VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ROZPOZNÁVÁNÍ DRUHU JÍDLA S POMOCÍ HLUBOKÝCH NEURONOVÝCH SÍTÍ
FOOD CLASSIFICATION USING DEEP NEURAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Michal Kuvik**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Radim Burget, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Michal Kuvik ***ID:*** 174340
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Rozpoznávání druhu jídla s pomocí hlubokých neuronových sítí

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou klasifikace obrazu s pomocí hlubokých konvolučních neuronových sítí a posledním vývojem v oblasti klasifikace obrazových dat. S použitím dat z databáze iFood 2018 Challenge (https://www.kaggle.com/c/ifood2018) natrénujte neuronovou síť, s pomocí které bude možné rozpoznávat jednotlivé kategorie jídla. Experimentujte s různými architekturami sítí včetně architektury FishNet. Dosažené výsledky diskutujte a vhodně je reprezentujte.

DOPORUČENÁ LITERATURA:

[1] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[2] Karpathy, Andrej, et al. "Large-scale video classification with convolutional neural networks." Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2014.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 16.5.2019

Vedoucí práce: doc. Ing. Radim Burget, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Cieľom tejto práce je naštudovať problematiku hĺbkových konvolučných neurónových sietí a s ňou spojenú klasifikáciou obrázkov, experimentovať s architektúrou zvolenej siete za účelom získania najväčšej presnosti na vybratom dátovom súbore. Práca je rozdelená do dvoch celkov, kde v prvom sú teoreticky priblížené vlastnosti a štruktúra neurónových sietí a stručne popísané zvolené siete. Druhá časť sa zaoberá experimentami s touto sieťou, ako je napr. vplyv rozšírenia dát, veľkosti dávky alebo vplyv vrstiev zahadzovania na presnosť siete. Následne sú všetky výsledky porovnané a diskutované, kde najlepší výsledok dosiahol presnosť $86,44\%$ na testovacích dátach.

KĽÚČOVÉ SLOVÁ

Python, Keras, TensorFlow, konvolučná neurónová sieť, InceptionV3, Kaggle, klasifikácia obrázkov

ABSTRACT

The aim of this thesis is to study problems of deep convolutional neural networks and the connected classification of images and to experiment with the architecture of particular network with the aim to get the most accurate results on the selected dataset. The thesis is divided into two parts, the first part theoretically outlines the properties and structure of neural networks and briefly introduces selected networks. The second part deals with experiments with this network, such as the impact of data augmentation, batch size and the impact of dropout layers on the accuracy of the network. Subsequently, all results are compared and discussed with the best result achieved an accuracy of $86,44\%$ on test data.

KEYWORDS

Python, Keras, TensorFlow, convolutional neural network, Inception, Kaggle, image classification

KUVIK, Michal. *Rozpoznávání druhu jídla s pomocí hlubokých neuronových sítí*. Brno, Rok, 55 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: prof. Ing. Radim Burget, CSc.

VYHLÁSENIE

Vyhlasujem, že som svoju diplomovú prácu na tému „Rozpoznávání druhu jídla s pomocí hlubokých neuronových sítí“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

Tato práce vznikla jako součást klíčové aktivity KA6 - Individuální výuka a zapojení studentů bakalářských a magisterských studijních programů do výzkumu v rámci projektu OP VVV Vytvoření double-degree doktorského studijního programu Elektronika a informační technologie a vytvoření doktorského studijního programu Informační bezpečnost, reg. č. CZ.02.2.69/0.0/0.0/16_018/0002575.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Projekt je spolufinancován Evropskou unií.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Jedným z najrozšírenejších ochorení v civilizovanom svete je nadváha ľudskej populácie a jej výskyt neustále stúpa. Monitorovaním stravy a následným dodržiavaním životosprávy, by sa mohlo podariť zmierniť následky tohoto trendu. Jedným z možných riešení je strojové učenie pomocou, ktorého by bolo možné rozpoznávať jednotlivé jedlá a teda príjem kalórií. Automatická detekcia potravín je však náročný proces. Veľký počet kategórií jedál a vysoká vizuálna podobnosť medzi rôznymi kategóriami výrazne sťažuje túto úlohu.

Problémom v oblasti strojového učenia bola dostupnosť kvalitných dát, ktorých je potrebné veľké množstvo pre správne natrénovanie neurónovej siete. Druhým hlavným problémom, ktorý bránil v rozvoji tejto oblasti bol výpočtový výkon. Pre trénovanie rozsiahlych neurónových sietí, na tak veľkom dátovom súbore, bolo potrebné veľké množstvo výpočtovej kapacity. Vývojom aj v tejto oblasti a to konkrétne v oblasti GPU (Graphics Processing Unit) dostávame potrebný výkon pre realizáciu konkrétnych úloh.

Táto práca sa zaoberá úlohou klasifikáciou jedál, kde za pomoci vybratého dátového súboru je trénovaná zvolená architektúra konvolučnej neurónovej siete za účelom získania čo najpresnejšej klasifikácie. Hlavným prínosom tejto práce je porovnanie dostupných architektúr konvolučných neurónových sietí, ktoré sú vhodné pre riešenie problému klasifikácie jedál, voľba optimálnej architektúry a experimentovanie s parametrami tejto siete. Vplyv jednotlivých parametrov na presnosť siete je graficky znázornený a následne popísaný a diskutovaný. Výsledný najpresnejší model je schopný s presnosťou v top-3 $86,44\%$ rozpoznávať jednotlivé kategórie jedál.

V prvej kapitole sú popísané základy neurónových sietí, princíp funkčnosti neurónových sietí a jedna z ich najdôležitejších vlastností a to učenie. Druhá kapitola sa zameriava konkrétnejšie na hĺbkové konvolučné neurónové siete, ktoré sú využívane v tejto práci. Sú v nej popísané základne časti sietí, spôsob spracovávania obrazu, pokrok v oblasti, ktorý zabezpečil zvýšenie presnosti konvolučných neurónových sietí a stručne popísaný princíp zvolených neurónových sietí. V ďalšej kapitole je popísané pracovné prostredie, ktoré bolo využívane, taktiež zvolený dátový súbor a jeho úprava, implementácia využívaných sietí a popis všetkých vykonaných experimentov. Posledná kapitola zhŕňa všetky výsledky a je v nej diskutovaný vplyv parametrov na presnosť konvolučnej neurónovej siete.

1 Problematika neurónových sietí

Prvá kapitola je zameraná na vlastnosti a princípy neurónových sietí. Nachádza sa v nej stručný popis základného stavebného prvku neurónových sietí, taktiež popis štruktúry neurónových sietí a princíp učenia.

1.1 Neurón

Ľudská centrálna nervová sústava sa skladá z biologických neurónov, ktoré sa stali predlohou pre vytvorenie umelého neurónu. Biologické neuróny sú schopné prenášať a spracovávať potrebné informácie. Každý neurón má svoj vstup (dendrit), telo (soma) a výstup (axon) na konci ktorého sa nachádza zakončenie (synapsia), ktoré slúži na prenos informácií. Jeden neurón môže byť prepojený s mnoho ďalšími a tým vzniká neurónová sieť, ktorej vlastnosti sa počas života človeka menia.[1]

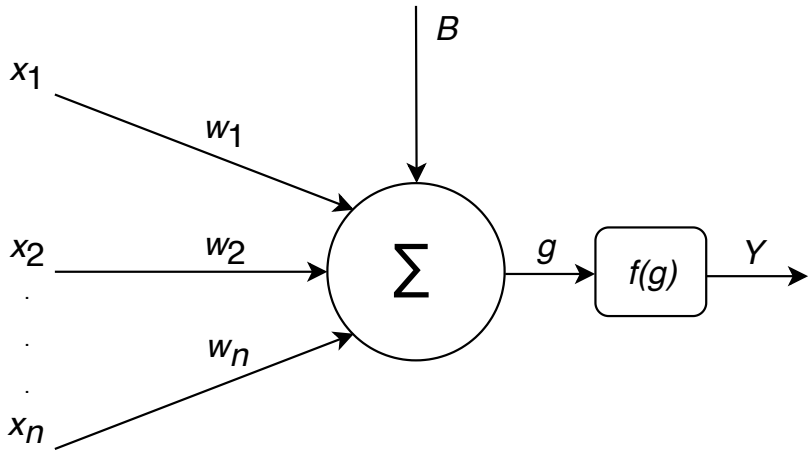
Ako už bolo spomínané umelý neurón vychádza z predlohy biologického neurónu a jeho štruktúru môžeme vidieť na obrázku 1.1[2]. Matematicky ho môžeme popísať nasledujúcim vzťahom:

$$Y = f(\sum_{i=1}^{n} x_i w_i + B), \tag{1.1}$$

kde

- x_i - vstupy neurónu,
- w_i - váhy vstupov,
- B - prahová funkcia,
- f - aktivačná funkcia,
- Y - výstup neurónu.

Neurón má určitý počet vstupov, ktoré sú označené ako x_1 až x_n a každý vstup má priradenú váhu w_1 až w_n. Váha jednotlivých vstupov reprezentuje významnosť vstupu a mieru v akom vstup ovplyvňuje neurón. Prahová funkcia (bias) je označená ako B a je to reálna konštanta, ktorá ovplyvňuje vnútorný potenciál neurónu a spracováva sa ako každná iná váha. Funkcia f sa nazýva aktivačná funkcia a samotný výstup z neurónu je označený, ako Y. Aktivačná funkcia môže byť lineárna alebo nelineárna. Pre neurónové siete je však nelinearita aktivačnej funkcie veľmi dôležitá pretože, ak by sieť obsahovala iba postupnosť lineárnych prvkov, výstup siete by bol ekvivalent jedného neurónu prípadne vrstvy. Existuje veľké množstvo aktivačných funkcií a tieto funkcie majú priamy vplyv na učenie neurónovej siete. V prípade konvolučných neurónových sietí je najpoužívanejšia ReLU (Rectified Linear Unit) aktivačná funkcia, ktorá zlepšuje rýchlosť učenia.[2]

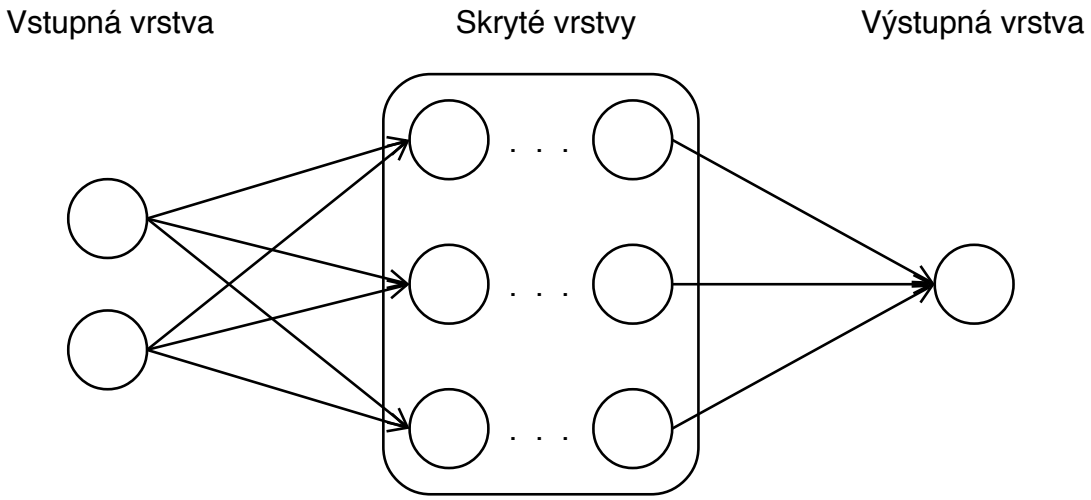


Obr. 1.1: Model umelého neurónu

1.2 Štruktúra neurónovej siete

Štruktúru neurónovej siete tvoria neuróny, ktoré sú určitým spôsobom medzi sebou prepojené. Tieto neuróny sú usporiadané do vrstiev a vytvárajú ohodnotený orientovaný graf, kde neuróny reprezentujú vrcholy grafu a prepojenia hrany s určitou váhou. Na obrázku 1.2 vidíme topológiu neurónovej siete s dopredným šírením signálu, ktorou sa bude táto práca zaoberať. Dáta v tejto topológii prechádzajú postupne od prvej vrstvy až k tej poslednej. Druhým typom sú rekurentné neurónové siete v ktorých sa nachádzajú spätné väzby a vstup nižšej vrstvy môže byť závislý na výstupe tej vyššej.

Každá neurónová sieť má svoju vstupnú vrstvu na ktorú prichádzajú vstupné dáta a zabezpečuje distribúcii týchto dát do nasledujúcich vrstiev. Tieto vrstvy sa nazývajú skryté vrstvy a je ich určitý počet v závislosti na topológii siete. Skryté vrstvy zabezpečujú transformáciu vstupov a teda samotnú funkčnosť neurónovej sieti. Neurónové siete, ktoré obsahujú viacej, ako jednu skrytú vrstvu nazývame hĺbkové neurónové siete a čím sa v hierarchii vrstiev dostávame hlbšie, tým je abstrakcia dát väčšia. S rastúcou hĺbkou rastie aj zložitosť siete a v rozsiahlych sieťach sa môžu nachádzať aj desiatky skrytých vrstiev. So zvyšujúcim sa počtom vrstiev a neurónov v skrytých vrstvách, vzrastie taktiež náročnosť siete na proces učenia a preto je potrebné vyhľadať kompromis medzi požadovanou presnosťou neurónovej siete a časom, ktorý je potrebný pre jej natrénovanie. Poslednou vrstvou neurónovej siete je výstupná vrstva, ktorá reprezentuje odozvu siete na vstupný signál a slúži na klasifikáciu jednotlivých tried. Konkrétne typy vrstiev a ich vlastnosti sú popísané v kap. 2.2.[3]



Obr. 1.2: Model neurónovej siete

1.3 Učenie neurónovej siete

Pre efektívne využívanie neurónovej siete je potrebné sieť tzv. natrénovať. Tento proces sa nazýva učenie neurónovej siete a môžeme ho rozdeliť do dvoch tried. Prvou triedou je učenie bez učiteľa, pri ktorom nie sú dostupné požadované výstupy a učenie prebieha na základe vnútornej štruktúry. Sieť sa sama rozhodne, ktorá odozva je pre daný vstup najlepšia. V prípade klasifikácie obrazu sa využíva druhý spôsob učenia a to učenie s učiteľom, ktorý je využívaný v našom prípade.[4]

Všetky dáta, ktoré sú predložené neurónovej sieti pri procese učenia majú svoj požadovaný výstup, teda triedu do ktorej chceme, aby dáta boli zaradené pri optimálnom natrénovaní. Dáta použité v tejto práci sú popísané v kap. 3.2. V prípade, že výstup neurónovej siete sa nezhoduje s výstupom priradeným k trénovacím dátam je vypočítaná chybová funkcia. Táto chybová funkcia nám určuje rozdiel medzi skutočným a požadovaným výstupom a slúži na adaptáciu váh neurónovej siete. Adaptácia váh je proces pri ktorej sú upravené váhy nerónovej siete pomocou optimalizačnej metódy za účelom zníženia chybovosti tejto siete. Často využívanou optimalizačnou metódou je znižovanie hodnoty gradientu, ktorá hľadá lokálne minimum chybovej funkcie. V ďalšom kroku je toto minimum distribuované do siete.

Jedným z najčastejších algoritmov je algoritmus so spätným šírením chyby. Tento algoritmu zabezpečuje úpravu váh neurónov v jednotlivých vrstvách na základe dosiahnutého výsledku na výstupe po prechode jednej dávky sieťou. Algoritmus využíva takzvané reťazové pravidlo so špecifickým poradím operácií.

V prípade, že máme dve funkcie f a g definované v obore reálnych čísel nasledovne $y = g(x)$ a $z = f(g(x)) = f(y)$, potom platí reťazové pravidlo

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}. \tag{1.2}$$

Avšak v oblasti neurónových sietí sa pohybujeme viac v oblasti vektorov, ako skalárov preto platí varianta $\vec{x} \in R^n$ a $\vec{y} \in R^m$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}. \tag{1.3}$$

Každý uzol vo výpočtovom grafe reprezentujúci funkciu je schopný vypočítať parciálnu deriváciu svojho výstupu podľa svojich vstupov. V procese spätného učenia sa potom určí vplyv vstupov uzlu na výstup celej siete.[5] Ďalším dôležitým parametrom pri učení je rýchlosť učenia, ktorý určuje o aký krok sa váhy upravia. Vyšší krok môže znamenať rýchlejšie učenie avšak výsledok pri nižšom kroku môže byť presnejší.[6] Trénovacie dáta sú rozdelené do jednotlivých dávok a proces učenia nastáva pri prechode každej dávky. Prechod všetkých trénovacích dát sieťou sa nazýva epocha a pri reálnom trénovaní sa tie isté dáta privádzajú na vstup niekoľko krát.

2 Hĺbkové konvolučné neurónové siete

Táto kapitola bližšie približuje princíp spracovania obrazu pomocou hĺbkových konvolučných neurónových sietí. Ďalej je tu popísaná architektúra týchto sietí z pohľadu jednotlivých vrstiev, vývoju v oblasti klasifikácie obrazu a v poslednej časti je stručne popísaný princíp zvolených sietí.

2.1 Princíp klasifikácii obrazu

Úlohou klasifikácie obrazu je na základe vstupného obrázku získať určitý výstup v podobe triedy do ktorej je obrázok priradený. Pre človeka je prirodzené tento problém vyriešiť, avšak pre stroj to nie je jednoduchá úloha. Klasifikáciu môžeme rozdeliť na binárnu a klasifikáciu do viacerých tried. Binárna klasifikácia je jednoduchšia a môže určovať napríklad výskyt určitého objektu na obrázku (auto, dom ...). Táto práca sa zaoberá druhým typom klasifikácii a v našom prípade môže byť obrázku priradená jedna z 211 tried.[7]

Dominantnú úlohu v klasifikácii obrazu hrajú konvolučné neurónové siete a ich architektúra, ktorá sa postupne osvedčuje, ako najúčinnejší spôsob vizuálnej reprezentácie. Model hĺbkového učenia využíva jednu neurónovú sieť trénovanú pomocou hodnôt bodov obrázka na klasifikovanie výstupu. Tento nápad bol prvý krát predstavený v práci [8] a potom vylepšený v nasledujúcej práci [9]. Príchodom možnosti trénovať neurónové siete na veľkom množstve trénovacích dát a efektívnejšou implementáciou na GPU, konvolučné neurónové siete prekonali niektoré iné konvenčné metódy a dokonca aj ľudské výkony súvisiace s klasifikáciou obrazu.[10]

Počítač vidí obrázok, ako pole hodnôt reprezentujúce jednotlivé body. Veľkosť poľa nám určuje rozlíšenie obrázka. Napríklad, ak sa jedná o farebný obrázok s rozlíšením 480×480 bodov naše pole má veľkosť $480 \times 480 \times 3$, pretože každý pixel sa skladá z troch zložiek RGB (Red Green Blue). Každé z týchto čísiel nadobúda hodnotu od 0 do 255, ktorá popisuje intenzitu bodu. Tieto hodnoty sú vstupom do konvolučnej neurónovej siete, ktorej úlohou je vyhľadať prvky nízkej úrovne, ako sú okraje, hrany, krivky a následne vytvárať abstraktnejšie koncepty prostredníctvom série konvolučných vrstiev. Tento proces v konvolučných vrstvách zabezpečuje konvolúcia na ktorej vstup je privádzané pole hodnôt bodov, taktiež nazývané tenzor.

2.2 Vrstvy konvolučnej neurónovej siete

2.2.1 Konvolučná vrstva

Základnou vrstvou v konvolučných neurónových sieťach je konvolučná vrstva, ktorá využíva lineárnu matematickú operáciu konvolúcie, ktorú môžeme definovať nasledujúcim vzorcom, ktorý je prebratý z literatúry[11]

$$h(x,y) = f * g(x,y) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] g[x - n_1, y - n_2], \tag{2.1}$$

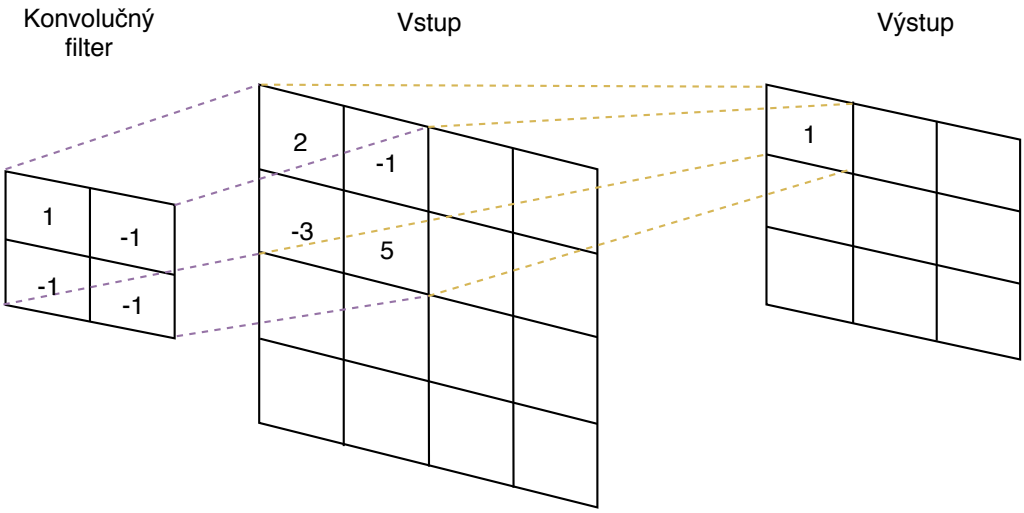
kde

$*$	-	označuje operáciu konvolúcie,
f	-	predstavuje konvolučný filter,
$g(x,y)$	-	predstavuje vstupní obrázok so súradnicami x, y,
$h(x,y)$	-	označuje výstup konvolúcie v danom bode.

Vrstva obsahuje niekoľko konvolučných filtrov, kde jeden filter je v našom prípade skupina váh. Majú rovnakú dimenziu ako vstupné pole, avšak sú rozmerovo niekoľko násobne menšie. Každý z filtrov slúži na detekciu príznakov určitej triedy. V nižších vrstvách sa môže jednať napríklad o hrany alebo krivky, vo vyšších vrstvách ide o konkrétne objekty.

Na obrázku 2.1 môžeme vidieť operáciu konvolúcie v konvolučnej vrstve. Každý z konvolučných filtrov je aplikovaný na vstupné pole a to takým spôsobom, že je počítaný skalárny súčin medzi konvolučným filtrom a určitým výsekom vstupného poľa, nazývaným recepčným poľom. Počiatok vstupného poľa je v ľavom hornom rohu a následne sa filter posúva definovaným krokom. V momente keď je filter aplikovaný na celé vstupné pole získavame aktivačnú mapu pre daný filter.[12]

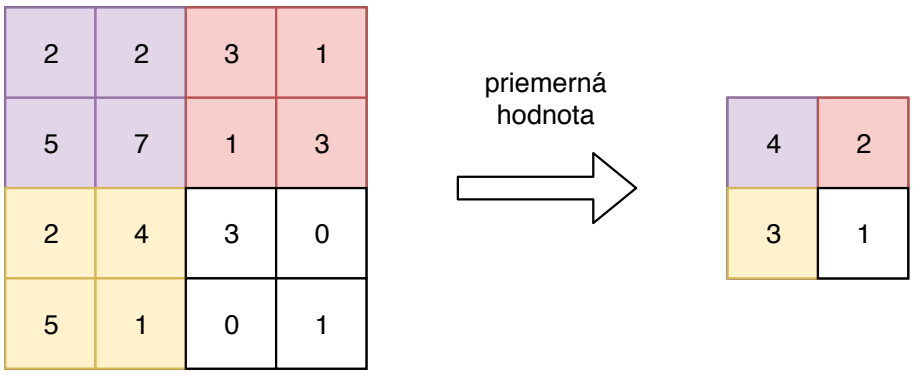
Z tejto operácie vychádza niekoľko dôležitých vlastností konvolučných neurónových sietí. Jednou z nich je riedka spojitosť, ktorá je dosiahnutá niekoľko násobne menším rozmerom filtra ako sú vstupné dáta. Konvolučný filter je aplikované len na recepčné pole, čím sa zníži zložitosť výpočtu. Druhou vlastnosťou je zdieľanie parametrov, kde rovnaký parameter je využívaný pre konvolúciu celého vstupu. Tento parameter je v našom prípade filter a výhodou tejto vlastnosti je zníženie pamäťovej náročnosti.[13]



Obr. 2.1: Operácia 2D konvolúcie v konvolučnej vrstve

2.2.2 Združovacia vrstva

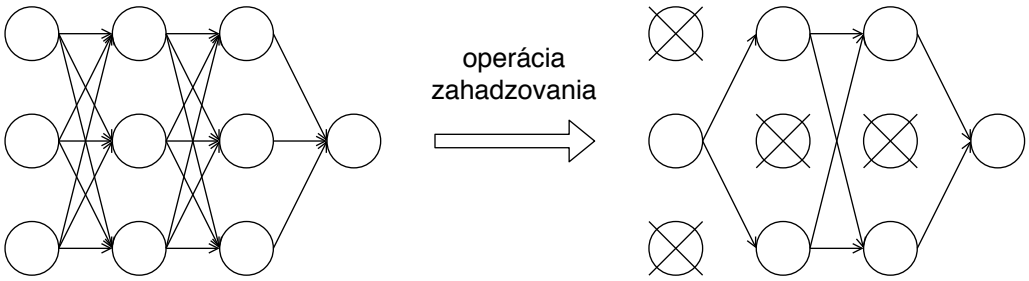
Združovacia vrstva sa stará o zmenšenie rozmerov vstupu do nasledujúcej vrstvy a tým počet parametrov siete. Podobne ako konvolučná vrstva obsahuje filtre, ktoré sú aplikované na aktivačné mapy, poprípade vstupné dáta neurónovej siete. Filter má určité rozmery $n \times n$ a jeho úlohou je zlúčiť hodnoty daného bloku do jedného výstupu. Zlučovanie prebieha podľa zvolenej metódy. Najčastejšie sa jedná o metódy výberu maxima alebo o priemernú hodnotu daného bloku. Pomocou združovacej vrstvy môžeme predchádzať pretrénovaniu sieti a taktiež sa do sieti zavádza väčšia nezávislosť na posunutí vstupného obrazu. Na obrázku 2.2 môžeme vidieť princíp združovacej vrstvy s metódou priemerovania hodnôt.[14]



Obr. 2.2: Princíp združovacej vrstvy s metódou priemerovania hodnôt

2.2.3 Vrstva zahadzovania

Vrstva zahadzovania sa v konvolučných neurónových sieťach využíva väčšinou v kombinácii s plne prepojenou alebo združovacou vrstvou. Úlohou tejto vrstvy je náhodne deaktivovať určité počet neurónov na základe zadaného parametru. Výsledkom tohoto procesu má byť ochrana konvolučnej neurónovej sieti pred pretrénovaním, čo je jeden z hlavných problémov týchto sietí. Pretrénovanie je stav pri ktorom si sieť na základe veľkého množstva vstupných parametrov vytvorí komplikované väzby, z modelu sa stratí generalizácia a stane sa príliš konkrétnym. Dôsledkom pretrénovania je negatívny vplyv zvýšenia chybovosti validačných dát aj v prípade, že na trénovacích dátach dostávame výrazne nižšiu chybovosť. Princíp vrstvy zahadzovania môžeme vidieť na obrázku 2.3 v pravo.[15]



Obr. 2.3: Princíp vrstvy zahadzovania

2.2.4 Ďalšie vrstvy konvolučnej neurónovej sieti

Medzi ďalšie využívané vrstvy v konvolučných sieťach patrí softmax vrstva a plne prepojená vrstva. Softmax vrstva sa najčastejšie využíva pri kategorizácii vstupných dát, kde výstupom je vektor hodnôt, ktorého súčtom získavame hodnotu 1 a môžeme ho interpretovať ako pravdepodobnosť.

Vlastnosti plne prepojenej vrstvy vychádzajú z jej názvu. V tejto vrstve je každý neurón spojený s každým neurónom nasledujúcej vrstvy. Plne prepojená vrstva sa v konvolučných neurónových sieťach väčšinou nachádza, ako výstupná vrstva a má za úlohu klasifikáciu vstupných dát do jednotlivých tried na základe príznakov získaných predchádzajúcimi vrstvami. Štruktúru takejto vrstvy môžeme vidieť na obrázku 2.3 v ľavo.[14]

2.3 Pokrok v oblasti klasifikácii obrazu

Klasifikácia obrazu pomocou konvolučných neurónových sietí sa neustále vyvíja. Cieľom je vylepšiť architektúru sietí, optimalizačné techniky a mnoho ďalších vlastností

pre efektívnejšiu prácu týchto sietí. Keďže táto tematika je veľmi rozsiahla v nasledujúcej sekcii sú stručne popísané niektoré moderné prístupy a vylepšenia poslednej doby v tejto oblasti.

2.3.1 Architektúra

Medzi najvýraznejšie vylepšenia na konvolučnej vrstve patrí vylepšenie s názvom sieť v sieti a dvojitá konvolúcia. **Sieť v sieti** sa zaoberá nahradením konvolučnej vrstvy viacvrstvovým perceptrónom. Dôvod tejto zámeny je, že konvolučná vrstva je vhodnejšia pre učenie skrytých vlastností, ktoré sú lineárne oddeliteľné, avšak nie pre extrahovanie abstraktnej reprezentácie z obrázku.[16] V druhom prípade, ako už z názvu vyplýva je v konvolučnej vrstve využívaná **dvojitá konvolúcia**. Myšlienka je naučiť sa zhluky filtrov, ktoré sa v rámci zhluku navzájom prekladajú. Aby sa toho dosiahlo konvolučná vrstva alokuje súbor meta-filtrov, ktoré majú väčšiu veľkosť ako je efektívna veľkosť filtra. Efektívne filtre sú potom extrahované z meta-filtrov a následne sú zhlukované.[17]

Vylepšovanie združovacej vrstvy a techniky, ktoré sú na nej použité je nevyhnutné z dôvodu zníženia výpočtovej náročnosti konvolučných sietí s ich zväčšujúcou sa architektúrou. Medzi tieto zlepšenia patrí $\mathbf{L_p}$ **združovanie** [18] , ktoré ukázalo, že dokáže lepšie generalizovať v porovnaní s metódou výberu maxima. Táto metóda sa teší stále väčšej obľúbenosti, pravdepodobne z dôvodu schopnosti lepšie zachytiť nemennosť v obraze. Metódy založené na stochastickom princípe, ako **stochastické združovanie** a **zmiešané združovanie**[18] dávajú výhodu nad metódou výberu maxima v predchádzaní pred pretrénovaním sieti. Nevýhodou je ich výpočtová náročnosť v porovnaní s inými deterministickými metódami. Ďalšími metódami používanými v združovacej vrstve môžu byť napríklad **spektrálne združovanie**, **združovanie priestorových pyramíd** alebo **transformačné invariantné združovanie**, ktore sú detailne popísané v literatúre[18] . Každá zo spomenutých metód má svoje výhody a nevýhody a voľba konkrétnej metódy bude závisieť od požiadaviek konkrétnej úlohy a od dostupných výpočtových zdrojov.

2.3.2 Nelineárne aktivačné funkcie

Voľba správne nelineárnej funkcie má veľký význam pre presnosť klasifikácie. Často využívaná funkcia s názvom sigmoid, ktorá má nevýhodu v nerovnomerných výstupoch, pretože vyhladzuje lokálny gradient, čo má vplyv na dynamiku sieti. V dôsledky týchto vlastností bola vyvíjaná funkcia s názvom **ReLU**, ktorá výrazne zrýchľuje konvergenciu. Avšak v situácii, ak táto funkcia pri trénovaní pracuje s veľkým gradientom, môže byť nenávratne stratený. Tieto dôvody viedli k inovácii ReLU aktivačnej funkcii, ako **PReLU**, **ELU**, **APL**, **SReLU**[18]. Napriek dobrým

empirickým výsledkom je potrebné testovať spoľahlivosť týchto funkcií na rozličných úlohách klasifikácie. Ďalšou funkciou môže byť **Maxout a Probout funkcia**, ktorá je výhodná pri trénovaní s vrstvou zahadzovania. Jej nevýhodou je však vysoký počet parametrov v dôsledku čoho je zvýšená výpočtová náročnosť siete.[18]

2.3.3 Stratové funkcie

Najznámejšou a najrozšírenejšou stratovou funkciou je softmax a to hlavne pre jej jednoduchosť a intuitívne výstupy. Napriek svojim výhodám a prijateľným výkonom nepodporuje vnútornú oddeliteľnosť medzi triedami, pretože využíva kosínusovú vzdialenosť pre určenie klasifikačného skóre, takže sa skôr jedná o uhlovú podobnosť. Aj z tohoto dôvodu boli vyvíjané ďalšie funkcie, ako **kontrastná** alebo **triplet stratová funkcia**, ktoré dokázali extrahovať viacej diskriminačných znakov.[18] Hoci tieto straty podporujú diskriminačné učenie, ich problémom je, že počet tréningových párov ale trojíc môže stúpnuť až na $O(n^2)$, kde n je počet tréningových vzoriek. Ďalšou funkciou je **zhluková stratová funkcia**, ktorej sa podarilo urýchliť konvergenciu sieť a stabilizovať tréningový proces, avšak je ešte potrebné vykonať dôkladné testy na klasifikačných úlohách.[18]

2.3.4 Mechanizmy regulácie

Ako už bolo spomínané veľkým problémom konvolučných neurónových sietí je ich pretrénovanie. O zamedzenie tohoto javu sa okrem iných možností stará vrstva zahadzovania popísaná v kapitole 2.2.3. Každá z týchto vrstiev využíva techniku, ktorá sa pokúša minimalizovať tento negatívny jav.

Rýchle zahadzovanie je technika, ktorá poskytuje konvolučnej neurónovej sieti zrýchlenie procesu trénovania s vyššou stabilitou. Nevýhoda je komplikovanejší tréning počas spätného prechodu, ktorý je potrebné zjednodušiť.[18] Ďalšími technikami sú napríklad **Standout** alebo **Priestorové zahadzovanie**, ktoré preukázali sľubné výsledky, ale je potrebné ich dôkladne testovať na klasifikačných úlohách.[18] Technika **Evolučného zahadzovania** má podobné charakteristiky ako Standout a môže zlepšiť konvergenčné vlastnosti a presnosť sieti, avšak jej komplementárne výpočty pravdepodobnosti zvyšujú výpočtové zaťaženie.[18] Poslednou spomenutou technikou je **Zahadzovanie spojení** umožňujúce trénovanie veľkých modelov bez problému pretrénovania, no je pomalšia. Najmä techniky, ktoré sa zameriavajú na zníženie výpočtového výkonu môžu mať v budúcnosti veľký prínos. Navyše vzhľadom na to že skutočná inteligencia má vysokú adaptačnú povahu predpokladá sa, že budúci vývoj bude zameraný na techniky podobné Standout alebo Evolučnému zahadzovaniu.[18]

2.3.5 Optimalizačné techniky

Tréning neurónových sietí sa začína počiatočnou inicializáciou váh. Rozumnou inicializáciou týchto váh môžeme predchádzať problému miznutia gradientu a tým zlepšiť konvergenciu siete.[19] Bežne rozšírenou inicializačnou technikou v neurónových sieťach je technika **Xavier**, ktorá urýchľuje konvergenciu, avšak nie je vhodná pre nelineárne aktivačné funkcie a tým pádom pre moderné hĺbkové konvolučné neurónové siete.[18] Tento problém viedol k nastavovaniu váh náhodne, čo má za následok zvýšenie tréningového času a náročnosti na výpočtový výkon. Nedávno navrhnutá technika s názvom **LSUV inicializácia**, ktorej prístup je založený na údajoch, preukázala sľubné empirické výsledky, no aj napriek tomu nie je vhodná pre veľké súbory údajov a má zložité postupy. Z týchto faktov vyplýva, že pri voľbe inicializačnej schémy je potrebné brať v úvahu parametre ako aktivačnú funkciu, hĺbku siete, dostupný výpočtový výkon a veľkosť dostupného súboru údajov.

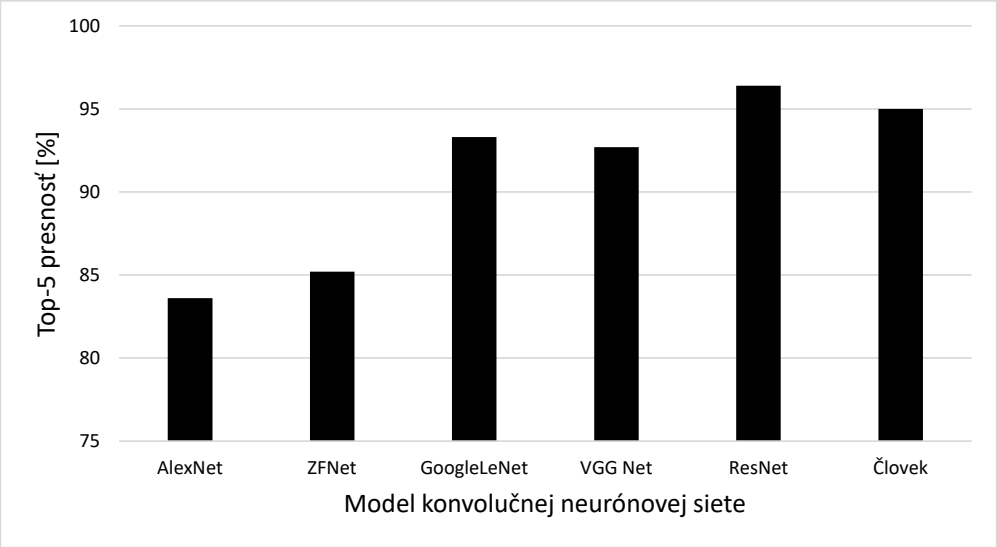
Ďalšou optimalizačnou technikou je **dávková normalizácia**[20], ktorá výrazne skrátila tréningové časy tým, že znižuje celkový počet opakovaní potrebných pre konvergenciu, pretože prostriedky v konvolučnej vrstve sa dajú zdvojnásobiť, čím sa eliminuje účinok vygenerovanej réžie. Cenou za tieto výhody je približne 30% nárast výpočtových nákladov. Spomínaná LSUV inicializácia má podobné normalizačnú procedúru, ako dávková normalizácia. Na základe tohoto sa môže považovať za inicializačnú schému doplnenú o túto normalizáciu. Účinnosť dávkovej normalizácie je závislá na veľkosti dávky, čo môže byť zmiernené **renormalizáciou**, ktorá je ľahko realizovateľná a významne zlepšuje trénovanie obmedzených dávok. Táto technika zavádza ďalšie parametre, ktoré je potrebné ladiť a ešte stále sa jedná o novú techniku.[18]

2.4 Konvolučné neurónové siete a ImageNet

2.4.1 Vývoj konvolučných neurónových sietí

Každý rok na celom svete vznikajú nové architektúry konvolučných neurónových sietí alebo sú vylepšované existujúce za účelom znižovať chybu klasifikácie obrazu. V mnohých súťažiach sa následne snažia tieto architektúry preukázať svoju presnosť. Súťažou s veľkým významom v tejto oblasti je ILSVRC (ImageNet Large Scale Visual Recognition Challenge)[21] v ktorej jednotlivé tímy súťažia v oblasti klasifikácii obrazu alebo detekcii objektov v obraze a videu. Súťaž prebieha od roku 2010 a jej ďalším cieľom je vytvoriť najväčší dátový súbor ručne označených obrázkov s názvom *ImageNet*.

Na obrázku 2.4 môžeme vidieť presnosť modelov , ktoré dosiahli významne výsledky na dátovom súbore ImageNet a posunuli vpred problematiku strojového učenia a konvolučných neurónových sietí. Modely týchto sietí sú zoradené podľa vývoja v rokoch 2012 až 2015, kde najstaršia sa nachádza na ľavej strane a zároveň sú porovnané s presnosťou človeka. Úspešný príbeh techník hĺbkového učenia a ich vývoj zavŕšila sieť s názvom ResNet v roku 2015, ktorej sa dokonca podarilo človeka prekonať. V tabuľke 2.1 sú detailnejšie zhrnuté informácie o týchto modeloch. Informácie zobrazené v spomínanej grafickej závislosti a tabuľke sú prebraté z literatúry[22]. Samozrejme vývoj v tejto oblasti neskončil v roku 2015 a aj naďalej vznikali nové verzie modelov alebo dokonca kombinácie viacerých modelov.



Obr. 2.4: Presnosť modelov konvolučných neurónových sietí trénovaných na dátovej sade ImageNet

Ešte pred samotnou sieťou AlexNet bola v roku 1998 navrhnutá sieť s názvom LeNet, ktorej algoritmus sa stal implementovateľný až v roku 2010 a to z dôvodu obmedzenej výpočtovej kapacity a pamäte. V neskoršej verzii LeNet-5 bol navrhnutý algoritmus spätného šírenia, ktorý dosiahol na ručne písaných číslicach najlepšiu presnosť. Konfigurácia tejto siete bola, dve konvolučné vrstvy, dve vrstvy zahadzovania, dve plne prepojené vrstvy a jedna výstupná. Celkový počet váh siete bol 431k a počet operácií násobenia a akumulácie $1,3$M. Ako nastal vývoj v oblasti hardwaru na rad sa dostávali nové a presnejšie modely.[22]

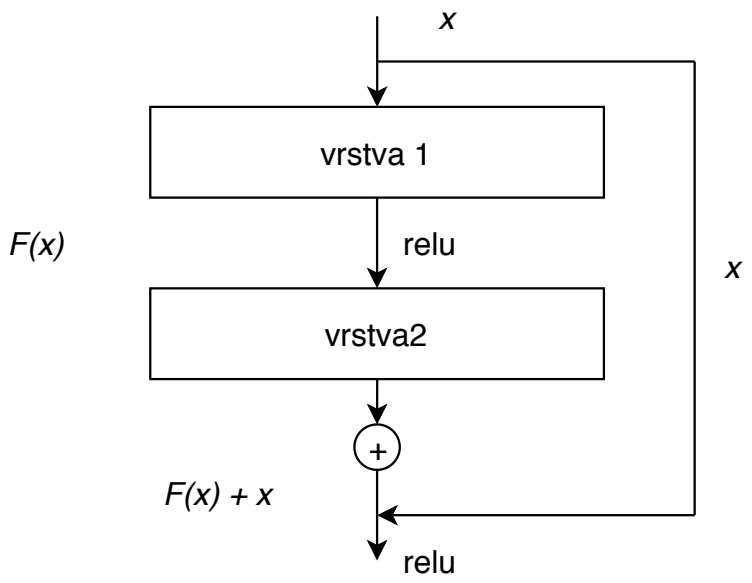
Na počiatku bol už spomínaný model AlexNet, ktorý bol širší a hlbší v porovnaní s LeNet. Tomuto modelu sa v roku 2012 podarilo vyhrať súťaž ILSVRC a stal sa významným prielomom v oblasti strojového učenia a počítačového videnia pre vizuálne

rozpoznanie a klasifikáciu úloh. Architektúra siete pozostávala z piatich konvolučných vrstiev a dvoch plne prepojených vrstiev. Počet váh tejto siete je 61M a počet operácií násobenia a akumulácie 724M. Nasledujúci model s názvom ZNFnet (Clarifai) sa stal víťazom v roku 2013 a vychádzal z princípov modelu AlexNet. Keďže konvolučné neurónové siete sú výkonovo nákladné je potrebné zabezpečiť optimálne využitie parametrov. To sa podarilo modelu ZFNet, ktorý využíva v konvolučných vrstvách jadrá z rozmermi 7×7 narozdiel od jadier 11×11 používaných v AlexNet. Táto zmena výrazne znížila počet sieťových parametrov a zlepšila celkovú presnosť rozpoznávania.[22]

V roku 2014 sa súťaže zúčastnili dva významne modely, model VGGNet a model GoogLeNet. Prvý model navrhla skupina s názvom Visual Geometry Group, ktorá na tejto súťaži obsadila druhé miesto. Najväčším prínosom tohoto projektu bolo preukázanie, že hĺbka modelu je jedným z kritických faktorov pre dosiahnutie lepších výsledkov. Boli navrhnuté tri verzie s názvom VGG-11, VGG-16 a VGG-19. Všetky tri verzie končili rovnako s tromi plne prepojenými vrstvami, avšak počet konvolučných vrstiev bol rozdielny. Model VGG-11 obsahoval 11 vrstiev, VGG-16 13 a model VGG-19 16 konvolučných vrstiev. Najrozsiahlejší model obsahoval 138M váh a $15,5$G operácií násobenia a akumulácie. Druhý model s názvom GoogLeNet v rovnakom roku súťaž vyhral. Vyvinul ho pán Christian Szegedy v spoločnosti Google s cieľom znížiť zložitosť výpočtov. Tento krok sa podarilo zrealizovať za pomoci takzvaných inception vrstiev a preto sa tento model nazýva taktiež Inception. Model obsahuje len 7M váh a $1,53$G operácii násobenia a akumulácii, čo je niekoľko násobne menej, ako modely AlexNet alebo VGG. Následne vzniklo viacej verzií siete alebo jej kombinácií. Keďže tento typ siete bol primárne využívaný v tejto práci, bližší popis jednotlivých verzií a taktiež princíp inception vrstiev je popísaný v kapitole 2.4.2.[22]

Residual Network alebo aj ResNet architektúru navrhol Kaiming He za účelom odstránenia problému miznúceho gradientu v ultra-hlbokých sieťach. Táto architektúra vyhrala súťaž v roku 2015 a bola vyvinutá s rôznym počtom vrstiev a to 34, 50, 101, 152 alebo až 1202 vrstiev. Jej princíp je zobrazený na obrázku 2.5. Populárna sieť ResNet50 obsahovala 49 konvolučných vrstiev a jednou plne prepojenou vrstvou. Celkový počet váh je $25,5$M A počet operácií násobenia a akumulácie $3,9$G.[22]

Najnovší príspevkom v oblasti konvolučných neurónových sietí je z apríla 2019 a to stavebný blok s názvom Res2Net, ktorý vytvára hierarchické pripojenie reziduálneho typu v rámci jedného bloku. Tento blok je možné využiť v najmodernejších architektúrach akými sú napríklad ResNet alebo ResNeXt, čím sa zabezpečí zvýšenie presnosti danej sieti. [23]



Obr. 2.5: Bloková schéma reziduálneho učenia

Tab. 2.1: Tabuľka modelov konvolučných neurónových sietí.

Názov	AlexNet	ZFNet	GoogleLeNet	VGG-16	ResNet50
Rok vydania	2012	2013	2014	2014	2015
Top-5 presnosť	83,6%	88,8%	93,3%	92,7%	96,4%
Počet konvolučných vrstiev	5	5	21	16	50
Počet parametrov	61M	-	7M	138M	25,5M
Počet operácií násobenia	724M	-	1,43G	15,5G	3,9G

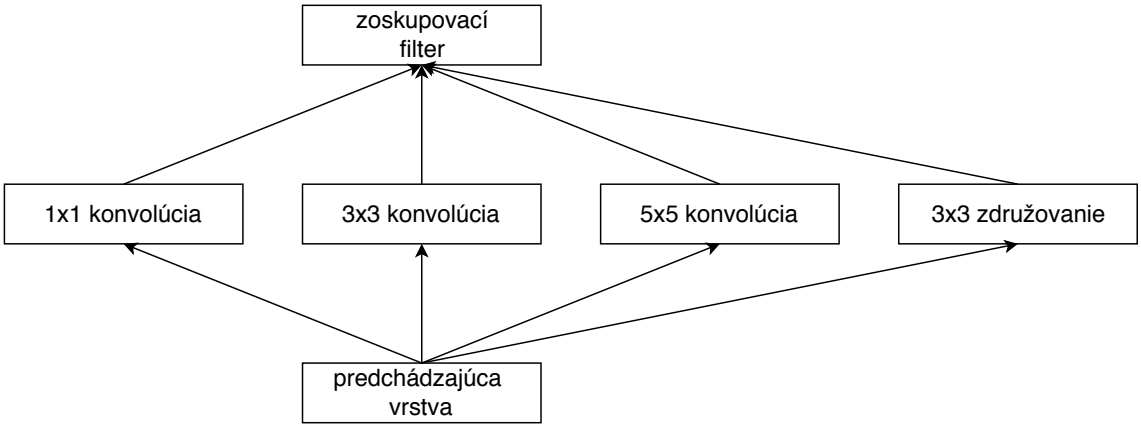
2.4.2 Využívané architektúry

V nasledujúcich riadkoch sú popísané zvolené architektúry na ktorých boli vykonávané všetky experimenty. Ako už bolo spomínané model Inception siete bol vyvinutý v roku 2014.[24] Podarilo sa jej prekonať doterajšie siete v oblasti rýchlosti učenia a taktiež v chybovosti. Tieto výsledky viedli k vývoju viacerých verzií, kde každá verzia vylepšovala tu predchádzajúcu. Dostupné verzie Inception sieti:

- InceptionV1
- InceptionV2 a InceptionV3
- InceptionV4 a Inception-ResNetV2

Základný problém, ktorý Inception architektúra rieši je, že významné oblasti obrázka môžu mať rôzne veľkosti. Napríklad, ak chcem zistiť, či sa na obrázku nachá-

dza mačka, môže byť táto mačka zobrazená na celom obrázku, poprípade len na jeho malej časti. V závislosti na tomto probléme je zložité zvoliť správnu veľkosť filtra, keďže väčšia veľkosť filtra je uprednostňovaná pre informácie, ktoré sú distribuované globálne a menšie filtre pre informácie vyskytujúce sa lokálne. Autor videl riešenie tohoto problému vo vytvorení viacerých filtrov, ktoré sa vyskytujú na rovnakej úrovni. Na obrázku 2.6 môžeme vidieť Inception modul, ktorý bol základom pre sieť InceptionV1. Nachádzajú sa v ňom tri filtre, ktorých rozmery sú 1×1, 3×3, 5×5 a jeden združovací filter. Výstupy týchto filtrov sú zoskupené a zaslané do nasledujúceho Inception modulu. Postupne bolo vytváraných viacej verzií modulov v závislosti na verzii Inception sieti.



Obr. 2.6: Inception modul

V tejto práci boli využité tri verzie Inception sietí a to InceptionV3 [25], InceptionV4 a Inception-ResNetV2.[26] Tieto siete boli zvolené na základe ich výkonnosti a zároveň boli dostupné pre moje účeli aj v predtrénovanej verzii na dátovom súbore ImageNet.

Ako už bolo spomínané každá nasledujúca verzia siete vylepšovala tú predchádzajúcu. V sieti InceptionV3 bola predstavená faktorizácia konvolučných vrstiev, ktorej hlavným účelom bolo redukovať počet parametrov siete bez zníženia jej efektivity. Ako môžeme vidieť na obr. 2.6 jeden z filtrov ma rozmer 3×3 a teda obsahuje 9 parametrov ($3 \times 3 = 9$). Pomocou faktorizácie sa podarilo znížiť počet parametrov na 6 a to spôsobom, že tento filter bol nahradený dvoma filtrami zapojenými za sebou s rozmermi 1×3 respektíve 3×1 ($3 \times 1 + 1 \times 3 = 6$). InveptionV4 priniesla zjednodušenú architektúru a viac vstupných modulov, ako bolo v InceptionV3 sieti [26].

Ďalšou použitou Inception sieťou bola Inception-ResNetV2, ktorá spájala vlastnosti Inception sietí a sietí ResNet. Princíp ResNet sietí je zobrazený na obr. 2.5 a táto vlastnosť bola použitá na Inception moduly. Táto sieť má výpočtové nákladu na

úrovni InceptionV4 sieti, avšak tréning bol rýchlejší a dosiala mierne lepšiu konečnú presnosť ako InceptionV4 sieť.[26]

Sieť ResNeXt je taktiež založená na princípoch sietí ResNet a Inception. Modul v tejto sieti vykonáva súbor transformácií s nízkym rozmerom, ktoré majú rovnakú topológiu a výstupy modulu sú združené sčítaním. Výhodou tohoto návrhu je možnosť rozšírenia o akýkoľvek veľký počet transformácií bez špeciálnych návrhov. Keďže každá cesta v sieti ResNeXt má rovnakú topológiu, napríklad na rozdiel od sieti Inception-ResNet, môžeme počet ciest určiť ako faktor, ktorý sa má skúmať. Reálne je pomerne jednoduchšie zvýšiť presnosť zvýšením šírky alebo hĺbky siete, avšak metód, ktoré sa pokúšajú zvýšiť presnosť pri zachovaní zložitosti je menej. Metóda siete ResNeXt naznačuje, že mohutnosť je merateľný rozmer, ktorý ma okrem šírky a hĺbky významný vplyv. Experimenty preukázali, že stúpajúca mohutnosť je efektívnejší spôsob získania lepších výsledkov, najmä ak šírka a výška začínajú znižovať výnosy siete. Sieť ResNeXt prekonala všetky doterajšie siete, ako ResNet a Inception na klasifikačnej množine údajov ImageNet a navyše vykazovala výrazne jednoduchšie vzory, ako tieto siete.[27]

3 Implementácia a experimenty

V tretej kapitole je popísané zvolené pracovné prostredie, využívané aplikačné rámce, súbor údajov na ktorom prebiehalo trénovanie a operácie s ním vykonané. V ďalšej časti je popísaná implementácia zvolených sietí a samotné experimenty, ktoré boli vykonané ako aj ich výsledky.

3.1 Pracovné prostredie

V súčasnej dobe existuje množstvo aplikačných rámcov implementovaných v rôznych programovacích jazykoch, ktoré je možné použiť pre prácu s hlbokými konvolučnými neurónovými sieťami. Tieto aplikačné rámce vývojárom výrazne uľahčujú prácu, pretože nie je potrebné zaoberať sa implementáciou prvkov najnižšej úrovni. Prvky potrebné pre výstavbu kompletnej neurónovej siete sú už pripravené a vývojár len využíva ich funkcionalitu. Veľké množstvo aplikačných rámcov umožňuje výpočet na grafických kartách, čo výrazne urýchľuje prácu. V našom prípade boli zvolené aplikačné rámce Keras a TensorFlow, ktoré sú detailnejšie popísané v nasledujúcich riadkoch a sú implementované v jazyku Python s voľne dostupnou licenciou.

Keďže zvolený dataset popísaný v kapitole 3.2 je pomerne rozsiahly, trénovanie architektúry a následne experimentovanie nebolo možné na bežne dostupných grafických procesoroch. Z tohoto dôvodu bola zvolená cloudová služba s názvom Google Collaboratory, ktorá je voľne dostupná a je založená na Jupyter Notebook pracovnom prostredí.[28] Služba podporuje výpočet na GPU a sú v nej nainštalované všetky potrebné knižnice, ako TensorFlow, Keras a ďalšie potrebné pre správnu funkčnosť. Služba nám dovoľuje si prenajať grafickú kartu *NVIDIA Tesla K80* s 12 GB RAM a taktiež je dostupných 12 GB operačnej RAM. Dáta potrebné pre trénovanie architektúry je možné uložiť na disk Google Drive.

3.1.1 Keras + TensorFlow

TensorFlow[29] bol vyvinutý firmou Google, ako nástroj určený pre numerické výpočty a strojové učenie, ktorý je voľne dostupný a ako už bolo spomínané je implementovaný v jazyku Python. TensorFlow pre výpočet využíva grafy dátových tokov, kde uzlami sú reprezentované matematické operácie a hrany reprezentujú tenzory, čiže viacrozmerné dátové polia. Výpočet môže prebiehať na viacerých CPU alebo GPU. Využívanie aplikačného rámcu TensorFlow môže byť pre nových užívateľov náročné, pretože sa jedná o nízko úrovňovú knižnicu. Z tohoto dôvodu bol vytvorený aplikačný rámec s názvom Keras[30], ktorý poskytuje jednoduché rozhranie využívajúce pre svoje výpočty TensorFlow, poprípade iné dostupné aplikačné rámce. Jeho

výhodou je možnosť pomerne rýchlo vytvoriť hlboké architektúry sietí s ktorými je možné experimentovať. Keras taktiež, ako TensorFlow umožňuje pre výpočet využívať CPU aj GPU.

3.2 Trénovacie, validačné a testovacie dáta

V tejto práci bol zvolený súbor údajov, použitý v súťaži *iFood 2018 Challenge* na serveri Kaggle.[31] Súťaž sa zaoberala kategorizáciou jedla, čo je dôležité pri monitorovaní príjmu potravy a zachovania zdravej výživy. Klasifikácia potravín je náročný proces, pretože existuje veľké množstvo kategórií a jedlá môžu byť veľmi podobné po vizuálnej stránke.

Súbor údajov pozostáva z farebných obrázkov rôznych rozlíšení, ktoré sú rozdelené do 211 tried. Každá z týchto tried obsahuje rôzny počet obrázkov. Trénovacie dáta tvorí 101 733 obrázkov, validačné 10 323 a testovacie 24 088 obrázkov získaných z webových vyhľadávacích nástrojov. Keďže Keras obsahuje funkciu ktorá dokáže určiť triedu obrázka na základe priečinku v ktorom je uložený, bolo potrebné tieto obrázky pre pohodlnejšiu manipuláciu rozdeliť podľa triedy. Trénovacia a validačná množina údajov má priradený súbor vo formáte *csv* v ktorom sa nachádzajú názvy obrázkov a im priradená trieda. Testovacia množina má priradený súbor len s názvami jednotlivých obrázkov. Pomocou týchto súborov a vytvoreného skriptu boli obrázky rozdelené podľa našich požiadaviek.

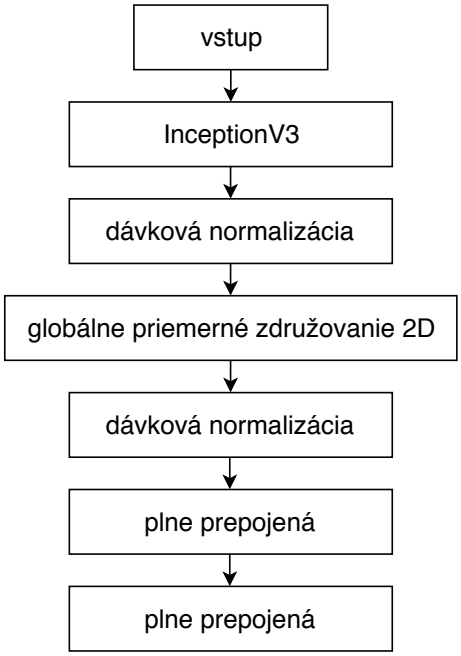
V tomto momente súbor údajov nebol iným spôsobom upravovaný a dáta boli nahrané do služby Google Drive, aby mohli byť využívané službou Google Collaboratory. Pre získanie presnosti na testovacích dátach bolo potrebné odoslať súbor vo formáte *csv* na server Kaggle. V tomto súbore boli triedy pre testovacie obrázky určené natrénovanou sieťou. Na generovanie tohoto súboru bol taktiež vytvorený skript a následne po odoslaní na server bola serverom určená chybovosť siete. Všetky výsledky, ktoré boli dosiahnuté na testovacej množine boli získané týmto spôsobom.

3.3 Implementácia

Pri voľbe sietí, ktoré budú využívané rozhodovali najmä vlastnosti, ako rozsiahlosť a výsledky, ktoré dosahovali v klasifikácii obrázkov. Pre implementáciu boli následne zvolené siete InceptionV3, InceptionV4, Inception-ResNetV2, ResNet50V2 a ResNeXt50 ktorých teoretický princíp je popísaný v kap. 2.4.2. Pre základnú konfiguráciu bola zvolená sieť InceptionV3. Spomedzi týchto sietí je rozsiahlosťou najmenšia a obsahuje viac ako 23M parametrov pri rozlíšení obrázku 299×299 pixelov s hĺbkou 159. Jedná sa o priemerne veľkú sieť, ktorej presnosť na datasete ImageNet je podľa

Keras dokumentácie 77, 9% pri top-1 presnosti a 93, 5% pri presnosti top-5. V ďalších experimentoch bola táto sieť porovnávaná s ďalšími spomenutými sieťami.[30]

Implementáciu sietí InceptionV3, Inception-ResNetV2, ResNEt50V2 a ResNeXt50 obsahuje Keras aplikačný rámec a ich využívanie bolo jednoduché. Jedná sa o jedno riadkový zápis, kde je potrebné špecifikovať základné vlastnosti siete, akým spôsobom sú inicializované váhy, zahrnutie plne prepojenej vrstvy na výstupe siete alebo rozmeru vstupného obrázka. Naopak Keras neobsahuje sieť InceptionV4 a preto bolo potrebné na serveri GitHub vyhľadať projekt, ktorý túto sieť implementuje.[32] Keďže v našom prípade sa jednalo o špecifický počet tried bolo potrebné zadefinovať výstupnú plne prepojenú vrstvu pre určenie triedy obrázka. Blokovú schému celej InceptionV3 siete je možné vidieť na obrázku 3.1 a pri ostatných sieťach topológia vyzerala rovnako až na zámenu s InceptionV3 sieťou.



Obr. 3.1: Bloková schéma implementovanej siete

Dôležitým procesom je spracovanie vstupných dát. V tomto prípade bola využitá trieda s názvom *ImageDataGenerator*, ktorá má množstvo parametrov za pomoci, ktorých je možné upravovať a rozšíriť vstupné dáta. Jednotlivé parametre a ich vplyv na dáta budú popísane pri vykonaných testoch. Táto trieda obsahuje funkciu zabezpečujúcu generovanie dávok pri trénovaní sieti. Názov funkcie je *flow_from_directory* a taktiež je potrebné špecifikovať niektoré parametre, ako je napríklad priečinok v ktorom sa dáta nachádzajú. Veľkou výhodou funkcie je, že si dokáže sama určiť triedu obrázka na základe priečinka v ktorom je uložený. Ďalšími parametrami sú napríklad požadované rozlíšenie obrázka, farebný režim v akom má byť obrázok načítaný, veľkosť dávky a náhodné načítanie dát.

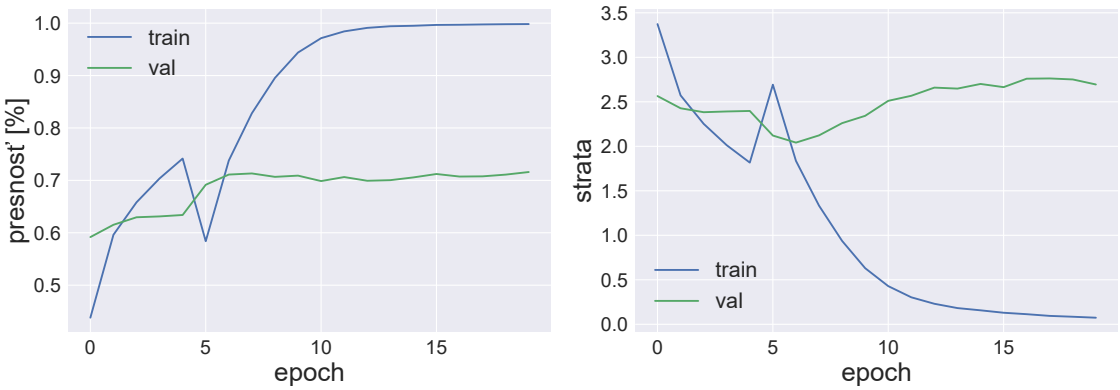
Posledným krokom bola príprava modelu k trénovaniu a samotné trénovanie. K tomuto bola využitá trieda *Model* a jej funkcie *compile* a *fit_generator*. Funkcia *compile* zabezpečuje konfiguráciu pre trénovanie. Definícia optimalizačného algoritmu sa líši na základe vykonaného testu, avšak stratová funkciu bola stále volená, ako *categorical_crossentropy* a metrika pre vyhodnotenia výsledkov modelu, ako top-3 presnosť. Druhá funkcia zabezpečovala trénovanie siete. Využíva už spomínanú funkciu *flow_from_directory* pre generovanie trénovacích a validačných dát, ďalej je potrebné definovať počet trénovacich epoch, počet krokov pre jednu epochu, maximálnu veľkosť fronty pre generovanie dát a maximálny počet procesov, ktoré sa využívajú pri generovaní dát.

3.4 Experimenty

3.4.1 Základná konfigurácia

Pri prvej konfigurácii je sieť InceptionV3 inicializovaná s váhami predtrénovanými na ImageNet súbore údajov a na výstupe sú zaradené vrstvy podľa obrázku 3.1. Vstupné obrázky sú upravené do rozlíšenia 256×256 bodov a ich hodnoty delené číslom 255, aby boli v rozmedzí 0 až 1, čo má následne výhody pri trénovaní. Vstupné dáta neboli iným spôsobom upravené ani rozšírené. Optimalizátor je pri konfigurácii nastavený, ako SGD (Stochastic Gradient Descent) [33], ktorého hodnoty rýchlosti učenia, spádu a úpadku ostali na základných hodnotách podľa Keras dokumentácie.

Trénovanie prebiehalo v 20 epochách s veľkosťou dávky 16. Prvých 5 epoch sú vrstvy siete InceptionV3 uzamknuté, teda ich váhy sa nemenia a sú trénované len pridané výstupné vrstvy, ktorých váhy sú inicializované náhodne. Následne sú odomknuté posledné dva bloky InceptionV3 siete a trénovanie pokračuje zostávajúcich 15 epoch. Na obrázku 3.2 môžeme vidieť priebeh presnosti a stratovej funkcii tejto konfigurácie. Približne na 13 epoche sme na trénovacích dátach dostali takmer 100% presnosť a sieť už nemala priestor na zlepšenie. Z grafov je tiež zrejmé, že už pri 6 alebo 7 epoche sme nedokázali zlepšiť presnosť siete na validačných dátach. V tomto momente začala presnosť validačných dát klesať a stratová funkcia naopak stúpať, čo je znakom preučenia siete. Predchádzať preučeniu môžeme rozšírením vstupných dát, regularizáciou alebo zaradením vrstvy zahadzovania medzi výstupné vrstvy siete. Tieto mechaniky boli použité v nasledujúcich experimentoch pre optimalizáciu siete. Maximálna presnosť modelu na validačných dátach bola $71,59\%$ pri top-3 presnosti s hodnotou stratovej funkcie $2,6948$ a presnosťou $70,31\%$ na testovacích dátach.



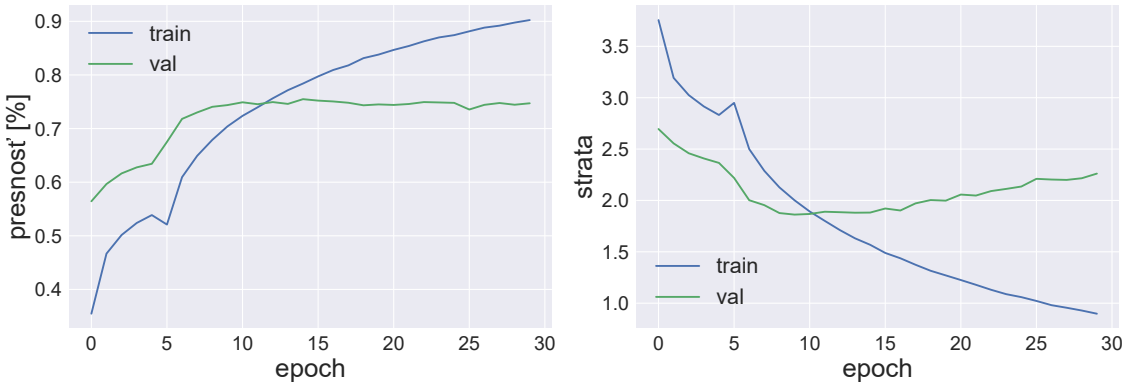
Obr. 3.2: Grafická závislosť základnej konfigurácie

3.4.2 Rozšírenie dát

Druhým experimentom bolo rozšírenie dát za pomoci triedy *ImageDataGenerator*. Cieľom tohoto kroku bolo znížiť vplyv preučenia siete a tým zvýšiť presnosť na validačných a testovacích dátach. Pre tento proces boli použité parametre, ktorých vplyv môžeme vidieť na obrázku 3.3 a umožňujú nám ľubovolne otočiť, priblížiť ale posunúť obrázok.

Obr. 3.3: Rozšírenie dát

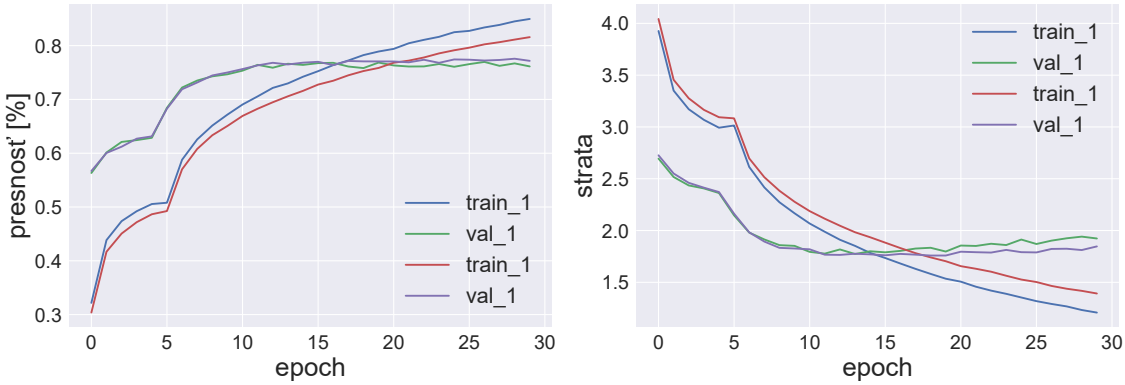
Trénovanie prebiehalo rovnakým spôsobom, ako pri predchádzajúcom experimente, avšak na 30 epochách a výsledky sú zobrazené na obrázkoch 3.4. V tomto prípade sme sieti predložili oveľa viac údajov pri trénovaní a tým sme proces učenia sťažili, čo je možné vidieť na krivke presnosti trénovacích dát. Dokonca po 30 epochách sa presnosť trénovacích dát priblížila len k hodnote okolo 90%. V učení nebolo potrebné pokračovať pretože znaky preučenia nastali pri jedenástej epoche, kedy sa presnosť na validačných dátach ustálila a stratová funkcia začala stúpať. Táto zmena zabezpečila, že presnosť na validačných dátach stúpla na $75,48\%$ s hodnotou stratovej funkcie $1,8819$ a presnosť na testovacích dátach na $74,09\%$.



Obr. 3.4: Grafická závislosť po rozšírení dát

3.4.3 Využitie vrstvy zahadzovania

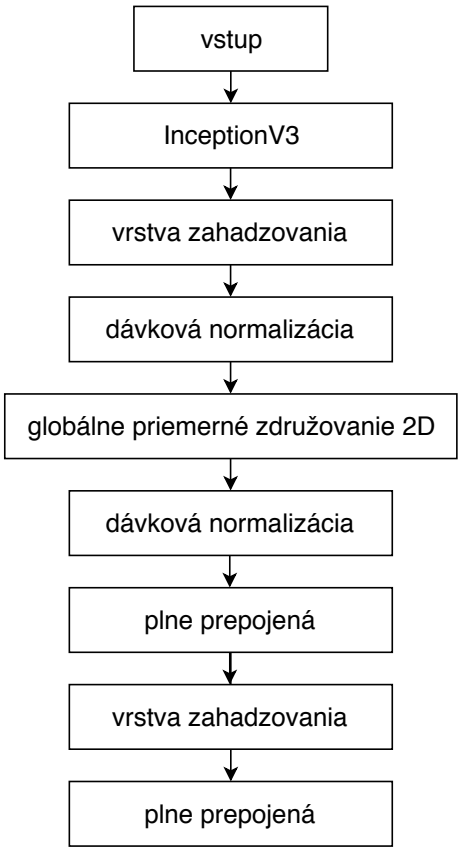
Nasledujúci experiment pozostával zo zaradenia vrstiev zahadzovania medzi výstupné plne prepojené vrstvy modelu. Všetky ostatné parametre ostali rovnaké, ako pri predchádzajúcom experimente. Po pridaní týchto vrstiev sa nám bloková schéma siete zmenila podľa obrázku 3.6. Tieto vrstvy majú taktiež prechádzať preučeniu siete a ich princíp je popísaný v kap. 2.2.3. Dôležitým údajom vrstvy je parameter, koľko percent neurónov má byť náhodne deaktivovaných. Vykonané boli dva experimenty s rôznymi hodnotami parametrov, aby bolo možné porovnať vplyv tejto vrstvy na výkonnosť neurónovej siete. Výsledky sú zobrazené na obrázkoch 3.5.



Obr. 3.5: Grafická závislosť po pridaní vrstiev zahadzovania

V prvej variante boli hodnoty parametrov týchto vrstiev nastavené na $0,3$ a $0,2$. V druhej to bolo $0,5$ a $0,3$. Z grafov môžeme vidieť, že sieť je dostatočne regularizovaná, keďže v prvých 15 epochách dostávame väčšiu validačnú presnosť ako trénovaciu. Rozdiel v presnosti na validačných dátach medzi týmito variantami je minimálny, len približne 1%. Vrstvy zahadzovania zabezpečili, že presnosť siete

sa nám opäť podarilo vylepšiť a to na hodnotu $77,57\%$ pri stratovej funkcii $1,8116$ na validačných dátach a $76,71\%$ na dátach testovacích.

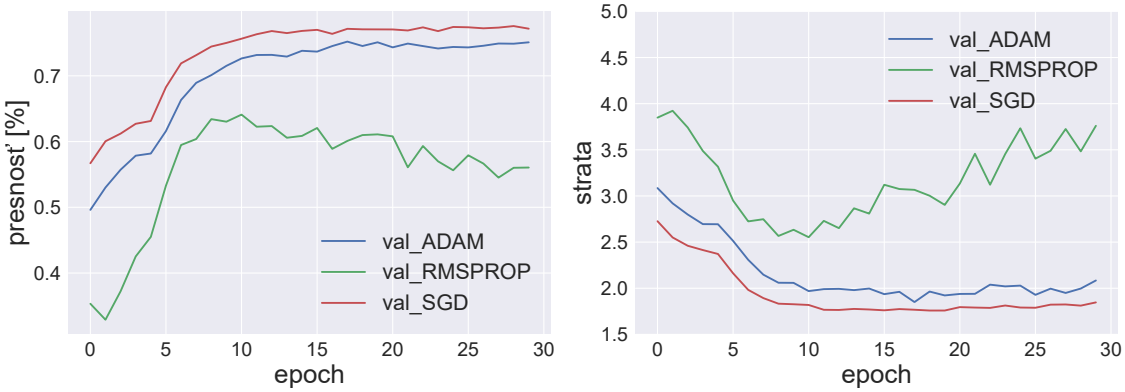


Obr. 3.6: Bloková schéma implementovanej siete s vrstvami zahadzovania

3.4.4 Vplyv optimalizačného algoritmu na neurónovú sieť

V tomto teste sme sa zamerali na vplyv optimalizačného algoritmu na výkon neurónovej siete. Optimalizačný algoritmus má veľký vplyv na proces učenia a pomáha minimalizovať stratovú funkciu vypočítanú z vnútorných parametrov siete pomocou, ktorých je určená trieda výstupu. Zamerali sme sa na tri typy optimalizačného algoritmu a to SGD [33], Adam [34] a RMSprop [35] algoritmus. Parametre týchto algoritmov ostali na základných hodnotách podľa Keras dokumentácie a ich vplyv na výkonnosť zvolenej siete môžeme vidieť na obrázku 3.7. Trénovanie siete prebiehalo na 30 epochách s veľkosťou dávky 16, kde boli dáta rozšírené rovnako, ako v kap. 3.4.2 a na výstup boli zaradené dve vrstvy zahadzovania s parametrami $0,5$ a $0,3$.

Pre prehľadnosť sú v grafe uvedené len presnosti a straty validačných dát. Z grafov je zrejmé, že najlepšiu výkonnosť mala sieť s optimalizačným algoritmom



Obr. 3.7: Grafická závislosť vplyvu optimalizačného algoritmu

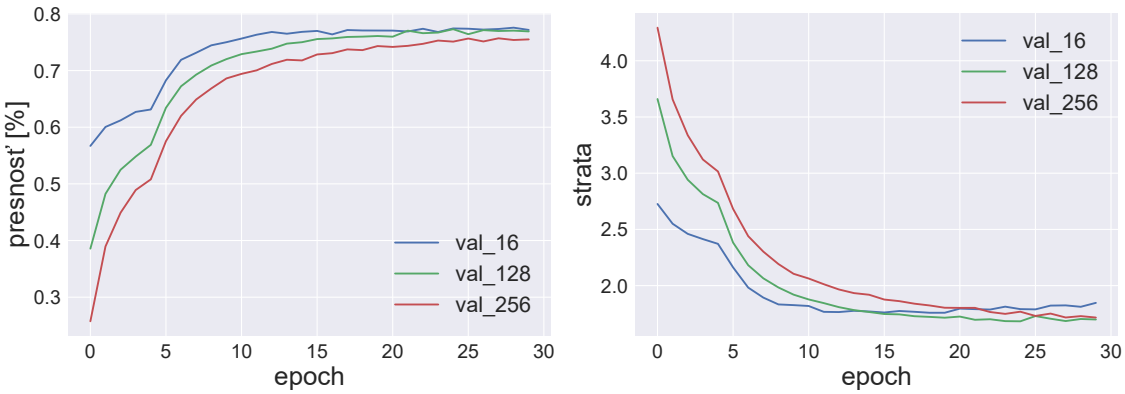
SGD, čo je zatiaľ najlepší dosiahnutý výsledok so $77,57\%$ presnosťou a $1,8116$ hodnotou stratovej funkcia na validačných dátach a s presnosťou $76,71\%$ na dátach testovacích. Druhý v poradí je algoritmus s názvom Adam. Tento algoritmus prakticky kopíroval priebeh algoritmu SGD, avšak s niekoľko percent nižšou presnosťou. Maximálna presnosť na validačných dátach pri tomto algoritme bola $75,28\%$ s hodnotou stratovej funkcie $1,9957$. Na dátach testovacích dosiahol algoritmus presnosť $74,23\%$. Pri obidvoch algoritmoch sa približne na týchto hodnotách presnosť a stratová funkcia ustálila a nezlepšovalo svoje výsledky. Posledná algoritmus s názvom RMSprop dosiahol maximálnu presnosť na testovacích dátach len $55,02\%$. V tomto prípade sa zdá, že algoritmus RMSprop nie je vhodný pre náš prípad.

3.4.5 Vplyv veľkosti dávky na neurónovú sieť

Veľkosť dávky ovplyvňuje proces učenia neurónovej siete a určuje počet vzoriek vstupných dáv po ktorých sú upravené váhy siete. Parametre siete ostali rovnaké, ako v kap. 3.4.3 s rozdielnymi hodnotami dávky. Zvolené boli tri hodnoty a to 16 s ktorou prebiehali všetky experimenty v tejto práci, ďalej 128 a 256. Trénovanie prebiehalo na 30 epochách a výslednú grafickú závislosť môžeme vidieť na obrázku 3.8.

V grafe sú opäť pre jeho prehľadnosť zobrazené len presnosti siete na validačných dátach a môžeme vidieť, že veľkosť dávky nám výrazne ovplyvnila počiatočný bod trénovania siete. V tabuľke 3.1 sú pre porovnanie zhrnuté hodnoty po prvej a poslednej trénovacej epoche. Pri veľkosti dávky 16 sme po prvej epoche dostali výrazne vyššiu presnosť siete oproti veľkosti dávky 256. Rozdiel bol približne 23% na trénovacích a 30% na validačných dátach v prospech menšej veľkosti. Presnosť dávky s veľkosťou 128 sa pohybovala medzi spomínnanými dvoma. Pri 30 epoche sme dostali takmer rovnaké presnosti pre všetky veľkosti dávky a môžeme predpo-

kladať, že ak by trénovanie pokračovalo ďalej presnosti by sa ustálili na rovnakej hodnote. V tomto momente sa nám nepodarilo presnosť siete vylepšiť, avšak sme získali poznatok, že je pre nás výhodnejšie voliť menšiu veľkosť dávke pre rýchlejšiu konvergenciu výsledku.



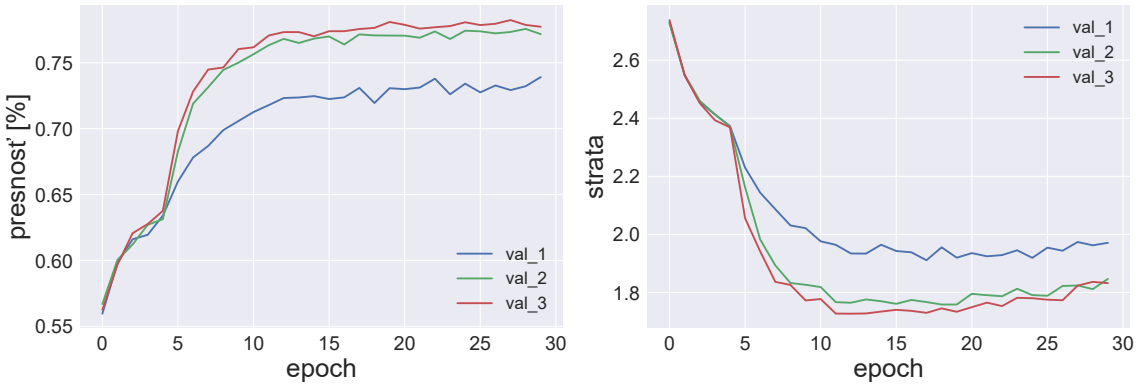
Obr. 3.8: Grafická závislosť vplyvu veľkosti dávky

Tab. 3.1: Tabuľka presnosti pre rôzne veľkosti dávky.

	trénovacia presnosť	trénovacia strata	validačná presnosť	validačná strata	testovacia presnosť
	1 epoch				
16	30,37%	4,0427	56,70%	2,7263	-
128	13,65%	4,9402	38,56%	3,6599	-
256	7,42%	5,3345	25,86%	4,2985	-
	30 epocha				
16	81,58%	1,3920	77,17%	1,8467	76,71%
128	81,18%	1,4130	77,30%	1,6825	76,53%
256	74,60%	1,7810	76,48%	1,7096	75,19%

3.4.6 Trénovanie rôzneho počtu blokov predtrénovanej siete

Ako už bolo spomínané pri všetkých experimentoch prebiehalo prvých 5 epoch trénovanie len pridaných výstupných vrstiev a všetky ostatné bloky InceptionV3 siete boli uzamknuté. Po týchto piatich epochách sa odomkli posledne dva bloky siete a trénovanie pokračovalo. V tomto experimente sme sa zamerali na vplyv počtu odomknutých blokov na výkon siete. Vykonali sme tri testy kedy bol odomknutý jeden, dva alebo tri bloky InceptionV3 siete. Grafickú závislosť týchto testov môžeme vidieť na obrázku 3.9.



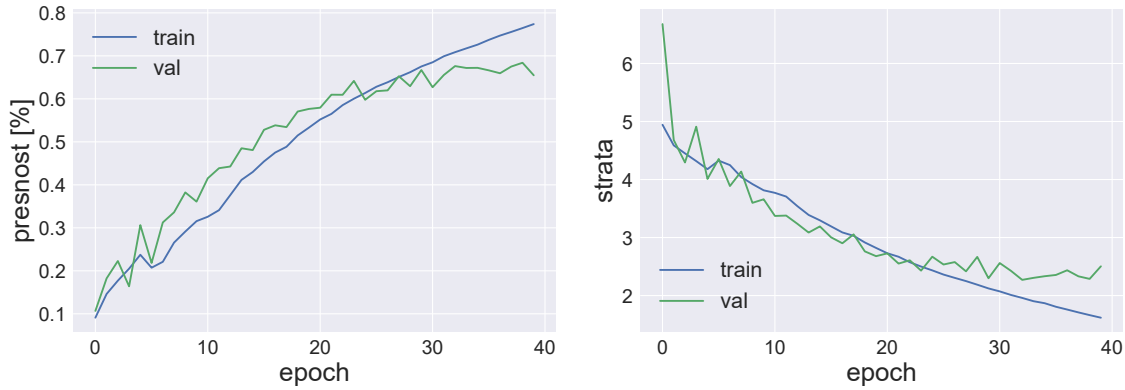
Obr. 3.9: Grafická závislosť pri trénovaní rôzneho počtu blokov

Z tejto grafickej závislosti je zrejmé, že počet odomknutých blokov siete má vplyv na jej výkonnosť. Najväčší rozdiel nastal pri odomknutí jedného a dvoch blokov. Rozdiel v presnosti na validačných dátach bol viac ako 4%. Následne po odomknutí tretieho bloku nastalo zvýšenie výkonu siete opäť, avšak len o niečo viac ako $0,5\%$. Ak sa pozrieme na priebehy presnosti siete pri odomknutí dvoch a troch blokoch môžeme vidieť, že tento nárast nebol náhodný a krivka počas celého priebehu dosahovala vyššej presnosti o spomínaného pol percenta. Aj napriek tomuto malému zvýšeniu sa nám podarilo dosiahnuť nové maximum výkonu siete a to $78,24\%$ pri strate $1,8237$ na validačných a $77,18\%$ na testovacích dátach.

3.4.7 Náhodná inicializácia váh

Tento experiment pozostával z trénovania siete, ktorej váhy boli inicializované náhodne. V tomto prípade bol použitý optimalizačný algoritmus SGD s hodnotou rýchlosti učenia $0,01$ a veľkosťou dávky 16. Pri učení boli dáta rozšírenie a štruktúra siete je rovnaká, ako na obrázku 3.6 s hodnotami parametrov vrstiev zahadzovania $0,5$ a $0,3$. Trénovanie prebiehalo na 40 epochách a výsledky môžeme vidieť na obrázku 3.10.

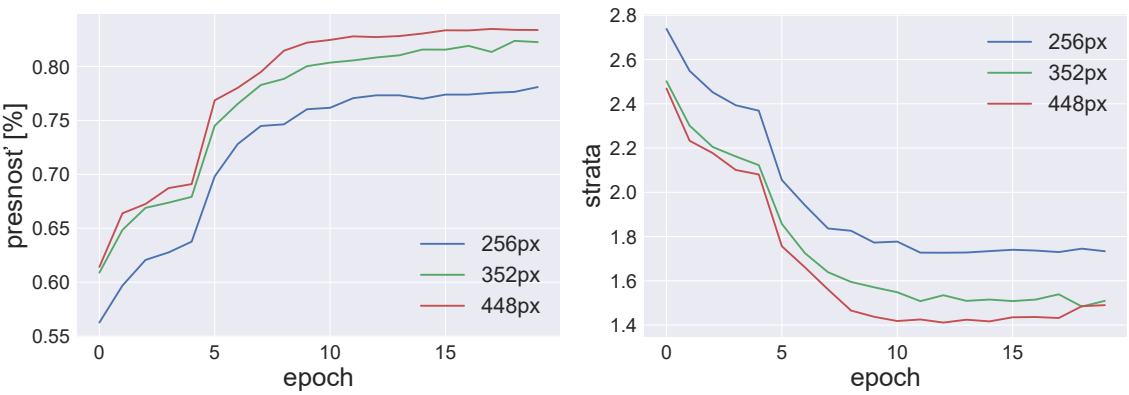
Tentoraz trénovanie prebiehalo úplne od začiatku a aj po 40 epochách sme sa dostali s presnosťou trénovacích dát len na hodnotu $77,40\%$. S trénovaním avšak nebolo potrebné pokračovať keďže sa presnosť validačných dát ustálila približne na hodnote 68%. Z tejto grafickej závislosti môžeme vidieť, že väzby ktoré má sieť vytvorené na veľkom dátovom súbore akým je ImageNet majú pre nás výhody v podobe rýchlejšieho trénovania a aj presnosti siete. Konečná presnosť siete na testovacích dátach bola len $64,68\%$.



Obr. 3.10: Grafická závislosť nepredtrénovanej siete

3.4.8 Vplyv rozlíšenia vstupného obrázku

Ďalším parametrom, ktorý by mohol zlepšiť výkonnosť neurónovej siete je rozlíšenie vstupného obrázka. Vo všetkých predchádzajúcich testoch bol tento parameter nastavený na 256×256 bodov. Teoreticky sa dá predpokladať, že čím bude rozlíšenie obrázka väčšie, tým budú konvolučné vrstvy schopné lepšie detekovať jednotlivé príznaky, čo bude mať za následok zvýšenie presnosti siete. Nevýhodou zvýšenia rozlíšenia je zvýšenie počtu vstupujúcich údajov pre každý obrázok a tým sa proces učenia predlžuje. V tomto experimente boli vykonané tri testy a to pre rozlíšenie 256×256, 352×352 a 448×448 bodov. Rozlíšenie samotných obrázkov nebolo upravované rozdiel nastal len pri spôsobe ich načítavania. Pre tieto účeli bola využitá už spomínaná trieda s názvom *ImageDataGenerator* a jej metóda *flow_from_directory*, ktorej jedným z parametrov je šírka a výška vstupného obrázka. Grafickú závislosť týchto testov môžeme vidieť na obrázku 3.11.



Obr. 3.11: Grafická závislosť rôzneho rozlíšenia vstupného obrázku

Ako sme predpokladali sieť na ktorej vstup sme priviedli obrázky s vyšším rozlíše-

ním dosiahla vyššiu presnosť. Z grafickej závislosti môžeme vidieť, že medzi prvým a druhým testom bol nárast presnosti na validačných dátach približne 4% a medzi druhým a tretím testom, už len okolo 1%. Samotné rozlíšenie obrázka má na výkonnosť sieti významný vplyv, avšak trénovanie sa v našom prípade predĺžilo o niekoľko hodín pri rovnakom type sieti. Z tohoto dôvodu trénovanie prebiehalo len na 20 epochách, čo však nemalo vplyv na maximálnu dosiahnutú presnosť. Ďalej som predpokladal, že ďalšie zvyšovanie rozlíšenia nie je potrebné a iba by proces trénovania opäť predĺžilo. Novým dosiahnutým maximom bola presnosť $83,48\%$ so stratou $1,4349$ na validačných dátach. Presnosť na testovacích dátach pri tomto modele bola $82,68\%$.

3.4.9 Voľba typu konvolučnej neurónovej siete

Všetky predchádzajúce experimenty prebiehali so sieťou InceptionV3, ktorá bola zvolená pre základnú konfiguráciu. Samozrejme voľba typu siete ovplyvňuje výsledky, ktoré dosiahneme na zvolenom dátovom súbore a z tohoto dôvodu boli vybraté ďalšie štyri typy sietí InceptionV4, Inception-ResNetV2, ResNet50V2 a ResNeXt50 ktorých princíp je popísaný v kap. 2.4.2. V tabuľke 3.2, ktorej hodnoty sú prebraté z literatury [30] môžeme vidieť presnosť týchto sietí, ktoré dosiahli na dátovom súbore ImageNet.

Tab. 3.2: Tabuľka presnosti sietí na dátovom súbore ImageNet.

Model	top-1 presnosť	top-5 presnosť
InceptionV3	77,9%	93,7%
InceptionV4	80%	95%
Inception-ResNetV2	80,3%	95,3%
ResNeXt50	77,7%	93,8%
ResNet50V2	76%	93%

Trénovanie prebiehalo 20 epoch, kde veľkosť rozlíšenia vstupného obrázka bola 448×448 bodov a ďalšie parametre, ako rýchlosť učenia, veľkosť vstupnej dávky ostali rovnaké, ako pri predchádzajúcich experimentoch. Výsledky tohoto experimentu môžeme vidieť na obrázku 3.12.

Na grafickej závislosti môžeme vidieť, že najlepšie výsledky dosiahli siete Inception-ResNetV2, ResNet50V2, ResNeXt50 a ostatné dve siete s presnosťou zaostávali. Presnosť troch najlepších sietí bola približne rovnaká a líšila sa len minimálne. Celkovo najlepšiu presnosť na testovacích dátach mala sieť ResNeXt50 a to $84,96\%$, avšak nedá sa určiť, že táto sieť je vhodnejšia, ako ostatné dve, keďže rozdieli boli minimálne a mohli byť spôsobené len náhodnosťou pri trénovaní. Cenou za toto

Obr. 3.12: Grafická závislosť rôzneho typu sietí

zlepšenie presnosti bolo opäť zvýšenie časovej náročnosti trénovania, keďže sa jedná o rozsiahlejšie siete ako je InceptionV3. Ďalej by sa dalo diskutovať o parametroch, ktoré boli zvolené pre tieto siete. Každá zo sietí mala rovnaké nastavenie. Avšak sa jedná o rôzne typy sietí a tým pádom by im mohlo vyhovovať rôzne nastavenie parametrov, čo by malo za následok dosiahnutie lepších výsledkov. Toto experimentovanie, by však bolo časovo náročné a nebolo možné sa mu detailnejšie venovať v tejto práci.

3.4.10 Rýchlosť učenia

V tomto experimente som sa venoval parametru rýchlosti učenia. Jeho vplyv počas trénovania neurónových sietí je taktiež popísaný v kapitole 1.3. Jedná sa o parameter na základe, ktorého sú upravované váhy siete so závislosťou na stratovú funkciu. Cieľom je vyhľadať lokálne minimum tejto funkcie. Samozrejme, ak zvolíme dostatočne nízku hodnotu rýchlosti učenia z veľkou pravdepodobnosťou sa nám podarí toto minimum nájsť. Problém nastáva, aký čas k tomu budeme potrebovať a môže nastať situácia kedy trénovanie prestane byť efektívne. Z tohoto dôvodu sa väčšinou volí rýchlosť učenia s klesajúcou tendenciou, prípadne s jej reštartami, v závislosti na vykonávanej iterácii.

V predchádzajúcich experimentoch bola hodnota rýchlosti učenia konštantná a mala hodnotu $0,01$. Taktiež aj v nasledujúcich testoch bola táto hodnota zvolená, ako počiatočná a prvé tri epochy nezmenená. V tomto momente prebiehalo trénovanie pridaných plne prepojených vrstiev. Následne rýchlosť učenia klesala na základe troch zvolených prístupov a bola využívaná sieť Inception-ResNetV2.

Prvým typom je časovo závislé klesanie rýchlosti učenia a je popísané nasledujúcim vzorcom.

$$lr = lr_0/(1 + ki), \tag{3.1}$$

kde

- lr - hodnota rýchlosti učenia,
- lr_0 - hodnota rýchlosti učenia predchádzajúcej epochy,
- k - parameter,
- i - poradie epochy,

Parameter k, ktorý určuje sklon klesania bol zvolený, ako podiel počiatočnej hodnoty rýchlosti učenia $(0, 01)$ a maximálneho počtu vykonaných epoch (17). Priebeh hodnôt môžeme vidieť na obrázku 3.13 a môžeme z neho usúdiť, že tento prístup je vhodnejší pre experimenty s väčším počtom iterácií, pretože v našom prípade bolo klesanie len minimálne a výsledky sa takmer zhodovali s hodnotami, kde bola rýchlosť učenia nastavená na konštantnú hodnotu.

Ďalší prístup má názov krokové klesanie. V tomto prípade rýchlosť učenia klesá o určitú hodnotu po zvolenom počte iterácií. Grafickú závislosť môžeme taktiež vidieť na obrázku 3.13 a tento prístup je možné popísať vzorcom:

$$lr = lr_0 k^{\lfloor \frac{i}{r} \rfloor}, \tag{3.2}$$

kde

- lr - hodnota rýchlosti učenia,
- lr_0 - počiatočná hodnota rýchlosti učenia,
- k - parameter,
- i - poradie epochy,
- r - poradie epochy, kedy dôjde ku klesaniu,

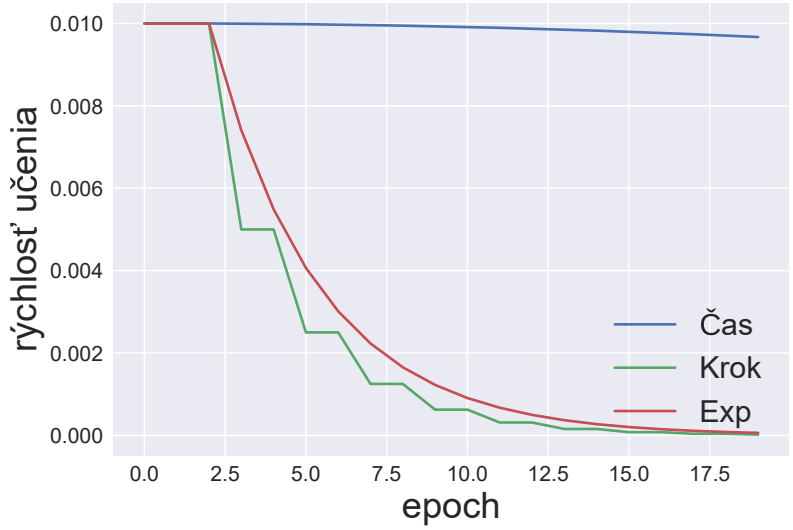
Parameter k nám v tomto prípade určuje o koľko percent má rýchlosť učenia klesnúť každú r-tú epochu. Tieto dva parametre boli v našom prípade zvolené, ako $k = 0, 5$ a $r = 2$, čo znamenalo, že každú druhú epochu sa rýchlosť učenia zníži o 50%.

Posledných typom je exponenciálne klesanie. Ako už z názvu vyplýva rýchlosť učenia klesá na základe exponenciálnej funkcii podľa nasledujúceho vzorca.

$$lr = lr_0 \exp(-ki), \tag{3.3}$$

kde

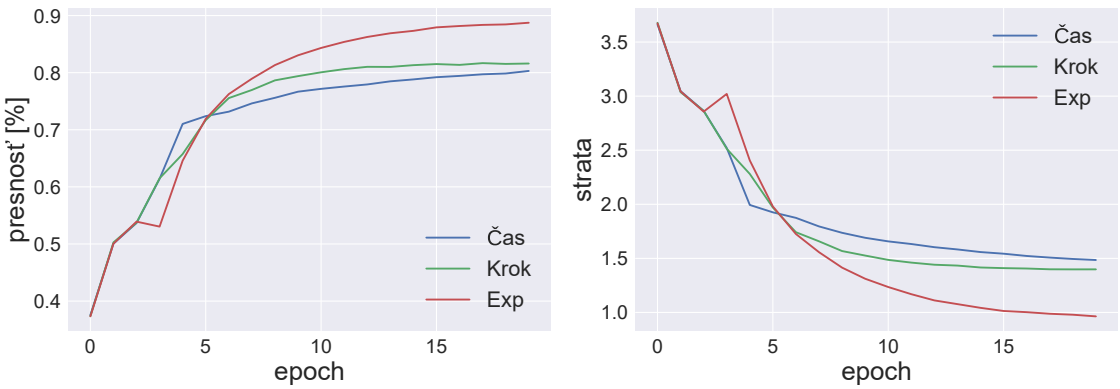
- lr - hodnota rýchlosti učenia,
- lr_0 - počiatočná hodnota rýchlosti učenia,
- k - parameter,
- i - poradie epochy,



Obr. 3.13: Závislosť rýchlosti učenia na epoche

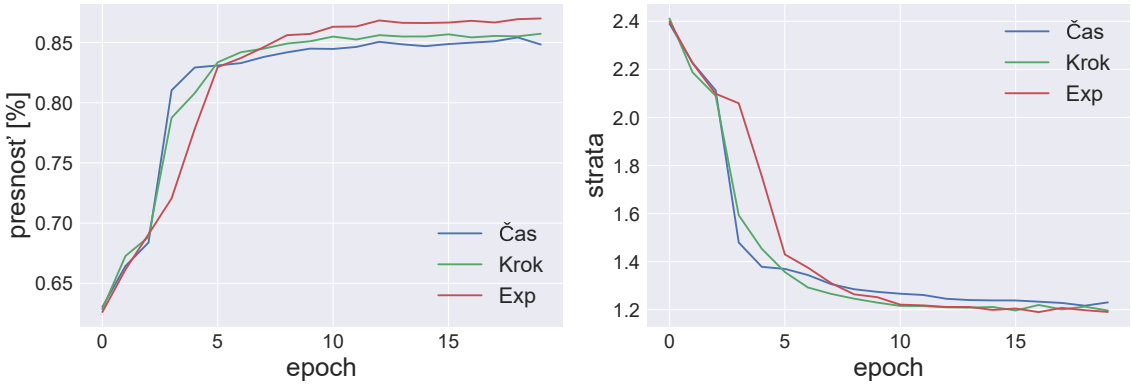
Závislosť je taktiež zobrazené na obrázku 3.13, kde môžeme vidieť jej priebeh a parameter k nám opäť určuje sklon tentoraz exponenciálnej funkcie.

Ako už bolo spomínané cieľom tohoto experimentu bolo vyhľadať čo najpresnejšie lokálne minimum stratovej funkcie a tým pádom spresniť dosiahnuté výsledky zvolenej siete. Na obrázkoch 3.14 a 3.15 sú zobrazené grafické závislosti presnosti na trénovacích a validačných dátach. Na trénovacích dátach sa táto úprava rýchlosti učenia prejavila viditeľnejšie a pri rovnakom počte epoch dosiahol exponenciálny prístup približne o 7% vyššiu presnosť, ako krokové nastavovanie rýchlosti učenia. Rovnako ako pri predchádzajúcich experimentoch ani tu pre nás nebola smerodajná presnosť na trénovacích dátach, pretože pri väčšine prípadoch táto presnosť rastie aj keď je model pretrénovaný.



Obr. 3.14: Grafická závislosť presnosti trénovacích dát na rýchlosti učenia

Na druhej grafickej závislosti a teda presnosti na validačných dátach vidíme, že exponenciálny prístup sa javil, ako najvhodnejší. Síce presnosť na validačných dátach bola vyššia len niečo viac ako o 1% celkovú presnosť na testovacích dátach sa oproti konštantnej rýchlosti učenia podarilo zlepšiť o skoro 2%. Novým najlepším výsledkom bola presnosť 86, 99% s hodnotou stratovej funkcie 1, 1906 na validačných dátach a presnosť 86, 44% na dátach testovacích.



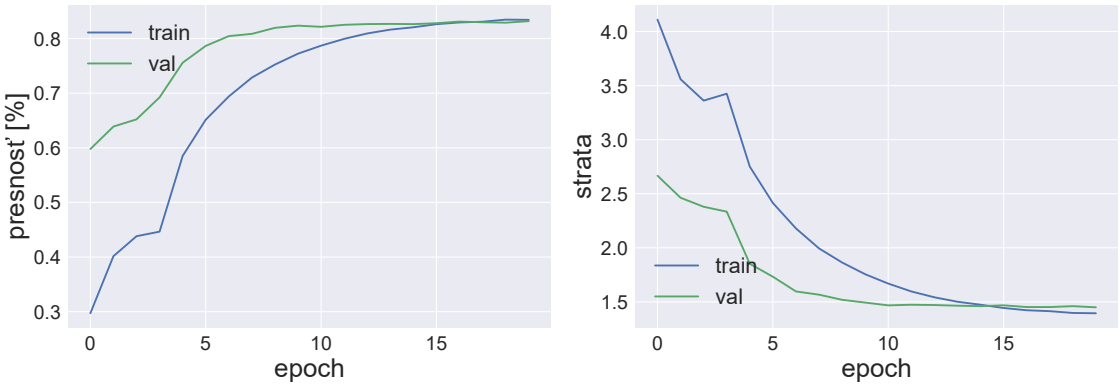
Obr. 3.15: Grafická závislosť presnosti validačných dát na rýchlosti učenia

3.4.11 Využitie testovacích dát

Posledný experiment pozostával zo zaradenia testovacích dát do procesu učenia. V tomto prípade som sa pokúsil rozšíriť trénovacie dáta o testovacie a tým pádom predložiť sieti viacej vzorov. Keďže testovacie dáta nie sú označené, teda zaradené do tried, bolo potrebné ich označiť čo najpresnejšie. Toto označenie zabezpečil model, ktorý dosiahol najväčšiu presnosť a to 86, 44%. Ovšem táto presnosť je top-3 a pre tento experiment bolo potrebné získať jednoznačné označenie obrázkov teda top-1 presnosť. Nevýhodou je že server Kaggle neposkytuje možnosť získať top-1 presnosť a teda som nevedel, akú presnosť daný model má. Podľa teoretických poznatkov modely na dátovom súbore ImageNet dosahovali približne o 15% nižšiu presnosť v top-1 presnosti, ako v top-3, čo môžeme vidieť aj v tab.3.2.

Samotné trénovanie prebiehalo štandardným spôsobom, ako v predchádzajúcom experimente a bola využívaná architektúra Inception-ResNetV2 s exponenciálnym klesaním rýchlosti učenia. Trénovacia množina dát sa rozšírila na 125 060 obrázkov a grafickú závislosť tohoto modelu môžeme vidieť na obrázku 3.16. Naše predpoklady sa nenaplnili a týmto spôsobom sa nám nepodarilo zlepšiť presnosť daného modelu. Presnosť klesla približne o 3% z 86, 44% na 83, 61% na testovacích dátach. Toto mohlo byť spôsobené nepresnou kategorizáciou testovacích dát, čo malo za následok zníženie presnosti. Na základe týchto výsledkov som predpokladal, že nie je

potrebné týmto spôsobom trénovať ďalšie architektúry a daný prístup nie je vhodný pre zlepšenie presnosti modelu.



Obr. 3.16: Grafická závislosť presnosti validačných dát pri využití testovacích dát pri trénovaní

4 Zhodnotenie výsledkov a diskusia

V práci bolo vykonaných celkovo 26 testov, ktoré sú rozdelené do 10 skupín. V každej z týchto skupín som sa zameral na určitý parameter alebo vlastnosť siete, ktorý má vplyv na presnosť zvolenej siete. V predchádzajúcej kapitole boli popísané nastavenia a parametre siete a taktiež graficky zobrazené výsledky všetkých experimentov. V tejto kapitole budú porovnané výsledky celej práce, diskutovaný vplyv jednotlivých parametrov a následne vybraná najpresnejšia konfigurácia siete. Výsledky všetkých experimentov sú zhrnuté v tabuľke 4.1, kde ich je možné porovnať.

Prvá konfigurácia bola akýmsi počiatočným bodom celej práce. Pri tomto experimente došlo k preučeniu siete hneď pri 6 alebo 7 epoche, kedy som dosiahol maximálnu presnosť na validačných dátach. Následne začala presnosť klesať a strata stúpať. Presnosť na trénovacích dátach dosiahla takmer 100% a sieť už nemala priestor, kde zlepšovať svoju presnosť.

V dôsledku tohoto bolo ďalším pokusom zmierniť následky preučenia siete za pomoci rozšírenia vstupných dát a tým predložiť sieti viacej vstupných informácií. Tento zámer sa podaril, keď presnosť na validačných dátach bola prvých 10 epoch dokonca väčšia, ako trénovacia a sieť mala väčší priestor zlepšovať svoje výsledky. Následkom tejto zmeny som dostal približne o 3% väčšiu presnosť na testovacích dátach. Ďalším spôsobom, ako zabezpečiť sieť pred preučením je zaradenie vrstiev zahadzovania do výstupného bloku siete. V konfigurácii som ponechal rozšírenie dát a pridal vrstvy zahadzovania. Pre porovnanie som vykonal dva testy s rôznymi parametrami týchto vrstiev. Týmto krokom som dosiahol, že presnosť na validačných dátach bola väčšia, ako na trénovacích prvých 15 epoch pri prvej verzii a pri druhej dokonca prvých 20 epoch. Opäť sa nám podarilo zvýšiť presnosť siete o viac ako 2%. Z grafickej závislosti sme následne usúdili, že sieť už nie je potrebné ďalej regularizovať.

V nasledujúcom experimente som sa zameral na vplyv optimalizačného algoritmu. Pre tento test boli vybraté tri algoritmy SGD, Adama a RMSprop, ktoré sú podľa teórie najvhodnejšie pre klasifikáciu obrázkov. Doteraz používaný algoritmus SGD sa preukázal, ako najvhodnejší a nepodarilo sa opäť zvýšiť presnosť siete. Možným nastavením vhodnej rýchlosti učenia a úpadku, by sa mohlo podariť zlepšiť výsledky, čomu bol venovaný samostatný experiment. Z tohoto testu vyplýva, že voľba vhodného optimalizačného algoritmu je dôležitý krok pre získanie presnej konvolučnej neurónovej siete.

Nielen voľba vhodného optimalizačného algoritmu ale aj voľba veľkosti dávky môže výrazne ovplyvniť proces učenia. V tomto experimente sme volili tri veľkosti dávky. Z grafickej závislosti sa zdá, že pri všetkých hodnotách sieť konvergovala k rovnakému výsledku, rozdiel bol v priebehu tejto závislosti. Kým dávka s menšou

veľkosťou konvergovala skorej ku finálnej hodnote, väčším dávkam to trvalo dlhšie. Tento jav mohol byť spôsobený, že pri voľbe menšej veľkosti dávky sú váhy počas jednej epochy oveľa viac krát upravované, čo vedie k väčšej presnosti siete. Z týmto súvisí aj časová náročnosť učenie, kedy sme potrebovali menej epoch na získanie maximálnej možnej presnosti.

Ďalším nastavením siete, ktoré mohlo ovplyvniť presnosť siete bol počet blokov sietí, ktoré boli trénované v druhej fáze. Ako už bolo spomínané vo všetkých experimentoch boli odomknuté posledné dva bloky siete. Z tohoto testu vyplýva, že čím viac blokov sme otvorili pre trénovanie tým väčšiu presnosť siete sme dostali. Taktiež z výsledkov môžeme predpokladať, že odomykanie ďalších blokov by už nemalo taký vplyv na presnosť a dokonca by mohlo prejsť aj do negatívneho dôsledku. S týmto dôsledkom by mohol súvisieť aj ďalší experiment, kedy boli váhy celej siete inicializované náhodne a žiadny z blokov nebol uzamknutý pre trénovanie. Trénovanie v tomto prípade prebiehalo dlhšie a presnosť na validačných dátach sa nepriblížila najlepšiemu možnému výsledku. Toto mohlo byť spôsobený tým, že predtrénovaná sieť, akou je InceptionV3 alebo ostatné siete, majú vytvorené väzby na tak veľkom dátovom súbore akým je ImageNet. Vytvorené väzby môžu byť pre nás výhodne a vytváranie nových väzieb alebo zmena vytvorených môže mať negatívny vplyv na presnosť siete.

Vo všetkých predchádzajúcich experimentoch mali obrázky rozlíšenie 256×256 bodov. Predpokladom tohoto experimentu bola závislosť presnosti siete na rozlíšení obrázka. Teoreticky som predpokladal, že čím bude obrázok vo väčšom rozlíšení, tým pádom budú mať možnosť konvolučné vrstvy extrahovať lepšie jednotlivé príznaky, čo bude mať za následok zvýšenie presnosti siete. Teoretické predpoklady sa potvrdili a s rozlíšením 448×448 som dosiahol zvýšenie presnosti modelu skoro o 6%. Ovšem tento parameter výrazne zvýšil počet informácií, ktoré boli predpokladané sieti a tým pádom sa výrazne predĺžil čas potrebný pre trénovania. Ďalšie zvýšenie rozlíšenia by mohlo poskytnúť opätovne aj navýšenie presnosti, avšak by sa čas trénovania opäť predĺžil. Z tohoto dôvodu by sa trénovanie na nami dostupnom hardvere stalo neefektívne a preto pre ďalšie experimenty bolo zvolené rozlíšenie 448×448 bodov.

Ako už bolo spomínané jednou z vecí, ktorá ma najväčší dopad na presnosť modelu je samotná architektúra siete. Sieť InceptionV3 bola zvolená z dôvodu jej rozsiahlosti, aby bolo možné dostatočne preukázať vplyv parametrov na presnosť siete. Následne boli volené architektúry sietí, ktoré sú rozsiahlejšie a časová náročnosť trénovania bola výrazne vyššia. Dalo sa predpokladať, že tieto siete dosiahnu presnejšie výsledky. Pre tento experiment boli vybraté okrem spomínanej siete InceptionV3 taktiež siete InceptionV4, Inception-ResNetV2, ResNeXt50 a ResNet50V2. Posledné tri dosiali približne rovnakej presnosti, ktorá bola vyššia takmer o 1% oproti sieti InceptionV3. Rozsiahlejšie verzie sietí, ako ResNeXt101 alebo ResNeXt152 nebolo

možné využívať z dôvodu nedostatku hardverového výkonu, keďže časová náročnosť narástla extrémne.

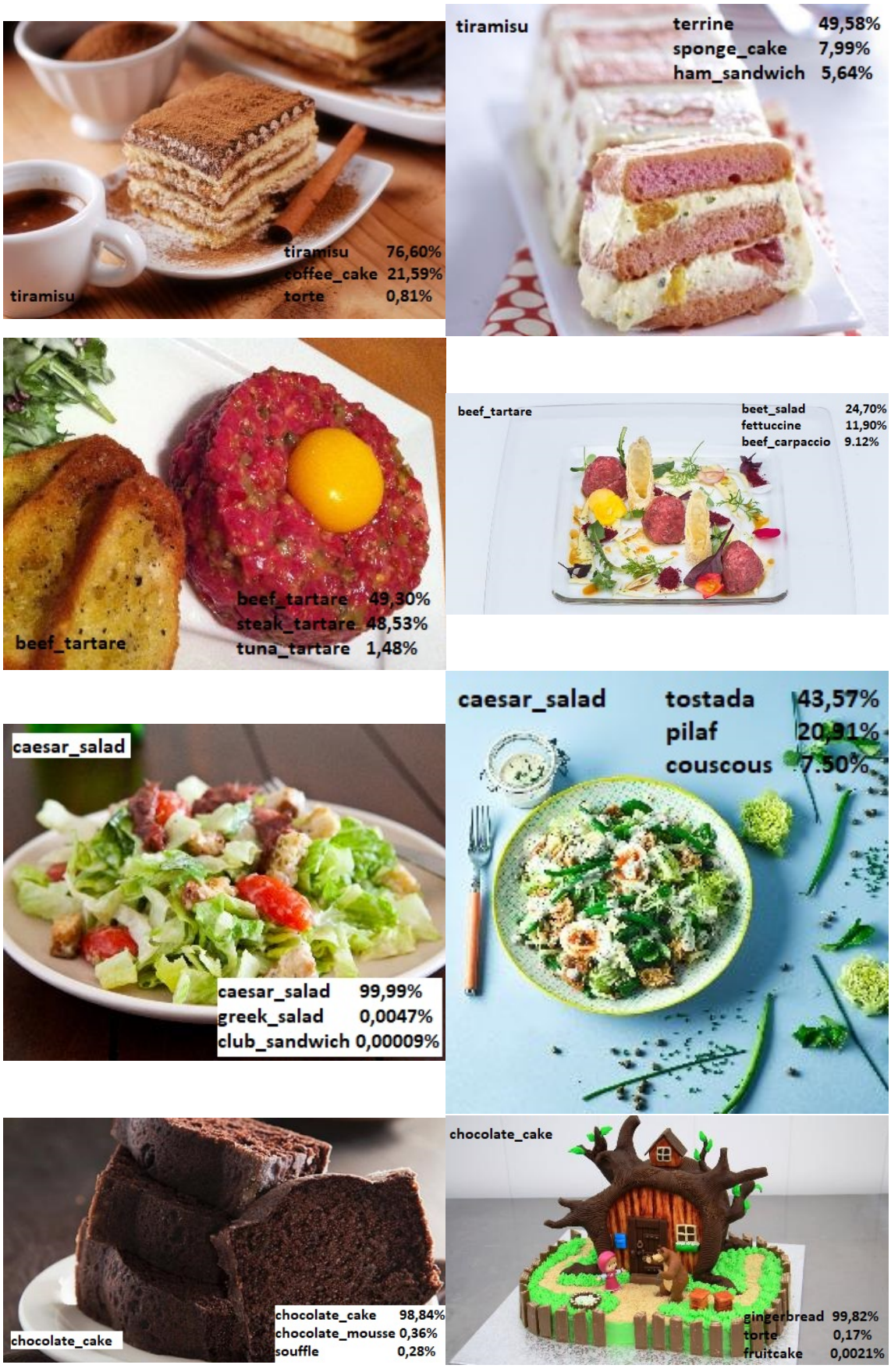
Predposledným experimentom a zároveň experimentom, kde sa podarilo dosiahnuť model s najvyššou presnosťou som skúmal vplyv rýchlosti učenia. Boli vybraté tri prístupy, akým spôsobom sa rýchlosť učenia bude nastavovať pri procese trénovania. Princípy zvolených prístupov sú popísané v kapitole 3.4.10. Boli nimi časová závislé klesanie, krokové klesanie a exponenciálne klesanie. Výsledný model siete Inception-ResNetV2 využíval exponenciálne klesanie a dosiahol finálnu presnosť na testovacích dátach 86, 44%. Posledný experiment, zaradenie testovacích dát do trénovania, nepreukázal dostatočné výsledky a z tohoto dôvodu mu nebolo venované viacej priestoru pri testovaní.

Tab. 4.1: Zhrnutie výsledkov

	trénovacia presnosť	trénovacia strata	validačná presnosť	validačná strata	testovacia presnosť
	základná konfigurácia				
	92,82%	0,0739	71,59%	2,0415	71,38%
	rozšírenie dát				
	92,23%	0,8976	75,48%	1,8819	74,32%
	vrstvy zahadzovania				
v1	83,38%	1,2911	76,97%	1,9025	76,48%
v2	81,58%	1,3920	77,57%	1,8116	76,63%
	optimalizačný algoritmus				
Adam	66,71%	2,2043	75,21%	1,8503	74.23%
RMSprop	47.87%	3,2216	64,10%	2,5530	55,45%
SGD	81,58%	1,3920	77,57%	1,8116	76,63%
	veľkosť dávky				
16	81,58%	1,3920	77,17%	1,8467	76,63%
128	81,18%	1,4130	77,30%	1,6825	76,51%
256	74,60%	1,7810	76,48%	1,7096	75,19%
	trénovanie rôzneho počtu blokov				
1 blok	74,46%	1,7663	73,92%	1,9711	73,26%
2 bloky	81,58%	1,3920	77,17%	1,8467	76,63%
3 bloky	82,72%	1,3285	78,24%	1,8237	78,07%
	nepredtrénovaná sieť				
	77,40%	1,6184	68,40%	2,2874	64,70%
	Vplyv rozlíšenia vstupného obrázku				
256px	82,74%	1,3285	78,24%	1,8237	78,07%

352px	83,02%	1,3181	82,37%	1,4839	81,98%
448px	84,19%	1,2562	83,48%	1,4319	83,95%
	Voľba typu konvolučnej neurónovej siete				
IncV3	84,19%	1,2562	83,48%	1,4319	83,95%
IncV4	83,36%	2,0803	83,01%	2,2113	82.52%
IncResV2	89,20%	0,9587	85,19%	1,3150	84,70%
ResNeXt50	86,23%	1,1408	85,41%	1,6113	84,96%
ResNet50V2	86,91%	1,0975	85,96%	1,6406	84,77%
	Vplyv rýchlosti učenia				
Čas	79,85%	1,4944	85,42%	1,2165	85,30%
Krok	81,60%	1,3984	85,72%	1,1963	85,47%
Exp	**88,74%**	**0,9637**	**86,99%**	**1,1906**	**86,44%**
	Využitie testovacích dát				
train + test	83,46%	1.3945	83,18%	1.4496	83,61%

Z výsledkov môžem konštatovať, že vyhľadanie správnych nastavení a architektúry siete pre riešenie konkrétneho problému je zdĺhavý proces. Natrénovaný model preukázal dostačujúcu presnosť pri kategorizácii jedál. Na nasledujúcich obrázkoch je možné vidieť jednotlivé výsledky, kedy sieť bola schopná jedlo správne kategorizovať a naopak kedy nie. Na ľavej strane obrázku je vždy uvedená trieda v ktorej bol zaradený v dátovom súbore a na pravej sú výsledné triedy do ktorých ho zaradil na trénovaný model.



Obr. 4.1: Kategorizácia obrázkov s natrénovaným modelom

5 Záver

Diplomová práca sa zaoberá problematikou klasifikácie obrazu pomocou hĺbkovej konvolučnej neurónovej siete. Zvolený bol dátový súbor obrázkov na ktorých sú zobrazené druhy jedál. Tento dátový súbor pozostáva približne zo 100 000 trénovacích, 10 000 validačných obrázkov rozdelených do 211 kategórií a približne 24 000 testovacích obrázkov. Vnútorná architektúra siete má významný vplyv na presnosť výsledkov a preto s ňou bolo potrebné experimentovať. Dosiahnutie uspokojivých výsledkov môže pomôcť napríklad pri dodržiavaní zdravotného životného štýlu a pri monitorovaní príjmu stravy.

Práca je rozdelená do dvoch častí. Prvá teoretická časť približuje vlastnosti konvolučných neurónových sietí a ich pokrok v oblasti klasifikácii obrazu. V druhej praktickej časti je popísaná implementácia využívaných hĺbkových konvolučných neurónovej sietí pomocou programovacieho jazyka Python a aplikačných rámcov Keras a TensorFlow. Následne boli vykonané experimenty s parametrami a architektúrou siete za účelom získať čo najlepšie výsledky pri klasifikácii obrazu. Experimenty boli rozdelené do 10 kategórií a to základná konfigurácia, rozšírenie dát, vplyv vrstvy zahadzovania, vplyv optimalizačného algoritmu, vplyv veľkosti dávky, trénovanie rôzneho počtu blokov, náhodná inicializácia váh neurónovej siete, vplyv rozlíšenia vstupného obrázka, voľba optimálneho typu konvolučnej siete, vplyv rýchlosti učenia a využitie testovacích dát pri trénovaní.

Hlavným prínosom práce je porovnanie architektúr konvolučných neurónových sietí vhodných pre riešenie problému kategorizácie jedál ďalej natrénovaný model Inception-ResNetV2 siete, ktorý dosiahol 86, 44% top-3 presnosť na testovacích dátach. Bolo vykonaných celkovo 25 experimentov, ktoré boli rôzne časovo náročné a ktoré preukázali vplyv jednotlivých parametrov na presnosť siete. Tieto experimenty sú zobrazené v prehľadných grafických závislostiach a následne diskutované a popísané. Podľa dosiahnutých výsledkov môžeme konštatovať, že presnosť siete a rýchlosť trénovania môžeme ovplyvniť mnohými parametrami a je potrebné vyhľadať kompromis pre dosiahnutie optimálneho výsledku.

Samozrejme nebolo možné odtestovať všetky dostupné techniky a parametre týchto sietí, čo by mohlo predstavovať ďalšie rozšírenie tejto práce. Taktiež nebolo možné využiť všetky architektúry konvolučných neurónových siete a to z dôvodu obmedzenia výpočtového výkonu, ktoré bolo dostupné. Najmä rozsiahlejšie siete, ako ResNeXt101 alebo dokonca ResNeXt152, by stáli za pokus a je možné, že by prekonali tento dosiahnutý výsledok.

Literatúra

[1] Eva Volná. Neuronové sítě 1. *Ostrava: Ostravská univerzita v Ostravě. Vydání: druhé*, 2008.

[2] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[3] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition. *Neural networks*, 1, 2016.

[4] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[5] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA:, 2015.

[6] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[7] Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin. Regionlets for generic object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 17–24, 2013.

[8] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Wei Hu, Yangyu Huang, Li Wei, Fan Zhang, and Hengchao Li. Deep convolutional neural networks for hyperspectral image classification. *Journal of Sensors*, 2015, 2015.

[11] Jamie Ludwig. Image convolution. *Portland State University*, 2013.

[12] Abhineet Saxena. "convolutional neural networks(cnns): An illustrated explanation, 2016.

[13] Yoshua Bengio, Ian J Goodfellow, and Aaron Courville. Deep learning. *Nature*, 521(7553):436–444, 2015.

[14] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[15] Haibing Wu and Xiaodong Gu. Towards dropout training for convolutional neural networks. *Neural Networks*, 71:1–10, 2015.

[16] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[17] Shuangfei Zhai, Yu Cheng, Zhongfei Mark Zhang, and Weining Lu. Doubly convolutional neural networks. In *Advances in neural information processing systems*, pages 1082–1090, 2016.

[18] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[19] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[21] Imagenet large scale visual recognition challenge. `http://image-net.org/`. Accessed: 2018-11-04.

[22] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: a comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[23] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *arXiv preprint arXiv:1904.01169*, 2019.

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[26] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, 2017.

[27] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[28] Jupyter notebook. `http://image-net.org/`. Accessed: 2018-11-04.

[29] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: `https://www.tensorflow.org/`.

[30] François Chollet et al. Keras. `https://keras.io`, 2015.

[31] ifood 2018 challenge. `https://www.kaggle.com/c/ifood2018`. Accessed: 2018-11-16.

[32] Github keras-inveptionv4. `https://github.com/kentsommer/keras-inceptionV4`. Accessed: 2019-03-12.

[33] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[35] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, page 14, 2012.

Zoznam symbolov, veličín a skratiek

APL	Adaptive Piecewise Linear
CPU	Central Processing Unit
CSV	Comma-Separated Values
ELU	Exponential Linear Unit
G	miliard
GB	Gigabyte
GPU	Graphics Processing Unit
k	kilo
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
LSUV	Layer-Sequential Unit-Variance
M	milion
PReLU	Parametric Rectified Linear Unit
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RGB	Red Green Blue
SGD	Stochastic Gradient Descent
SReLU	S-shaped Rectified Linear Unit
VGG	Visual Geometry Group
VGGNet	Visual Geometry Group Net

Zoznam príloh

A Obsah priloženého DVD

Zložka Modely obsahuje všetky natrénované modely, ktoré sú rozdelené do jednotlivých zložiek. Názov zložky obsahuje *X*, ktoré udáva podkapitolu experimentu, ktorého sa daný model týka a *Y* udáva poradie experimentu v danej podkapitole. Tieto zložky obsahujú súbory s príponou *.json* v ktorých je uložená história trénovania modelu, súbor *.7z*, kde je komprimovaný samotný model v súbor *.ipynb* je skript pomocou ktorého prebiehalo trénovanie a súbor *.csv* obsahuje výsledky modelu na testovacích dátach. V zložke Latex sú zdrojové súbory práce.