, bylo nyní zvoleno postupné snižování učící rychlosti, a to za pomoci příkazu PiecewiseConstantDecay. Tento příkaz je nastaven tak, aby se síť prvních patnáct epoch (tedy celkem 960 kroků) trénovala se 100% rychlostí, poté se po dobu šesti epoch učící rychlost sníží na 50 % původní rychlosti, po dalších třech epochách se rychlost opět sníží na 30 % původní rychlosti a poslední epochu se síť učí už jen na 10 % své původní rychlosti učení. Trénink sítě proběhl úspěšně, nejlepší model dosahuje schopnosti klasifikace 99,46 % na trénovací množině dat, model je tedy ověřen na testovací množině dat a to za pomoci příkazu model.evaluate().

Obrázek 8: Ukázka kódu - přesnost sítě na testovací množině dat

```
[ ]  print(model.evaluate(x_test, y_test, verbose=1)[1])

     14/14 [==============================] - 4s 316ms/step - loss: 0.1607 - sparse_categorical_accuracy: 0.9708
     0.9707865118980408
```

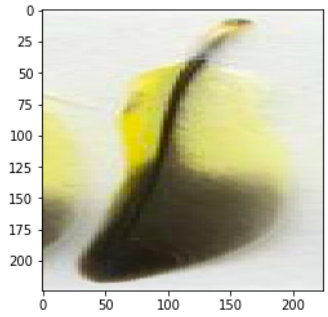
zdroj: autor

Výsledek tohoto příkazu je zobrazen na obrázku výše, zde je patrné, že na testovací množině dat dosáhla síť schopnosti klasifikace s 97,08% úspěšností. Stanovený cíl byl tedy dosažen již při třetím experimentu.

Pro ověření je také zobrazen náhodný snímek z množiny validačních dat a je pozorováno, zda je označení přiřazené sítí správné – zda dokáže trénovaná síť snímek správně klasifikovat. Z deseti náhodně zvolených snímků síť ukázala správné označení u všech deseti.

Obrázek 9: Ukázka kódu - označení snímku určené sítí

```
[ ]  index=44

     plt.imshow(x_val[index])
     plt.show ()

     print('label', y_val[index])

     y=model.predict (np.expand_dims(x_val[index], 0))[0]
     print(y)
     print(np.argmax(y))
```



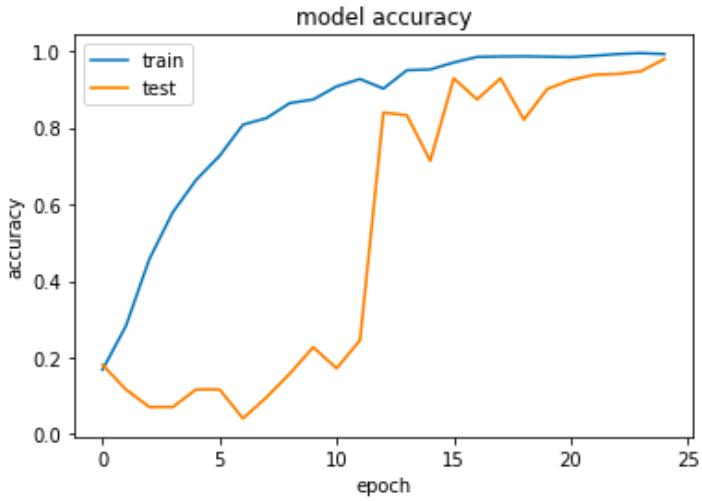

```
label 4.0
[3.3873434e-08 3.3749888e-07 1.5815071e-07 3.5931316e-04 9.9910885e-01
 2.8003674e-10 7.1751403e-11 3.9573528e-11 5.4894230e-07 7.3087875e-08
 5.3070718e-04 2.5853761e-10]
4
```

zdroj: autor

Také jsou za pomoci knihovny Matplotlib vyobrazeny grafy mapující celý průběh učení. Osa X zobrazuje počet epoch, osa Y jak dobře se síť učí. První graf ukazuje přesnost modelu pro trénovací i validační data, druhý graf sleduje ztrátovou funkci. Na základě analýzy grafů je možné vypozorovat některé další vlastnosti učení. Například je zde vidět, že prvních 10 epoch má síť v průběhu učení na tréninkové množině dat jen malé výkyvy, učení tedy probíhá bez problémů. Na validační křivce jsou po dobu prvních 10 epoch patrny velké výkyvy. To je způsobeno tím, že model neměl zpočátku nastaveny žádné váhy a nyní

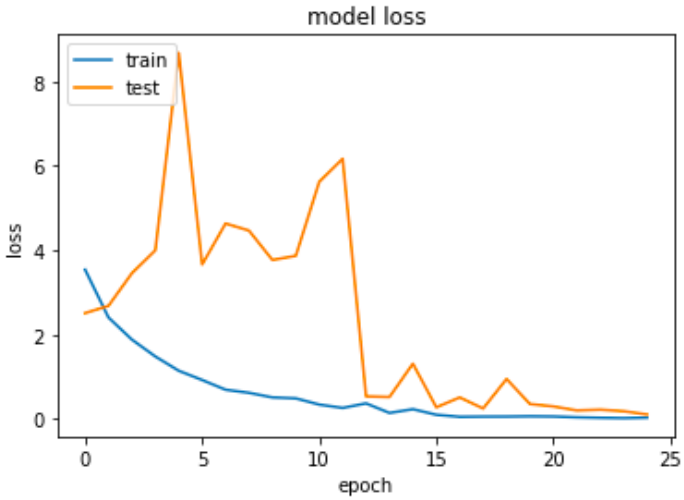
si je během procesu učení tvoří sám. Ke konci učícího procesu jsou křivky u obou grafů téměř souběžné, což je známkou optimálního nastavení parametrů.

Obrázek 10: Graf - přesnost modelu



zdroj: autor

Obrázek 11: Graf - ztrátová funkce



zdroj: autor

Přestože cíl byl splněn, byl spuštěn čtvrtý experiment, jehož cílem bylo zjistit, zda augmentace (náhodné grafické transformace) vstupních dat pozitivně ovlivní výsledek učení sítě. Snímkům v tréninkové množině byly nastaveny následující parametry, které se na snímky aplikují náhodně: vodorovné a svislé překlápění, natočení, přiblížení,

úprava jasu, zkosení a posunutí obrázku po ose X nebo po ose Y. Následuje ukázka kódu pro augmentaci dat.

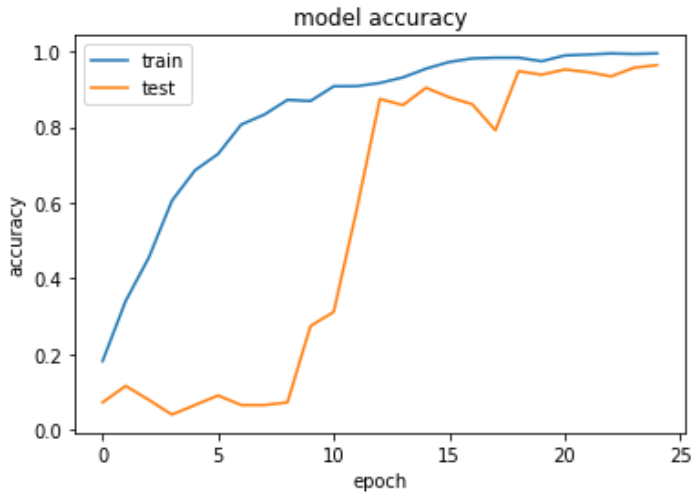
Obrázek 12: Ukázka kódu - augmentace dat

```
[ ]  gen_t = ImageDataGenerator(
         horizontal_flip = True,
         vertical_flip = True,
         rotation_range = 360,
         zoom_range = 0.2,
         brightness_range = [0.5, 1.3],
         shear_range = 0.1,
         width_shift_range = 0.1,
         height_shift_range = 0.1
     )

     gen_v = ImageDataGenerator ()
```

zdroj: autor

Čtvrtý experiment tedy obsahoval augmentaci dat, ostatní vstupní parametry byly ponechány z předchozího experimentu, aby bylo možné pozorovat, jak augmentace ovlivní proces učení. Po dotrénování neuronové sítě má nejlepší model schopnost klasifikace s 99,41% přesností na tréninkové množině dat, na testovací množině je tato přesnost 97,53 %. Pro testovací množinu je tento výsledek o 0,45 % lepší než u experimentu třetího, výsledný graf ukazuje, že celý proces učení probíhal lépe. Křivky u obou grafů ukazují po desáté epoše menší výkyvy a po sedmnácté epoše již nedochází k velkým výkyvům.

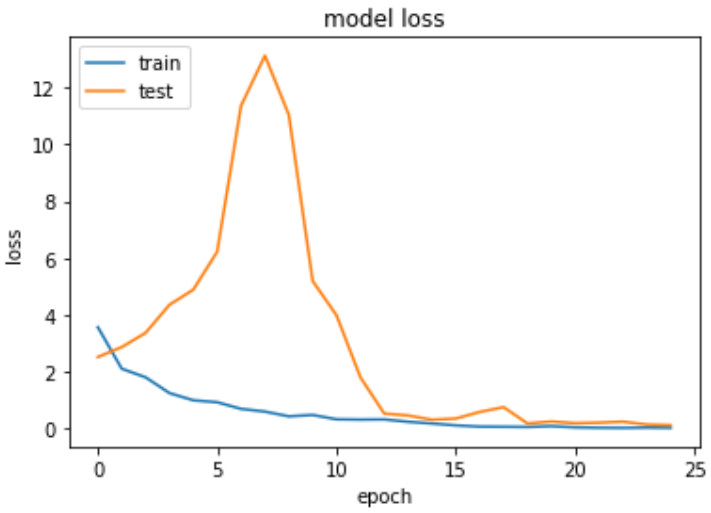
Obrázek 13: Graf - přesnost modelu s augmentací dat



zdroj: autor

Na obrázku č. 13 je zobrazen graf průběhu učení čtvrtého experimentu. Graf na obrázku č. 14 pak zobrazuje jeho ztrátovou funkci.

Obrázek 14: Graf ztrátové funkce modelu s augmentací dat



zdroj: autor

Tento model je tedy také optimální a pro účely této práce bude považován za finální. Podrobněji bude rozebrán v kapitole výsledky.

Proveden byl ještě pátý experiment. Velikost dávky, rychlost učení i počet epoch byl nastaven stejně jako u experimentu čtvrtého. Augmentace byla ale tentokrát nastavena pro stejné parametry (vodorovné a svislé překlápění, natočení, přiblížení, úprava jasu, zkosení a posunutí obrázku po ose X nebo po ose Y), ale na vyšší hodnoty. Výsledky tohoto experimentu ale vykazují mírné zhoršení průběhu učení a výsledný model disponuje nižší rozpoznávací schopností – viz. Tabulka 1. Také je potřeba více času pro trénink sítě. Příliš velká augmentace dat může snímky zkreslit a síť už není při jejich klasifikaci tak úspěšná.

5 Výsledky a diskuse

V této kapitole dojde k vyhodnocení finálního modelu, z analýzy výsledků experimentů budou vyvozeny závěry pro vliv jednotlivých hyperparametrů na proces učení konvoluční neuronové sítě. Také zde budou uvedeny snímky, které se nepovedlo síti správně zařadit a budou analyzovány možné důvody těchto chyb.

5.1 Vyhodnocení finálního modelu

Během čtvrtého experimentu byl vytvořen model, který je schopen klasifikovat s přesností 97,53 % na testovací množině dat, čímž byl splněn stanovený cíl. Tento model je označen za optimální a za finální.

Díky převážně jednolitému pozadí a absenci rušivých elementů na všech snímcích v datasetu nedocházelo během klasifikace k žádným větším potížím se zařazením fotografií, do příslušných kategorií. Další problém by mohl nastat v případě, že by na jedné fotografii bylo vyobrazeno peří od více druhů ptáků. Vzhledem k absenci takové fotografie v datasetu se takovému problému předešlo. Některé snímky obsahovaly více per téhož druhu, ale dle výsledků je síť byla schopna rozpoznat a správně klasifikovat. Také díky augmentaci dat, kdy byly jednotlivé snímky náhodně graficky upraveny (překlopeny, zkoseny, vychýleny z osy apod.), nedocházelo ani k problému, kdy síť nerozpozná obrázek jen proto, že je například převrácen.

Konkrétní příklady snímků, které síť nedokázala správně zařadit, jsou vyobrazeny na následujících fotografiích. Prvním špatně klasifikovaným snímkem bylo pero výra velkého (Bubo Bubo), které síť chybně označila za peří patřící zvonku zelenému (Chloris Chloris). Toto pero bylo výrazně tmavší a užší než pera vyobrazená na ostatních snímcích. Ve chvíli, kdy bylo takto odlišné pero ještě augmentováno, síť jej nedokázala zařadit správně.

Obrázek 15: Špatně klasifikovaný snímek č.1

(Belko & Kuznetsov, & Dobratulin, 2020)

Druhým nesprávně klasifikovaným snímkem bylo pero čížka lesního (Spinus Spinus). Letky i krovky čížka jsou charakteristické díky svým výrazným žlutě zbarveným plochám, zde se ale jednalo o pero pokrývající jinou část těla ptáka (pravděpodobně břicho) a místo žluté plochy má pouze světlý, nevýrazný okraj. Síť tedy toto pero snadno zaměnila za peří sýkory koňadry (Parus Major). Příčinou této chyby tedy může být špatně zvolený snímek v datasetu.

Obrázek 16: Špatně klasifikovaný snímek č. 2

(Belko & Kuznetsov, & Dobratulin, 2020)

Třetím příkladem je záměna peří stehlíka obecného (Carduelis Carduelis) za čížka lesního (Spinus Spinus). Oba druhy disponují výrazně žlutými plochami na perech, navíc oba patří do stejné čeledi pěnkavovití (Fringillidae), rod Carduelis. Jedná se tedy o druhy příbuzné a proto mohou vykazovat stejné morfologické znaky, proto je síť v některých případech zamění.

Obrázek 17: Špatně klasifikovaný snímek č.3

(Belko & Kuznetsov, & Dobratulin, 2020)

Vzhledem k dosažení stanoveného cíle pro tuto bakalářskou práci a také vzhledem k nalezení optimálního řešení již další experimenty, mimo těch popsaných v předchozí

kapitole, neproběhly. Do budoucna je ale možné upravovat různé parametry sítě a dále pozorovat její zlepšení.

Analýza provedených experimentů ukazuje, že augmentace dat je důležitá hlavně pro menší datasety, kdy napomáhá lepšímu průběhu učení neuronové sítě a předcházení přeučení. Také ale bylo experimentem prokázáno, že příliš velká augmentace výkon sítě naopak snižuje.

V prvním a druhém experimentu byla nastavena na konstantní hodnotu, a byly změněny pouze hodnoty parametru velikost dávky a počet epoch. Během těchto experimentů došlo pouze k malému zlepšení, přibližně o 6 %, které poukazuje na malý vliv parametrů pro proces učení neuronové sítě. Je ale nezbytné zmínit, že při použití jiného, různorodějšího a obsáhlejšího datového korpusu by i tyto parametry mohly mít pro trénink signifikantní význam.

Rychlost učení byla pro experimenty v této práci významným faktorem. Analýza výsledků experimentů ukazuje, že po nastavení postupného snižování učící rychlosti síť vykazuje výrazně lepší výsledky (přibližně o 63 %) na konci svého učení než při rychlosti konstantní. Pokud by byla při experimentech zvolena konstantní rychlost nižší, než rychlost výchozí pro optimizer Adadelta, výsledky by mohly být jiné.

Pro přehlednost jednotlivých experimentů a příslušných hyperparametrů byla vytvořena následující tabulka.

Tabulka 1: Přehled výsledků experimentů

Experiment číslo	1	2	3	4	5
Počet epoch	200	50	25	25	25
Velikost dávky	32	16	32	32	32
Rychlost učení	100%	100%	15 epoch: 100% 6 epoch: 50% 3 epochy: 30% 1 epocha: 10%	15 epoch: 100% 6 epoch: 50% 3 epochy: 30% 1 epocha: 10%	15 epoch: 100% 6 epoch: 50% 3 epochy: 30% 1 epocha: 10%
Augmentace dat	ne	ne	ne	ano	ano
Doba učení sítě	100 min	27 min	30 min	15 min	30 min
Přesnost nejlepšího modelu – tréninková data	24,02 %	14,56 %	99,46 %	99,41 %	99,36 %
Přesnost nejlepšího modelu – testovací data	-	-	97,08 %	97,53 %	96,17%

zdroj: autor

Práce na daném tématu by mohla pokračovat experimenty s jinými typy architektur konvolučních neuronových sítí a pozorováním jejich vlivu na výsledném procesu učení. Další možností by bylo zařadit do datového setu i snímky obsahující více druhů ptačích per na jedné fotografii a síť vytrénovat pro jejich detekci. Také by bylo možné stávající dataset rozšířit o další kategorie, aby byla síť schopna rozpoznávat desítky až stovky druhů ptáků, čímž by se zvýšila její využitelnost. Možný rozvoj této sítě by byl její použití jako doplněk již existujících aplikací, které v současnosti rozpoznají pouze snímek celého živočicha.

6 Závěr

V teoretické části práce došlo k vymezení základních pojmů souvisejících se strojovým učením a s tematikou neuronových sítí. Došlo k charakterizování pojmu konvoluční neuronová sít a krátce byly představeny typy ptačích per a jejich funkce.

V praktické části bylo v rámci předzpracování dat připraveno prostředí Google Colab, jehož nezbytnou součástí je nahrání knihoven Keras, TensorFlow, Matplotlib i NumPy a nahrání vstupních dat do prostředí Colab za pomocí Google Disku. Došlo k rozdělení všech snímků do dvanácti kategorií, které odpovídají dvanácti zástupcům ptačí říše, které má síť klasifikovat. Tato data se dále rozdělila na tréninkovou, validační a testovací množinu v poměru 70:15:15 *(Draelos, 2019)* a síť byla za pomoci tohoto datového korpusu úspěšně vytrénována.

Dále byly vymezeny vhodné hyperparametry, mezi které řadíme rychlost učení, počet epoch a velikost dávky. Vzhledem k tomu, že každá umělá neuronová síť se liší a také záleží na použitém datovém korpusu, neexistuje jednotný garantovaný postup pro nalezení optimálního řešení, existují pouze doporučení a praxí ověřené praktiky. Počáteční parametry byly tedy zvoleny náhodně na základě daných doporučení. K zjištění vlivu hodnot těchto parametrů na celý proces učení bylo provedeno několik experimentů a jejich výsledky byly analyzovány. Na základě těchto výsledků byly parametry pro následující experiment poupraveny. Experimentů proběhlo pět, během nich bylo nalezeno optimální řešení – tedy takový model, který je schopen klasifikace s minimálně 90% přesností na testovací množině dat. Finální model má hyperparametry nastaveny takto: velikost dávky = 32, počet epoch = 25, rychlost učení se postupně zmenšovala z původních 100 % až na 10 %. Byla zde také použita augmentace dat.

Finální model dosáhl přesnosti 97,08 % na testovací množině dat a je zahrnut do přílohy této práce.

7 Seznam použitých zdrojů

About Keras. (2015). *Keras*. Retrieved from: https://keras.io/about/

Aggarwal, C. (2018). *Neural Networks and Deep Learning: A Textbook.* (1). Switzerland: Springer.

Artificial Neuron Model. In: *Theoretical & Computational Chemistry*. (2005). Nijmegen: -. Retrieved from: https://www.theochem.ru.nl/~pwormer/Knowino/knowino.org/w/images/thumb/ArtificialNeuronModel_english.png/800px-ArtificialNeuronModel_english.png

Belko, A., Kuznetsov, A., & Dobratulin, K. (2020). Feathers dataset for Fine-Grained Visual Categorization. *CoRR, 2004*(-), pp. -.

Brownlee, J. (2019). How to Control the Stability of Training Neural Networks With the Batch Size. *Machine Learning Mastery*. Retrieved from: https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/

Brownlee, J. (2020). Understand the Impact of Learning Rate on Neural Network Performance. *Machine Learning Mastery*. Retrieved from: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Čihák, R. (2011-2016). *Anatomie*. (Třetí, upravené a doplněné vydání). Praha: Grada.

Draelos, R. (2019). Best Use of Train/Val/Test Splits, with Tips for Medical Data. *Glass Box*. Retrieved from: https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/

Epoch. (2017). *DeepAI*. Retrieved from: https://deepai.org/machine-learning-glossary-and-terms/epoch

Featherbase. (2015). Retrieved from: https://www.featherbase.info/uk/home

Gaisler, J., & Zima, J. (2007). *Zoologie obratlovců.* (Vyd. 2., přeprac). Praha: Academia.

Gupta, A. (2021). A Comprehensive Guide on Deep Learning Optimizers. *https://www.analyticsvidhya.com/*. Retrieved from: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, -(-), pp. 1026-1034.

Hook, D., & Norman, J. (2002). *Origins of Cyberspace: A Library on the History of Computing and Computer-Related Telecommunications (Limited Edition).* (Limited edition). USA: Jeremy Norman Co.

Hyperparameter. (2017). *DeepAI*. Retrieved from: https://deepai.org/machine-learning-glossary-and-terms/hyperparameter

Chen, M., Xie, W., Jiang, S., Wang, X., Yan, H., & Gao, C. (2020). Molecular Signaling and Nutritional Regulation in the Context of Poultry Feather Growth and Regeneration. *Frontiers in Physiology*, *10*(-), pp. 1609.

Chollet, F. (2019). *Deep learning v jazyku Python: knihovny Keras, Tensorflow.* (1. vydání). Praha: Grada Publishing.

Kimura, N., Yoshinaga, I., Sekijima, K., Azechi, I., & Baba, D. (2020). Convolutional Neural Network Coupled with a Transfer-Learning Approach for Time-Series Flood Predictions. *Water*, *12*(1), pp. 96.

Koďousková, B. (2021). Co je strojové učení a jak souvisí s umělou inteligencí?. *Rascasone*. Retrieved from: https://www.rascasone.com/cs/blog/strojove-uceni-ml-metody-klasifikace

Legris, A. (2017). How many different bird species are there?. *Birding World: About Birds*. Retrieved from: http://birding-world.com/many-different-bird-species/

Malý, M. (2007). *Vícevrstvé dopředné neuronové sítě: úvod do teorie a aplikací.* (1. vydání). Ústí nad Labem: Univerzita J.E. Purkyně, Přírodovědecká fakulta.

Matplotlib: Visualization with Python. (2021). *matplotlib*. Retrieved from: https://matplotlib.org/

Mitsa, T. (2019). How Do You Know You Have Enough Training Data?. *https://medium.com/*. Retrieved from: https://towardsdatascience.com/how-do-you-know-you-have-enough-training-data-ad9b1fd679ee

Neuron Structure. In: *shutterstock*. (2003-2022). New York: Shutterstock, Inc.. Retrieved from: https://image.shutterstock.com/image-vector/structure-motor-neuron-anatomy-brain-600w-1507344530.jpg

NumPy Introduction. (2005). *W3Schools*. Retrieved from: https://www.w3schools.com/python/numpy/numpy_intro.asp

Olej, V., & Hájek, P. (2010). *Úvod do umělé inteligence: moderní přístupy : distanční opora*. Pardubice: Univerzita Pardubice.

Procházka, L. (2021). Konvoluční neuronové sítě pomáhají detekovat a blokovat nevhodný obsah sociálních sítí. *toxin*. Retrieved from: https://www.toxin.cz/blog/index.php/blog/konvolu%C4%8Dn%C3%AD-

neuronov%C3%A9-s%C3%ADt%C4%9B-pom%C3%A1haj%C3%AD-detekovat-a-blokovat-nevhodn%C3%BD-obsah-soci%C3%A1ln%C3%ADch-s%C3%ADt%C3%AD

Šťastný, K., Bejček, V., & Hudec, K. (1998). *Ptáci.* (1.). Praha: Albatros.

Tumiel, T. (2016). From MobileNet to EfficientNet. *ttumiel*. Retrieved from: https://ttumiel.github.io/blog/mobilenet-to-efficientnet/

Uldrich, M., & Jurczyk, T. (2014). Neuronové sítě a jejich využití. *IT SYSTEMS*, *2014*(3).

Why TensorFlow. (2015). *TensorFlow*. Retrieved from: https://www.tensorflow.org/about

8 Přílohy

Příloha 1: Finální model neuronové sítě

```
[ ] from google.colab import drive
    drive.mount ('/content/drive/')

    import os

    drive_path = '/content/drive/MyDrive/novadatadoplnena'
    local_path = '/content'
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_remount=True).
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_generator = ImageDataGenerator (rescale = 1./255).flow_from_directory(
    '/content/drive/MyDrive/novadatadoplnena/training',
    target_size=(224, 224),
    batch_size=2040,
    class_mode='binary',
    seed=12)

val_generator = ImageDataGenerator (rescale = 1./255).flow_from_directory(
    '/content/drive/MyDrive/novadatadoplnena/validation',
    target_size=(224, 224),
    batch_size=436,
    class_mode='binary',
    seed=12)

test_generator = ImageDataGenerator (rescale = 1./255).flow_from_directory(
    '/content/drive/MyDrive/novadatadoplnena/test',
    target_size=(224, 224),
    batch_size=445,
    class_mode='binary',
    seed=12)

x_train, y_train = next (train_generator)
x_val, y_val = next (val_generator)
x_test, y_test = next (test_generator)

train_generator.class_indices
```

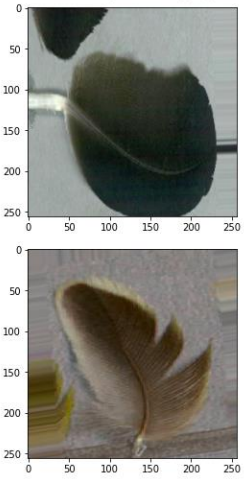
```
Found 2040 images belonging to 12 classes.
Found 436 images belonging to 12 classes.
Found 445 images belonging to 12 classes.
{'anser_anser': 0,
 'bubo_bubo': 1,
 'buteo_buteo': 2,
 'carduelis_carduelis': 3,
 'chloris_chloris': 4,
 'columba_palumbus': 5,
 'corvus_frugilegus': 6,
 'cygnus_olor': 7,
 'hirundo_rustica': 8,
 'parus_major': 9,
 'spinus_spinus': 10,
 'tyto alba': 11}
```

```
[ ] gen_t = ImageDataGenerator(
        horizontal_flip = True,
        vertical_flip = True,
        rotation_range = 360,
        zoom_range = 0.2,
        brightness_range = [0.5, 1.3],
        shear_range = 0.1,
        width_shift_range = 0.1,
        height_shift_range = 0.1
    )

    gen_v = ImageDataGenerator ()
```

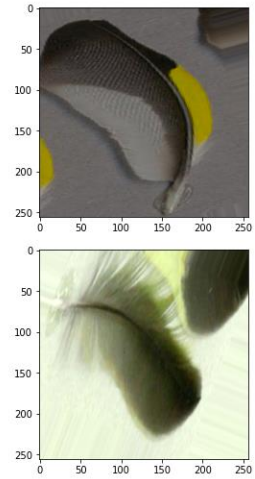
```
[ ] import numpy as np

    from tensorflow.keras.applications.efficientnet import EfficientNetB1 as eff
    from tensorflow.keras.optimizers import Adadelta, schedules

    import matplotlib.pyplot as plt
```

```
[ ] xs, _ = gen_t.flow_from_directory ('/content/drive/MyDrive/novadatadoplnena/training', batch_size=10).next()
    for x in xs:
        plt.imshow (x.astype('int'))
        plt.show()
```

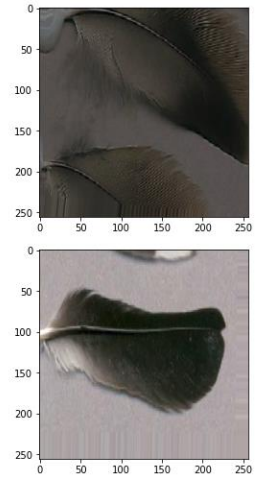
```
Found 2040 images belonging to 12 classes.
```



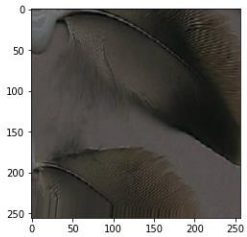
[ ]

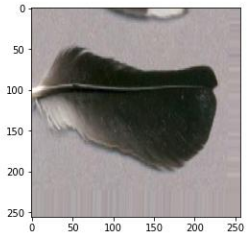


[ ]

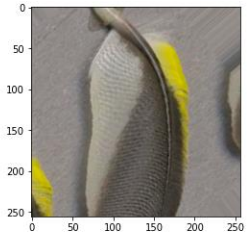


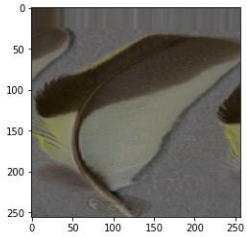
[ ]





[ ]





```
[ ] lr = schedules.PiecewiseConstantDecay([960,1344,1536], [1.0,0.5,0.3,0.1])
```

```
[ ] model = eff(classes=12, weights=None, input_shape=(224, 224, 3), include_top=True)
    model.compile(Adadelta(lr), loss='sparse_categorical_crossentropy', metrics='sparse_categorical_accuracy')
```

```
history = model.fit(x_train, y_train, epochs=25, batch_size=32, validation_data=(x_val, y_val))
```

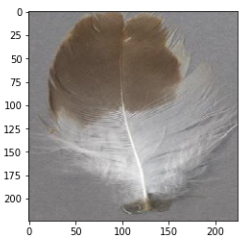
```
Epoch 1/25
64/64 [==============================] - 49s 520ms/step - loss: 3.5636 - sparse_categorical_accuracy: 0.1824 - val_loss:
2.5131 - val_sparse_categorical_accuracy: 0.0734

Epoch 2/25
64/64 [==============================] - 28s 441ms/step - loss: 2.1031 - sparse_categorical_accuracy: 0.3412 - val_loss:
2.8643 - val_sparse_categorical_accuracy: 0.1170

Epoch 3/25
64/64 [==============================] - 28s 445ms/step - loss: 1.8004 - sparse_categorical_accuracy: 0.4554 - val_loss:
3.3650 - val_sparse_categorical_accuracy: 0.0803

Epoch 4/25
64/64 [==============================] - 29s 448ms/step - loss: 1.2451 - sparse_categorical_accuracy: 0.6054 - val_loss:
4.3530 - val_sparse_categorical_accuracy: 0.0413

Epoch 5/25
64/64 [==============================] - 29s 448ms/step - loss: 0.9906 - sparse_categorical_accuracy: 0.6858 - val_loss:
4.8856 - val_sparse_categorical_accuracy: 0.0665

Epoch 6/25
64/64 [==============================] - 29s 453ms/step - loss: 0.9230 - sparse_categorical_accuracy: 0.7289 - val_loss:
6.2223 - val_sparse_categorical_accuracy: 0.0917

Epoch 7/25
64/64 [==============================] - 29s 452ms/step - loss: 0.6895 - sparse_categorical_accuracy: 0.8064 - val_loss:
11.3584 - val_sparse_categorical_accuracy: 0.0665

Epoch 8/25
64/64 [==============================] - 29s 454ms/step - loss: 0.5938 - sparse_categorical_accuracy: 0.8328 - val_loss:
13.1240 - val_sparse_categorical_accuracy: 0.0665

Epoch 9/25
64/64 [==============================] - 29s 457ms/step - loss: 0.4272 - sparse_categorical_accuracy: 0.8716 - val_loss:
11.0414 - val_sparse_categorical_accuracy: 0.0734
```

```
Epoch 10/25
64/64 [==============================] - 29s 456ms/step - loss: 0.4745 - sparse_categorical_accuracy: 0.8686 - val_loss:
5.1916 - val_sparse_categorical_accuracy: 0.2752

Epoch 11/25
64/64 [==============================] - 29s 457ms/step - loss: 0.3215 - sparse_categorical_accuracy: 0.9074 - val_loss:
3.9827 - val_sparse_categorical_accuracy: 0.3119

Epoch 12/25
64/64 [==============================] - 29s 456ms/step - loss: 0.3114 - sparse_categorical_accuracy: 0.9078 - val_loss:
1.8012 - val_sparse_categorical_accuracy: 0.5826

Epoch 13/25
64/64 [==============================] - 29s 455ms/step - loss: 0.3170 - sparse_categorical_accuracy: 0.9162 - val_loss:
0.5236 - val_sparse_categorical_accuracy: 0.8739

Epoch 14/25
64/64 [==============================] - 29s 457ms/step - loss: 0.2349 - sparse_categorical_accuracy: 0.9309 - val_loss:
0.4564 - val_sparse_categorical_accuracy: 0.8578

Epoch 15/25
64/64 [==============================] - 29s 458ms/step - loss: 0.1741 - sparse_categorical_accuracy: 0.9539 - val_loss:
0.3063 - val_sparse_categorical_accuracy: 0.9037

Epoch 16/25
64/64 [==============================] - 29s 460ms/step - loss: 0.1093 - sparse_categorical_accuracy: 0.9716 - val_loss:
0.3487 - val_sparse_categorical_accuracy: 0.8784

Epoch 17/25
64/64 [==============================] - 29s 457ms/step - loss: 0.0645 - sparse_categorical_accuracy: 0.9809 - val_loss:
0.5802 - val_sparse_categorical_accuracy: 0.8601

Epoch 18/25
64/64 [==============================] - 29s 458ms/step - loss: 0.0599 - sparse_categorical_accuracy: 0.9828 - val_loss:
0.7510 - val_sparse_categorical_accuracy: 0.7913

Epoch 19/25
64/64 [==============================] - 29s 457ms/step - loss: 0.0536 - sparse_categorical_accuracy: 0.9828 - val_loss:
0.1758 - val_sparse_categorical_accuracy: 0.9472

Epoch 20/25
64/64 [==============================] - 29s 455ms/step - loss: 0.0810 - sparse_categorical_accuracy: 0.9735 - val_loss:
0.2445 - val_sparse_categorical_accuracy: 0.9381

Epoch 21/25
64/64 [==============================] - 29s 455ms/step - loss: 0.0382 - sparse_categorical_accuracy: 0.9887 - val_loss:
0.1852 - val_sparse_categorical_accuracy: 0.9518

Epoch 22/25
64/64 [==============================] - 29s 457ms/step - loss: 0.0244 - sparse_categorical_accuracy: 0.9912 - val_loss:
0.2039 - val_sparse_categorical_accuracy: 0.9450

Epoch 23/25
64/64 [==============================] - 29s 457ms/step - loss: 0.0196 - sparse_categorical_accuracy: 0.9941 - val_loss:
0.2386 - val_sparse_categorical_accuracy: 0.9335

Epoch 24/25
64/64 [==============================] - 29s 459ms/step - loss: 0.0301 - sparse_categorical_accuracy: 0.9926 - val_loss:
0.1383 - val_sparse_categorical_accuracy: 0.9564

Epoch 25/25
64/64 [==============================] - 29s 457ms/step - loss: 0.0215 - sparse_categorical_accuracy: 0.9941 - val_loss:
0.1162 - val_sparse_categorical_accuracy: 0.9633
```

```
[ ] print(model.evaluate(x_test, y_test, verbose=1)[1])
```

```
14/14 [==============================] - 1s 87ms/step - loss: 0.0870 - sparse_categorical_accuracy: 0.9753
0.9752808809280396
```

```
[ ] index=370


    plt.imshow(x_val[index])
    plt.show ()

    print('label', y_val[index])

    y=model.predict (np.expand_dims(x_val[index], 0))[0]
    print(y)
    print(np.argmax(y))
```

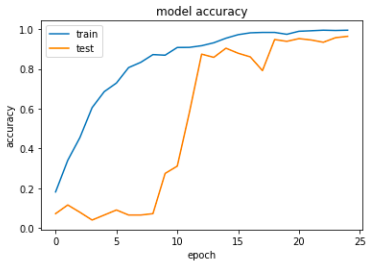



```
label 2.0
[5.6001124e-09 2.7771241e-10 9.9999952e-01 2.0785598e-12 8.6338719e-14
 3.6149149e-07 6.1725729e-11 7.3728046e-08 4.7135795e-10 3.8820553e-15
 2.4385900e-11 1.4025283e-09]
2
```

```
[ ] print(history.history.keys())
```

```
dict_keys(['loss', 'sparse_categorical_accuracy', 'val_loss', 'val_sparse_categorical_accuracy'])
```

```
[ ] plt.plot(history.history['sparse_categorical_accuracy'])
    plt.plot(history.history['val_sparse_categorical_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
```




```
[ ] plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
```