Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Metody hlubokého učení, rozpoznávání obrazu s použitím nástroje Google Colaboratory

Bc. Daniel Charvát

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Daniel Charvát

Systémové inženýrství a informatika
Informatika

Název práce

Metody hlubokého učení, rozpoznávání obrazu s použitím nástroje Google Colaboratory

Název anglicky

Deep learning methods, image recognition with use of Google Colaboratory

---

Cíle práce

Hlavním cílem práce je vytvoření neuronové sítě v nástroji Google Colaboratory. Tato neuronová síť by měla umožnit rozeznávat pomeranče v jednotlivých snímcích obrazu. Dílčím cílem je získání a správná úprava dat, pro učení neuronové sítě.

Metodika

Metodika řešení problematiky diplomové práci je založena na studiu a analýze odborných informačních zdrojů, ale také na praktickém využití jednotlivých nástrojů, pro dosažení cíle diplomové práce. Pomocí této metodiky je vytvořena neuronová síť, schopna rozeznávat pomeranče v obrazu. Na základě teoretických poznatků a vlastního řešení neuronové sítě budou formulovány závěry Diplomové práce.

-Prostudujte současné technologie hlubokého učení neuronových stítí

-Navrhněte vhodnou technologii umělé inteligence použitelnou pro danou úlohu.

-Úlohu vypracujte, stroj natrénujte.

-Proveďte test.

-Definujte závěry.

Doporučený rozsah práce

60-80

Klíčová slova

Konvoluční neuronové sítě, Počítačová vize, YOLO, Strojové učení, Keras, Hluboké učení

Doporučené zdroje informací

Pavlíček, J. ; Jarolímek, J. ; Pavlíčková, P. ; Dvořák, S. ; Pavlík, J. ; Hanzlík, P.: Automated Wildlife Recognition

---

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. Josef Pavlíček, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 4. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 1. 4. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 03. 04. 2020

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Metody hlubokého učení, rozpoznávání obrazu s použitím nástroje Google Colaboratory" jsem vypracoval(a) samostatně pod vedením vedoucího diplomové 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 autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne datum odevzdání _____

Poděkování

Rád bych touto cestou poděkoval své rodině, která mě podporovala nejen při tvorbě této diplomové práce, ale také v celém mém studiu. Dále bych chtěl poděkovat vedoucímu své diplomové práce panu Ing. Josefu Pavlíčkovi Ph.D., za vstřícnou spolupráci, pomoc při řešení problémů s daty v informačním systému, cenné rady a trpělivost při jejím vypracování. Jako poslední bych chtěl poděkovat Ing. Kateřině Pouskové za pomoc s korekcí této diplomové práce.

Metody hlubokého učení, rozpoznávání obrazu s použitím nástroje Google Colaboratory

Abstrakt

Tato diplomová práce se zaobírala neuronovými sítěmi, konkrétně konvolučními neuronovými sítěmi, rozpoznáváním obrazu a možnostmi využití Google Colaboratory, které zdarma poskytuje prostředí pro tvorbu a trénování neuronových sítí.

V rámci teoretické části práce byly podrobně představeny stěžejní prvky, tedy umělá inteligence, význam a princip strojového a hlubokého učení. Ze zjištěných informací vyplynulo, že přestože se využívání neuronových sítí v praxi potýká i s několika dílčími problémy, například s hrozbou přeučení se nebo s dlouhou dobou učení se, má tento typ sítí mnoho předností.

V praktické části bylo představeno prostředí Google Colaboratory a názorně zde byla popsána možnost propojení s prostředím úložiště Google Drive a jeho následné propojení s lokálním počítačem. Dále byl popsán postup získávání a úpravy dat pro učení CNN. Tato data byla využita při procesu učení první CNN. Následně byla zpracována druhá CNN, která byla naučena rozpoznávat pomeranče na datech získaných ze služby Open Images V6. Dosažené výsledky byly v závěru porovnány.

Klíčová slova: konvoluční neuronové sítě, CNN, počítačová vize, YOLOv3, strojové učení, hluboké učení, umělé neurony, Google Colaboratory, Colab

Deep learning methods, image recognition with use of Google Colaboratory

Abstract

This diploma thesis was focused on neural networks, namely convolutional neural networks, image recognition and possibilities of using Google Colaboratory, which provides free environment for creating and training neural networks.

In the theoretical part of the thesis, the main elements, namely artificial intelligence, the meaning and principle of machine and deep learning, were introduced in detail. The findings have shown that although the using of neural networks in practice is faced with several partial problems, such as the threat of overfitting or long time of learning, this type of network has many advantages.

The practical part introduces the Google Colaboratory environment and illustrates the possibility of linking it to the Google Drive storage environment and the subsequent connection to the local computer. Next there was described the procedure for obtaining and modifying data for the CNN learning. This data was used in learning process of the first CNN. Then a second CNN processed, this time was used for the learning process a dataset from Open Images V6. Then the obtained results were compared.

Keywords: convolutional neural networks, CNN, computer vision, YOLOv3, machine learning, deep learning, artificial neurons, Google Colaboratory, Colab

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Za poslední desetiletí se posunuly naše technické a softwarové možnosti mílovými kroky. Díky tomu se rozšířila možnost rozpoznávání obrazu i do každodenních životů většiny lidí, ať už se jedná o detekci obličejů při focení, nebo rozpoznávání obličejů, které se v dnešní době používá při odemykání obrazovky u chytrých telefonů, nebo rozpoznávání znaků a přepis vyfoceného papíru s psaným textem do PDF podoby ve vybraném fontu. Pro potřeby identifikace subjektů z obrazu se v praxi velmi často používá implementace pomocí různých typů neuronových sítí, které jsou k tomuto vhodné díky jejich schopnosti abstrakce a učení se. Téma neuronových sítí je velmi zajímavé a přínosné, a proto si tato práce klade za cíl představit fungování konvolučních neuronových sítí, způsob, jakým se s nimi dá pracovat, a nakonec praktickou ukázku jejich naučení na konkrétním případu.

Tato práce je rozdělena na dvě hlavní části – teoretickou a praktickou. Teoretická část práce je věnována vysvětlení pojmů umělá inteligence, strojové učení a hluboké učení, jejichž správná interpretace je pro text této práce nezbytná. Dále je zde řešena problematika neuronových sítí, jednotlivé architektury a princip jejich fungování. Součástí je také popis fungování umělého neuronu a reprezentace dat v neuronových sítích. Také jsou rozebrány jednotlivé typy a modely neuronových sítí a jejich vlastnosti, princip fungování, architektura a jejich využití. V příslušné podkapitole teoretické části jsou probrány konvoluční neuronové sítě, jejich charakteristika, princip, čím jsou tvořeny, způsob učení se a jejich využití. Jako poslední je v teoretické části krátce představena služba Google Colaboratory, která je k dispozici jako funkční platforma, kterou lze využít pro vytváření vlastních konvolučních sítí s využitím technického zázemí společnosti Google.

V rámci praktické části je pak prostředí Google Colaboratory podrobněji představeno. Tato služba je následně názorně propojena se službou Google Drive. Hlavní a zároveň také nejdůležitější pasáž praktické části je věnována vytvoření funkční konvoluční neuronové sítě, a to právě s využitím služby Google Colaboratory. Nechybí zde také popis způsobu získání potřebných dat a jejich následná úprava do vhodného formátu pro samotný proces učení konvolučních neuronových sítí. Poté je zde detailně popsáno vyhotovení dvou ukázkových konvolučních neuronových sítí pomocí vhodně zvolených nástrojů a knihoven, jejichž naučenou schopností je rozpoznávat v obrazu určený objekt. Pro účely této práce byl vybrán jako rozpoznávaný objekt pomeranč, protože má jasně dané a neměnné rysy, díky

kterým předpokládám, že se jedná o velmi dobrou volbu pro tuto práci.

V poslední kapitole praktické části diplomové práce jsou obě vytvořené sítě popsány, stejně tak jako nástroje použité při jejich tvorbě. V závěru celé práce jsou pak shrnuty dosažené výsledky a způsob dosažení vytyčených cílů.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je vytvoření funkční neuronové sítě v nástroji Google Colaboratory. Tato neuronová síť by měla umožnit rozeznávat pomeranče v jednotlivých snímcích obrazu. Dílčí cíle pak jsou představení prostředí Google Colaboratory a metody získání a správné úpravy dat, pro učení neuronové sítě.

2.2 Metodika

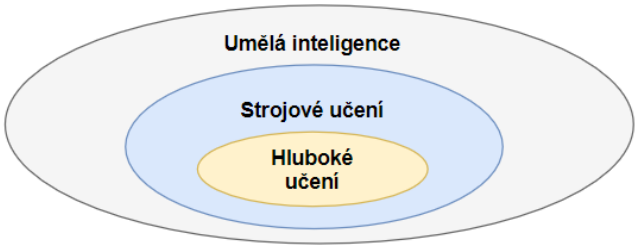
Metodika řešení problematiky diplomové práce je založena nejen na studiu a analýze odborných informačních zdrojů, ale také na praktickém využití jednotlivých nástrojů, vhodných pro dosažení cíle diplomové práce. Pomocí této metodiky je vytvořena neuronová síť schopná rozeznávat pomeranče v obrazu. Na základě teoretických poznatků a vlastního řešení neuronové sítě budou formulovány závěry diplomové práce.

3 Teoretická část

Teoretická část práce se zabývá problematikou hlubokého učení, neuronových sítí a dále je více rozebrán typ konvolučních neuronových sítí, které se nejvíce hodí k rozpoznávání obrazu. Hlavním cílem teoretické části je upřesnění základních pojmů, jako jsou například neurony, umělá inteligence, tensory a další. Dílčím cílem je poté představení nástroje Google Colaboratory a celkového principu rozpoznávání obrazu, který v dnešní době hraje velkou roli a dostává se do čím dál většího množství oborů.

3.1 Hluboké učení, strojové učení a umělá inteligence

Tato kapitola poskytuje základní informace o umělé inteligenci, strojovém učení a hlubokém učení a jejich vzájemném vztahu.



Obrázek 1 - AI obsahující strojové učení, jehož částí je hluboké učení[1]

3.1.1 Umělá inteligence (artificial intelligence)

Termín umělá inteligence odkazuje k simulaci lidské inteligence. Zrodil se v padesátých letech, kdy John McCarthy zorganizoval konferenci pro všechny zajímající se o strojovou inteligenci a o to, zda můžeme přimět počítače, aby „myslely“. V té době sice ještě nebyl zaveden termín umělá inteligence, ale již se na ní pracovalo. Mezi první výskyty tohoto termínu patří práce Alana Turinga, který napsal studii na téma strojů a jejich možnosti simulace lidí a schopnostech dělat inteligentní věci, jako je například hraní šachů. [1]

Tento termín lze aplikovat ke každému stroji, který ukazuje vlastnosti lidské mysli, jako je například schopnost učit se a řešit problémy. Ideální charakteristikou umělé inteligence je schopnost racionalizovat a podnikat akce, které mají nejlepší šanci k dosažení specifického cíle. [2]

[1] Zdroj: Vlastní zpracování ve free softwaru draw.io

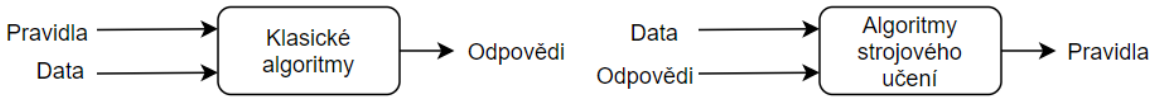
AI[2] nezahrnuje pouze strojové a hluboké učení, ale také mnoho dalších přístupů, které neobsahují žádné učení. Příkladem je výše zmíněné hraní šachů, první takovéto šachové programy byly vytvořeny pouze zakódováním pravidel programátory. Toto bychom nemohli označit za strojové učení. [2]

AI je založena na principu, že lidská inteligence může být definována takovým způsobem, který dokáže stroj jednoduše napodobit a realizovat jednotlivé úlohy, a to od těch nejjednodušších, až po ty, které jsou více složité. Cíl umělé inteligence zahrnuje učení se, uvažování a vnímání. [2]

3.1.2 Strojové učení

Strojové učení je podskupina umělé inteligence, která dovoluje softwarovým aplikacím, aby byly více správné v předpovídáním výstupů bez toho, aby k tomu byly přímo naprogramovány. Algoritmy strojového učení využívají mnoho příkladů pro danou úlohu, díky kterým se algoritmus natrénuje. Na základě těchto příkladů algoritmus vyvodí statistickou strukturu, kterou poté využije pro pravidla k automatizaci daného řešení. Pokud by se měl systém například naučit na fotkách označovat auta, stačí takovému programu dát mnoho příkladů obrázků kde již lidé auta označili. Systém se z nich poté naučí statistická pravidla pro označování aut na obrázku.

Definice dle Joseph Rocca „*Strojové učení je věda, která přiměje počítače dělat nějaký úkol bez toho, aby byli na daný úkol naprogramováni.* [3]“ Klasické algoritmy je třeba naprogramovat na danou úlohu a tím jim dát veškerá pravidla k tomu, aby dokončily danou úlohu. Naopak algoritmům strojového učení jsou dána základní pravidla, která definují daný model a data. Tato data by měla obsahovat chybějící informaci potřebnou pro model k dokončení úlohy. Díky tomuto může algoritmus strojového učení může upravit daný model dle poskytnutých dat tak, aby splnil zadaný úkol. Tomuto kroku se říká, že „**trénování modelu na datech**“. [3] Rozdíl mezi těmito algoritmy lze jednoduše zpozorovat z obrázku 2.



Obrázek 2: rozdíl mezi algoritmy klasickými a strojového učení[3]

[2] AI je zkratka pro Artificial Inteligence (česky umělá inteligence)
[3] Zdroj: Vlastní zpracování ve free softwaru draw.io

Existují základní tři metody strojového učení:

- Učení s učitelem (Supervised Learning): Tento typ učení zajišťuje model, který je schopen určit zařazení daného vstupu, a to dle jeho vlastností. Aby se správně naučil zařazovat vstupy, musíme ho naučit spojitosti mezi vlastnostmi vstupů a jednotlivými typy výstupů. Toto ho naučíme tím, že danému učícímu se systému dáme příklady vlastností a k nim patřící typy výstupů. Vstupy se pustí do sítě, poté se výstup porovná s požadovaným výstupem, a pokud se nerovnají, tak se provede korekce (změna vah a prahů). Poté, co je model natrénován, může sám rozeznávat další data, která mu dáme. Například systém naučený na množině jablek a pomerančů dokáže rozlišovat, zda se jedná o jablko, nebo pomeranč.
- Učení bez učitele (Unsupervised Learning): V učení bez učitele dáme učícímu se systému pouze vstupní data, ve kterých si automaticky najde jednotlivé vzory a spojitosti tím, že si v nich vytvoří jednotlivé skupiny (shluky). Tento model nedokáže rozeznávat o jaký výstup se jedná, ale dokáže jej přiřadit do správné skupiny. Například systém, který jsme učili na obrázcích jablek a pomerančů, dokáže rozeznat jednotlivé vlastnosti a rozřadí obrázky do skupin (U kterých neví, že to jsou jablka a pomeranče, jen skupiny se stejnými vlastnostmi.). Poté co dostane nová data, tak je zařadí do správné skupiny.
- Zpětnovazebné učení (Reinforcement Learning): U zpětnovazebného učení učící systém zpracovává poskytnutá data, aniž by byl předem naučen a dostává průběžnou odezvu, zda byl jeho výstup správný či nikoliv. Systém následně upravuje svůj model dle správných a špatných odpovědí. Například systém, kterému jsme dali data jablek a pomerančů. Ze začátku dostane obrázek pomeranče, který špatně vyhodnotí jako jablko. V tuto chvíli ho opravíme, že se jedná o pomeranč, načež si upraví model a zkusí vyhodnotit obrázek znovu. Tímto způsobem se naučí správně rozpoznávat vstupy. [4]

3.1.3 Hluboké učení

Hluboké učení je podskupinou umělé inteligence a strojového učení. Využívá algoritmy inspirované strukturou a funkcionalitou lidského mozku. Využívá vícevrstvé umělé neuronové sítě, aby zpracovávalo velmi přesně náročné úkoly, jako je rozeznávání obrazu, řeči, překládání řeči a další. Tyto vrstvy umělých neuronových sítí jsou postaveny jako lidský mozek se sítí neuronů, které jsou spolu vzájemně propojeny. Toto je více rozebráno v kapitole 3.2 o neuronových sítích. [5]

Hluboké učení se liší od strojového učení především v tom, že se dokáže automaticky naučit z reprezentací z dat jako jsou obrázky, videa nebo text bez toho, aniž by programátoři programovali složitá pravidla pro jejich rozpoznávání. Takovéto architektury jsou velmi flexibilní a dokážou se naučit přímo z poskytnutých dat. Dokáží zlepšit svou přesnost, když jsou jim poskytnuta další data, ze kterých se mohou učit.

Hluboké učení je důvodem mnoha nedávných průlomů v technologiích umělé inteligence, jako je například Google DeepMind's AlphaGo samo-řídící se vozidla, inteligentní hlasoví asistenti, jako je Cortana, Bixbi, rozpoznávání obličejů a další. [6]

3.2 Neuronové sítě

Neuronové sítě vychází z poznatků neurofyziologů, kteří se snažili vysvětlit zpracování informací v mozku pomocí nervových buněk. Za počátek neuronových sítí se považuje práce neurofyziologa Warrena McCullocha a matematika Waltera Pittse z roku 1943, ve které psali o tom, jak neurony mohou fungovat. Společně také vymodelovali jednoduchou neuronovou síť za pomocí elektrických okruhů. Za další milník se považuje rok 1949, ve kterém Donald Hebb posílil koncept neuronů ve své knize „*The Organization of Behavior*[4]“. V této knize bylo řečeno, že neuronové spoje se posílí vždy, když jsou použity. [7]

Tyto neurofyziologické poznatky umožnily vytvoření zjednodušených matematických modelů, které se dali využít pro řešení praktických úloh v umělé inteligenci. Neurofyziologie zde tedy slouží jen jako zdroj inspirace a navržené modely neuronových sítí se dále rozvíjely bez ohledu na interpretaci fungování lidského mozku.

[4] Česky: Organizace chování

Jak již bylo zmíněno, tak neuronová síť je algoritmus inspirovaný činností lidského mozku. Jedná se o analytickou metodu spadající do tzv. data miningu, která získává skryté informace z dat. Z dřívějších poznatků je již známé, že mozek tvoří mnoho vzájemně propojených buněk, kterým se říká neurony. Ty jsou mezi sebou propojeny za pomoci takzvaných synapsích, které mezi jednotlivými neurony posílají informace díky elektrickým impulzům. Jednotlivé synapse si lze tedy představit jako drátky spojující neurony. [8]

3.2.1 Model neuronu

Z předchozí kapitoly již víme, že první matematický model neuronu byl vytvořen již v roce 1943 a popsán McCullochem a Pittsem[5]. Jedná se o nepřesnou abstrakci mechanismu, jak jsou informace zpravovány pomocí nervových buněk. Model umělého neuronu je nepřesný, protože v dnešní době nelze vytvořit přesnou kopii modelu biologického neuronu kvůli jeho složitosti a doposud také není přesně definováno, jakým způsobem neurony zpracovávají informace. Tento model používá pouze základní funkce neuronu, které spočívají ve sběru elementárních informací, vyhodnocení jejich souhrnného stavu, a nakonec zodpovězení reakcí na výstupu. To stačí k tomu, aby se i dodnes používal pro běžné aplikace.

Neuron jako takový si lze představit jako velmi jednoduchý počítač, který se všemi vstupními informacemi provádí nějakou jednoduchou operaci, jako je například sečtení. Výsledek poté posílá výstupem do dalších neuronů, které provádí další jednoduché operace. Samotná synapse obvykle signál pouze přenáší, ale v některých případech může signál deformovat, například oslabením, nebo naopak zesílením signálu. [8]

Schématický model jednoho neuronu lze vidět na obrázku 3, jeho fungování lze vysvětlit i pomocí matematické funkce níže. První částí jsou synapse, které jsou ohodnocené váhami (w), podle jednotlivých vah lze jednotlivé vstupy zvýhodnit, či naopak potlačit. Synapse přivádějí vstupy (x) o několika (n) prvních do těla neuronu, do kterého také vstupuje prahová hodnota (θ). V těle neuronu se získává vnitřní potencionál neuronu. Vnitřní potencionál neuronu je hodnota vzniklá porovnáním vážených vstupů s prahovou hodnotou. Dále je signál poslán do bloku aktivační přenosové funkce (f), která provádí obecně nelineární transformaci vnitřního potencionálu na jednu hodnotu, tedy výstup (y). Výstup je obecně opět reálná hodnota a to 1 nebo 0 v závislosti na tom, zda váhová suma vstupních

[5] Říká se mu McCullock-Pitts (MCP) neuron. [12]

signálů překročila prahovou hranici, nebo ne. Samotný neuron neprovádí žádné složité operace, jedná se tedy spíše o vykonávání funkcí na úrovni regresní analýzy. [9]

Matematicky je funkce popsána následovně [10]:

$$y = f(q) = f\left(\sum_{i=1}^{n} x_i w_i + \theta\right)$$

Kde:

- y představuje vstupní signál neuronu,
- f představuje aktivační funkci (nebo také přenosová funkce) neuronu,
- q představuje potenciál neuronu,
- n představuje celkový počet vstupů,
- x_i představuje hodnotu na i-tém vstupu,
- w_i představuje váhu i-tého vstupu,
- θ představuje prahovou hodnotu (označováno také jako bias)

Mezi nejčastěji použité přenosové funkce jsou zařazeny tyto [11]:

- Skoková funkce (hardlim) – jedná se o funkci prahovou, nabývá tedy pouze hodnot 0 nebo 1.

$$a(n) = \begin{cases} 1 \; pro \; n \geq 0 \\ 0 \; pro \; n \; \leq 0 \end{cases}$$

- Lineární funkce (purelin) – jedná se o jednoduchý vztah, který říká, že vstup je kopírován beze změny na výstup.

$$a(n) = n$$

- Sigmoidní funkce (logsig) – je využita především u vícevrstvých perceptronových sítí.

$$a(n) = \frac{1}{1 + e^{-n}}$$

- Funkce signum (hardlims) – podobná funkci skokové, nabývá pouze hodnot 1 a -1.

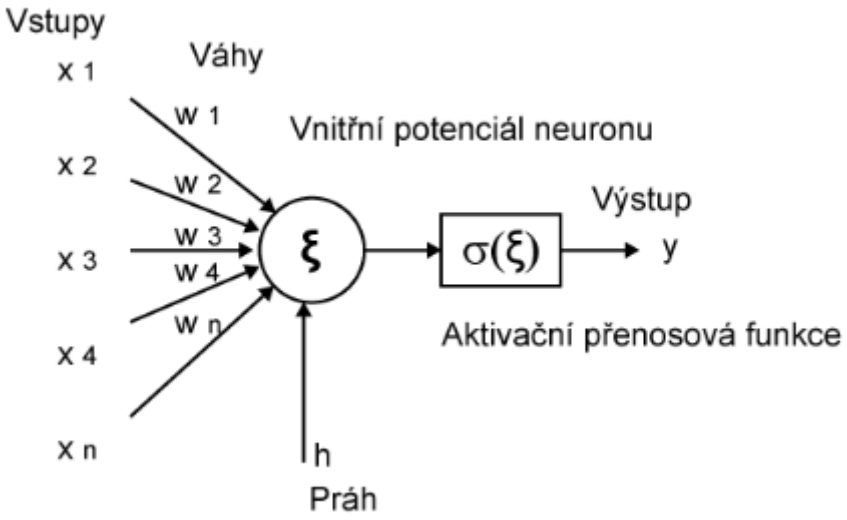
$$a(n) = \begin{cases} 1 \; pro \; n \geq 0 \\ -1 \; pro \; n < 0 \end{cases}$$

- Funkce ReLU (Rectified linear unit) – funkce jednoduchá, ale velmi využívaná právě pro modely hlubokého učení. Tato funkce vrací 0 pro každý negativní vstup, ale pro každý positivní vstup x vrací tu samou hodnotu zpět. [12]

$$f(x) = \max(0, x)$$

- Funkce Softmax – využíván k přeměně hodnot na pravděpodobnosti, tudíž její výstup je v intervalu $\langle 0,1 \rangle$. Je často využívána u klasifikačních úloh, kde udávají pravděpodobnost toho, že vstup patří do třídy určené výstupním neuronem. Typicky je použitá ve výstupní vrstvě klasifikační neuronové sítě. [13, 14]

$$f(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}$$



Obrázek 3: jednoduchý model neuronu[6]

Jednotlivé neurony, které jsou složeny ze tří hlavních částí, a to vstupní, funkční a výstupní, jsou mezi sebou propojeny vazbami. Neuron sám o sobě nevykonává žádné složité procesy, ale v neuronových sítích, které využívají principu propojení jednotlivých neuronů do větších struktur a do několika za sebou jdoucích vrstev, díky tomu struktura získává na síle systému.

[6] Na obrázku je znak pro funkci **σ** roven znaku **f** v matematickém vzorci a znak pro prahovou hodnotu **h** je roven $\boldsymbol{\theta}$ ve vzorci.
Zdroj: https://portal.matematickabiologie.cz/res/image/Umela%20inteligence/obr-4-5-model-neuronu.png

3.2.2 Princip neuronové sítě

Jak bylo uvedeno v předchozí kapitole, neuronové sítě se skládají z jednotlivých vrstev neuronů, které jsou propojeny takzvanými synapsemi. Každá neuronová síť se skládá z jedné vstupní vrstvy, jedné výstupní a minimálně jedné mezi nimi, ve která zajišťuje takzvané strojové myšlení.

Hlavní myšlenkou neuronových sítí je schopnost učit se. Díky tomu se mohou učit pomocí dat, ze kterých se naučí různé kombinace, které vedly ke správnému výsledku. Pro další vstupní data poté použijí nejlepší kombinaci vnitřního nastavení, kterou v kroku učení se získaly a odvozují tak nový výsledek. Takováto schopnost učit se je považována za jednu z definic umělé inteligence. Díky tomu nemusí programátor znát fungování neuronů, ani hodnoty jejich propojení, to vše zvládá neuronová síť sama. Programátorovi jen stačí navrhnout počet neuronů, jejich rozdělení do vrstev a vzájemné propojení synapsemi. [8]

V začátcích měly neuronové sítě, jako například první perceptrony, velmi málo vrstev. Zpravidla se skládaly z jedné vstupní, jedné výstupní a na nejvíš z jedné skryté (střední) vrstvy. Říkalo se jim mělké, neuronové sítě s více jak třemi vrstvami se již kvalifikují jako hluboké učení. Z tohoto lze usoudit, že slovo „hluboké“ není pouze k tomu, aby tyto algoritmy zněly chytře, ale značí, že obsahují více jak jednu skrytou vrstvu. [15]

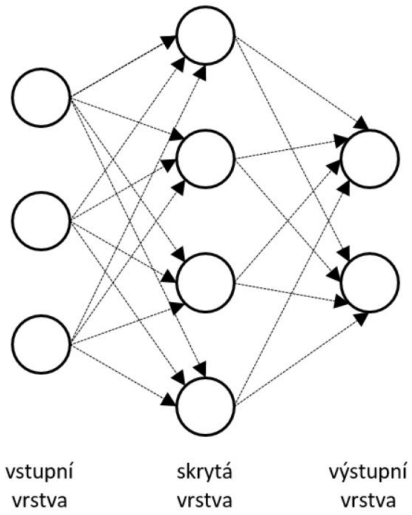
3.2.3 Fungování neuronových sítí

Neuronové sítě nám pomáhají shlukovat a klasifikovat. Lze si je představit jako vrstvu, která je nad daty, co jsou uložená a která spravujeme. Tato vrstva právě shlukuje a klasifikuje data. Pomáhá dávat neoznačená data do skupin díky jejich podobnostem mezi jednotlivými příklady vstupů, a klasifikuje data, pokud se to naučila z olabelovaných[7] dat. [15]

Neuron je základním prvkem lidské i umělé neuronové sítě. Neurony jsou mezi sebou propojeny synapsemi, které mají za úkol předávat signály. Zde platí, že každý neuron má pouze jeden výstup, který může být poslán i více dalším neuronům a zároveň může mít více vstupů. Každý ze vstupů má určitou hodnotu váhy. Na obrázku 4 lze vidět orientovaný graf sítě typu 3-4-2. Tato síť má vstupní vrstvu se třemi vstupy, které jen opakují hodnoty vstupů. Ty se předají všem neuronům ve vrstvě další, tzv. skryté vrstvě. Skrytá vrstva je takto nazývána proto, protože vstupy ani výstupy zde se nacházející, nezasahují přímo do vnějšího

[7] Olabelovaná data – labeled data (více v odstavci o úpravě dat v praktické části)

prostředí. Výstupy ze skryté vrstvy se poté stanou vstupy pro dva neurony ve výstupní vrstvě. Výsledné hodnoty z výstupní vrstvy jsou pak výstupem celé neuronové sítě.



Obrázek 4: Síť typu 3-4-2[8]

Je-li pevně daný počet neuronů ve vrstvách, jejich spojení a transformační funkce, tak převod vstupů na výstupy závisí na hodnotě vah a prahů. Tyto hodnoty se stanovují v tzv. procesu učení. Velkou roli v učení hraje nastavený počet neuronů, který by neměl být příliš velký, ale ani příliš malý. Dále hraje roli reprezentativní množina vstupů a výstupů. Díky tomu je neuronová síť schopna řešit úlohy, se kterými se během učení přímo nesetkala. [16]

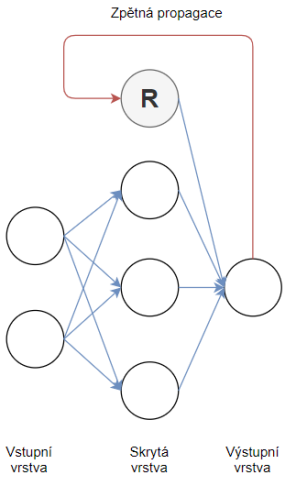
Jednou z vlastností neuronových sítí je topologie neuronové sítě neboli způsob učení, nebo model učení. Topologie neuronových sítí se dělí na základní dva typy, a to na neuronové sítě s dopředným šířením signálu a se zpětnovazebným šířením signálu.

U neuronových sítí s dopředným šířením signálu nejsou žádné smyčky, signál tedy vždy putuje ze vstupní vrstvy do skryté vrstvy postupně až k vrstvě výstupní. Přitom každý neuron v dané vrstvě přijme všechny vypočítané výstupy z neuronů u vrstvy předchozí. Poté vypočítají svůj výsledek a pošlou ho do další vrstvy. Takto se šíří signál až k vrstvě výstupní bez jakýchkoli zpětných vazeb. [17]

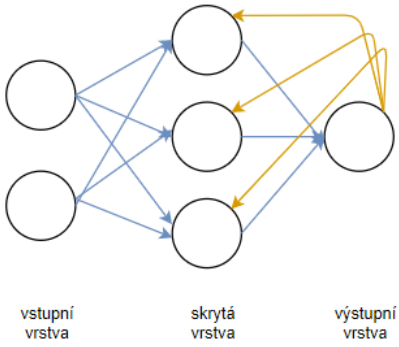
U sítí se zpětnovazebným šířením signálu se u jednotlivých neuronů objevují vstupy, které jsou závislé na výstupu sítě z předchozího cyklu. Jedná se o snahu eliminovat fakt, že stejný vstup vede vždy na stejnou odezvu sítě. Kvůli tomu je třeba se snažit implementovat časový kontext, kdy odezva sítě nebude dána pouze aktuálním vstupem, ale bude odrážet i vliv vstupů, které mu předcházely. Oproti sítím s dopředným šířením signálu se signál nešíří

[8] Zdroj: https://www.napocitaci.cz/images/onlibpc/17_19neuronovesite_03.png

pouze od vstupní vrstvy k výstupní, ale dochází i ke zpětnovazebnému šíření dat od i-té vrstvy k vrstvám předchozím (i–1-tých). Tento přenos dat je řešen pomocí takzvaných rekurentních neuronů a příkladem této sítě je následující obrázek. [18]



Obrázek 5: Graf neuronové sítě se zpětnovazebným šířením signálu[9]



Obrázek 6: Backpropagation ve grafu sítě 2-3-1[10]

Učení u neuronových sítí zajišťuje změna vah (pomocí tzv. metody *backpropagation*), která je nutná pro optimalizaci řešení. Toho lze zajistit pomocí ztrátového skóre počítaného ztrátovou funkcí. Ztrátové skóre je následně použito k úpravě vah a prahových hodnot. Úkolem učení je co nejvíce zmenšit ztrátové skóre. Menší ztrátové skóre vede k více přesným výsledkům neuronové sítě, a to díky tomu, že se rozdíl mezi výslednou hodnotou a požadovanou hodnotou zmenšuje. Minimalizace ztrátové funkce je tím hlavním cílem, jakého se neuronové sítě snaží dosáhnout. Existují dvě hlavní ztrátové funkce, co by měly vyřešit skoro každý problém, na který lze narazit. Tyto funkce jsou:

[9] Zdroj: vlastní zpracování pomocí app.diagrams.net

[10] Zdroj: vlastní zpracování pomocí app.diagrams.net

Ztráta křížové entropie (Cross-Entropy Loss):

$$L(n) = -\sum_{i=0}^{N} \hat{y}_i * \log(y_i)$$

$$y_i: vstupuje\ jako\ vektor\ výstupu\ \vec{y}$$

$$\hat{y}_i: vstupuje\ jako\ vektor\ očekávaných\ hodnot\ \vec{\hat{y}}$$

Ztráta střední kvadratické chyby (Mean Squared Error Loss):

$$L(n) = \frac{1}{N}\sum_{i=0}^{N} (y_i - \hat{y}_i)^2$$

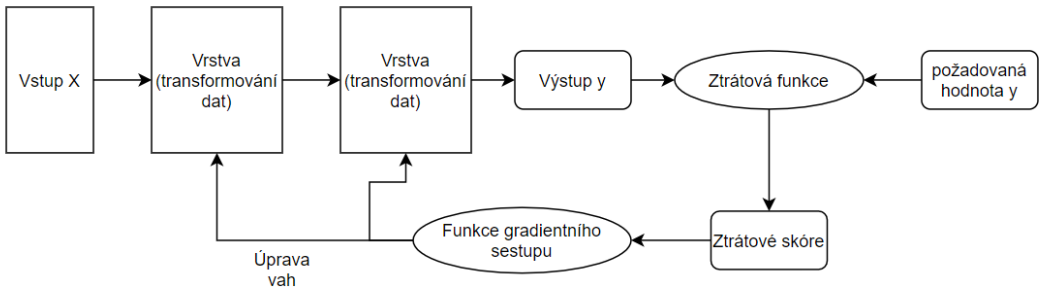
$$y_i: vstupuje\ jako\ vektor\ výstupu\ \vec{y}$$

$$\hat{y}_i: vstupuje\ jako\ vektor\ očekávaných\ hodnot\ \vec{\hat{y}}$$

Kvůli tomu, že ztráta záleží na nastavení vah, je nutné najít určité nastavení vah s co nejmenší možnou ztrátovou hodnotou. Metody minimalizace ztrátové funkce je docíleno matematicky pomocí takzvaného gradientního sestupu (Gradient Descent).

Během gradientního sestupu se využívá gradient ztrátové funkce (jinými slovy derivace ztrátové funkce) k vylepšení vah neuronové sítě.

Celkový model fungování zpětnovazebné neuronové sítě z obrázku 6 by vypadal takto:



Obrázek 7: Model fungování backpropagation u neuronové sítě[11]

3.2.4 Reprezentace dat v neuronových sítí

Reprezentace dat ve strojovém učení musí být spíše číselná, a to platí dvojnásob, když se bavíme o reprezentaci dat u neuronové sítě. Toho je docíleno díky jakýmsi kontejnerům pro data, označují se pojmem **tenzory**. Tenzory jsou kontejnery, které dokážou uchovávat data v N dimenzích, také označované jako osy (anglicky axis). Mnoho lidí pravděpodobně zná 2D tenzory aniž by o tom vědělo - 2D tenzory jsou matice. [19, 20]

[11] Zdroj: vlastní zpracování pomocí app.diagrams.net

0D tenzory obsahují pouze jedno číslo neboli skalár. Takovému tenzoru se říká skalární tenzor. Jako 0D tenzor lze uvést čísla typu float32/float64 v knihovně NumPy. Nebo jakékoli pole o jedné hodnotě. Příkladem je „array(12)“

1D tenzory jsou poté pole s několika hodnoty, tedy vektory. Tento tenzor má přesně jednu osu. Příkladem je „array ([12,25,5,8])“. Tento vektor má čtyři prvky, a proto ho lze označit jako čtyřdimenzionální vektor. Je důležité si uvědomit, že 4D vektor není totéž, co 4D tenzor. 4D vektor má pouze jednu osu, kdežto 4D tenzor má osy čtyři a může mít libovolný počet rozměrů podél každé své osy. Dimenze tak lze přiřadit jak k vektorům i tenzorům, ale je nutné nezaměňovat jakou informaci nám sdělují.

2D tenzory jsou, jak již bylo zmíněno, matice. Jedná se o pole vektorů. Každá matice má dvě osy, které jsou často označované jako řádky a sloupce. V reálném světě se jedná o vektorová data a jsou ve tvaru (příklady, funkce). Příkladem je: „array([12,25,5,8], [2,13,4,7])“. Pro lepší přehlednost:

$$\begin{pmatrix} 12 & 25 & 5 & 8 \\ 2 & 13 & 4 & 7 \end{pmatrix}$$

3D tenzory lze vytvořit zabalením 2D tenzorů do pole. V reálném světě se jedná o časové řady a jsou ve tvaru (příklady, časové údaje, funkce). To lze ukázat na příkladu „array([[12,25,5,8], [2,13,4,7]], [[45,2,12,4], [23,13,5,2]])“. Pro lepší přehlednost:

$$\left(\begin{pmatrix} 12 & 25 & 5 & 8 \\ 2 & 13 & 4 & 7 \end{pmatrix} \begin{pmatrix} 45 & 2 & 12 & 4 \\ 23 & 13 & 5 & 2 \end{pmatrix}\right)$$

Tvoření **4D** a **5D tenzorů** funguje podobně jako u 3D, pouze zabalením předchozí dimenze tenzoru do další vrstvy. 4D tenzory jsou v reálném světe obrázky a jsou ve tvaru (příklady, výška, šířka, barevné kanály) nebo (příklady, barevné kanály, výška, šířka). 5D tensory jsou videa a jsou ve tvaru (příklady, rámy, výška, šířka, barevné kanály) nebo (příklady, rámy, barevné kanály, výška, šířka). Například u tenzoru tvaru (128, 128,128, 1) lze určit, že bude obsahovat dávku 128 obrázků o velikosti 128x128 a v stupních šedi. Stupně šedi označuje jednička jako poslední číslo tenzoru. Ta nám říká, že bude pouze jeden barevný kanál. [19]

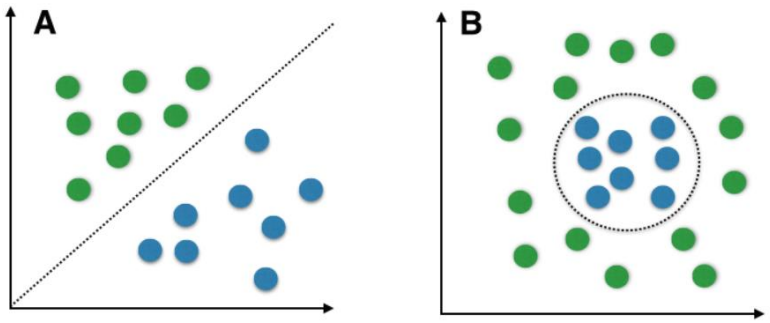
3.3 Příklady modelů neuronových sítí

3.3.1 Perceptron

Perceptron je základní model neuronové sítě, která provádí určité funkce k detekování vlastností dat. Byl vynalezen roku 1957 psychologem Frankem Rosenbalttem, který se

zabýval umělou inteligencí. Ten navrhnul pravidla učení Perceptronu na základě originálního MCP neuronu, pro který tehdy nebyla vypracována žádná tréninková metoda. Dále Původně se mělo jednat o model zrakové soustavy, ale později se přišlo na jeho limitace. Je možné ho použít pouze na množiny, které jsou lineárně separovatelné.

Jedná se o jednovrstvou neuronovou síť, v níž se využívá učení s učitelem, kdy je Perceptronu nejdříve předložena množina příkladů se správným chováním. Učící pravidlo následně nastaví váhy a práh tak, aby došlo k přiblížení výstupu sítě k cíli. Učení se zahajuje inicializací úvodních hodnot, ve které se nastaví váhy na náhodné hodnoty. Poté je síti předložen vstupní vektor ze souboru učících dat, který projde sítí a pomocí funkce je zjištěn výstup. Tento výstup porovná se správnou výstupní hodnotou. Pokud se nerovnají, tak se musí adaptovat váhové hodnoty. Například pokud je vstupní vektor vyhodnocen jako 1 i když má být 0, tak je nutné odečíst vstupní vektor od vah, a naopak, pokud se špatně vyhodnotí výstup jako 0 a měl by být 1, musí se vstupní vektor přičíst k vahám. Tímto se získá nová hodnota vah a lze přejít k dalšímu vektoru v učících datech. Jednovrstvé Perceptrony se mohou naučit pouze lineárně oddělitelné vzory. Jsou to tedy takzvané binární klasifikátory, které mapují vektor vstupů, na výstupní hodnoty, které jsou rozděleny do dvou skupin. Tento algoritmus učení je konečný, pouze pokud řešení existuje a učící množina lze lineárně separovat. Složitější funkce pomocí Perceptronu nelze řešit a v takovýchto případech je nutné rozšířit základní model do vícevrstvé dopředné neuronové sítě. [21, 22]



Obrázek 8: A- lineárně separovatelná množina; B- lineárně neseparovatelná množina[12]

Již naučený obsahuje *n* vstupů a jeden neuron. Prvním vstupem je hodnota 1, která se násobí s hodnotou vah w_0. Tato hodnota vah představuje prahovou hodnotu neuronu. Jeho ostatní vstupy jsou vynásobeny příslušnými hodnotami vah, tyto hodnoty jsou poté sečteny a výsledná hodnota a vstupuje do aktivační funkce. Výstupem je pak hodnota dle aktivační

[12] Zdroj: http://sebastianraschka.com/Articles/2014_naive_bayes_1.html

funkce, která může být binární (tj. s výstupem 0 nebo 1), nebo bipolární (tj. s výstupem -1,0 nebo 1). [21, 22]

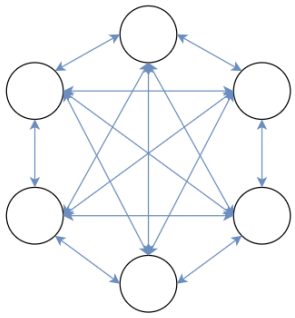
3.3.2 Vícevrstvá dopředná neuronová síť

Byla představena v roce 1986 a jejími autory jsou G.E. Hinton, D.E. Rumelhart a R.J. Williams. Měla za cíl vyřešit omezené schopnosti perceptronu. Jak již bylo zmíněno, vícevrstvé sítě se skládají minimálně ze tří vrstev, a to z jedné vrstvy vstupní[13], jedné vrstvy výstupní a alespoň jedné vrstvy skryté. Lze tedy říci, že graf sítě na obrázku 4 na straně 20 je grafem vícevrstvé dopředné neuronové sítě.

V tomto modelu neexistují žádná spojení mezi neurony ve stejné vrstvě a každý neuron v i-té vrstvě je spojen s každým neuronem ve vrstvě následující (i + 1-té). Spojení mezi neurony, které zajišťují šíření signálu, jsou orientované, takže se signál šíří jen jedním směrem, a to od vstupu k výstupu. Každé z těchto spojení je ohodnocené váhami, kde případnou neexistenci spojení lze vytvořit nastavení váhy rovné nule. [23, 24]

3.3.3 Hopfieldova síť

Hopfieldova síť byla představena roku 1982. Jejím autorem je John Hopfield, který se zabýval studiem neuronů podobných perceptronům zmíněných výše. Tyto neurony měly některé podstatné odlišnosti od normálních perceptronů. Cílem bylo použití energické funkce svázané s neuronovou sítí, jak je to běžné u jiných fyzikálních systémů. Jak lze vidět na obrázku níže, tato síť je vytvořena z množiny navzájem propojených neuronů, kde se signál šíří v obou směrech. [18]



Obrázek 9: Diagram neuronů v Hopfieldově síti

[13] Některá literatura nebere vrstvu vstupní jako součást sítě, jelikož neupravuje data. Toho si lze všimnout na obrázku 7, který zobrazuje fungování modelu znázorněného v obrázku 6. Je zde zobrazen vstup X, který je vstupní vrstvou a neupravuje data, pouze zajišťuje jejich přeposlání do sítě.

Hodnoty jednotlivých vah jsou uloženy ve váhové matici W, která reprezentuje stav sítě a váhy jsou v síti symetrické, ve smyslu rovnosti ($w_{ij} = w_{ji}$). Každý neuron má stejně tak jako Perceptron svůj vlastní práh a aktivační funkci, do které vstupuje vnitřní potenciál daný váhovou sumou výstupů okolních neuronů. Stav neuronů může být standartní (přesněji 0 nebo 1) nebo častěji bipolární (-1 nebo 1).

U častějšího případu, tedy u Hopfieldova modelu s bipolárním stavem neuronů je hlavním rozdílem oproti ostatním modelům neuronových sítí to, že vstup je aplikován na všechny neurony sítě ve formě hodnot -1 a +1. Poté následuje cyklus postupných změn neuronů, až do okamžiku dosažení stabilního stavu. Jinak se dá říct, že výstupy z předchozího kroku jsou novými vstupy současného kroku. Inicializační stav představuje různorodost výsledů neuronů, které se navzájem ovlivňují s ostatními neurony. Díky tomu je možné říct, že se jednotlivé neurony snaží ostatní excitovat, nebo naopak inhibovat[14]. [18]

Princip kódování a následného vyvolávání vzorů se dá nejlépe vysvětlit pomocí funkce energie Hopfieldovy sítě. Lze si představit energetickou plochu, která má svá lokální minima reprezentující předložené vzory ve fázi adaptace sítě. Následně vstupem definujeme bod na této energetické ploše, který iteračním procesem relaxace sítě hýbeme po dané ploše až k některému stabilnímu bodu (atraktoru). Stabilní body jsou reprezentované lokálními minimy. [18]

3.3.4 Bolzmannovy stroje

Mezi hlavní problémy algoritmu Hopfieldovy sítě patří fakt, že je zde vysoká pravděpodobnost uvíznutí v některém z nežádoucích lokálních minim energie. O řešení tohoto problému se pokusil S. Kirpatrick, který vymyslel tento model. Základní myšlenkou je tedy umožnění systému dočasné zvýšení energie, která vedou k opuštění nežádoucích lokálních minim energie.

Hezky se k procesu relaxace sítě u Boltzmannova stroje vyjádřil Ivo Vondrák ve svých skriptech. „*Na proces relaxace sítě se pak můžeme dívat jako na proces přesunu kuličky po energetické ploše tím způsobem, že s celým systémem je "natřásáno" a kulička má tak možnost vyskočit i ze stabilních míst. Čím větší bude natřásání tím hlubší energetická minima kulička může opustit. Budeme-li však proces "natřásání" postupně snižovat, má kulička menší šanci opustit hluboká minima, zatímco minima mělká jsou ještě překonatelná.* [18]*“*

[14] Excitovat – zvyšovat jejich hodnotu; Inhibovat – snižovat jejich hodnotu

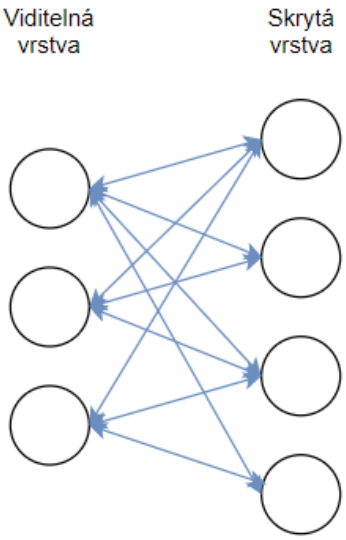
Díky tomu je mnohem častější přechod z mělčích míst do hlubších než přechod z míst hlubších do mělčích. Kirpatrickův nápad spočívá v tom, že začneme s velkým natřásáním, které budeme zvolna zmírňovat až do úplného zastavení. To lze realizovat principem žíhání, kdy se materiál po zahřátí postupně ochlazuje, jeho struktura se dostává do energetického minima a odstraňuje se jeho vnitřní pnutí. Takováto metoda se nazývá simulované žíhání.

Hopfieldův model byl rozšířen Hintonem, Sejnovskim a Ackleym ua použití výše zmiňované metody simulovaného žíhání. Svůj nový model nazvali Boltzmannův stroj, na počest zakladatele statistické fyziky. Měl dva typy uzlů, skryté a viditelné. Princip natřásání je zde realizován stochastickým charakterem stanovení stavu excitace neuronu kde se počítá s teplotou systému. Čím větší bude teplota systému, tím pravděpodobnější je, že excitace neuronu bude opačná k jeho potenciálu. Dochází tak k přechodnému zvýšení energie systému. Naopak pro teplotu blížící se nule přechází vztah v obvyklé pravidlo pro aktivitu Hopfieldova modelu, kde energie systému monotónně klesá až ke stabilnímu energetickému stavu. Takovýto systém se po nějaké době dostane do tepelné rovnováhy, a tím je dosažení nižšího energetického stavu pravděpodobnější než dosažení stavu o vyšší energii. [18]

3.3.5 Omezené Boltzmannovy stroje

Implementace Boltzmanova stroje jako takového je velmi obtížná, a proto byla vytvořena nová varianta, Omezené Boltzmanovy stroje (RBM). Patří také mezi stochastické typy neuronových sítí, což znamená, že budou mít nějaké náhodné chování, během jejich aktivace. Mají pouze jeden rozdíl, a to že jednotlivé uzly ze stejné vrstvy, ať už viditelné nebo skryté, mezi sebou nejsou vzájemně propojeny. Jedná se tedy o dvouvrstvé umělé neuronové sítě. Byly vynalezeny Geoffrym Hintonem a mohou být využity k řadě věcí jako je klasifikace nebo regrese. [25]

Díky omezení v podobě omezení na propojení mezi viditelnými a skrytými uzly, jsou RBM výrazně lépe implementovatelné než obyčejné Boltzmanovy stroje. Jak bylo zmíněno, jsou zde dvě vrstvy, které jsou navzájem obousměrně spojeny, což lze vidět na grafu níže. Existují tu dva vektory hodnot bias (prahových hodnot). Skryté bias, které jsou využity při výpočtech aktivační funkce směrem vpřed, a viditelné bias pomáhají rekonstruovat vstup při šíření signálu zpět směrem k viditelné vrstvě. Rekonstruovaný vstup je vždy odlišný od reálného vstupu, protože neexistují vazby mezi jednotlivými viditelnými uzly, kvůli čemuž si nemohou mezi sebou posílat informace. [26]



Obrázek 10: Diagram Omezeného Boltzmanova stroje[15]

První kroky v trénování RBM s vícero vstupy lze vidět na obrázku 11. Jako první jsou vstupy vynásobeny váhami, poté je přidána hodnota bias. Výsledek je poté dán do sigmoidní aktivační funkce a její výstup určí, zda se skrytá vrstva aktivuje nebo ne. Tento krok bychom tedy reprezentovali takovouto rovnicí [26]:

$$h^{(1)} = -S(v^{(0)T}W + a)$$

Kde:

- $h^{(1)}$ – vektor hodnot skryté vrstvy
- $v^{(0)}$ – vektor hodnot počátečních vstupních dat
- S – sigmoidní aktivační funkce
- T – tepelná hodnota
- W – vektor vah
- b - vektor prahových hodnot (bias) viditelné vrstvy
- a – vektor prahových hodnot (bias) skryté vrstvy

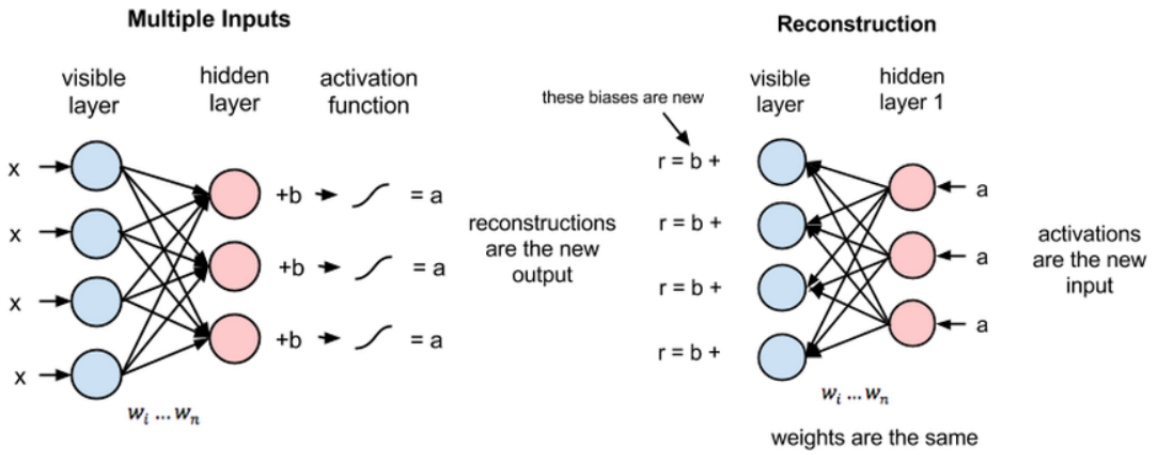
Dalším krokem je rekonstrukční fáze, která je podobná předchozí, pouze v opačném směru. Reprezentace rovnicí by byla:

$$v^{(1)} = -S(h^{(1)}W^T + a)$$

Kde:

- $v^{(1)}$ – vektor hodnot viditelné vrstvy vypočítaných rekonstrukcí

[15] Zdroj: vlastní zpracování pomocí app.diagrams.net



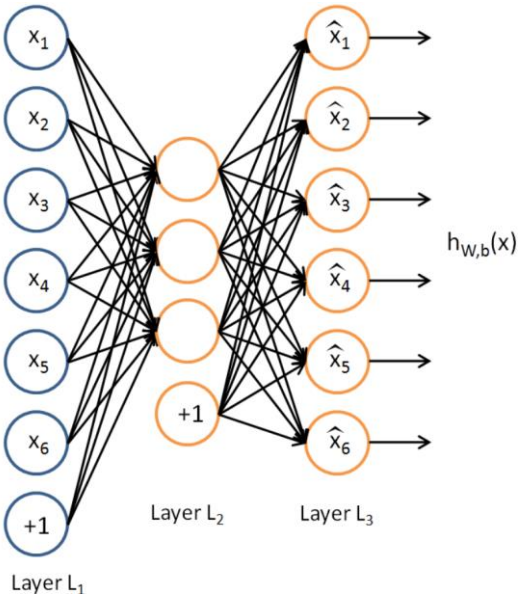
Obrázek 11: Fungování Omezeného Boltzmanova stroje[16]

Rozdíl $v^{(0)}$ - $v^{(1)}$ lze považovat za chybu rekonstrukce, kterou se v následujících krocích trénovacího procesu snažíme zmenšit. Každou iteraci se upravují hodnoty vah, abychom snížili danou chybu. O tom je vlastně celý trénovací proces. [26]

3.3.6 Autoenkóder

Jedná se o typ neuronové sítě, kde není potřeba mít jednotlivé prvky pro učení označkované. Jedná se tedy o typ sítě s učením bez učitele, kde k učení sítě využívají pouze vlastní vstup. Jsou to jednoduché učící se okruhy, které dokážou transformovat vstupy na výstupy s co nejmenším zkreslením. Snaží se zjistit funkci, která by vytvořila výstup co nejblíže podobný vstupním datům, tedy $\hat{x} \approx x$. Zjištění takové funkce se může zdát triviální, ale tím, že nastavíme nějaké omezení sítě, jako je například omezení počtu jednotek ve skryté vrstvě, můžeme přijít na zajímavé struktury dat. Jako příklad si vezmeme vstup *x*, který bude intenzita pixelů z obrázku 10 x 10 (100 pixelů). Takže $n = 100$, ale máme jen polovinu (50) jednotek ve skryté vrstvě L_2. Takováto síť se bude muset naučit „zkompresovanou“ reprezentaci vstupních dat. Poté se z hodnot skryté vrstvy musí naučit rekonstrukci původních dat. Výstupy z třetí vrstvy jsou poté výstupy z autoenkóderu, které mohou být vstupy pro další autoenkóder. Lze je tedy řetězit. [27]

[16] Zdroj: https://skymind.ai/images/wiki/multiple_inputs_RBM.png a https://skymind.ai/images/wiki/reconstruction_RBM.png



Obrázek 12: Graf autoenkóderu[17]

3.3.7 Nevýhody neuronových sítí

Neuronové sítě mají mnoho předností, díky kterým jsou velmi oblíbené při řešení hned několika problémů. Existuje ovšem několik problémů, co se k nim vážou. Tato podkapitola se věnuje právě několika těmto problémům.

- Přeučení (overfitting) – Mezi hlavní problémy neuronových sítí patří hrozba přeučení sítě. Obecně pro neuronové sítě platí, že pokud obsahují malý počet neuronů, jejich schopnost vystihování a popsání závislostí v trénovacích datech je slabší. Na druhé straně, pokud bude obsahovat až moc velký počet neuronů, tak taková síť bude dobře hledat a reprezentovat závislosti v trénovacích datech. Její schopnost zobecňovat (generalizace) ale může být horší. Právě tento jev se nazývá přeučení sítě. Dochází k němu, když model obsahuje příliš mnoho neuronů a vrstev, díky kterým se přesně naučí trénovací vstupy, nebo když se síť učí na až moc trénovacích datech, díky čemuž se spíše přizpůsobí konkrétním vstupům. [28]
- Černá skříňka (black box) – Neuronové sítě jsou modely černých skříněk, znamená to, že nedokážeme přesně říct, co se odehrává uvnitř. Nevíme, jak jednotlivé nezávislé proměnné ovlivňují proměnné závislé. Nelze přesně říct, jak síť dochází ke svým závěrům.

[17] Zdroj: http://ufldl.stanford.edu/tutorial/images/Autoencoder636.png

- Doba učení – Podle velikosti sítě, trénovacích dat a výkonu PC, na kterém učíme, se jednotlivé doby učení liší. Některé doby učení mohou trvat i několik měsíců. Platí, že s větším počtem vstupních vektorů a skrytých vrstev, se doba učení zvětšuje.

3.4 Konvoluční neuronové sítě

Tato kapitola se zabývá speciálním typem neuronových sítí, a to konvolučními neuronovými sítěmi (angl. Convolutional Neural Network[18]). V této kapitole jsou dále popsány rozdíly a výhody oproti normálním vícevrstvým neuronovým sítím a jejich použití. Tento typ byl vynalezen, aby umožnili strojům vidět svět více jako lidé, a aby využili své znalosti pro úlohy jako je rozeznávání videa a obrázků, analýza a klasifikace, rozeznávání hlasu, rozeznávání jazyku, a mnoho dalších.

3.4.1 Historie CNN

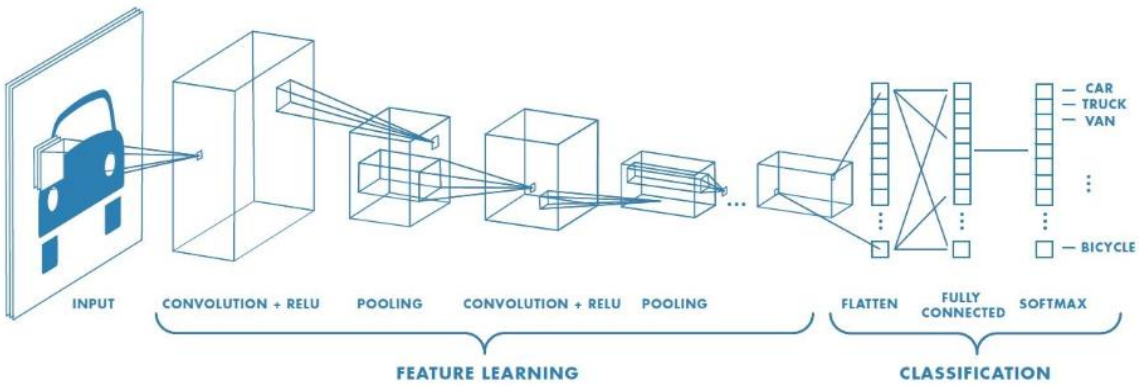
První práce na moderních CNN se stala v devadesátých letech devatenáctého století, kdy Yann LeCun vydal svou publikaci „Gradient-Based Learning Applied to Document Recognition“. Tato publikace byla inspirována neocognitronovým modelem, který byl představen v osmdesátých letech devatenáctého století Dr. Kunihiko Fukushimou. Neocognitorový model je vícevrstvý a obsahuje nové komponenty označované jako „S-cells“ (Simple cells – jednoduché buňky) a „C-cells“ (Complex cells – složité buňky), kde jednotlivé vrstvy podléhají hierarchické struktuře, kde „S-cells“ jsou vždy před „C-cells“, se kterými jsou propojeny. Celková myšlenka Neocognitronového modelu je zachytit koncept „od jednoduchého ke složitému“ a díky tomu vytvořit výpočetní model pro zjišťování vzorů u obrazových dat. Specificky LeCunpoužil databázi MINST s ručně psanými číslicemi. Okolo roku 2012 vzrostla popularita CNN, protože *Alex Krizhevsky*, *Ilya Sutskever* a *Geoff Hinton* představili konvoluční neuronovou síť, ta obsahovala více než tisíc filtrů v pěti konvolučních vrstvách, ReLu[19] aktivační funkci a pooling vrstva, která řešila overfitting. [29] Jejich popularita pokračuje až dodnes. [30]

[18] Známé také zkratkami CNN nebo ConvNet.
[19] Aktivační funkce vysvětlena v kapitole 3.2.1.

3.4.2 Základní charakteristika

Konvoluční neuronové sítě jsou algoritmem hlubokého učení, který dokáže vzít vstupní obrázek, přiřadit důležitost (učení vah a hodnot bias/prahových hodnot) k několika aspektům nebo objektům na obrázku. Dokáže také rozlišit různé objekty na obrázku. Příprava dat předem je také menší než u ostatních klasifikujících algoritmů.

Tyto sítě se skládají z různého množství za sebou jdoucích vrstev, mezi které by vždy měla patřit vrstva konvoluční, po které následuje pooling vrstva, která redukuje velikost *příznakových map* (feature map). Pomocí funkce flatten je výstup z konvolučních a poolingových vrstev převeden na vektor hodnot. Ukončení CNN je většinou řešené plně propojenou vrstvou. Tato struktura je velmi dobře viditelná na obrázku 13. [31, 32]



Obrázek 13: Příklad sítě s mnoha konvolučními vrstvami[20]

3.4.3 Konvoluce

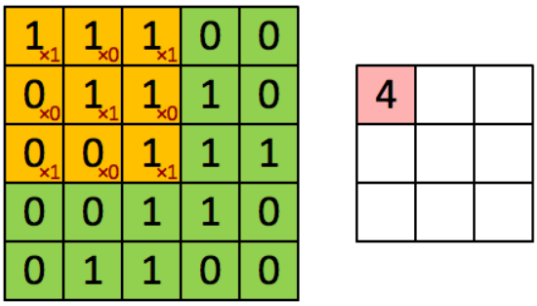
Hlavní komponentou konvolučních sítí jsou konvoluční vrstvy, které využívají takzvané operace konvoluce. Jedná se o speciální druh lineární operace, během které se posouvá po obrázku takzvaný *kernel*[21] (filtr, konvoluční matice), dle kterého jsou vypočítané nové hodnoty výstupu. [31, 32]

Jednoduše si lze tuto operaci prováděnou v konvoluční vrstvě představit jako posouvání menšího okna po vstupní matici a násobení dvou shodných hodnot, přičemž zpravidla platí, že při posouvání po vstupu k sobě středové „dlaždice“ kernelu přiléhají. Posouvají se tedy o jedno pole. Každé posunutí *kernelu*, má jednu výstupní hodnotu ve

[20] Zdroj: https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network/_jcr_content/mainParsys/band_copy_copy_14735_1026954091/mainParsys/columns_1606542234_c/2/image.adapt.full.medium.jpg/1575485682772.jpg

[21] Kernel by měl být přetáčet, u konvolučních neuronových sítí se tento krok často vynechává. [32]

výstupní matici, která je nazvána *feature map (mapa příznaků)*. Platí, že čím více má konvoluční vrstva filtrů, tím více rysů se dokáže naučit detekovat. Matice kernelu má jednotlivé hodnoty, které fungují jako váhy. Pomocí této operace se nedíváme na vstup jako na celek, ale jako na množinu jeho částí. Na obrázku 14 lze vidět celou operaci konvoluce, kde zelená matice představuje hodnoty jasu vstupního obrázku, žlutá matice je poté kernel a vpravo vidíme výslednou matici. Hodnoty kernelu jsou uvedeny v pravých dolních rozích jednotlivých buněk v matici kernelu. [31, 32]



Obrázek 14: Operace konvoluce[22]

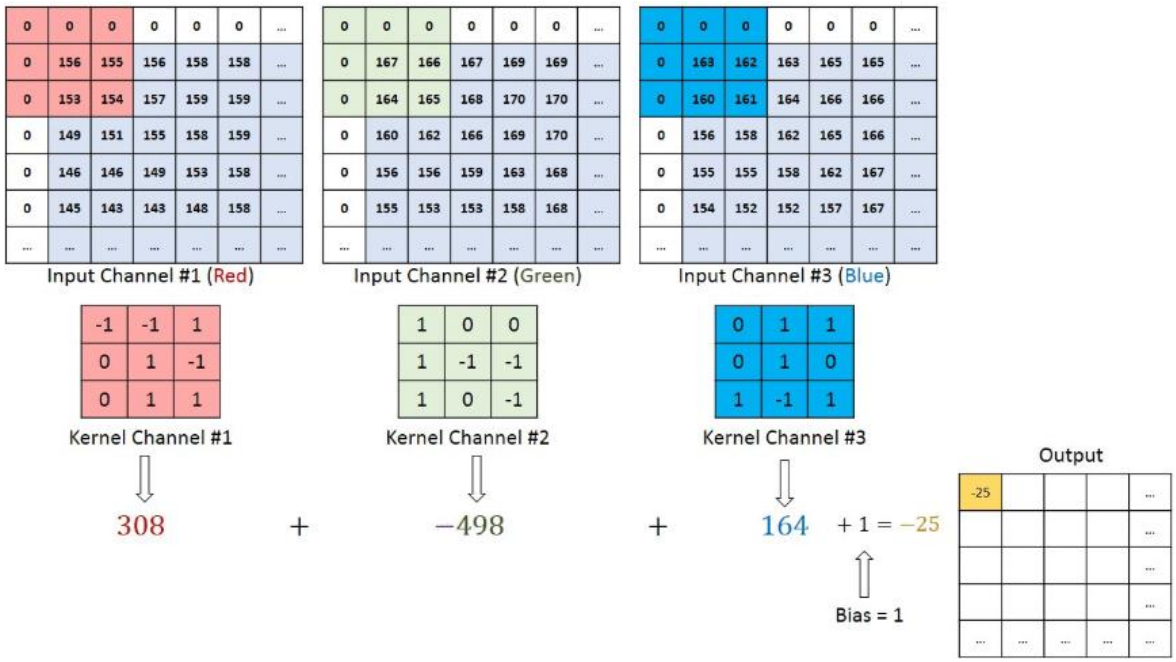
3.4.4 Konvoluční vrstva

Konvoluční vrstvy jsou nejhlavnějším aspektem CNN. Tyto vrstvy se učí lokální vzory, jako jsou hrany, textury a podobně. Tyto vzory jsou invariantní, pokud se tedy naučí určitý vzorec na jedné části obrázku, mohou ho poté použít i na jakékoli jiné části obrázku. Díky tomu dokáže CNN efektivně zpracovávat data při analýze obrazu a zvládá učit se z méně trénovacích dat se zachováním schopnosti generalizovat. Hlavní částí je filtr (kernel), který má tři dimenze. Šířka a výška bývá malá, zpravidla vždy menší[23], než jsou rozměry vstupu. Jeho hloubka je poté vždy stejná, jako hloubka vstupu, protože tento rozměr označuje osu kanálů obrazu. Pokud by měl vstup hodnotu osy kanálů 1, jednalo by se o obrázek ve stupních šedi. Pokud by měl hodnotu osy kanálů 3, jednalo by se o RGB obrázek, kde každá z vrstev reprezentuje hodnoty jasu u jednotlivé barvy. Operaci konvoluční vrstvy u RGB obrázku lze vidět na obrázku 15. [33, 19]

Dokážou se naučit prostorovou hierarchii vzorů, kde se první konvoluční vrstva učí malé lokální vzory, druhá konvoluční vrstva se bude učit větší vzory v první vrstvě a tak dále. Toto si lze představit na příkladu obrázku kočky, kde první konvoluční vrstva bude rozpoznávat určité vzory, jako jsou hrany obličeje, obrys nosu, zaoblení uší. Následující

[22] Zdroj: https://miro.medium.com/max/658/1*GcI7G-JLAQiEoCON7xFbhg.gif
[23] Většinou jsou používány filtry o velikosti 3x3xN nebo 5x5xN.

konvoluční vrstva by se poté učila složitější vzory, jako je celé oko, nos nebo celé ucho. Filtry na vysoké úrovni by následně kódovaly vlastnosti dat jako je koncept „přítomnost oka na vstupu“. Pokud tedy později takto naučená síť dostane obrázek splňující tyto vlastnosti (např. právě tvar obličeje, uší, nosu a očí), označí ho síť jako kočku. [19]



Obrázek 15: Počítání konvoluce pro MxNx3 obrázek s 3x3x3 kernelem[24]

Jako konkrétní příklad, na kterém si ukážeme fungování konvoluční vrstvy, si vezmeme černobílý obrázek 10x10x1, u kterého bychom dali velikost kernelu 3x3x1. Každý z pixelů v dané oblasti je vstupem do neuronu, který rozpoznává nějakou vlastnost. Rozpoznávané vlastnosti mohou být svislé, horizontální nebo pokřivené čáry. Takovýto neuron bude mít 10 vstupů (3x3 = 9 k čemuž přičteme hodnotu bias). Tento neuron je následně rozkopírován a aplikován na ostatní oblasti obrázku tak, že se jednotlivé oblasti mohou překrývat. Toto umožňuje takzvané sdílení vah, kde všechny neurony v dané *příznakové mapě* sdílejí váhy i hodnotu bias. Takovýchto neuronů se svými podoblastmi bude 68. Příznaková mapa v tomto příkladu má 10 vah i přesto, že obsahuje 68 neuronů. Takto lze detekovat lokální charakteristiky i při jejich posunu. [33, 19]

Jedna takováto příznaková mapa určuje pouze jednu charakteristiku, proto je využito více různých příznakových map, kde každá z nich obsahuje 68 neuronů a 10 vah. Dvě různé příznakové mapy mezi sebou nesdílí váhy ani hodnoty bias. V případě detekování šesti

[24] Zdroj: https://miro.medium.com/max/1600/1*ciDgQEjViWLnCbmX-EeSrA.gif

charakteristik lze určit, že daná konvoluční vrstva bude obsahovat celkem 408 (68 x 6) neuronů a 60 (10 x 6) unikátních vah. [33, 19]

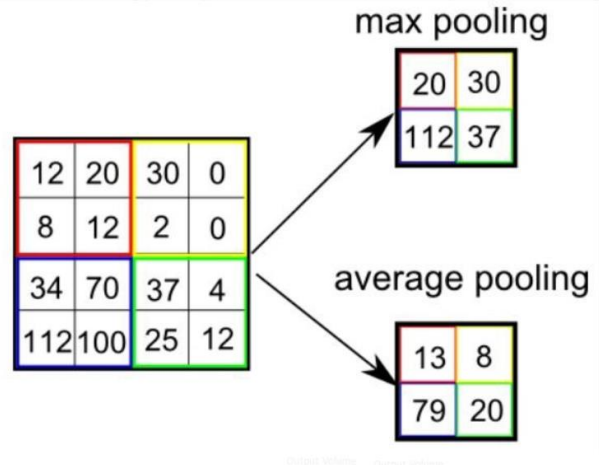
Pro změnu velikosti příznakových map lze použít dvě možnosti. Použití *padding* („vycpávek“) u konvoluce využívá přidání požadovaného počtu hodnot okolo vstupu, aby se upravil počet hodnot výstupu. Když se nevyužije *paddingu*, jsou vždy příznakové mapy menší než vstup. Druhý faktor, který ovlivňuje velikost výstupu, jsou *strides* (kroky), kde nastavený počet kroků udává, o kolik se posouvá kernel.

Konvoluční vrstvy jsou často zakončeny tím, že se data projedou přes *ReLU* algoritmus, jako je tomu vidět na obrázku 13.

3.4.5 Poolingová vrstva

Jedná se o operaci, která vždy stojí za konvoluční vrstvou. Lze ji vidět na obrázku 13. Má za cíl snižovat velikost vstupu, což jsou příznakové mapy, a díky tomu mají následné kroky menší časovou i paměťovou náročnost. Je velmi užitečná k extrakci dominantních charakteristik, které jsou invariantní k rotacím nebo umístění. Díky tomu lze udržovat efektivní trénování modelu.

Existují dva typy poolingu. Max pooling (operace sdružování dle maxima), a average pooling (sdružování dle průměru). Max pooling vrací maximální hodnotu z porce vstupu, která je překrytá kernelem, zatímco average pooling vrací průměr všech hodnot, které jsou překryté kernelem. Názornou zjednodušenou ukázku fungování lze vidět na obrázku 16. [33]



Obrázek 16: Max pooling a average pooling[25]

[25] Zdroj: https://miro.medium.com/max/745/1*KQIEqhxzICU7thjaQBfPBQ.png

Po využití poolingové vrstvy je snížena velikost příznakových map na polovinu. Max pooling má tendenci fungovat lépe než alternativa, kterou je average pooling, proto se dále budeme bavit spíše o metodě max pooling. Je to podobné konvoluci s tím rozdílem, že se oblasti transformují pomocí zabudované operace tensorového maxima, místo naučené lineární transformace. Dalším rozdílem oproti konvoluci je fakt, že se tato operace obvykle provádí za pomoci oken 2x2 a kroku 2. díky tomu se velikost příznakových map sníží o polovinu. Toto zajišťuje, vytvoření hierarchie prostorových filtrů. [19]

3.4.6 Plně propojená vrstva

Jak lze vidět na obrázku 13, plně propojená vrstva (anglicky fully connected layer) se typicky nachází na konci CNN (lze vidět i několik takových vrstev za sebou). Vstupuje do ní výstup z předchozí třídimenzionální vrstvy, který je linearizován do vektoru. Výstup z plně propojené vrstvy je výstupem z konvoluční neuronové sítě. Dá se říct, že se jedná o plně propojenou neuronovou síť bez skrytých vrstev [33]

3.4.7 Učení CNN

Jak již bylo v předchozích kapitolách řečeno, konvoluční neuronové sítě se skládají z konvolučních vrstev, které následují vrstvy poolingové. Tyto vrstvy se mohou řetězit a jejich konečný výstup je napojen na jednorozměrnou neuronovou síť. Taková síť se učí obdobou metody backpropagation, která je upravena pro jednotlivé vrstvy sítě.

Trénovací proces začíná fází dopředného šíření signálu, kde se vstup šíří od začátku sítě ke konci. V tomto kroku každá vrstva načte nějaká data (vstupy, náhodné hodnoty vah), která bude potřebovat pro zpětnou fázi. To znamená, že jakákoli zpětná fáze, musí být použita až po dopředné fázi. Zpětná fáze je nejčastěji implementována metodou backpropagation, ve které každá vrstva dostane gradient a zároveň také vrátí gradient. Vrstvy získávají gradient ztráty v souvislosti ke svým výstupům ($\partial L/\partial out$) a vrací gradient ztráty v souvislosti na své vstupy ($\partial L/\partial in$).

Jako první v brakpropagaci vypočteme cross-entropy loss[26]:

$$L = -ln(P_c)$$

Kde p_c je předpovězená hodnota pro správnou třídu.

[26] Křížová entropie mezi dvěma rozděleními pravděpodobnosti.

Dále je dtřeba vypočítat vstup do zpětné fáze softmaxové vrstvy($\partial L/\partial out_s$) kde out_s je výstup z vrstvy softmax, kde se počítá pouze se správnou třídou.

$$\frac{\partial L}{\partial out_s(i)} = \begin{cases} 0 & if\ i \neq c \\ -\frac{1}{P_i} & if\ i = c \end{cases}$$

Takto získáme náš základní gradient.

V dalším kroku musíme nacachovat 3 důležité věci, které se nám budou hodit v implementaci zpětné fáze. Tyto věci jsou:

- vstup poté co jej převedeme na vektor,
- tvar vstupu, než jej převedeme na vektor,
- vstupy do aktivační funkce softmax (totals)

Po tomto kroku můžeme začít derivovat gradienty pro fázi zpětného šíření. Již jsme derivovali vstup do softmaxové zpětné fáze ($\partial L/\partial out_s$). Díky jeho vlastnosti, že se nerovná nule jen u *c*, což je jeho správná třída, můžeme ignorovat vše, krom out_s(c).

Nyní se spočte gradient z out_s(c) v souvislosti na vstupy do aktivační funkce softmax. Řekněme, že t_i je hodnota vstupu pro třídu *i*.

$$Out_s(c) = \frac{e^{t_c}}{\Sigma_i e^{t_i}} = \frac{e^{t_c}}{S}$$

Nyní si představme nějakou třídu *k*, která není třídou *c*. a využít pravidla zřetězení:

$$Out_s(c) = e^{t_c} S^{-1}$$

$$\frac{\partial out_s(c)}{\partial t_k} = -e^{t_c} S^{-2} \left(\frac{\partial S}{\partial t_k}\right) = -e^{t_c} S^{-2} (e^{t_k}) = \frac{-e^{t_c} e^{t_k}}{S^2}$$

Musíme si pamatovat, že *k* se nerovná *c*. Dále uděláme derivaci pro *c*, pomocí podílového pravidla.

$$\frac{\partial out_s(c)}{\partial t_c} = \frac{Se^{t_c} - e^{t_c}\frac{\partial S}{\partial t_C}}{S^2} = \frac{Se^{t_c} - e^{t_c} e^{t_c}}{S^2} = \frac{e^{t_c}(S - e^{t_c})}{S^2}$$

Díky tomu, že $\partial L/\partial out_s$ se nerovná nule pouze pro správnou třídu *c*, budeme tuto třídu hledat tak, že se budeme koukat po gradientu, který není nula. Poté co ho najdeme, vypočteme gradient $\partial out_s(i)/\partial t$ pomocí výsledků, které jsme derivovali výše:

$$\frac{\partial out_s(k)}{\partial t} = \begin{cases} \frac{-e^{t_c} e^{t_k}}{S^2} & if\ k \neq c \\ \frac{e^{t_c}(S - e^{t_c})}{S^2} & if\ k = c \end{cases}$$

Snažíme se získat gradient ztráty pro váhy, hodnoty bias a vstup. Abychom mohli dosadit do rovnice, kde jsou vstupy do softmax na jedné straně a váhy, bias a vstupy na straně druhé.

$$t = w * input * b$$

Jednotlivé gradienty:

$$\frac{\partial t}{\partial w} = input$$

$$\frac{\partial t}{\partial b} = 1$$

$$\frac{\partial t}{\partial input} = w$$

Složení všeho dohromady:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial out} * \frac{\partial out}{\partial t} * \frac{\partial t}{\partial w}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial out} * \frac{\partial out}{\partial t} * \frac{\partial t}{\partial b}$$

$$\frac{\partial L}{\partial input} = \frac{\partial L}{\partial out} * \frac{\partial out}{\partial t} * \frac{\partial t}{\partial input}$$

Tato část byla zpracována podle [34].

3.4.8 Plně konvoluční síť

Jak již název napovídá, plně konvoluční síť se od CNN odlišuje tím, že všechny její vrstvy jsou konvoluční, nebo poolingové. Neobsahuje tedy žádné plně propojené vrstvy, které by ji zakončovaly.

Plně konvoluční síť (FCN) se učí a rozhoduje pomocí lokálních charakteristik. Zatímco u CNN připojení plně propojených vrstev zajišťuje síti učit se i z globálních charakteristik. Nevýhodou plně propojených vrstev je, že potřebují vstup v určitém formátu a díky jejich absenci u FCN lze zajistit zpracování obrázku v téměř libovolné velikosti.

Další výhodou u FCN je, že neztrácejí informace o umístění. Plně propojené vrstvy tyto informace ztrácejí. Je tomu způsobeno stejně jako u předchozí nevýhody, nutnou transformací dat. [35, 36]

3.4.9 Existující implementace

V dnešní době již existuje velké množství implementací konvolučních neuronových sítí. Jde často o knihovny nebo toolboxy, které programátorům poskytují řadu základních nástrojů. V dnešní době lze pro výpočty, místo procesoru (CPU), využívat grafický procesor (GPU). GPU využívá dva druhy programování, a to CUDA a OpenCL.

CUDA (anglicky Computer Unified Device Architecture) je platforma vytvořená společností nVidia. Využití CUDA je velmi dobře popsáno Františkem Doupalem v jeho článku NVIDIA CUDA – využití grafické karty naplno z roku 2009 „*Každá grafická karta v dnešní době v závislosti na svém typu a cenové kategorii obsahuje určité množství výpočetních jednotek – stream procesorů. Nabízí se tedy otázka, proč jich nevyužít i jinak než v graficky náročných operacích nebo při hraní her. A právě o tomto NVIDIA CUDA je*. [37]“. Tento takzvaný stream procesorů se využívá, když není potřeba pro jinou činnost, pro kterou byl primárně určen, jako je práce s grafikou. Díky tomu, že klasické procesory fungují jinak a ani jejich hrubý výpočetní výkon nelze porovnávat s GPU, u kterých je podstatně vyšší díky jejich množství výpočetních jednotek. U moderních karet tak můžeme mluvit až o několika stech výpočetních jednotek. CUDA při svých výpočtech dělí složitější výpočty na jednodušší, které následně rozděluje mezi jednotlivé stream procesory. Největší nevýhodou CUDA je, že funguje pouze na grafických kartách od společnosti nVidia. [37, 38]

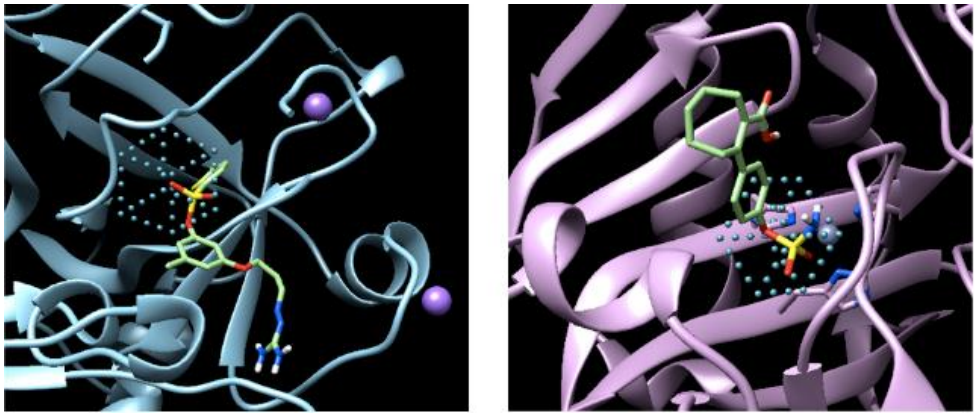
OpenCL (anglicky Open Computing Language) je otevřeným frameworkem pro psaní programů, které se spouští v heterogenním prostředí (CPU, GPU, DPU a jiných procesorech). Již z tohoto lze říct, že se nejedná o variaci CUDA. Díky němu je možné nechat vypočítávat kód na několika procesních jednotkách. Například, pokud potřebujete vypočítat sin(x) na velkém vektoru, řekněme miliony čísel. OpenCL detekuje jaké procesní jednotky lze využít, dá na výběr jejich statistiky a nechá vás vybrat nejlepší možnost nebo i několik procesních jednotek. Následně rozdělí data pro jednotlivé procesní jednotky a pošle jim je. Díky tomu, je výpočetní čas o mnoho kratší. [39]

3.4.10 Využití konvolučních neuronových sítí

Konvoluční neuronové sítě byly navrženy pro rozpoznávání obrazových vzorů z obrazu s co nejmenším předzpracováním. V posledních letech konvoluční neuronové sítě

zažívají obrovský rozvoj a jejich využití tak narůstá. Tato kapitola se zabývá právě několika možnostmi využití CNN.

Mezi ta zajímavější využití CNN patří určování nemocí. Roku 2018 United States Food and Drug Administration povolil využívání stroje využívající Konvoluční neuronové sítě k detekci diabetické retinopatie [27]. Zpracování obrazu a celkově využití umělé inteligence v medicíně je teprve v začátcích, ale slibuje velký pokrok pro lidstvo. V budoucnu by tak mohla CNN klasifikovat nemoci, což by doktorům mohlo uvolnit čas na důležitější úkoly. Pokud by se CNN zařadily do zdravotní péče správně, mohlo by to zlepšit péči o pacienty a snížit ceny za zdravotní úkony. [40] Další využití v medicíně je pro hledání nových léků, kde CNN hledá nové kombinace molekul a biologických proteinů a predikuje jejich vlastnosti a interakce. V roce 2015 byl představen AtomNet, první CNN pro designování léků. Tento systém se trénuje rovnou na trojrozměrných reprezentacích chemických interakcí. Stejně jako se konvoluční neuronové sítě učí rozeznávat z malých jednotlivých rysů obrazu větší a komplikovanější struktury, tak i AtomNet objevuje chemické vlastnosti a nové látky skládáním z menších struktur. [41, 42]



Obrázek 17: AtomNet, který se učí rozeznat skupiny sulfonyl

Další využití CNN je v rozeznávání lidské řeči. Cílem automatického rozpoznání řeči (anglicky Automatic Speech Recognition, zkráceně ASR), je přepsání lidské řeči do textové podoby. Sdružuje vědomosti lingvistiky a informatiky. Jedná se o velmi obtížný úkol, protože je lidská řeč pro každou osobu odlišná a skládá se z několika aspektů, jako je intonace, tón hlasu a dalších. [43]

[27] Jedná se o nezánětlivé onemocnění oční sítnice, vzniklé důsledkem celkového postižení cév u diabetu mellitu.

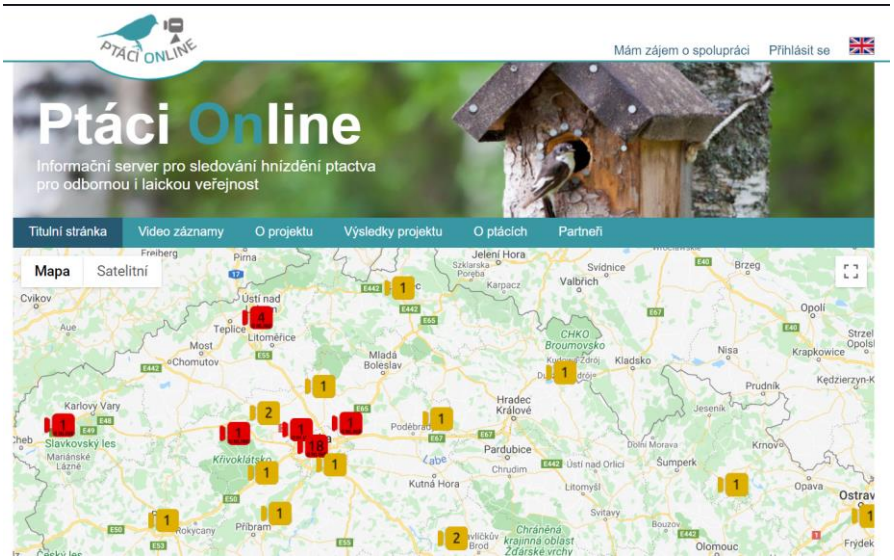
DeepDream (česky hluboký sen) je modifikace obrazu, která využívá reprezentace naučené konvoluční neuronové sítě. Byla vydána společností Google v roce 2015 a stala se internetovou senzací díky zvláštním obrázkům, které dokáže generovat. Takový obrázek lze vidět na obrázku 18. Upravené obrázky pomocí DeepDream jsou plné ptačího peří, psích očí a celkově zvířecích motivů. Toto bylo vedlejším produktem toho, že CNN DeepDream byla trénována na datasetu ImageNet, ve kterém jsou v nadměrné většině zastoupeny různá plemena ptáků a psů. [19]

Obrázek 18: Pomeranče upravené pomocí DeepDream generátoru[28]

Jako praktické využití CNN lze ukázat na projekt Bird`s Online. Tento projekt je realizován od roku 2014 Fakultou životního prostředí České zemědělské univerzity v Praze. Na tomto projektu se podílí i pan Ing. Josef Pavlíček, Ph.D., který je vedoucí této diplomové práce. Hlavním cílem tohoto projektu, je přiblížit vědeckou činnost široké veřejnosti. Základem projektu jsou tzv. „chytré ptačí budky“. Tyto závěsné ptačí byly vyvinuty pro monitorování průběhu hnízdění sýce rousného. Jedná se o závěsné ptačí budky. V konečné fázi má být projekt schopný sledovat ptáky v budkách, sledování jejich vývoje, jejich potravy a také pro ně možného nebezpečí. Tyto záznamy lze následně použít jako učební pomůcka, pro ochranu ohrožených druhů, nebo pro vědecké účely. Celé to funguje pomocí již výše zmíněných inteligentních hnízdech, ve kterých je umístěn počítač, snímač pohybu umístěný u vstupního otvoru, jedna či dvě kamery s nočním osvětlením, mikrofon, vnitřní i venkovní snímač teploty a osvětlení. Přenos dat a napájení elektroniky je zajištěno pomocí ethernetového kabelu, jehož doporučená vzdálenost od budky k ethernetové zásuvce je do 50 metrů. Tento projekt dovoluje sledovat online video přenos z jednotlivých hnízd. Hnízdo pro sledování lze jednoduše vybrat na hlavní stránce viz Obrázek 19. Jednotlivá hnízda mají

[28] Zdroj: vlastní obrázek upraven v https://deepdreamgenerator.com/

u sebe informace o tom, zda dovolují živý přenos, zda probíhá hnízdění a jestli je tu možnost prohlédnout si aktivity u daného hnízda. [44]



Obrázek 19: Titulní stránka projektu Birds Online s mapou jednotlivých hnízd[29]

3.5 Google Colaboratory

Také známé jako Colab, jedná se o produkt od společnosti Google. Colab dovoluje komukoli psát a pouštět python kód přímo v prohlížeči a je velmi dobře využitelný pro strojové učení, analýzu dat a učení se. Přímo řečeno je to služba poskytující hostovaný Jupyter notebook, který nepotřebuje žádné nastavení, nebo výpočetní sílu, jako například GPU, a je zdarma. Nemusíte tedy provádět jakékoli výpočty na svém počítači, vše běží na straně Googlu, na virtuálních strojích. Je to tedy skvělá varianta pro ty, co chtějí začít s hlubokým učením a nemají k tomu potřebné vybavení. [45]

3.5.1 Využívání Colabu

Notebooky Colabu jsou uložené na Google Drive, nebo mohou být nahrány přímo z GitHubu. Lze je využívat zdarma pouze s účtem od společnosti Google, stačí Vám tedy pouze být přihlášen pod svým účtem a přejít na webové stránky Colabu, které jsou *https://colab.research.google.com/*. Zde své notebooky můžete nejen využívat, ale také sdílet. Jednotlivé virtuální stroje při sdílení neobsahují žádné upravené knihovny ani složky, je tedy potřeba vytvořit kusy kódu, které je nahrají.

[29] Zdroj: Snímek obrazovky z titulní strany webové stránky https://www.birdsonline.cz/cz/homepage

Kód je spouštěn na virtuálním stroji, který je privátní a svázaný s Vaším účtem. Tyto virtuální stroje jsou smazány, pokud nějakou dobu nic neuděláte a mají maximální dobu životnosti, nastavenou společností Google. Vytvořené notebooky Jupyter lze uložit na Google Drive ve formátu .ipynb, což je formát Jupyter notebooků. [45]

3.5.2 Jupyter notebook

Projekt Jupyter je open-source software, podporující několik programovacích jazyků. Jméno Jupyther bylo vytvořeno krom jiného ze spojení tří hlavních programovacích jazyků, které podporuje a to Julia, Python a R. Kombinuje schopnost spouštět kód Pythonu, možnosti úpravy textu a anotaci toho, co děláte. Kód pythonu zde lze také rozdělit na jednotlivé menší kousky, které lze spouštět samostatně. Díky tomu nemusíme spouštět celý předchozí kód, když se někde stane chyba. [19]

3.5.3 Limitace

Možnosti Colabu nejsou neomezené, a protože jsou poskytovány zdarma, tak nemusí být vždy dostupné v té nejlepší kvalitě. Mezi limitace patří ukončení virtuálního stroje pokud se na něm dlouho nic neudělalo, maximální časová životnost virtuálních strojů, typy GPU k dispozici, a další faktory, které se v čase využívání mění. Colab nepublikuje přesné specifikace těchto limitů, protože se mohou měnit velmi často. Výpočetní síla je většinou přidělována především uživatelům, kteří využívají Colab více interaktivně, než těm, kteří ho využívají pro náročné a dlouho trvající výpočty. Dále bývají upřednostňováni uživatelé, kteří v poslední době využívali méně zdrojů Colabu. Kvůli tomu uživatelé, kteří využívají Colab pro dlouho trvající výpočty, a využívají tak více zdrojů, mají větší pravděpodobnost, že jim přidělené zdroje budou dočasně omezeny. Pokud tedy chcete pracovat s Colabem častěji, zkuste zavírat okna Colabu, a již nepotřebujete, dále si nevybírejte GPU na výpočty, pokud to není nezbytně nutné pro vaši práci. Takto lze snížit možnost, že narazíte na omezení využívání Colabu. [45]

Typy GPU k dispozici se v průběhu času mění. To je důležité, aby byla tato služba zdarma. Často ovšem zahrnují Nvidie K80, T4, P4 a P100. Neexistuje ovšem způsob, jak si zvolit typ přidělené GPU.

Virtuální stoje jsou přiděleny nejvíce na 12 hodin. Pokud je ovšem virtuální stroj déle bez jakékoli akce, tak se může odpojit i dříve, než za daných maximálních 12 hodin.

Maximální doba propůjčení virtuálního stroje se ovšem může měnit, a to podle využití služby uživateli.

Jednotlivé virtuální stroje mají přidělenu určitou velikost paměti, která se může lišit při každém přidělení. Někdy může dojít k automatickému přidělení větší paměti, pokud Colab detekuje, že ji budete s větší pravděpodobností potřebovat.

Veškeré tyto limity řeší placený Colab Pro, jedná se o předplatné na měsíc za zhruba 255,- Kč. Colab Pro zajišťuje životnost virtuálních strojů po až na 24 hodin, méně odpojení kvůli nečinnosti stroje, vyšší prioritu na přidělení rychlejší GPU, a dále je možnost nastavení větší paměti u svých virtuálních strojů. Colab Pro je zatím k dispozici pouze ve Spojených státech amerických. [45]

4 Vlastní práce

Cílem praktické části je vytvoření konvoluční neuronové sítě za pomoci služby Google Colaboratory, která by měla být schopná rozeznávat pomeranče s co největší přesností. V této kapitole bude popsána služba Google Colaboratory, její prvky a možnosti, jak v ní pracovat. Dále bude popsán postup získávání a úpravy dat pro učení. Postup získání dat z Open Images V6 a následně bude vytvořena druhá konvoluční neuronová síť schopna rozeznávat pomeranče.

4.1 Google Colaboratory

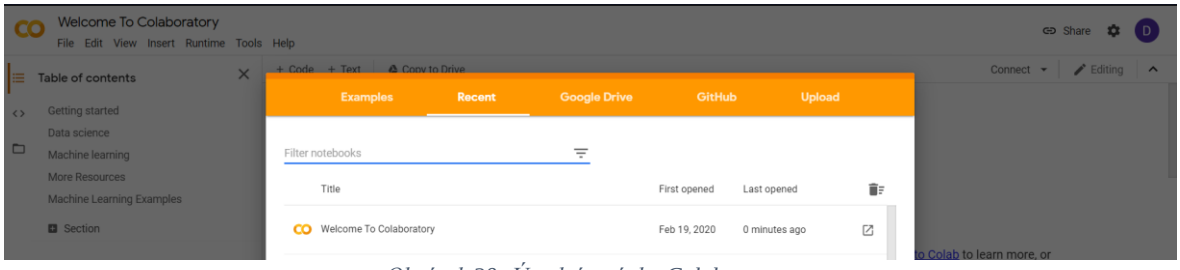
Jak bylo zmíněno v teoretické části, jedná se o službu poskytující zdarma notebooky Jupyter, ve kterých můžeme spouštět kód pythonu.

Pokud chcete využívat služeb Colabu, musíte vlastnit Google účet a být na něm přihlášeni.

4.1.1 Uživatelské rozhraní

První stránka, kterou uvidíte hned po přihlášení, je vidět na obrázku 20. Zde je možné vybrat virtuální notebook Jupyter (VM), který se vám má otevřít.

V záložce examples, jak již název napovídá, najdete příklady a návody, které lze v Colabu využít.



Obrázek 20: Úvodní stránka Colaboratory

Jednotlivé VM jsou listy buněk, do kterých píšete. Existují dva základní typy buněk. Oba typy buněk lze vidět na obrázku 21.

Prvním typem jsou textové buňky, které můžete editovat pouhým dvojklikem na danou buňku. Textové buňky slouží pro popsání kódu, psaní návodů. Dále je lze využít pro

matematické výpočty s využitím LaTeX[30]. Matematické výpočty jsou renderované pomocí MathJax[31]. Do textových buněk lze psát a přidávat obrázky, ale nelze je nijak spouštět.

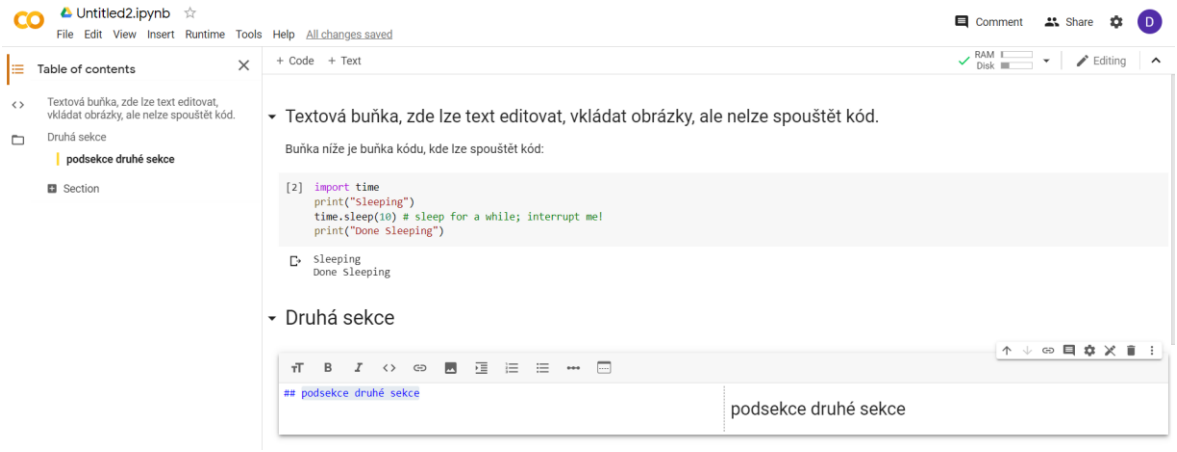
Druhým typem jsou buňky kódu, do nich se zapisuje kód, který jde následně spouštět. Toto spouštění jde zajistit těmito způsoby:

- Kliknutím na ikonku přehrání u dané buňky (vlevo nahoře u každé buňky).
- Zmáčknutím kláves Ctrl+Enter se spustí pouze daná buňka.
- Zmáčknutím Shift+Enter se spustí daná buňka a přesune se na buňku další, pokud další neexistuje, tak se vytvoří nová.
- Zmáčknutím Alt+Enter se spustí daná buňka a za ní se vytvoří nová.

Mezi kód, který lze spouštět patří i notace magics from Jupyter[32]. Komentování na buňkách kódu je pomocí znaku „#“, kde vše za tímto znakem v jednom řádku je komentářem.

Vlevo na obrázku 21 si můžeme všimnout okna „*Table of contents*“ (česky tabulka obsahu). Tato tabulka se strukturuje pomocí textových buněk, ve kterých je jako první možnost nad danou buňkou tlačítko *Toogle heading*. Při každém kliknutí na toto tlačítko se označený text posune o jednu úroveň nadpisu od nadpisu 1 až po podnadpis úrovně 3. Pod tabulkou obsahu je *Code snippets*[33], ve kterých lze vyhledávat. Jednotlivé snippety pak lze jedním kliknutím přidat přímo do VM. Mezi těmito snippety lze najít například kód pro stahování a nahrávání souborů na Google Drive, stahování souborů na lokální počítač, nebo i ukládání snímků z kamery. Poslední záložka v levém panelu je průzkumník souborů na dané VM. Průzkumník souborů lze využívat pro stahování souborů z VM na lokální počítač, nebo naopak pro upload souboru z lokálního počítače na VM. Pro stažení souboru na něj stačí kliknout pravým tlačítkem a dát *download*. Pro upload souboru se musí kliknout pravým tlačítkem na složku, do které chceme soubor nahrát, a zvolit *upload*. V okně, které se otevře následně vybereme soubor z lokálního počítače, který chceme nahrát.

[30] LaTeX je balík maker typografického systému TeX. Dokument LaTeXu se píše jako text, do kterého se vpisují formátovací příznaky, podobně jako tomu je v HTML.
[31] MathJax je JavaScriptový display engine pro matematiku.
[32] Magics je systém příkazů IPythonu, který efektivně poskytuje mini-příkazový jazyk.
[33] Snippet je znovu využitelný kód, který provádí nějakou funkci. Často se jedná o malou část kódu, která lze využít ve větším a komplikovanějším kódu.



Obrázek 21:Typy buněk a základní orientace v Google Colaboratory

Nejčastěji se v buňkách kódu pracuje s pythonem. Notebooky Jupyter obsahují zkratky pro základní operace, jako je ls, mv, rm, rmdir. Tyto operace můžete znát například z Linuxu. Pokud nějakou z nich chcete využít, je nutné dodržet syntaxi, která říká, že před každou takovouto operací musí být uveden znak vykřičníku. Příklad operace ls, a tabulku podporovaných operací, lze vidět na obrázku 22.

Obrázek 22: Využití operace ls, která vypsala obsah bin[34]

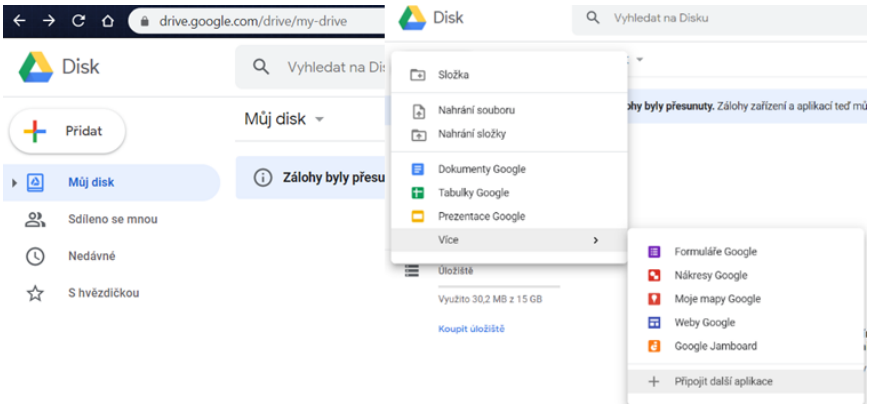
4.1.2 Propojení Google Drive s Colabem

Abychom si mohli ukládat postup a rychleji načítat naši práci, je dobré si propojit Colab s Google Drive. Tímto zajistíme, že se dostaneme rychleji k již dosaženým výsledkům. Pro toto propojení stačí být přihlášen pod Google účtem, jít na adresu drive.google.com v levém menu kliknout na tlačítko **přidat**, dále na **více**, **připojit další**

[34] Ve složce bin jsou uloženy jednotlivé podporované základní operace, jako je ls, cp, cat.

aplikace. V okně, které se po tomto otevře, se v pravém horním rohu dá vyhledat **colaboratory**. Nyní stačí pouze vybrat Colaboratory, kliknout na **instalovat** a vybrat účet. Takto zajistíme možnost spouštět uložené notebooky Jupyter na Google drive, kde se po rozkliknutí samy spustí v Colabu. Toto si nepleťme s propojením Colabu s Google Drive, pro které je potřeba spustit určitý kus kódu v Colabu.

Nyní je možné spouštět vaše uložené notebooky přímo z Google Drive.



Obrázek 23: Propojení Google Drive s Colaboratory

4.1.3 Propojení Google drive se složkou v PC

V tomto kroku je potřeba nainstalovat Backup and Sync from Google. Tento program je určen pro zálohování fotek, videí a důležitých dokumentů a umožňuje tak jejich zobrazení na všech propojených zařízeních. My ho ovšem využijeme pro propojení složky v počítači s Colaboratory.

V tomto programu nastavíme složku, kterou chceme automaticky propojit s Google Drive. Pokud něco vložíme do této složky, automaticky se daný soubor přenese i na Google Drive, kam můžeme přistoupit z Colabu.

4.1.4 Propojení Colabu s Google Drive

Pro propojení Colabu s Google drive je nutný na dané VM spustit tento kód:

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

Po spuštění tohoto kódu se vypíše „ Go to this URL in a browser: „ a odkaz, na který je nutné kliknout. Jedná se totiž o ověřovací proces. Na nově zobrazeném okně vyberete svůj

účet, povolíte práva pro Colab a nakonec zkopírujete kód. Tento autorizační kód následně vložíte do tabulky pod spuštěným kódem v Colabu, která má nad sebou text „Enter your authorization code:“ a zmáčkene tlačítko Enter. Takto jste nastavili propojení Colabu s Google Drive. Nyní by měla být viditelná složka Drive v levém okně pod záložkou *Files*.

4.2 Vlastní zpracování dat pro učení

4.2.1 Získání obrázků pomerančů

Obrázky byly získány z několika zdrojů, protože ani jeden ze zdrojů samostatně neobsahoval dostatečný počet kvalitních obrázků, vhodných pro tuto diplomovou práci.

Prvním zdrojem byl volně dostupný soubor fotek jednoho pomeranče ze všech úhlů[35]. Jedná se o dataset vysoké kvality i s jinými obrázky, než jsou jen pomeranče. Obsahuje celkově 82213 obrázků a 120 druhů ovoce a zeleniny. Z tohoto výběru jsem si vybral pouze 7 obrázků z co nejvíce odlišných úhlů.

Tento soubor se mi zdál nedostačující, tak jsem se rozhodl rozšířit data o mnou pořízené fotky pomerančů, které jsem se snažil dělat co nejvíce různorodé. U jednotlivých fotek jsem se snažil měnit pozadí, úhel fotek, osvětlení, počet a seskupení pomerančů. Jednotlivé obrázky mají rozměry 2268x4032 pixelů nebo 4032x3024 pixelů. Příklady pořízených obrázků lze vidět na obrázku 24. Některé z těchto obrázků jsem následně ořízl, aby obsahovaly co nejméně přebytečného pozadí.

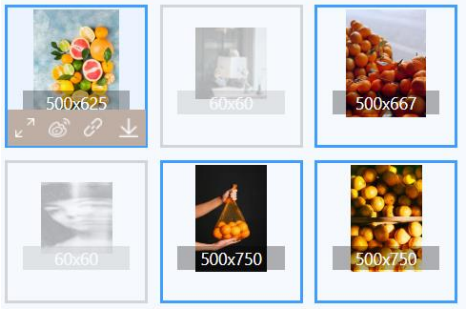
Obrázek 24: Příklady pořízených obrázků

Dalším zdrojem obrázků byla webová stránka Pexels[36], na které jsou k dispozici obrázky s licencí k volnému užití zdarma. Jsou zde obrázky ve vysoké kvalitě a možnost vyhledávat podle klíčových slov. Možnost vyhledávání ovšem nezajistí, že ve výsledcích vyhledávání naleznete pouze relevantní obrázky. K tomuto také přispívá fakt, že

[35] Tento dataset je volně dostupný na adrese: https://github.com/Horea94/Fruit-Images-Dataset
[36] https://www.pexels.com/about/

vyhledávám pomeranč v anglickém jazyce, ve kterém slovo odpovídající pomeranči (orange) určuje také oranžovou barvu. Ve výsledcích vyhledávání jsou tedy často i obrázky, které v sobě obsahují oranžovou barvu.

Pro stažení obrázků ze stránky Pexels jsem nestahoval obrázky po jednom, ale využil jsem plugin Fakun Batch Image Download, který je zdarma dostupný pro prohlížeč Google Chrome. Poskytuje jednoduchou možnost stáhnout obrázky vyskytující se na zvolené kartě, případně obrázky ze stránek na všech otevřených kartách. Lze zde třídit obrázky dle šířky, výšky, filtrovat obrázky pomocí klíčových slov, a hlavně se zde dá vybrat, které z obrázků stáhnete a které ne. Takto jsem protřídil 942 obrázků ze stránky Pexels na pouhých 88, které vyhovovaly pro účely práce. Vybrané obrázky obsahují nerozříznutý, neoloupaný pomeranč a daný obrázek není upravený žádným filtrem, aby se nevhodně neupravila informace o barvě pomeranče. Výběr obrázků lze vidět na obrázku 25, kde modře ohraničené jsou vybrané pro stáhnutí a zašedlé se nestáhnou.



Obrázek 25: Vybrané obrázky pomerančů v pluginu Fakun Batch Image Download

Jako poslední zdroj dat jsem vybral Google vyhledávání obrázků. Zde jde filtrovat dle několika parametrů jako je velikost, barva, typ, čas a také dle práv k užití obrázků. Zde jsem po vyfiltrování stáhl obrázky pomocí výše zmíněného Fakun Batch Image Download pluginu. Pro poslední zdroj dat jsem se rozhodl kvůli nedostatečnému množství dat pro učení z předchozích zdrojů.

4.2.2 Labelování

Labelování je označení vstupních dat, zda, případně kde, obsahují nějaký prvek. V našem případě se jedná o označování pomerančů na obrázku. Takto označkovaná data pomáhají počítači rozpoznat jednotlivé aspekty obrázku a pro učení CNN se jedná o zásadní krok.

Jedny z nejčastějších typů označení jsou takzvané „*bounding boxes*“ které na obrázcích ohraničují objekty a určují jejich třídy. V našem případě budou označené pomeranče, a každý ohraničený pomeranč bude v *bounding boxu* s třídou „pomeranc“. Vynechávám diakritiku, abych si byl jist, že nebude při celém řešení diplomové práce dělat problémy s jakýmkoli z potřebných kroků.

Pro labelování jsem vybral jeden z nejpopulárnějších grafických nástrojů pro labelování obrázků, který se jmenuje LabelIMG. Je napsán v Pythonu, a tak uživatelé operačních systémů, jako jsou Ubuntu, Linux nebo Max potřebují alespoň verzi Pythonu 2.6, doporučuje se ale Python 3, nebo výš. Pro své grafické rozhraní využívá Qt, což je framework pro vytvoření uživatelského rozhraní. [46]

Pro využití LabelIMG na operačním systému Windows je nutné nainstalovat *Python*, *PyQt5* a *instal lxml*. Další možností využití LabelIMG je s nainstalováním Anaconda (Python 3+). Tuto možnost jsem si zvolil já. Po instalaci jsem si otevřel příkazový řádek Anacondy a šel do adresáře s labelImg. Dále jsem do příkazového řádku Anacondy napsal následující kód, který spouští LabelImg.

conda install pyqt=5

pyrcc5 -o libs/resources.py resources.qrc

python labelImg.py

python labelImg.py [IMAGE_PATH] [PRE-DEFINED CLASS FILE]

Tabulka klávesových zkratek pro LabelIMG:

Klávesa	**Její činnost**
Ctrl + u	Načtení všech obrázků ze složky
Ctrl + r	Změna defaultní složky, kam se ukládají anotace
Ctrl + s	Uložení
Ctrl + d	Zkopírování zvoleného labelu a bounding boxu
Space	Označení obrázku za ověřený
w	Vytvoření nového bounding boxu
d	Další obrázek
a	Předchozí obrázek
del	Vymazání vybraného bounding boxu
Ctrl ++	Přiblížení

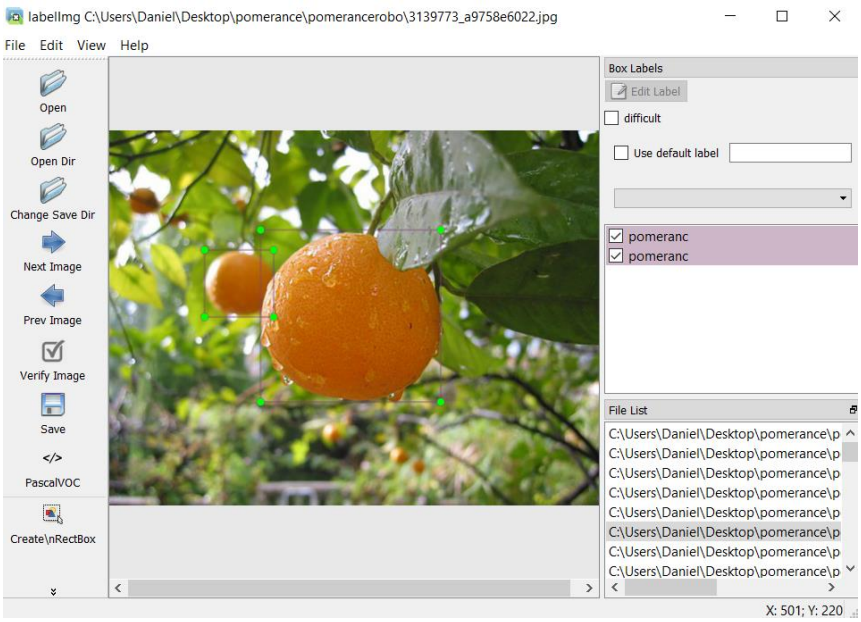
Ctrl --	Oddálení
Klávesové šipky	Posunutí vybraného bounding boxu

Tabulka 1: Klávesové zkratky pro LabelIMG

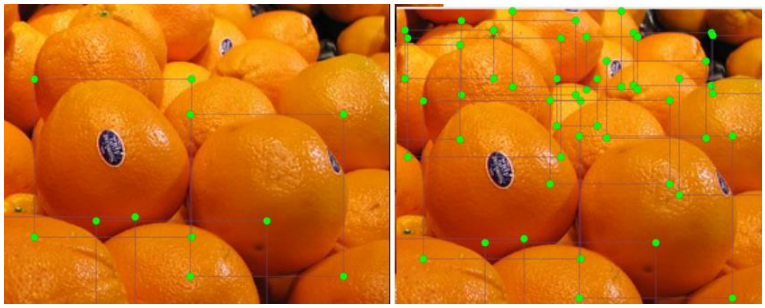
Výše zmíněné klávesové zkratky nám velmi pomohou při labelování, jelikož se jedná o sadu několika set obrázků, na kterých je třeba označit jednotlivé pomeranče. Klávesové zkratky tedy tuto činnost dokážou značně urychlit a zjednodušit. Nejčastěji používané klávesové zkratky pak budou *Ctrl+s*, *w*, *d* a *a*.

Dávejte pozor, pokud zavřete příkazový řádek Anakondy, zavře se i LabelIMG, při zpracovávání této diplomové práce se mi to několikrát stalo. Nic se neděje a žádná data neztrácíte, protože po každém označeném obrázku ukládáte soubor s bounding boxy, pouze musíte znovu otevírat Anakondu a LabelIMG, otevřít složku s obrázky a najít poslední olabelovaný obrázek.

Grafické rozhraní a jednotlivé označování lze vidět na obrázku 26. Vedoucí této diplomové práce pan Ing. Josef Pavlíček, Ph.D. mi poradil, že při labelování nemám označovat malé a vzdálené pomeranče, stejně tak nemám označovat pouze malé viditelné kousky pomerančů, protože by se pak konvoluční neuronová síť při trénování naučila rozeznávat pomeranč i v malém kousku pomeranče. Konvoluční neuronová síť naučená na takovýchto datech, by poté mohla například najít a označit tři různé pomeranče na jednom skutečném pomeranči. Vlevo na obrázku 27 lze vidět správné labelování, naopak vpravo je vidět, jak by se labelovat nemělo.



Obrázek 26: Labelování obrázku v LabelIMG



Obrázek 27: Vlevo - správné labelování; Vpravo - nesprávné labelování

Po každém olabelovaném obrázku je před pokračováním na obrázek další potřeba uložit výstup do souboru se stejným názvem, jako má obrázek a s .xml příponou. V tomto XML souboru jsou uložené jednotlivé bounding boxy k danému obrázku s jejich třídami. Každý z těchto bounding boxů má u sebe čtyři hlavní hodnoty, a to souřadnice pro každý z jeho rohů, které určují daný bounding box na obrázku. Jeden takový bounding box ve formátu XML vypadá následovně:

```
<object>
    <name>pomeranc</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
        <xmin>232</xmin>
        <ymin>67</ymin>
        <xmax>349</xmax>
        <ymax>184</ymax>
    </bndbox>
</object>
```

Obrázek 28: Bounding box v XML formátu

Díky obrázku 28 můžeme říct, že k labelování nepotřebujeme žádný nástroj a můžeme labelovat ve formátu XML ručně. Takovéto labelování by ale zabralo nesmírně času a práce, proto byly vymyšleny grafické nástroje, které takovouto práci značně usnadňují.

4.2.3 Předpříprava pro učení

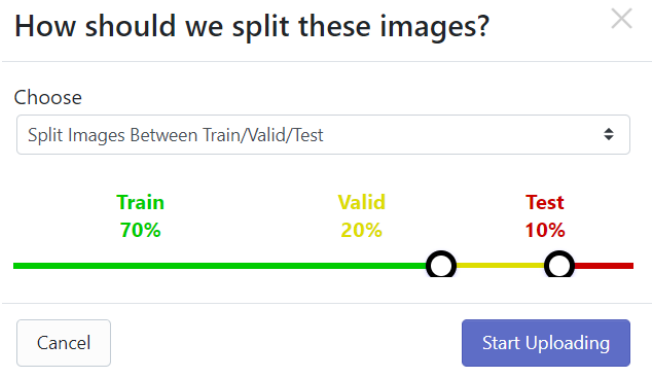
Dalším krokem je příprava obrázků pro učení. Jedná se o úpravu fotografií na vhodný formát.

Pro tento krok jsem si vybral webovou aplikaci zvanou RoboFlow[37], která umožňuje nahrát veškeré olabelované obrázky a následně je automaticky upravit. Tato aplikace je zdarma a pro její využívání si musíte vytvořit účet.

[37] Dostupná na https://app.roboflow.ai/

Po přihlášení se klikne na horní liště na záložku **Datasets**. Zde se klikne vpravo nahoře na tlačítko **Create Dataset**, dále se vyplní jméno datasetu. Jedná se pouze o jméno uložené v RoboFlow a nijak neovlivňuje následné učení. Dataset Type se nastaví na Object Detection (Bounding Box). A na závěr vytváření nového datasetu se vyplní třídy, které využíváme. V našem případě se vyplní „pomeranc“.

Do nově zobrazeného okna se nahrají jednotlivé obrázky s jejich textovými soubory, které obsahují bounding boxy. Soubory s bounding boxy byly vytvořeny v kapitole 4.2.2. Labelování. RoboFlow zobrazí obrázky k nahrání i s jejich bounding boxy. Nahoře nad nahrávanými obrázky je vidět počet všech obrázků, počet anotovaných a počet neanotovaných. V tomto kroku lze vyhodit obrázky které nechcete dát do datasetu. Pro nahrání obrázků se klikne vpravo nahoře na **Start Uploading**. Zde je možnost rozdělit obrázky na trénovací, validační a testovací množinu. Pro náš příklad byla vybrána možnost „Split images Between Train/Valid/Test“ v procentuálním nastavení a klikne se na **Start Uploading**. Okno s možností výběru rozdělení datasetu lze vidět na obrázku 29.



Obrázek 29: Rozdělení datasetu v RoboFlow

4.3 Vytvoření konvoluční neuronové sítě

V této kapitole jsou pospány použité způsoby vytvoření CNN, které dokážou rozeznávat pomeranče z vložených obrázků.

Jako první bude představena možnost využití výše zmíněného Roboflow s vlastními daty na modelu YOLOv3 za pomocí nástrojů a knihoven PyTorch.

Následně bude rozebrána tvorba CNN za pomoci Darknetu. V této části je popsána možnost získání dat pro učení z Open Images Dataset V6, která jsou v této části využita

pro dotrénování hodnot vah. V každé z obou kapitol bude ukázán a popsán výsledek trénování.

4.3.1 RoboFlow YOLOv3 PyTorch

V této kapitole je rozebráno vytvoření CNN, určené k rozpoznávání pomerančů, za pomocí nástrojů a knihoven PyTorch a RoboFlow. Dále jsou zde využita data pro učení získaná v kapitolách výše. Celý Jupyter notebook tvoří soubor příloha1.ipynb této diplomové práce.

Jako první je nutné u virtuálního stroje nastavit výpočet pomocí GPU. Toho je dosáhnuto kliknutím na **Runtime** v horním menu a následné zakliknutí **GPU** u **Hardware accelerator**. Po kliknutí na tlačítko **SAVE** se automaticky připojí GPU jako hlavní výpočetní jednotka. Využití GPU je v Google Colaboratory omezené, ale pro tento příklad dostačující.

Nyní můžeme začít provádět požadované operace. Spuštěním následujícího kódu se importuje prostředí a jsou vypsány informace o stroji se kterým pracujeme.

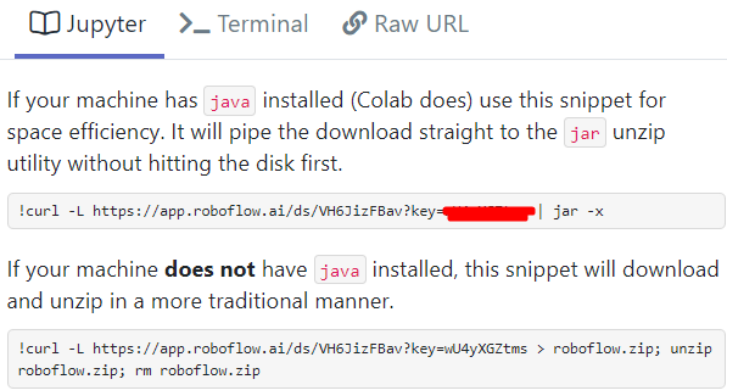
```
import os
import torch
from IPython.display import Image, clear_output
print('PyTorch %s %s' % (torch.__version__, torch.cuda.get_device_properties(0) if torch.cuda.is_available() else 'CPU'))
```

Dalším důležitým krokem je naklonování YOLOv3 pro využití s RoboFlow

```
!git clone https://github.com/roboflow-ai/yolov3
```

Nyní se získají data pro učení z RoboFlow. Toho dosáhneme vložením kódu vygenerovaného přímo v RoboFlow. Vybere se dříve dataset a v oddílu **Pre-Processing Options** se zvolí možnosti **Auto-Orient** a **Resize** (u které bylo vybráno Fit (black edges) in 416 x 416. Toto automaticky upraví obrázky na námi požadovanou velikost, aniž by to obrázky deformovalo. Případné nové plochy budou na obrázku vyplněny černou barvou. Po uložení těchto změn tlačítkem Save, byl vygenerován kód kliknutím na tlačítko Generate

v pravém horním rohu. Tato akce chvilku trvá, jelikož se upravují jednotlivé obrázky dle námi nastavených možností. Nyní byl vybrán Formát „YOLO v3 Darknet“ a zaškrtnuta možnost „show download code“. Zkopírujeme první kód, jelikož Google Colaboratory má na svých noteboocích nainstalovanou javu. Vygenerovaný kód nikomu neukazujte, protože obsahuje klíč, který slouží jako identifikace uživatele pro RoboFlow. Na obrázku 30 lze vidět příklad vygenerovaného kódu, a také která část kódu obsahuje uživatelský klíč.



Obrázek 30: Získání snippetu pro získání dat z RoboFlow

Po získání dat z RoboFlow byla data upravena a byla dána do struktury, která vyhovuje YOLOv3. Následně byly vytvořeny textové soubory nutné pro trénování. Příklad kódu pro vytvoření takového textového souboru, nutného pro učení je vidět níže.

```
file = open("train_images_roboflow.txt", "w")
for root, dirs, files in os.walk("."):
    for filename in files:
      if filename == "train_images_roboflow.txt":
        pass
      else:
        file.write("../train/images/" + filename + "\n")
file.close()
```

Tento kód otevře soubor "train_images_roboflow.txt", pokud neexistuje, tak jej sám vytvoří. Poté prochází soubory v dané složce, a pokud se daný soubor nejmenuje "train_images_roboflow.txt", tak jej zapíše ve formátu "../train/images/" + jméno_souboru + "\n".

Dalším krokem je konfigurace souborů pro trénování. Byl tedy vytvořen soubor s názvy tříd a upraven soubor „roboflow.data“, aby obsahoval správný počet tříd. V našem případě se jedná o jednu třídu.

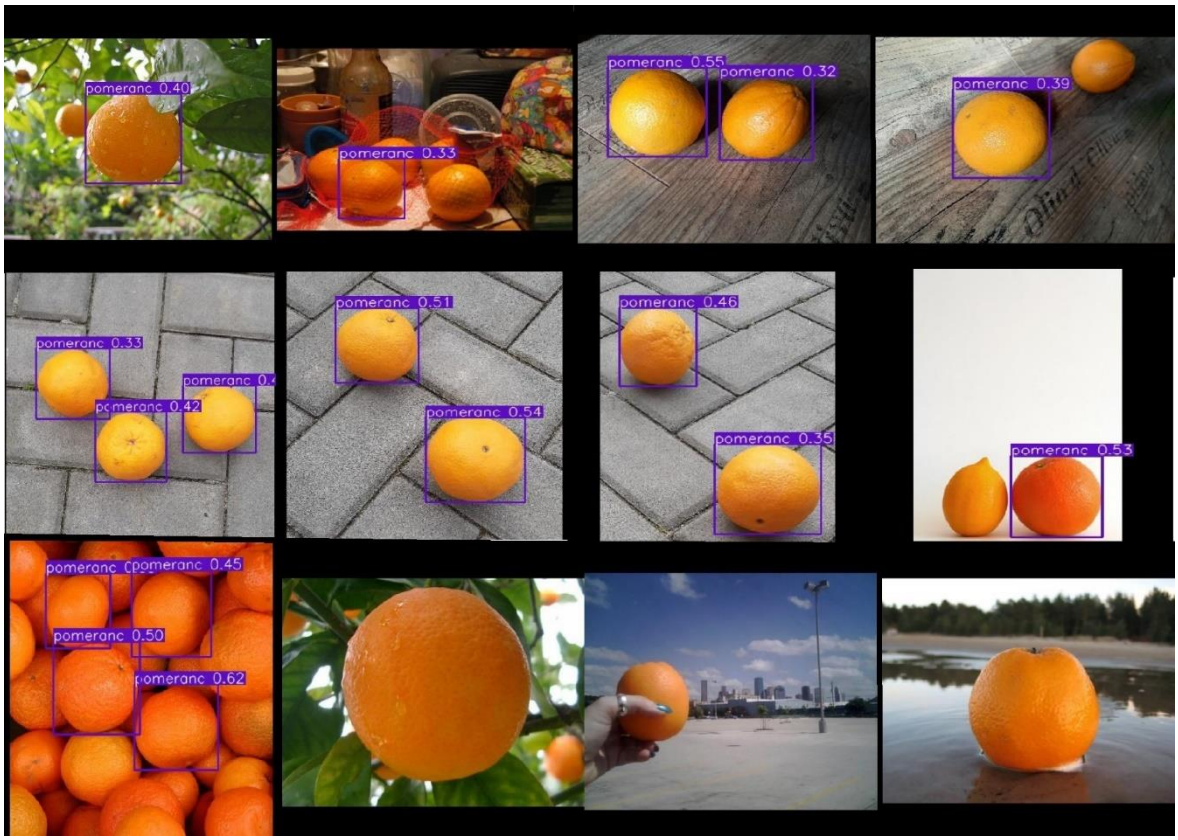
Nyní lze přejít rovnou na trénování modelu, které se spustí následujícím kódem.

!python3 train.py --data data/roboflow.data --epochs 300

Tímto byly natrénovány váhy za přibližně 2 hodiny a 15 minut. Takto krátký čas lze připisovat zvolení malého počtu epoch. Tyto váhy byly následně uloženy a dále využity k predikci výskytu pomerančů na testovacích obrázcích.

!python3 detect.py --weights weights/last.pt --source=../test --names=../train/roboflow_data.names

Výše zmíněný kód vezme natrénované váhy z předcházejícího kroku a predikuje, zda obrázky ve složce „test“ obsahují natrénovanou třídu („pomeranc“).



Obrázek 31: Obrázky s predikcí z YOLOv3 za pomoci RoboFlow a PyTorch

Obrázek 31 obsahuje ukázku obrázků s vytištěnou predikcí. Všechny obrázky lze najít v příloze příloha4.zip.

4.3.2 Získání dat z Open Images V6

V této kapitole je popsán postup pro získání dat z Open Images V6, což je dataset zhruba devíti milionů obrázků s anotacemi. Neobsahuje pouze bounding boxy pro detekci obrazu, ale také segmentaci obrazu a další.

Obrázky jsou svojí tématikou rozsáhlé a jednotlivé bounding boxy byly vytvořeny ručně profesionálními anotátory, což zajišťuje vysokou přesnost a konzistenci. Celý set s bounding boxy obsahuje přesně 15 851 536 bounding boxů a 600 tříd. [47]

Pro získání dat a jejich následnou úpravu byl využit toolkit OIDv4_ToolKit, který byl naklonován na nový Colab Notebook pomocí následujícího kódu. Celý Colab notebook je přiložen jako příloha3.ipynb.

!git clone https://github.com/theAIGuysCode/OIDv4_ToolKit.git

Dále byly nainstalovány důležité knihovny, po kterých byl spuštěn kód na stažení dat s třídou Orange (česky pomeranč).

!python main.py downloader --classes Orange --type_csv train --limit 300

Kde za „--classes“ je vypsána třída, kterou chceme stáhnout a za „--type_csv“ je jaký typ souborů chceme. Jelikož chceme trénovat CNN na datech s bounding boxy, je zde „train“. Poslední parametr je „--limit“, za kterým se udává maximální počet obrázků, které se stáhnou. Po spuštění tohoto kódu se třikrát potvrdilo stahování pomocí vepsání znaku „y“ a zmáčknutí tlačítka enter. Dále byl změněn soubor classes.txt v souboru OIDv4_ToolKit. V tomto souboru bylo napsáno pouze „Orange“ jako název jediné třídy. Po tomto kroku byl spuštěn následující kód.

!python convert_annotations.py

Kód výše generuje textové soubory s anotacemi ve správném formátu pro Darknet. Jednotlivé textové soubory se generují přímo k obrázkům.

Následně byla odebrána složka „Label“, která se nacházela u obrázků a poté byl soubor s obrázky zazipován a stáhnut na lokální PC. Stažení bylo provedeno jednoduše kliknutím na soubor „obj.zip“ a následně kliknutím na Download. Takto byl daný soubor stažen na lokální PC, kde byl rozbalen a složka, ve které byly obrázky, byla přejmenována na „obj“. Tato nová složka byla znovu zazipována a uložena do složky propojené s Google Drive.

4.3.3 YOLOv3 za pomocí Darknet

V této kapitole je rozebráno vytvoření CNN, určené k rozpoznávání pomerančů, za pomocí nástrojů a knihoven Darknetu. Dále jsou zde využita data pro učení získaná z Open Images V6. Celý Jupyter notebook lze nalézt jako příloha2.ipynb této diplomové práce.

Stejně tak jako v předchozí kapitole, i tady bylo jako první nutné u virtuálního stroje nastavit výpočet pomocí GPU. Po nastavení GPU byl spuštěn kód, který zkopíroval Darknet[38].

```
!git clone https://github.com/AlexeyAB/darknet
```

Dále byly provedeny změny v souboru Makefile, díky čemuž byla zajištěna podpora OpenCV[39] a práce s GPU. Po změnách v souboru Makefile je inicializován darknet pomocí jednoduchého příkazu „*!make*“.

V dalším kroku byl připojen Colab ke Google Drive, tento postup je popsán v kapitole 4.1.4. Ihned potom byl přiřazen symbolický link, abychom do „MY\ Drive/“ přistupovali jednodušeji pomocí „/mydrive“. Zároveň tak bylo zabráněno potížím, které by nastaly při udávání cesty s mezerou.

Kód níže zkopíruje zazipovanou složku s daty k trénování z Google Drive.

```
!cp /mydrive/yolov3/obj.zip ../
```

[38] Darknet je open source framework pro neuronové sítě. Je psán v jazyce C a CUDA. Dovoluje výpočty, jak na CPU, tak na GPU. V této diplomové práci byla použita jedna z nejznámějších verzí od AlexeyAB, dostupná z https://github.com/AlexeyAB/darknet

[39] Open Computer Vision knihovna, která obsahuje několik set algoritmů pro práci s počítačovou vizí.

Tato data byla následně rozbalena a uložena do složky /content/darknet/data.

V další části byl upraven soubor yolov3.cfg, který obsahuje strukturu neuronové sítě. Jsou zde popsány jednotlivé vrstvy a nastavení trénování. První byly vytvořeny složky na Google Drive, do kterých byl tento soubor následně zkopírován pomocí kódu níže.

```
!cp cfg/yolov3.cfg /mydrive/yolov3/yolov3_custom.cfg
```

Tento kód při kopírování rovnou soubor přejmenovává na yolov3_custom.cfg. Ten byl otevřen přímo v Google Drive za pomoci aplikace Text Editor. Následně byl upraven začátek tohoto souboru tak, aby byl připraven pro trénování. Přesněji řečeno, byly zakomentovány dva řádky pro testování a odkomentovány dva řádky pro trénování. Dále zde byl změněn počet **max_batches**, který udává maximální počet iterací trénování na jednom, nebo více vzorcích. Na konci každé „batch" jsou predikce z trénování porovnány s očekávanými hodnotami a je vypočítána chyba sítě, díky čemuž jsou následně upraveny vnitřní parametry modelu. [48] Počet **max_batches** by měl být nastaven na hodnotu získanou ze vzorce *2000 * x*, kde *x* značí počet tříd, na které trénujeme. Radí se zde ovšem neudávat hodnoty menší než 4000, což je hodnota, která byla námi nastavena. Řádek **steps** byl upraven na hodnoty 80 % z hodnoty **max_batches** a 90 % z hodnoty **max_batches**. Další úprava proběhla u YOLO vrstev, u kterých byl nastaven počet tříd (**classes**) na 1 a hodnota filtrů (**filters**), v konvolučních vrstvách nad YOLO vrstvami, byla nastavena dle vzorce (x + 5) * 3, tedy na hodnotu 18[40]. Toto shrnuje úpravu souboru yolov3_custom.cfg, který byl následně stažen z Google Drive do složky Darknet v Colabu.

Dále byly na Google Drive pomocí aplikace Text Editor vytvořeny dva soubory. Prvním z nich je obj.names, který obsahuje naši třídu, tedy pouze jeden řádek se slovem „Orange". Druhým vytvořeným souborem pak je obj.data, ten obsahuje počet tříd, na které CNN trénujeme a cesty k důležitým souborům. Jeho obsah je následující:

```
classes = 1
train = data/train.txt
valid = data/test.txt
```

[40] U YOLO vrstev se nachází položka **random,** která má nastavenu hodnotu 1. Takovéto nastavení zajišťuje že se při učení po každých 10 iteracích náhodně změní velikosti obrázků. Díky tomuto si síť zvykne na různé velikosti obrázků a poté má lepší predikce. Pokud bychom to chtěli zakázat, je zde nutné nastavit hodnotu 0, takováto síť by ale měla horší rozpoznávání u obrázků jiných velikostí, než na kterých byla natrénována. [49]

names = data/obj.names

backup = /mydrive/yolov3/backup/

Oba nově vytvořené soubory se následně ukládají do složky data ve složce darknet na Colab notebooku. Upravený soubor yolov3_custom.cfg je zkopírován do složky darknet/cfg.

Dále byl z adresy https://github.com/theAIGuysCode/YoloGenerateTrainingFile stažen soubor generate_train.py a uložen na Google Drive, odkud byl stažen do složky darknet na Colab notebooku. Tento soubor byl následně spuštěn a vygeneroval do složky darknet/data soubor train.txt. Ten obsahuje relativní cesty k našim trénovacím datům. Cestu k tomuto souboru jsme uvedli ve vytvořeném souboru obj.data.

Jako další byly stáhnuty již předtrénované váhy pro konvoluční vrstvy YOLOv3 sítě. Využití předtrénovaných vah zajistí vyšší přesnost a rychlejší trénování CNN.

Samotné trénování sítě se začne spuštěním tohoto kódu:

!./darknet detector train data/obj.data cfg/yolov3_custom.cfg darknet53.conv.74 -dont_show

Tím voláme spustitelný soubor darknet. Parametr train zde značí, že se bude jednat o trénování sítě, dalším parametrem je soubor .data, ve kterém je uveden počet trénovaných tříd a cesty k důležitým souborům. Jako další parametr je soubor .cfg s nastavením učení a struktury sítě. Další parametr je soubor s přednastavenými váhami, který jsme stáhli v předchozím kroku. Posledním parametrem je dont_show, který zajišťuje, že se nebudou ukazovat grafy učení. Tomu je zabráněno, kvůli faktu, že Google Colab nepodporuje možnost otevírat obrázky během chodu kódu.

Trénování sítě trvá přibližně 7 až 8 hodin. Během trénování může dojít k přerušení ze strany Colabu, ať už kvůli neaktivitě, nebo z důvodu překročení životnosti notebooku Colabu. V tomto případě bychom přišli o celý proces trénování. Bylo tedy využito propojení s Google Drive a každých 100 „batchů“ se na Drive do složky yolov3/backup uloží kopie doposud natrénovaných vah. Tato kopie se jmenuje yolov3_custom_last.weights a byla využita v momentech, kdy se přerušilo trénování. Po restartování daného Colab notebooku byly spuštěny buňky kódu v daném notebooku až na buňky se stažením předtrénovaných vah a buňky s kódem pro spuštění trénování. Následně byl spuštěn tento kód pro trénování:

!./darknet detector train data/obj.data cfg/yolov3_custom.cfg /mydrive/yolov3/backup/yolov3_custom_last.weights -dont_show

Jedná se o podobný kód jako v předchozí části, ale místo stažených předtrénovaných vah použijeme poslední uložené váhy na Google Drive z našeho trénování. Takto bylo navázáno na předešlé trénování, bez větších ztrát.

Průběh trénování začíná s vysokou hodnotou průměrné ztrátové hodnoty, kde na začátku může dosahovat až několika tisíc. Tato hodnota se ze začátku velmi rychle snižuje a síť je již použitelná při hodnotě pod 2. Na obrázcích níže lze vidět průběh trénování sítě, kde první hodnota určuje na kolikátém „batchy“ z 4000 jsme. Poslední hodnota, která udává kolik času schází je pouze orientační.

```
87: 918.273315, 1094.153687 avg loss, 0.000000 rate, 4.117647 seconds, 5568 images, 5.212327 hours left
```

Obrázek 32: Trénování sítě - batch 87

Na obrázku 33, kde byl proveden batch číslo 150, můžeme vidět, že nám klesla průměrná chyba, z 1094 na pouhých 73. Dále je na tomto obrázku vidět, že proběhla změna velikosti obrázku, jak bylo nastaveno v souboru yolov3_custom.cfg.

```
 150: 57.325935, 73.743240 avg loss, 0.000001 rate, 3.966468 seconds, 9600 images, 4.402596 hours left
Resizing, random_coef = 1.40

 576 x 576
```

Obrázek 33: Trénování sítě - batch 150

Obrázek níže je začátek výpočtů, při navázání po restartu VM.

```
1300: 2.139724, 2.107430 avg loss, 0.001000 rate, 5.808878 seconds, 83200 images, 3.030610 hours left
```

Obrázek 34: Trénování sítě - batch 1300, navázání po restartu VM

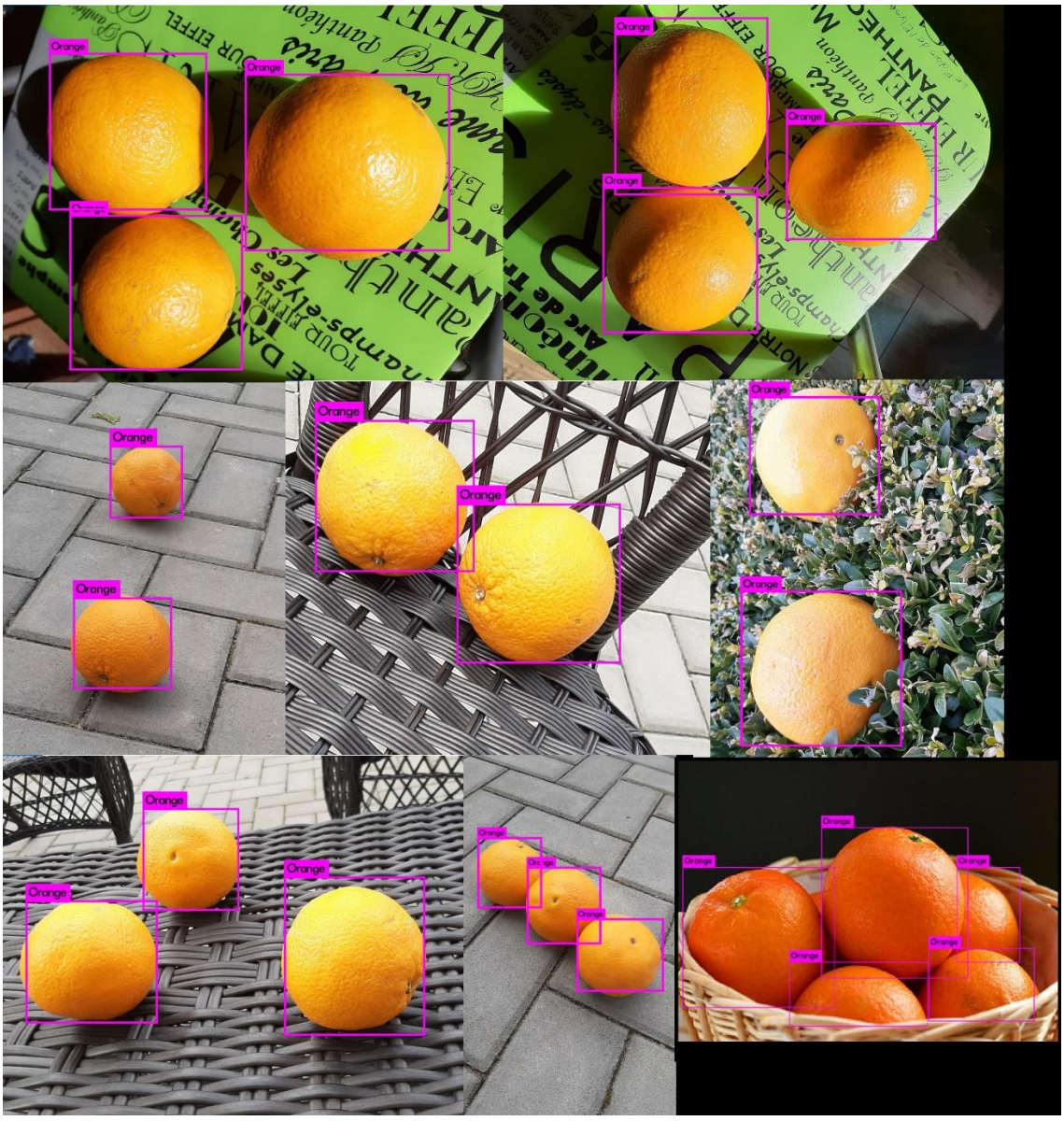
Po natrénování sítě byla provedena úprava souboru yolov3_custom.cfg k jeho nastavení do módu pro testování.

Následná detekce byla provedena za pomoci následujícího kódu:

!./darknet detector test data/obj.data cfg/yolov3_custom.cfg /mydrive/yolov3/backup/yolov3_custom_last.weights /mydrive/images/pomstul.jpg -thresh 0.8

kde parametr test uvádí, že chceme testovat. Dalším parametrem je soubor .data, ve kterém je uveden počet trénovaných tříd a cesty k důležitým souborům. Jako další parametr je soubor .cfg s nastavením učení a struktury sítě. Další parametr je soubor s

posledními natrénovanými váhami, uložený na Google Drive. Předposledním parametrem je poté obrázek, který chceme testovat. Poslední parametr **-thresh** udává, s jakou procentuální hodnotou predikce, se mají „vytisknout“ bounding boxy na obrázek. V kódu výše je nastavena hodnota 0.8, což značí, že se budou „tisknout“ pouze bounding boxy, které mají více jak 80% jistotu, že se jedná o danou třídu. Příklady obrázků, které byly vyzkoušeny na této CNN lze vidět na obrázku 35. Všechny obrázky, které byly testovány touto sítí, lze nalézt v souboru příloha5.zip.

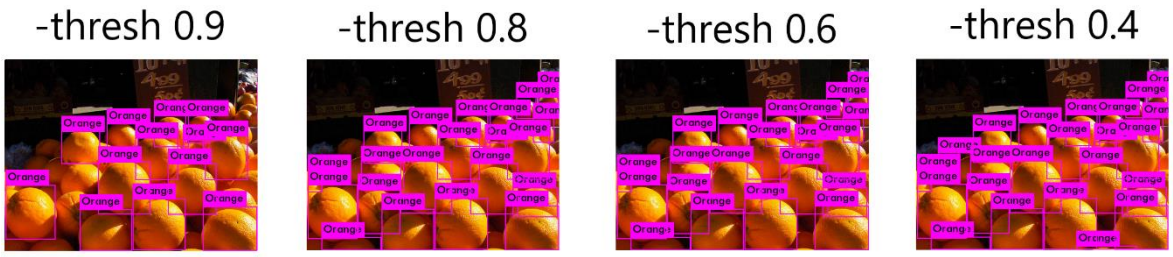


Obrázek 35: Obrázky s predikcí z YOLOv3 za pomoci Darknetu a dat z Open Images V6

Jako poslední bylo vyzkoušeno testování jednoho obrázku s různými parametry -tresh. Pro toto testování byl vybrán obrázek s velkým množstvím pomerančů, které jsou na sobě nakupeny. Originální obrázek lze vidět níže.

Obrázek 36: Originální obrázek použit k testování parametru -thresh

Dále byl u testování obrázku postupně měněn parametr -thresh na různé hodnoty, přičemž se sledovaly změny v predikci. Tam, kde byla nastavena hodnota 0.9, tedy „vytisknutí“ bounding boxů s pravděpodobností větší než 90 %, bylo nejméně predikovaných pomerančů. Predikce s nastavenými hodnoty 0.8 a 0.6 byly totožné. Poslední nastavená hodnota byla 0.4, kde bylo nalezeno o dva pomeranče více, než u predikce s parametrem 0.6.



Obrázek 37: Zkouška parametru -thresh na obrázku pomerančů

5 Výsledky a diskuse

V této kapitole jsou rozebrány výsledky kapitoly 4.

5.1 Google Colaboratory

Během využívání Google Colaboratory jsem došel k závěru, že se jedná o skvělou možnost, jak začít s neuronovými sítěmi. Ne všichni mají výkonný počítač, na kterém by mohli zkoušet vytvářet a trénovat své neuronové sítě. Díky tomu, že Colab nabízí virtuální stroje zdarma k využití pro každého s účtem od společnosti Google, může každý z jakéhokoli počítače, nebo notebooku, vyzkoušet tvorbu a učení své vlastní CNN. Jediná podmínka je internetové připojení.

Má to ale i své nevýhody. Mezi ty patří omezené zdroje, pokud je na Colabu využíváte až moc často, nebo váš kód více zatěžuje poskytovaná zařízení. Další nevýhodou je, že Váš virtuální stroj, poskytovaný Colabem, může být kdykoli odpojen kvůli neaktivitě. Musíte tedy fyzicky být u počítače a sem tam kliknout na buňky kódu, abyste průběžně projevovali aktivitu. Já sám jsem si při trénování nastavil časovač na 10 minut, který mi vždy připomněl, abych provedl nějakou akci. Dále je zde nevýhoda v životnosti poskytovaných virtuálních strojů, která je pro uživatele zdarma přibližně 12 hodin.

5.2 Vlastní zpracování dat pro učení

Získání a zpracování dat pro učení není v dnešní době složité. Existuje spousta nástrojů, které pomáhají s labelováním a předzpracováním dat pro učení. Nejvíce času zabírá ruční labelování, při kterém je nutné vědět do jaké míry označovat pro dosažení co nejlepšího výsledku. Využívání klávesových zkratek tuto činnost do jisté míry usnadňuje.

5.3 První konvoluční neuronová síť

První CNN byla učena za pomoci vlastnoručně olabelovaných dat, která byla následně upravena pomocí webové aplikace Roboflow. Ta velmi usnadňuje práci s automatickou změnou velikostí obrázků, které si zachovávají svůj poměr stran. Pokud by si nezachovaly svůj poměr stran, došlo by k zdeformování informace o tvaru, v tomto případě pomeranče, při učení.

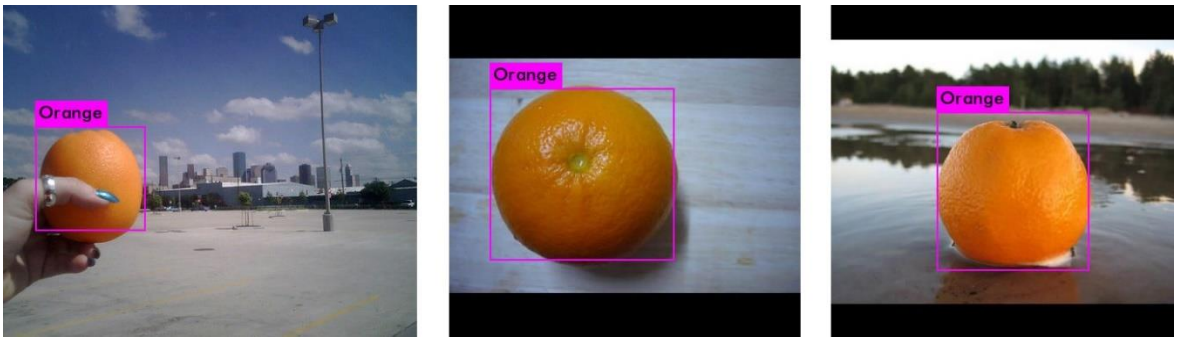
Jak lze z obrázku 33 vypozorovat, tak tato CNN nedosahovala dobrých výsledků. Pravděpodobnostní hodnoty, že se jedná o pomeranče, se pohybovali okolo 40 %, a některé

pomeranče měly tuto hodnotu i pod 30 % a nebyly u nich vykresleny bounding boxy. Přisuzuji to k nedostatečnému počtu trénovacích dat.

Dále byla natrénována na obrázcích o velikosti 416 x 416, což vedlo k tomu, že si síť „navykla“ na tuto velikost obrázků. Při predikci obrázku jiné velikosti nemusí mít tak dobré výsledky, jako při detekci obrázku o rozměrech 416 x 416. Proto jsem se rozhodl pro vytvoření druhé CNN, využít Darknetu a získat data pro učení z Open Images V6.

5.4 Druhá konvoluční neuronová síť

Tato CNN byla natrénována za pomoci dat získaných z Open Images V6. Díky tomuto datasetu se zvedl počet obrázků, na kterých se síť trénuje. Zároveň se zde využívá již předučených vah na rozsáhlé skupině dat. To zajišťuje rychlejší učení a kvalitnější začátek učení. V průběhu trénování je obrázkům náhodně změněna velikost, aby síť v kroku predikce byla přesná pro všechny velikosti obrázků. Jak můžeme vidět na obrázku 35, tato CNN je velice přesná. Pro kontrolu jsem dal síti predikovat obrázky pomerančů, které první CNN nedokázala rozpoznat (měly pravděpodobnost výskytu pomeranče pod 30 %). Výsledky lze vidět níže.



Obrázek 38: Predikované obrázky druhou CNN, které první CNN nerozeznala

Lze tedy říct, že druhé CNN jde rozeznávání pomerančů na obrázku velmi dobře i v případech, kde první CNN pomeranče nerozeznala.

6 Závěr

Díky rychlému rozvoji hardwarových a softwarových prostředků se posouvají kupředu i konvoluční neuronové sítě. Ty lze stále rychleji učit a vznikají také nové knihovny pro jejich tvorbu. Konvoluční neuronové sítě se tedy dostávají nejen do našich každodenních životů, ale také do čím dál více odvětví. Počínaje mobilními telefony, kde dokážu rozpoznávat obličeje, přes auta, která díky počítačové vizi dokážou sama řídit, nebo zabránit nehodám, až po využití v medicíně, kde pomáhají detekovat a rozeznávat stádia nemocí.

Tato diplomová práce se zaobírala právě neuronovými sítěmi, konkrétně konvolučními neuronovými sítěmi, rozpoznáváním obrazu a možnostmi využití Google Colaboratory, které zdarma poskytuje prostředí pro tvorbu a trénování neuronových sítí. V rámci teoretické části práce byly podrobně představeny stěžejní prvky, tedy umělá inteligence, význam a princip strojového a hlubokého učení. Ze zjištěných informací vyplynulo, že přestože se využívání neuronových sítí v praxi potýká i s několika dílčími problémy, například s hrozbou přeučení se nebo s dlouhou dobou učení se, má tento typ sítí mnoho předností. Díky tomu je možné tento typ sítí dobře využít i třeba pro soudobý vědecký výzkum léků nebo rozpoznávání nemocí.

Jak už bylo zmíněno, důležitou roli při zpracování praktické části této diplomové práce sehrála právě služba Google Colaboratory. Jako nesporná výhoda používání Google Colaboratory se ukázala možnost propojení s prostředím úložiště Google Drive a následné propojení s lokálním počítačem, což bylo také v práci názorně popsáno. Ukázalo se, že díky tomu lze předejít ztrátám dat a možným přerušením procesů učení CNN, které se bohužel s používání Colabu pojí.

Cílem práce tedy bylo v tomto prostředí vytvořit konvoluční neuronovou síť, která by byla schopná rozpoznávat zadaný objekt, v tomto případě byl zvolen pomeranč. Celkem byly vytvořeny dvě konvoluční neuronové sítě, přičemž pro učení každé z nich bylo využito rozdílných dat. Pro učení první sítě byla využita mnou získaná a připravená data, druhá síť pak byla naučena rozpoznávat pomeranče na datech stažených ze služby Open Images V6. S využitím vlastních dat je ale spojeno několik nevýhod. Takto naučená síť si například navykla na konkrétní velikost obrázků, což nevyhnutelně ve výsledku vedlo ke snížení přesnosti predikce na obrázcích jiné velikosti. Další nevýhodou bylo malé množství dat k učení, což vedlo k nepřesným predikcím natrénované sítě. Z tohoto důvodu byla vytvořena síť druhá, která byla trénována na datech dostupných z již zmíněné Open Images V6. Díky

tomuto datasetu se zvedl počet obrázků a zároveň bylo využito již předučených vah na rozsáhlé skupině dat, což zajistilo kvalitnější začátek a větší rychlost během celého procesu učení. Náhodné změny velikosti obrázků zvýšily pravděpodobnost přesnější predikce, z čehož vyplývá, že pro dosažení lepších výsledků při učení CNN je vhodná větší variabilita datasetu, na kterých se CNN natrénuje. Závěrem lze nicméně říct, že i přes zjištěné limity první vytvořené CNN byly cíle práce ve výsledku splněny.

Budoucnost pro CNN vypadá příznivě. Jak již bylo zmíněno, CNN se dodnes rozvíjí, což bude dle mého názoru pokračovat. Dostanou se více do našich každodenních životů, jako je automatické rozpoznávání a odemykání u dveří, chytré obchody, kde CNN rozpozná věci, které byly vložené do košíku a při odchodu z obchodu systém automaticky odečte peníze za dané zboží z účtu. Začnou se více využívat v ostatních oborech od zdravotnictví, přes automobilní průmysl, po využití při zajištění bezpečí občanů a zpříjemní náš život v mnoha dalších ohledech.

7 Seznam použitých zdrojů

[1] C. Smith, B. McGuire, T. Huang a G. Yang, „University of Washington,“ Prosinec 2006. [Online]. Available: https://courses.cs.2006/courses/csep590/06au/projects/history-ai.pdf. [Přístup získán 15 Leden 2020].

[2] J. Frankenfield, „Investopedia,“ 22 Leden 2020. [Online]. Available: https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp. [Přístup získán 26 Leden 2020].

[3] J. Rocca, „Towards data science,“ 23 Prosinec 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-machine-learning-f41aabc55264. [Přístup získán 13 leden 2020].

[4] Atul, „edureka,“ 22 Květen 2019. [Online]. Available: https://www.edureka.co/blog/what-is-machine-learning/. [Přístup získán 29 Prosinec 2019].

[5] M. Hargrave, „Investopedia,“ 30 Duben 2019. [Online]. Available: https://www.investopedia.com/terms/d/deep-learning.asp. [Přístup získán 13 Únor 2020].

[6] „Nvidia Developer,“ [Online]. Available: https://developer.nvidia.com/deep-learning. [Přístup získán 28 Prosinec 2019].

[7] K. Strachnyi, „Medium,“ 23 leden 2019. [Online]. Available: https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec. [Přístup získán 4 Únor 2020].

[8] T. Doseděl, „svetchytre,“ 31 Květen 2018. [Online]. Available: https://www.svetchytre.cz/a/itswE/jak-se-neuronove-site-snazi-napodobit-lidsky-mozek. [Přístup získán 19 Únor 2020].

[9] I. b. a. a. L. f. M. univerzity, „Matematická biologie,“ [Online]. Available: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--jednotlivy-neuron--matematicky-model-a-aktivni-dynamika-neuronu. [Přístup získán 14 Leden 2020].

[10] K. Molnár, „elektrorevue,“ 22 Únor 2000. [Online]. Available: http://www.elektrorevue.cz/clanky/00013/index.html#typy. [Přístup získán 21 Leden 2020].

[11] „Mendelova univerzita v Brně,“ [Online]. Available: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=21471. [Přístup získán 8 Leden 2020].

[12] DanB, „Kaggle,“ 7 Květen 2018. [Online]. Available: https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning. [Přístup získán 6 Leden 2020].

[13] V. Zhou, „A Simple Explanation of the Softmax Function,“ 26 Prosinec 2019. [Online]. Available: https://victorzhou.com/blog/softmax/. [Přístup získán 24 Únor 2020].

[14] „Multi-Class Neural Networks: Softmax,“ [Online]. Available: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax. [Přístup získán 24 Únor 2020].

[15] C. Nicholson, „pathmind,“ [Online]. Available: https://pathmind.com/wiki/neural-network. [Přístup získán 24 Leden 2020].

[16] „NaPočítači,“ 8 Září 2017. [Online]. Available: https://www.napocitaci.cz/33/neuronove-site-a-princip-jejich-fungovani-uniqueidgOkE4NvrWuNY54vrLeM670eFNQh552VdDDulZX7UDBY/. [Přístup získán 5 Leden 2020].

[17] B. Tóth, „tech trustpilot,“ 6 Červen 2018. [Online]. Available: https://tech.trustpilot.com/forward-and-backward-propagation-5dc3c49c9a05. [Přístup získán 17 Leden 2020].

[18] I. Vondrák, „Technická univerzita Ostrava,“ Duben 2009. [Online]. Available: http://vondrak.cs.vsb.cz/download/Neuronove_site.pdf. [Přístup získán 25 Únor 2020].

[19] F. Chollet, Deep learning v jazyku python, USA: GRADA, 2018.

[20] „TensorFlow,“ [Online]. Available: https://www.tensorflow.org/guide/tensor. [Přístup získán 15 Březen 2020].

[21] „simplilearn,“ [Online]. Available: https://www.simplilearn.com/what-is-perceptron-tutorial. [Přístup získán 18 Leden 2020].

[22] S. Sharma, „towards data sciene,“ 9 Září 2017. [Online]. Available: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53. [Přístup získán 18 Leden 2020].

[23] D. E. Rumelhart, G. E. Hinton a R. J. Williams, „Stanford,“ 1986. [Online]. Available: https://web.stanford.edu/class/psych209a/ReadingsByDate/02_06/PDPVolIChapter8.pdf. [Přístup získán 20 Únor 2020].

[24] I. b. a. a. L. f. M. univerzity, „MatematickaBiologie,“ [Online]. Available: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--uvod-do-neuronovych-siti--koncept-umele-neuronove-site. [Přístup získán 20 Únor 2020].

[25] K. Cho, T. Raiko a A. Ilin, „Enhanced Gradient and Adaptive Learning Rate,“ 2011. [Online]. Available: http://www.icml-2011.org/papers/98_icmlpaper.pdf. [Přístup získán 26 Únor 2020].

[26] A. Sharma, „towards data science,“ 2 Říjen 2018. [Online]. Available: https://towardsdatascience.com/restricted-boltzmann-machines-simplified-eab1e5878976. [Přístup získán 24 Únor 2020].

[27] „UFLDL Standford,“ [Online]. Available: http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/. [Přístup získán 25 Únor 2020].

[28] M. Uldrich a T. Jurczyk, „Neuronové sítě a jejich využití,“ 2014. [Online]. Available: http://www.statsoft.cz/file1/PDF/Statsoft_neuronove_site.pdf. [Přístup získán 29 Únor 2020].

[29] A. Krizhevky, I. Sutskever a G. E. Hinton, „ImageNet Classification with Deep Convolutional,“ 2012. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. [Přístup získán 26 Únor 2020].

[30] R. Draelos, „Glass Box,“ 13 Duben 2019. [Online]. Available: https://glassboxmedicine.com/2019/04/13/a-short-history-of-convolutional-neural-networks/. [Přístup získán 18 Březen 2020].

[31] I. Goodfellow, Y. Bengio a A. Courville, „Deep Learning,“ 2016. [Online]. Available: http://www.deeplearningbook.org/. [Přístup získán 24 Březen 2020].

[32] „Convolution,“ [Online]. Available: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolution.html. [Přístup získán 24 Březen 2020].

[33] S. Saha, „A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,“ 15 Prosinec 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Přístup získán 24 Březen 2020].

[34] V. Zhou, „towards data science,“ 7 Červen 2019. [Online]. Available: https://towardsdatascience.com/training-a-convolutional-neural-network-from-scratch-2235c2a25754. [Přístup získán 25 Únor 2020].

[35] J. Bouvrie, „Notes on Convolutional Neural Networks,“ 22 Říjen 2006. [Online]. Available: http://cogprints.org/5869/1/cnn_tutorial.pdf. [Přístup získán 21 Únor 2020].

[36] Y. Kashef, „Quora,“ 9 Červen 2016. [Online]. Available: https://www.quora.com/How-is-Fully-Convolutional-Network-FCN-different-from-the-original-Convolutional-Neural-Network-CNN. [Přístup získán 21 Únor 2020].

[37] F. Doupal, „NVIDIA CUDA - využití grafické karty naplno,“ 8 Duben 2009. [Online]. Available: https://notebook.cz/clanky/technologie/2009/nvidia-cuda-vyuziti-graficke-karty-skutecne-naplno. [Přístup získán 24 Březen 2020].

[38] „GPU Accelerated Computing with C and C++,“ [Online]. Available: https://developer.nvidia.com/how-to-cuda-c-cpp. [Přístup získán 24 Březen 2020].

[39] „What is OpenCL?,“ [Online]. Available: https://streamhpc.com/knowledge/what-is/opencl/. [Přístup získán 26 Březen 2020].

[40] G. Rosati, „Medical Diagnosis with a Convolutional Neural Network,“ 28 Srpen 2019. [Online]. Available: https://towardsdatascience.com/medical-diagnosis-with-a-convolutional-neural-network-ab0b6b455a20. [Přístup získán 29 Březen 2020].

[41] O. Engkvist, H. Chen, Y. Wang, M. Olivecrona a T. Blaschke, „The rise of deep learning in drug discovery,“ 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1359644617303598. [Přístup získán 29 Březen 2020].

[42] „Introducing AtomNet – Drug design with convolutional neural networks,“ 5 Prosinec 2015. [Online]. Available: https://www.atomwise.com/2015/12/02/introducing-atomnet-drug-design-with-convolutional-neural-networks/. [Přístup získán 29 Březen 2020].

[43] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn a D. Yu, „Convolutional Neural Networks,“ Říjen 2014. [Online]. Available:

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/CNN_ASLPTrans2-14.pdf. [Přístup získán 29 Březen 2020].

[44] „O projektu,“ [Online]. Available: https://www.birdsonline.cz/cz/o-projektu. [Přístup získán 29 Březen 2020].

[45] Google, „Google research,“ [Online]. Available: https://research.google.com/colaboratory/faq.html. [Přístup získán 19 Leden 2020].

[46] S. Venkat, „Best Image Labeling Tools For Computer Vision,“ 24 Říjen 2019. [Online]. Available: https://medium.com/tektorch-ai/best-image-labeling-tools-for-computer-vision-393e256be0a0. [Přístup získán 24 Březen 2020].

[47] „Open Images Dataset V6 + Extensions,“ [Online]. Available: https://storage.googleapis.com/openimages/web/index.html. [Přístup získán 31 Březen 2020].

[48] J. Brownlee, „Difference Between a Batch and an Epoch in a Neural Network,“ 20 Červenec 2018. [Online]. Available: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. [Přístup získán 30 Březen 2020].

[49] alexbe, „what is mean random = 1? in cfg file?,“ 19 Květen 2017. [Online]. Available: https://groups.google.com/forum/#!msg/darknet/7X0mNQv14z8/0Ibkd4_hAgAJ. [Přístup získán 31 Březen 2020].

[1] C. Smith, B. McGuire, T. Huang a G. Yang, „University of Washington,“ Prosinec 2006. [Online]. Available: https://courses.cs.2006/courses/csep590/06au/projects/history-ai.pdf. [Přístup získán 15 Leden 2020].

[2] J. Frankenfield, „Investopedia,“ 22 Leden 2020. [Online]. Available: https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp. [Přístup získán 26 Leden 2020].

[3] J. Rocca, „Towards data science,“ 23 Prosinec 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-machine-learning-f41aabc55264. [Přístup získán 13 leden 2020].

[4] Atul, „edureka,“ 22 Květen 2019. [Online]. Available: https://www.edureka.co/blog/what-is-machine-learning/. [Přístup získán 29 Prosinec 2019].

[5] M. Hargrave, „Investopedia,“ 30 Duben 2019. [Online]. Available: https://www.investopedia.com/terms/d/deep-learning.asp. [Přístup získán 13 Únor 2020].

[6] „Nvidia Developer,“ [Online]. Available: https://developer.nvidia.com/deep-learning. [Přístup získán 28 Prosinec 2019].

[7] K. Strachnyi, „Medium,“ 23 leden 2019. [Online]. Available: https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec. [Přístup získán 4 Únor 2020].

[8] T. Doseděl, „svetchytre,“ 31 Květen 2018. [Online]. Available: https://www.svetchytre.cz/a/itswE/jak-se-neuronove-site-snazi-napodobit-lidsky-mozek. [Přístup získán 19 Únor 2020].

[9] I. b. a. a. L. f. M. univerzity, „Matematická biologie,“ [Online]. Available: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--jednotlivy-neuron--matematicky-model-a-aktivni-dynamika-neuronu. [Přístup získán 14 Leden 2020].

[10] K. Molnár, „elektrorevue,“ 22 Únor 2000. [Online]. Available: http://www.elektrorevue.cz/clanky/00013/index.html#typy. [Přístup získán 21 Leden 2020].

[11] „Mendelova univerzita v Brně,“ [Online]. Available: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=21471. [Přístup získán 8 Leden 2020].

[12] DanB, „Kaggle,“ 7 Květen 2018. [Online]. Available: https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning. [Přístup získán 6 Leden 2020].

[13] V. Zhou, „A Simple Explanation of the Softmax Function,“ 26 Prosinec 2019. [Online]. Available: https://victorzhou.com/blog/softmax/. [Přístup získán 24 Únor 2020].

[14] „Multi-Class Neural Networks: Softmax,“ [Online]. Available: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax. [Přístup získán 24 Únor 2020].

[15] C. Nicholson, „pathmind,“ [Online]. Available: https://pathmind.com/wiki/neural-network. [Přístup získán 24 Leden 2020].

[16] „NaPočítači,“ 8 Září 2017. [Online]. Available: https://www.napocitaci.cz/33/neuronove-site-a-princip-jejich-fungovani-uniqueidgOkE4NvrWuNY54vrLeM670eFNQh552VdDDulZX7UDBY/. [Přístup získán 5 Leden 2020].

[17] B. Tóth, „tech trustpilot,“ 6 Červen 2018. [Online]. Available: https://tech.trustpilot.com/forward-and-backward-propagation-5dc3c49c9a05. [Přístup získán 17 Leden 2020].

[18] I. Vondrák, „Technická univerzita Ostrava,“ Duben 2009. [Online]. Available: http://vondrak.cs.vsb.cz/download/Neuronove_site.pdf. [Přístup získán 25 Únor 2020].

[19] F. Chollet, Deep learning v jazyku python, USA: GRADA, 2018.

[20] „TensorFlow,“ [Online]. Available: https://www.tensorflow.org/guide/tensor. [Přístup získán 15 Březen 2020].

[21] „simplilearn,“ [Online]. Available: https://www.simplilearn.com/what-is-perceptron-tutorial. [Přístup získán 18 Leden 2020].

[22] S. Sharma, „towards data sciene,“ 9 Září 2017. [Online]. Available: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53. [Přístup získán 18 Leden 2020].

[23] D. E. Rumelhart, G. E. Hinton a R. J. Williams, „Stanford,“ 1986. [Online]. Available: https://web.stanford.edu/class/psych209a/ReadingsByDate/02_06/PDPVolIChapter8.pdf. [Přístup získán 20 Únor 2020].

[24] I. b. a. a. L. f. M. univerzity, „MatematickaBiologie,“ [Online]. Available: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--uvod-do-neuronovych-siti--koncept-umele-neuronove-site. [Přístup získán 20 Únor 2020].

[25] K. Cho, T. Raiko a A. Ilin, „Enhanced Gradient and Adaptive Learning Rate,“ 2011. [Online]. Available: http://www.icml-2011.org/papers/98_icmlpaper.pdf. [Přístup získán 26 Únor 2020].

[26] A. Sharma, „towards data science,“ 2 Říjen 2018. [Online]. Available: https://towardsdatascience.com/restricted-boltzmann-machines-simplified-eab1e5878976. [Přístup získán 24 Únor 2020].

[27] „UFLDL Standford,“ [Online]. Available: http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/. [Přístup získán 25 Únor 2020].

[28] M. Uldrich a T. Jurczyk, „Neuronové sítě a jejich využití,“ 2014. [Online]. Available: http://www.statsoft.cz/file1/PDF/Statsoft_neuronove_site.pdf. [Přístup získán 29 Únor 2020].

[29] A. Krizhevky, I. Sutskever a G. E. Hinton, „ImageNet Classification with Deep Convolutional,“ 2012. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. [Přístup získán 26 Únor 2020].

[30] R. Draelos, „Glass Box,“ 13 Duben 2019. [Online]. Available: https://glassboxmedicine.com/2019/04/13/a-short-history-of-convolutional-neural-networks/. [Přístup získán 18 Březen 2020].

[31] I. Goodfellow, Y. Bengio a A. Courville, „Deep Learning,“ 2016. [Online]. Available: http://www.deeplearningbook.org/. [Přístup získán 24 Březen 2020].

[32] „Convolution,“ [Online]. Available: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolution.html. [Přístup získán 24 Březen 2020].

[33] S. Saha, „A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,“ 15 Prosinec 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Přístup získán 24 Březen 2020].

[34] V. Zhou, „towards data science,“ 7 Červen 2019. [Online]. Available: https://towardsdatascience.com/training-a-convolutional-neural-network-from-scratch-2235c2a25754. [Přístup získán 25 Únor 2020].

[35] J. Bouvrie, „Notes on Convolutional Neural Networks,“ 22 Říjen 2006. [Online]. Available: http://cogprints.org/5869/1/cnn_tutorial.pdf. [Přístup získán 21 Únor 2020].

[36] Y. Kashef, „Quora,“ 9 Červen 2016. [Online]. Available: https://www.quora.com/How-is-Fully-Convolutional-Network-FCN-different-from-the-original-Convolutional-Neural-Network-CNN. [Přístup získán 21 Únor 2020].

[37] F. Doupal, „NVIDIA CUDA - využití grafické karty naplno,“ 8 Duben 2009. [Online]. Available: https://notebook.cz/clanky/technologie/2009/nvidia-cuda-vyuziti-graficke-karty-skutecne-naplno. [Přístup získán 24 Březen 2020].

[38] „GPU Accelerated Computing with C and C++,“ [Online]. Available: https://developer.nvidia.com/how-to-cuda-c-cpp. [Přístup získán 24 Březen 2020].

[39] „What is OpenCL?,“ [Online]. Available: https://streamhpc.com/knowledge/what-is/opencl/. [Přístup získán 26 Březen 2020].

[40] G. Rosati, „Medical Diagnosis with a Convolutional Neural Network,“ 28 Srpen 2019. [Online]. Available: https://towardsdatascience.com/medical-diagnosis-with-a-convolutional-neural-network-ab0b6b455a20. [Přístup získán 29 Březen 2020].

[41] O. Engkvist, H. Chen, Y. Wang, M. Olivecrona a T. Blaschke, „The rise of deep learning in drug discovery,“ 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1359644617303598. [Přístup získán 29 Březen 2020].

[42] „Introducing AtomNet – Drug design with convolutional neural networks,“ 5 Prosinec 2015. [Online]. Available: https://www.atomwise.com/2015/12/02/introducing-atomnet-drug-design-with-convolutional-neural-networks/. [Přístup získán 29 Březen 2020].

[43] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn a D. Yu, „Convolutional Neural Networks,“ Říjen 2014. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/CNN_ASLPTrans2-14.pdf. [Přístup získán 29 Březen 2020].

[44] „O projektu,“ [Online]. Available: https://www.birdsonline.cz/cz/o-projektu. [Přístup získán 29 Březen 2020].

[45] Google, „Google research,“ [Online]. Available: https://research.google.com/colaboratory/faq.html. [Přístup získán 19 Leden 2020].

[46] S. Venkat, „Best Image Labeling Tools For Computer Vision,“ 24 Říjen 2019. [Online]. Available: https://medium.com/tektorch-ai/best-image-labeling-tools-for-computer-vision-393e256be0a0. [Přístup získán 24 Březen 2020].

[47] „Open Images Dataset V6 + Extensions,“ [Online]. Available: https://storage.googleapis.com/openimages/web/index.html. [Přístup získán 31 Březen 2020].

[48] J. Brownlee, „Difference Between a Batch and an Epoch in a Neural Network,“ 20 Červenec 2018. [Online]. Available: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. [Přístup získán 30 Březen 2020].

[49] alexbe, „what is mean random = 1? in cfg file?,“ 19 Květen 2017. [Online]. Available: https://groups.google.com/forum/#!msg/darknet/7X0mNQv14z8/0Ibkd4_hAgAJ. [Přístup získán 31 Březen 2020].

8 Přílohy

příloha1.ipynb
příloha2.ipynb
příloha3.ipynb
příloha4.zip
příloha5.zip