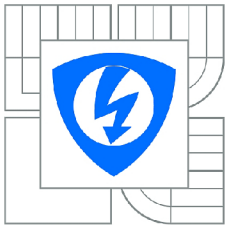
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

HLUBOKÉ UČENÍ PRO KLASIFIKACI TEXTŮ

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN KOLAŘÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUKÁŠ POVODA

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

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

Student: Bc. Martin Kolařík ***ID:*** 146862
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Hluboké učení pro klasifikaci textů

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte existující rámce hlubokého učení, které se v současné době využívají na klasifikaci a porovnejte jejich chování na testovací množině dat. Získané poznatky aplikujte při návrhu vlastní struktury hluboké neurónové sítě. Návrh sítě popište a zdůvodněte. Funkčnost vhodně prezentujte pomocí dosažených výsledků.

DOPORUČENÁ LITERATURA:

[1] SZEGEDY, Christian, et al. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. p. 1-9.

[2] ZHANG, Xiang; LECUN, Yann. Text understanding from scratch. arXiv preprint arXiv:1502.01710, 2015.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: Ing. Lukáš Povoda
Konzultant:

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

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizovaný v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systémů (SIX); registrační číslo CZ.1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

Abstrakt
Práce se zabývá rozborem současných metod strojového učení používaných pro emoční klasifikaci textových dat a testováním různých architektur neuronových sítí na problému binární klasifikace textů na pozitivní a negativní. Výstupem práce je návrh vlastní architektury hluboké konvoluční neuronové sítě, která je optimalizovaná pro textovou klasifikaci a dosáhla úspěšnosti 79,9 procent. Navrhovaná metoda není závislá na použitém jazyce a je možno ji aplikovat i při využití méně detailně vytvořených vstupních trénovacích databází. Součástí práce je také přehled teoretického základu pro práci s konvolučními neuronovými sítěmi a historie neuronových sítí. Trénovací a testovací množina dat se skládá z kratších amatérských filmových recenzí v češtině a angličtině. Skripty jsou psány v programovacím jazyce Python, využita byla knihovna pro modelování neuronových sítí Keras a výpočetní knihovna Theano. Kvůli zvýšení rychlosti výpočtu byly početní operace prováděny přes architekturu CUDA na grafické kartě.

Summary
Thesis focuses on analysis of contemporary machine learning methods used for text classification based on emotion and testing several deep neural nework architectures. Outcome of this thesis is a neural network architecture, which is tuned for using with text data and which had the best result of 79,9 percent. Proposed method is language independent and it does not require as precisely classified training datasets as current methods. Training and testing datasets were consisted of short amateur movie reviews in Czech and in English. Thesis contains also overview of theoretical basics of convolutional neural networks and history of neural networks and language processing. Scripts were written in Python, neural networks were simulated using Keras library and Theano framework. We used CUDA for better performance.

Klíčová slova
CUDA, emoce, hluboké učení, keras, klasifikace, neuronové sítě,strojové učení, theano

Keywords
classification, CUDA, deep learning, emotion, keras, machine learning, neural networks, theano

KOLAŘÍK, M.*Hluboké učení pro klasifikaci textů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 50 s. Vedoucí Ing. Lukáš Povoda.

Prohlášení:
Prohlašuji, že svou diplomovou práci na téma "Hluboké učení pro klasifikaci textů"jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonu (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne .....

..... podpis autora

Bc. Martin Kolařík

Děkuji vedoucímu práce Ing. Lukáši Povodovi za projevenou trpělivost, vlídný přístup a odbornou pomoc. Dále mojí rodině, která mě podporovala na studiích a umožnila mi výjimečně krásná studentská léta.

Bc. Martin Kolařík

Obsah

1 Teorie hlubokého učení a analýzy textů **3**
1.1 Historie neuronových sítí . . . 3
1.2 Teorie neuronových sítí . . . 4
1.2.1 Biologické základy umělých neuronů . . . 4
1.2.2 Matematický model neuronu . . . 5
1.2.3 Neuronové sítě a jejich učení . . . 6
1.2.4 Hluboké učení . . . 8
1.3 Analýza emocí v textech . . . 9
1.3.1 Význam emoční textové analýzy . . . 9
1.3.2 Dnešní metody emoční textové analýzy . . . 10
1.3.3 Metoda hlubokých konvolučních neuronových sítí . . . 10

2 Konvoluční neuronové sítě a technologie výpočtu **12**
2.1 Vlastnosti konvolučních neuronových sítí . . . 12
2.1.1 Výpočetní vrstvy . . . 12
2.1.2 Optimizační funkce . . . 16
2.1.3 Hyperparametry . . . 18
2.2 Architektury použitých neuronových sítí . . . 20
2.2.1 Dopředná konvoluční neuronová síť . . . 20
2.2.2 Inception modul a jeho použití . . . 21
2.2.3 Multi-column architektura . . . 22
2.2.4 Residuální sítě . . . 23
2.3 Vývojové prostředky . . . 23
2.3.1 Přehled dnes používaných vývojových prostředků . . . 24
2.3.2 Použité vývojové prostředky . . . 25
2.4 Databáze textů . . . 26

3 Výsledky **27**
3.1 Přehled testovaných sítí . . . 27
3.2 Porovnání testovaných sítí dle parametrů . . . 32
3.2.1 Porovnání dle velikosti konvolučního jádra . . . 32
3.2.2 Porovnání dle hloubky konvolučních vrstev . . . 34
3.2.3 Porovnání dle rozměru plně propojených vrstev výstupní části . . . 36
3.2.4 Porovnání dle optimizační funkce . . . 38
3.2.5 Porovnání dle použití inception modulu . . . 39
3.2.6 Porovnání dle použití residuální architektury . . . 40
3.3 Výsledná architektura . . . 41
3.3.1 Rozbor architektury . . . 41
3.3.2 Naměřené výsledky . . . 43

4 Závěr **44**

Úvod

Snaha modelovat chování lidského mozku stojí u vzniku umělých neuronových sítí. Díky jejich odlišnosti od klasických exaktních výpočetních algoritmů, se právě neuronové sítě hodí pro využití v situacích, které dříve dokázal zastávat pouze člověk. Jedná se především o situace, kde je potřeba generalizovat velké množství informace či klasifikovat objekty s vysokou úrovní abstrakce. Jedním z takových problémů je i klasifikace emocí. Jelikož se jedná o silně subjektivní záležitost, nelze ji dostatečně exaktně definovat.
Díky masivnímu vzestupu každodenního používání informačních technologií dnes roste množství elektronicky uložených dat exponenciálně. Většinou jde o data nestrukturovaná a kvůli jejich obrovskému množství již jejich zpracování pomocí lidské síly není možné. Proto je důležité vytvářet nové možnosti automatizovaného třídění a klasifikace dat. Z takto upravených dat můžeme poté vyčíst relevantní informace. Může se jednat o ekonomicky cenné informace, kdy dokážeme předem vyhodnotit, který zákazník by měl o jaké zboží zájem, sběr dat z mobilních zařízení a jejich periferií (chytré hodinky), může napomáhat hledání a včasnému upozornění na nepřirozenou zdravotní anomálii, či v tomto případě například jednoduše zpracovat množství textových dat, které denně tvoříme na sociálních sítích.

Cílem této práce bylo vytvoření specializované architektury konvoluční neuronové sítě, která bude optimalizovaná na problém klasifikace emocí v textech. V práci se kromě toho věnujeme testování několika architektur neuronových sítí s cílem najít správné parametry pro výslednou síť.
V první kaptiole je probrána historie neuronových sítí, jejich předobraz v lidském mozku a základy teorie umělých neuronových sítí. Také se v ní věnujeme historii a dnešním metodám zpracování přirozeného jazyka s přihlédnutím k emoční analýze. Druhá kapitola obsahuje popis vlastností konvolučních neuronových sítí, různé přístupy k tvorbě jejich architektur a přehled vývojových prostředků pro práci s nimi.
Ve třetí kapitole jsou probrány testované sítě, vyhodnocení jejich vlastností při aplikaci na emoční analýzu a také detailní popis výsledné specializované sítě, která je výsledkem této práce.

1. Teorie hlubokého učení a analýzy textů

V rámci kapitoly je probrán historický a teoretický aspekt umělých neuronových sítí a zpracování přirozeného jazyka s přihlédnutím k emoční analýze

1.1. Historie neuronových sítí

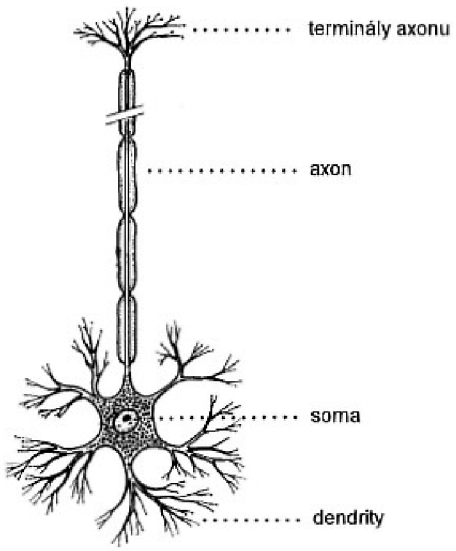
Poprvé se s myšlenkou modelování umělých neuronů setkáváme v roce 1943 publikováním práce Waltera Pittse a Warrena McCulloha, kteří vytvořili jednoduchý umělý model neuronu. Umělý neuron vycházel z funkce lidské mozkové buňky a měl bipolární výstup. V rámci práce byly představeny hypotetické první architektury jednoduchých neuronových sítí, které měly aproximovat libovolnou aritmetickou či logickou funkci. Přestože se jednalo spíše o teoretickou práci bez přímého praktického využití, položila základ nového vědního oboru. V roce 1949 byla vydána kniha Donalda Hebba, ve které vycházel ze studia podmíněných reflexů. V této publikaci bylo představeno Hebbovo učení. Jedná se o učící pravidlo pro umělé neurony, které pracuje s inhibičními a excitačními vazbami neuronů neboli umělými synapsemi.
Důležitý milník nastal v roce 1957 kdy Frank Rosenblatt vynalézá zjednodušený model umělého neuronu - perceptron, teorie perceptronu viz část 1.2.1. Zásadní rozdíl oproti původnímu umělému neuronu je ten, že pro perceptron byl představen učící algoritmus, který dokáže v konečném počtu kroků najít, v případě jeho existence, váhový vektor pro daná tréninková data. V následujících letech byl na principu perceptronu sestrojen první počítač pracující na principu umělých neuronů, neuropočítač, Mark 1 Perceptron. Byl určen k automatické klasifikaci znaků promítaných na speciální pole. Díky tomuto úspěchu byl obor neuronových sítí novým středem vědeckého zájmu do poloviny 60. let minulého století.
Velkým problémem až do 80. let ovšem byl fakt, že pro vícevrstvou síť nebyl znám učící algoritmus. Odpůrci dané problematiky často argumentovali neschopností perceptronu řešit logickou funkci xor. Problém byl vyřešen až v roce 1986 vydáním článku pánů Rumelharta, Hintona a Williamse, který popisoval algoritmus backpropagation - učení vícevrstvých sítí za pomocí zpětného šíření chyby, více o backpropagation viz část 1.2.3. Tento algoritmus stále využívá většina sítí i v dnešní době. Díky objevu zpětného šíření chyby bylo možno rozvíjet složitější vícevrstvé modely neuronových sítí a začala éra praktického využití tohoto oboru. Příkladem může být systém NETtalk, který konvertoval anglicky psaný text do mluveného slova. Tento systém během pár měsíců vývoje dokázal svojí kvalitou předčit systémy postavené na pravidlech výslovnosti. [1]
Díky vědeckému pokroku v oblasti matematických algoritmů a dnešní dostupnosti velkého výpočetního výkonu se v posledním desetiletí začalo využívat hlubokých neuronových sítí, které v mnoha ohledech již dokáží automatizovat činnosti, které byly dříve vyhrazené pouze pro člověka. Jednou z mnoha takových činností je klasifikace dat, kterou se budeme zaobírat v této práci.

1.2. Teorie neuronových sítí

V této části jsou vysvětleny základní teoretické poznatky důležité k pochopení fungování neuronových sítí. Následně je popsán termín deep learning, neboli hluboké učení.

1.2.1. Biologické základy umělých neuronů

Nervová soustava člověka nám umožňuje vnímat svět okolo nás pomocí mnoha receptorů. Její centrální prvek, lidský mozek, dokáže takto získané obrovské množství informací v reálném čase zpracovat. Inspirací pro tvorbu matematických modelů, které simulují lidské myšlení, jsou proto lidské nervové buňky - neurony. Abychom pochopili souvislost mezi biologickým neuronem a perceptronem, popíšeme si základní vlastnosti těchto stavebních kamenů lidského mozku. Obecná struktura neuronu je znázorněna na obrázku 1.1.



Obrázek 1.1: Lidský neuron [1]

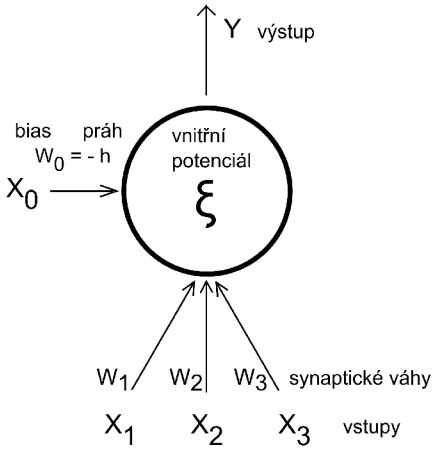
Neuron se skládá ze základních tří částí - dendritů, těla (neboli somatu) a axonu. Tělo neuronu zprostředkovává základní funkci buňky. Pro komunikaci s ostatními neurony slouží výběžky neuronu - dendrity a axon. Dendrity jsou dostředivé výběžky. Pomocí nich neuron přijímá vzruchy z ostatních neuronů. Množství dendritů se liší podle typu neuronu. Existují dokonce neurony, které žádné dendrity nemají (tzv. unipolární neurony). Dlouhý výstupní výběžek neuronu se nazývá axon. Zprostředkovává výstup informací z nervové buňky a synapsemi se napojuje na dendrity dalších neuronů. Axon má neuron vždy jeden, na konci rozdělený do množství tzv. terminálů. Chemické rozhraní, které zprostředkovává výměnu informací, vzruchů, mezi neurony se nazývá synapse. Synapse můžeme rozdělit na dva základní druhy

[1] Obrázek převzat z [1]

- excitační a inhibiční. Pokud vzruch projde synapsí excitační, je zesílen. V opačném případě, přes inhibiční synapsi, je utlumen. Předpokládá se, že lidská paměť je tvořena právě jedinečným nastavením synapsí v mozku každého člověka. Tento fakt lze brát jako analogii k učení umělých neuronových sítí a nastavováním vah pro jednotlivé perceptrony. [1]

1.2.2. Matematický model neuronu

Základním prvkem umělých neuronových sítí je umělý neuron (dále jen neuron). Schéma je znázorněno na obrázku 1.2.



Obrázek 1.2: Matematický model umělého neuronu

Lze si povšimnout jasné souvislosti s biologickým neuronem (viz obrázek 1.1). Model má obecně n reálných vstupů x. Ty simulují dostředné výběžky, dendrity. Každý vstup má přiřazen svou synaptickou váhu w. Dle hodnoty váhy se poté určuje charakter vstupu x. Pokud je váha záporná, vstup je inhibiční, v opačném případě se jedná o vstup excitační. Suma vstupních hodnot představuje vnitřní potenciál neuronu ξ.

$$\xi = \sum_{i=1}^{n} w_i x_i - h \tag{1.1}$$

Výpočet vnitřního potenciálu neuronu [1]

Dle typu přenosové funkce poté neuron reaguje na potenciál odezvou $y = \delta(\xi)$ kde y je výstup neuronu a δ je přenosová funkce neuronu. Základní bipolární skokovou funkci, ostrou nelinearitu můžeme popsat:

$$\delta(\xi) = \begin{cases} 1, & \text{jestliže } \xi \geq \text{h} \\ 0, & \text{jestliže } \xi \leq \text{h} \end{cases} \tag{1.2}$$

Ostrá nelineární aktivační funkce δ [1]

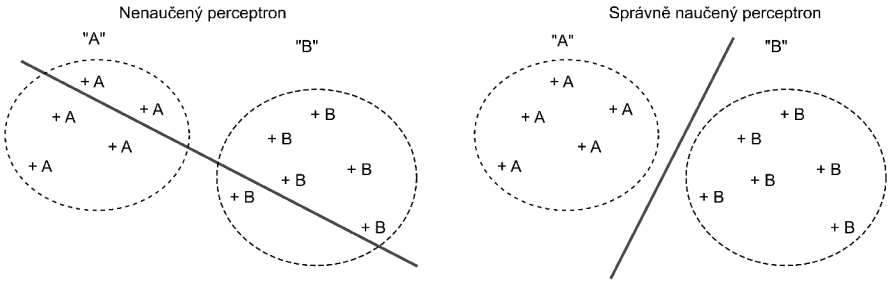
Kromě skokových aktivačních funkcí je v praxi časté použití funkcí spojitých. V praktické části této práce jsme používali převážně aktivační funkce softmax a rectified linear unit. Jejich popis a průběh viz kapitola 2.3.

Důležitou vlastností neuronů je jejich schopnost samostatně nalézt rozdělení pro lineárně separabilní množiny v konečném počtu kroků. Celý proces se skládá ze dvou základních částí - aktivní a adaptivní fáze. V rámci aktivní fáze neuron vypočítá svůj výstup pomocí hodnot na vstupu, jejich vah a svého prahu. V případě, že je potřeba váhy upravit, nastává fáze adaptivní. Učící pravidlo pro perceptron (umělý neuron s učící funkcí) bylo představeno Frankem Roseblattem - viz kaptiola 1.1. V rámci adaptivní fáze jsou perceptronu postupně předávány na vstup jednotlivé vzory a vyhodnocuje se, který byl určen špatně. Nastavení jednotlivých vstupních vah w se určuje následujícím vzorcem:

$$w_i(t+1) = w_i(t) + \eta(d-y)x_i \tag{1.3}$$

Výpočet hodnoty váhy w_i v rámci adaptivní fáze, kde t je číslo iterace, η je parametr rychlosti učení, d je očekávaný výstup a y je výstup perceptronu [1]

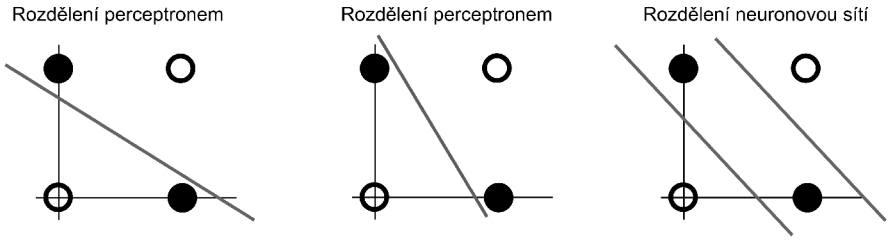
Princip je založen na faktu, že pokud je předložený vstup klasifikován správně, to jest že se výstup perceptronu y rovná očekávanému výstupu d, poté se příslušná váha nemění. Abychom si celou situaci mohli jednoduše graficky představit, následující obrázek ukazuje perceptron nenaučený s náhodně inicializovanými váhami a poté správně naučený perceptron. [1]



Obrázek 1.3: Grafické znázornění nenaučeného a správně naučeného perceptronu

1.2.3. Neuronové sítě a jejich učení

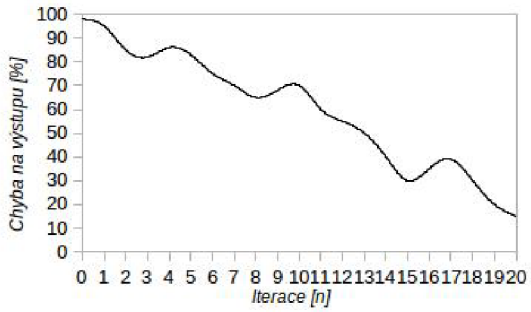
Fakt, že samotný perceptron dokáže realizovat pouze rozdělení lineárně separabilní množiny vedl k úpadku tohoto vědního oboru v 70. letech. Argumentace oponentů často stavěla na faktu neschopnosti realizace funkce XOR - exkluzivní disjunkce. Funkci XOR je možné vyřešit pomocí neuronové sítě o pouhých třech neuronech, ovšem do roku 1986 nebyla pro tyto sítě známá učící funkce. Příklad demonstrujeme graficky na následujícím obrázku. Vyplněné body znázorňují hodnotu 1, body prázdné hodnotu 0.



Obrázek 1.4: Neschopnost perceptronu aproximovat funkci XOR

Abychom plně využili schopností perceptronů, začaly se jednotlivé umělé neurony spojovat do sítí, což si můžeme graficky představit přidáváním dalších dělících přímek v rovině a tím zpřesnění možnosti separovat složitější třídy dat. V neuronových sítích se můžeme kromě skokových přenosových funkcí často setkat se spojitými sigmoidálními funkcemi. V tom případě již jednotlivé perceptrony nedělí rovinu přímkami, ovšem se spojitou křivkou, což umožňuje zvýšení přesnosti výpočtu.
Obecně lze říci, že neuronové sítě jsou skupiny perceptronů zapojených způsobem, že výstup jednoho perceptronu slouží jako vstup dalšímu, popřípadě více perceptronů. Perceptrony v síti můžeme rozdělit na vstupní, pracovní a výstupní. Vstupní neurony mají na vstupy připojené správně zformátovaná vstupní data. Jejich výstupy jsou připojeny na vstupní synapse pracovních perceptronů. Takto zapojené vstupní perceptrony tvoří vstupní vrstvu neuronové sítě. Pracovní perceptrony jsou uspořádány dle funkce sítě do jedné či více skrytých vrstev. Výstupní perceptrony nám zajišťují reprezentaci výstupních dat. Jejich uspořádání se volí dle požadovaného výstupu sítě. V praktické části jsme klasifikovali data do dvou tříd a výstupní vrstva byla tvořena dvěma perceptrony. Výsledek byl interpretován podle toho, který perceptron byl po předložení adekvátního vstupu aktivní.
Množství a způsob propojení jednotlivých perceptronů poté tvoří architekturu neuronové sítě. Základní dvě architektury neuronových sítí, se kterými se setkáváme, jsou sítě dopředné a cyklické. Dopředná architektura je zapojena tak, aby vzruchy sítí postupovaly stále dopředu - výstupy jednotlivých vrstev jsou zapojeny jako vstupy vrstvy další, vzruchy se žádným kanálem nevracejí zpět. V případě cyklického zapojení nejsou neurony uspořádány do vrstev. Příkladem cyklického zapojení může být například rekurentní topologie, kdy výstup neuronu je zpetně přiveden na jeho vstup a vzniká zpětná vazba. Těchto zapojení se často používá při různých typech simulací paměti.
Průběh výpočtu neuronových sítí lze rozdělit do tří základích fází - organizační, aktivní a adaptivní. V organizační fázi neuronová síť vytváří svoji topologii na základě nastaveného algoritmu. Aktivní fáze je samotný výpočet požadovaných dat. V adaptivní fázi poté síť zpětně spočítá svoji chybu při použití jednotlivých vstupů a tím nastavuje váhy na jednotlivých synapsích, neboli síť se v této fázi učí. Správným nastavením jednotlivých fází poté určujeme výslednou funkci neuronové sítě. V praktické části jsme používali struktury s pevnou architekturou, po vytvoření modelu již síťe pracovaly pouze ve fázích aktivních a adaptivních.
Rozvoj složitých neuronových sítí byl umožněn objevením agoritmu backpropagation. Jedná se o iterativní algoritmus, který je určen k nalezení optimálního na-

stavení vstupních vah perceptronů pro vícevrstvé sítě. Celý popis algoritmu je nad rámec této práce, bližší informace najdete [1, str. 53] . Pro učení hlubokých neuronových sítí se nejčastěji používá backpropagation ve spojení s algoritmem Stochastic gradient descent [1, str. 64] . V adaptivní fázi síť porovná výstup vygenerovaný a výstup očekávaný a ze zjištěné chyby se zpětně počítají váhy jednotlivých neuronů. Cílem algoritmu je směřovat k nejmenší možné chybě. V praktické části této práce jsme použili nástroje s vyšším stupněm abstrakce, proto je popis učících algoritmů spíše teoretický, zaměřený na interpretaci výsledků.[1]



Obrázek 1.5: Příklad učení sítě pomocí algoritmu backpropagation

Zde je důležité popsat dvě základní situace, kterým se při učení neuronových sítí snažíme vyhnout - přeučení a skončení učícího algoritmu v lokálním minimu.
Jak je vidět na obrázku 1.5 síť v průběhu učení prochází často lokálním minimem, ovšem my hledáme globální minimum chyby. V případě chybného nastavení parametru λ, koeficientu rychlosti učení, se nám může stát, že síť není schopna v průběhu adaptivní fáze překročit hranice lokálního minima. Dalším způsobem řešení tohoto stavu je funkce dropout (viz kapitola 1.3.6) nebo jiný algoritmus pracující na podobném principu. Síť v každé iteraci ignoruje určitou část vstupů a je šance, že díky jiné kombinaci vstupních dat překročí hranice lokálního minima.
Přeučení (anglicky overfitting) neuronové sítě je nežádoucí stav nastavení vah neuronové sítě. Častým úkolem neuronových sítí je generalizace problému z omezené trénovací množiny, při přeučení ovšem síť až moc přesně aproximuje trénovací data a pro ostatní vstupy je nepoužitelná. Přeučení se dá vyhnout podobně jako zastavení na lokálnímu minimu. Kromě toho je důležité dodání správného množství učících dat. V tomto případě platí čím více, tím lépe. [1]

1.2.4. Hluboké učení

Hluboké učení, častěji používaný anglický termín deep learning či machine learning, je odnož strojového učení zaměřeného na algoritmy, které dokáží hledat a reprezentovat informace v datech strukturovaných i nestrukturovaných. Vstupní data mohou mít různou podobu - textová data, obrazová data, hodnoty akcií v čase atd. Dnešní rozmach hlubokého učení je způsoben jednak obrovským výpočetním výkonem dnešních počítačů a také pokrokem ve studiu aplikovaných algoritmů. V této práci se zaměřujeme na hluboké učení ve smyslu aplikace hlubokých neuronových sítí (dále

již DNN z anglického Deep neural network) na textová data metodou učení s učitelem. [2]
Termínem DNN můžeme označit jakoukoli neuronovou síť o jedné a více skryté vrstvě. Ovšem drtivá většina DNN se skládá spíše ze stovek vrstev, přičemž už nejsme limitování klasickou perceptronovu sítí. Vzorem při tvorbě DNN je mozek a jeho části, ovšem na vyšší úrovni, než tomu bylo u klasických neuronových sítí. Například konvoluční neuronové sítě (dále CDNN z anglického Convolutional deep neural network) vycházejí z výzkumu zabývajícím se funkcí očního nervu u primátů. Velice zajímavé jsou aplikace DNN na problémy porozumnění obsahu dat a jejich klasifikace. V dnešní době jsou počítače schopny v reálném čase analyzovat a popsat například objekty a scény na videu, odezírat ze rtů s lepšími výsledky než lidé nebo kolorovat černobílé filmy. [2] Velký virální úspěch měla minulý rok například práce [3] na níž autoři navázali populární aplikací Prisma na aplikování výtvarných uměleckých stylů na fotografie.
Oblast hlubokého učení v dnešní době probíhá bouřlivým vývojem a je velice pravděpodobné, že přinese ještě velmi mnoho zajímavých objevů. Cílem této práce je koneckonců výrok v minulé větě splnit.

1.3. Analýza emocí v textech

V rámci kapitoly je probrán současný stav automatického zpracování emocí v textech a způsob jeho využití. Součástí je také srovnání ostatních používaných metod s metodou hlubokých konvolučních neuronových sítí.

1.3.1. Význam emoční textové analýzy

Metody automaticky zpracovávající emoce v textech jsou vytvářeny již od 60. let minulého století. Jedná se o odnož vědeckého oboru Zpracování přirozeného jazyka, která se zaměřuje především na metody určování subjektivních pocitů autora textu. V dnešní době je mimojiné hojně využívaná jako marketingový nástroj pro získávání zpětné vazby od zákazníků pomocí dolování dat z internetových diskusí a recenzí. Jelikož společnosti jako Facebook či Google denně uloží obrovské množství dat, je jejich cílem data zpracovat do strukturované podoby pro širší užití. Množství generovaných dat časem exponenciálně narůstá a metody klasifikace těchto informací budou mít stále větší využití. [4]
V praxi se můžeme setkat s textovou analýzou například při experimentu, kdy sledujeme výskyt zpráv s hastagem #bitcoin a vliv jejich sentimentu na hodnotu virtuální měny. V práci [5] autor analyzuje všechny příspěvky na sociální síti twitter s hashtagem #bitcoin po jejich emocionální stránce a sleduje korelaci vývoje hodnoty této měny v závislosti na množství pozitivních a negativních příspěvků.[5]
V této práci klasifikujeme texty pouze do dvou tříd, na pozitivní a negativní, ovšem v praxi se často setkáváme s metodami určujícími širší spektrum emocí - strach, radost atd. [6]

1.3.2. Dnešní metody emoční textové analýzy

Jedna z metod používaných pro klasifikaci vlastností v textech je metoda Bag of words. Touto metodou se po vypuštění slov, které nemají žádný emoční význam (spojky, krátké předložky) počítá výskyt jednotlivých slov v textu a tato informace je pak předána určité klasifikační funkci. Metoda vyžaduje kvalitní databázi, ve které je každému slovu přiřazen jeho emoční význam. Vytvořit takovou databázi je poměrně náročný úkol. Existují modifikace této metody, které sledují výskyt n-gramů. N-gramy si obecně můžeme představit jako spojení slov o délce n (dvojslovná spojení jsou označovány jako bigramy, trojslovná jako trigramy atd.). [4]
V dnešní době je jednou z nejpoužívanějších metod Support vector machine. Jedná se v podstatě o lineární klasifikátor vycházející z perceptronu. I tato metoda vyžaduje poměrně kvalitní oklasifikovanou vstupní databázi. Při splnění těchto podmínek však má velmi dobré výsledky. Článek [6] popisuje aplikace SVM na problém klasifikace emocí do pěti kategorií s průměrnou úspěšností 80 procent. Lidská úspěšnost je v tomto případě na hodnotě 86 procent, tudíž jde o velmi dobrý výsledek.

1.3.3. Metoda hlubokých konvolučních neuronových sítí

Jak bylo zmíněno v minulé podkapitole, v praxi se dnes používá několik různých přístupů k problému automatické textové analýzy sentimentu. Jejich společnou nevýhodou je nutnost vytváření rozsáhlých vstupních databází, ve kterých je většinou nutno ručně přiřazovat slova či slovní spojení jejich emočnímu výrazu. Kvůli tomuto přístupu jsou vždy závislé na jazyku vytvořených databází.
V závislosti na použité metodě také narážíme na problém, kdy se v textu začnou objevovat složitější slovní struktury, např. rozsáhlejší slovní spojení, nebo speciální literární či řečnické formy typu ironie, satira apod. V případě metody Bag of words tyto formy často analýza neodhalí.
Použití neuronových sítí nám přináší hned několik výhod. Dva hlavní důvody, proč je použití této metody výhodné, jsou především komplexní porozumnění textu a zadruhé jednodušší tvorba vstupních dat a následná nezávislost metody na použitém jazyku a jeho formě v podobě různé diakritiky či například v dnešní době populárních emotikon.
První výhoda, komplexní porozumnění textu, je dána způsobem zpracování a formátem vstupních dat. Na vstupu sítě máme v případě této práce krátké texty o maximální délce 512 znaků (více viz kapitola Databáze textů). Celý tento vstupní text je zpracován vstupní vrstvou, která dle velikosti konvolučního jádra hledá souvislosti mezi daným počtem vedlejších znaků. Tyto souvisloti dále postupují do hlubších vrstev neuronové sítě a jsou zpracovávány mezi sebou. Tímto přístupem dosáhneme větší míry porozumnění textu jako celku, ne pouze každého slova zvlášť, nebo slov sousedních, jako např. v případě použití metody založené na zkoumání n-gramů.
Další již zmíněnou výhodou je jednodušší tvorba vstupní databáze. Jelikož pracujeme s mnohem většími vstupními celky textu, odpadá nutnost klasifikovat emoce

u každého slova, popřípadě n-gramu zvlášť. Díky tomu je možné automatizovaně vytvářet databáze z různých jazyků mnohem snadněji a rychleji a tudíž navrhovanou metodu přenášet nezávisle na požadovaném jazyku. Dále je velkou výhodou sítí schopnost zpracovat nestandardní části psaného neformálního textu, jako jsou např. emotikony. Tyto jazykové grafické symboly složené z interpunkčních znaků v sobě nesou zásadní informaci o emocích daného textu a každý uživatel jejich použití a formu cítí nějak jinak.

2. Konvoluční neuronové sítě a technologie výpočtu

Vlastnosti umělých neuronových sítí jsou typicky určeny třemi parametry - jejich architekturou (způsob, jakým jsou propojeny jednotlivé perceptrony), metodou výpočtu zpětného šíření chyby a použitou aktivační funkcí. V rámci této kapitoly jsou probrány vlastnosti a stavební prvky použitých neuronových sítí. Kromě teoretických základů jsou nastíněny přístupy ke tvorbě jednotlivých architektur. Součástí je také přehled dnes běžně používaných vývojových prostředků pro práci s neuronovými sítěmi a rozbor těch, které byly použity v praktické části této práce.

2.1. Vlastnosti konvolučních neuronových sítí

Hlubkoké konvoluční neuronové sítě se skládají z několika druhů vrstev s rozdílnými vlastnostmi a účelem. Kromě jejich přehledu jsou v následující podkapitole probrány také jejich parametry a doporučené nastavení. Následuje přehled optimizačních funkcí a nastavení hyperparametrů, mezi které se řadí především rychlost učení, proměna její hodnoty v závislosti na výpočetní epoše, jejím výsledku, a koeficient funkce dropout. V následující části Dopředná plně propojená vrstva je také krátky přehled použitých aktivačních funkcí.

2.1.1. Výpočetní vrstvy

Vlastnosti klasických dopředných či rekurentních neuronových sítí se dají vylepšovat přidáváním dalších neuronů a vrstev do modelované architektury. Velmi hluboké sítě ovšem mají problém s učením. Kvůli postupnému mizení gradientu již u sítí s desítkami plně propojených vrstev je velmi obtížné sítě doučit do požadované podoby. S přidáváním prvků také velmi roste náročnost výpočtu.
Proto byly vytvořeny nové architektury vrstev, které se vyznačují lepšímy parametry při menší výpočetní náročnosti. V této kapitole probereme vrstvy použité v praktické části, které se obecně používají pro klasifikaci v rámci hlubokých neuronových sítí.

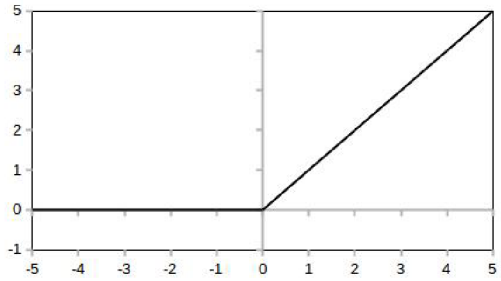
Plně propojená dopředná vrstva

Jedná se o klasickou dopřednou vrstvu (anglicky feed forward fully conected, dále zkráceně FC), která je plně propojena se svými vedlejšími vrstvami. Výstup neuronu je napojen na vstup každého neuronu v další vrstvě, obdobně jsou zapojeny vrstvy předešlé. Při použití FC vrstev v architektuře CDNN je jejich nejčastější pozice na místě dvou posledních skrytých vrstev a na vrstvě výstupní. V případě sítě určené pro klasifikaci se na výstupní vrstvě používá aktivační funkce softmax. Tato funkce

dokáže z k-dimensionálního vektoru obecně reálných hodnot vypočítat k-dimensionální vektor s hodnotami v rozmezí (0,1) o vlastnosti, že součet všech výsledků je jedna. Mezi další často používané aktivační funkce patří Rectified linear unit.

$$f(x) = max(0, x) \tag{2.1}$$

Rectified linear unit [7]

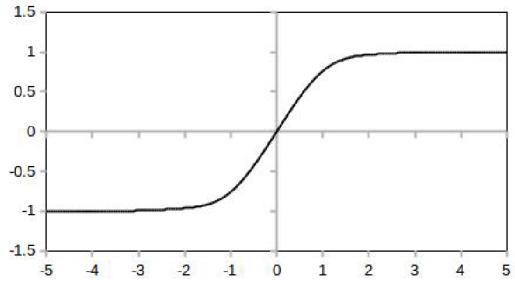


Obrázek 2.1: Rectified linear unit

Můžeme se také setkat s použitím funkce funkce TanH.

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.2}$$

TanH funkce [7]

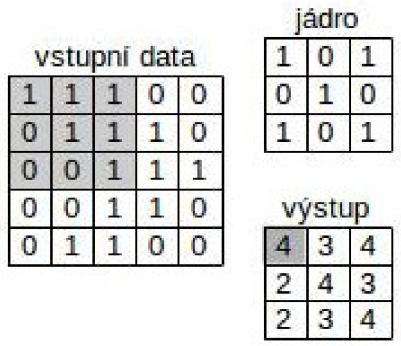


Obrázek 2.2: Aktivační funkce TanH

Výčet aktivačních funkcí tímto nekončí, ale v rámci práce jsme použili pouze tyto tři.
V případě použití FC jako výstupní vrsty je důležité správně zvolit její velikost dle požadované funkce. Bipolární klasifikátor se často realizuje pomocí jednoho výstupního neuronu, kdy se kontroluje jeho stav na výstupu, popřípadě pomocí neuronů dvou a kontroluje se, který je aktivní. [7]
V rámci práce jsme ověřili, že nastavení a velikost výstupních FC vrstev velkým dílem ovlivňují výslednou úspěšnost sítě. Více v podkapitole Výsledky-Porovnání dle rozměru plně propojených vrstev vý- stupní části.

Konvoluční vrstva

Jedná se o dopřednou neuronovou vrstvu, jejíž architektura je inspirována funkcí zrakového nervu. Jednotlivé neurony reagují na vstup aktivací okolních neuronů podle zadané velikosti konvolučního jádra a výstupem je počítání operace konvoluce posouváním jádra přes celou množinu dat. Příklad výstupu konvoluční funkce na dvoudimensionálních datech můžete vidět na následujícím obrázku. Operací konvoluce je v tomto případě násobení vstupních dat konvolučním jádrem.



Obrázek 2.3: 2D konvoluce

Při tvorbě konvolučních vrstev se setkáme s následujícími základními parametry, popřípadě metodami zpracování vstupních dat:

Hloubka Množství konvolučních vrstev poskládaných za sebou, výstup jedné vrstvy slouží jako vstup druhé. Neurony, které se zaměřují na stejnou oblast vstupních dat (jsou na stejné pozici) můžeme označit jako depth column (hloubkový sloupec)

Velikost jádra Velikost konvolučního jádra (na obrázku 1.7 vidíme jádro 3x3)

Stride Parametr stride určuje, zda konvoluční jádro projde všechny pozice (stride=0), nebo bude procházet data s určitým skokem n (stride=n). Zde je důležité si uvědomit, že při procházení konvolučního jádra daty se nám vstupy opakují (například na obrázku 1.7 se můžeme setkat se situací, že jedna vstupní pozice bude konvolučním jádrem počítána devětkrát). Proto je dobré parametr stride zohlednit podle velikosti konvolučního jádra.

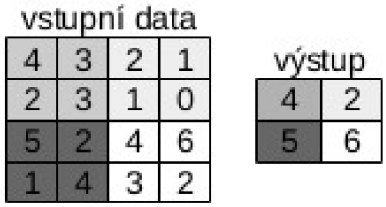
Zero padding Doplnění vstupních dat nulami na požadovanou velikost. Tuto vlastnost jsme použili především při navrhování sítí s inception modulem a multi-column architektur. Přes rozdílnou velikost jádra se po průchodu každou vrstvou výsledky doplnily na stejnou velikost a mohly po spojení dále pokračovat do hlubších vrstev.

Aktivační funkce Aktivační funkce mají u konvolučních vrstev stejnou funkci, jako u vrstev plně propojených. V práci jsme používali u konvolučních vrstev výhradně aktivační funkci Rectified linear unit.

Konvoluční sítě se velmi úspěšně používají na klasifikaci obrazových dat a hledání souvislostí v datech metodou učením s učitelem. Dokáží generalizovat data při mnohem menším počtu neuronů, než by na podobný problém potřebovala FC síť. [7]

Poolingová vrstva

Poolingové vrstvy se používají v konvolučních sítích pro zmenšení množství výstupů, výpočetní náročnosti a předcházení přeučení sítě. Jelikož při výpočtu konvoluční vrstvy se při posouvání konvolučního jádra jednotlivé vstupy několikrát opakují, vznikají nadbytečná data, které se pomocí poolingu odstraňují.
Nejčastější použití poolingu je ve formě maxpoolingových vrstev. Maxpooling data rozdělí na oblasti dle zadané velikosti svého jádra a z každé oblasti vybere největší hodnotu. Příklad pro maxpooling s jádrem o velikosti 2x2 a parametrem stride=1 vidíte na obrázku 1.8:



Obrázek 2.4: Výsledek operace maxpooling

Při tvorbě poolingových vrstev se setkáme s následujícími základními parametry:

Typ Jak jsme uvedli v předchozím odstavci, nejpoužívanějším typem poolingu je maxpooling. Používá se ovšem také average pooling - hledání průměrné hodnoty, min pooling hledání minimální hodnoty.

Velikost jádra Velikost oblasti pro výpočet poolingu

Stride Parametr stride, stejný jako u konvoluční vrstvy

Poolingové metody můžeme ještě rozdělit na overlapping (překrývající) a non-overlapping (nepřekrývající) pooling. V případě overlapping poolingu použijeme např. hodnoty jádra 3x2 a stride=1,0, poolingové oblasti se budou vzájemně překrývat. Pro non-overlapping pooling nastavíme hodnoty jádra a stride tak, aby se nám jednotlivé oblasti výpočtu nepřekrývaly, pouze sousedily (viz obrázek 1.8). Nejčastější je použití poolingových hodnot jádra 2 a 3 (popřípadě ve více dimenzích 2x2, 3x3). Při větších hodnotách již většinou výpočtem ztratíme důležitá data. [7]

Dropout

Dropout (česky vyřazení) je operace, která při každé iteraci zajišťuje ignorování dané procentuální části vstupů. Jedná se spíše o algoritmus, než vrstvu neuronové sítě,

ovšem v námi použité výpočetní knihovně Keras se dropout modeluje jako vrstva, pro přehlednost další kapitoly ho proto vyjádříme takto.
Použití dropout funkce se používá z více důvodů:

Předcházení přeučení Přeučení neuronové sítě je obecný problém, díky změně vstupních dat při každé iteraci vlastně používáme vždy jiné vstupní hodnoty a při správném nastavení hodnoty dropout předcházíme tomuto jevu

Lepší aplikace výsledné sítě na reálná data Jak bylo uvedeno u předchozí kategorie, použití různých vstupů u každé iterace nám přinese lepší výsledky sítě při aplikaci na reálná data. Teoreticky sice získáme horší výsledek na testovacích datech, ale v praxi bude síť fungovat lépe.

Často se dropout v rámci CDNN používá u koncových FC vrstev. Většinou se volí parametr funkce od 0.05 do 0.5. Při menší hodnotě už dropout přestává mít význam, při větší již začne vynechávat příliš velké množství dat. [7]

Flatten

Vrstva flatten (česky zploštění) upraví vícerozměrné data do jednorozměrného vektoru, který očekávají na vstupu FC vrstvy na konci sítě. Funkci si můžeme představit na následujícím jednoduchém příkladu:

$$Flatten([[1,2],[3,4]],[[5,6],[7,8]]) = [1,2,3,4,5,6,7,8] \tag{2.3}$$

Flatten [8]

2.1.2. Optimizační funkce

Optimizací v matematice rozumíme procesu výběru nejlepšího elementu z dané množiny elementů podle určitého kritéria. V rámci neuronových sítí rozumíme pod pojmem optimizace hledání správného nastavení vah pro vazby jednotlivých neuronů takovým způsobem, abychom v dalším kroku snížili výslednou chybu sítě a postupně hledali její ideálně globální minimum.
Optimizační funkce dnes v hlubokých konvolučních neuronových sítích vylepšují klasickou formu backpropagation. V práci jsme testovali hned několik dnes používaných optimizačních funkcí, především Stochastic gradient descent a funkci Adam. Praktické výsledky naleznete v podkapitole Dosažené výsledky-Porovnání testovaných sítí dle parametrů-Optimizační funkce.
Jelikož se v dnešní době nasazují umělé neuronové sítě na mnoho různých výpočetních problémů, postupně bylo vytvořeno několik optimizačních funkcí vhodných pro různé použití. Při samotném modelování se často optimizační funkce a jejich parametry používají v rámci výpočetních knihoven jako uzavřená funkce a uživatel nemusí kromě jejich správného nastavení, což se často testuje metodou pokus-omyl, znát jejich bližší vlastnosti. V následujících kapitolách budou proto popsány některé nejpoužívanější optimizační funkce z teoretického hlediska. [9] [10]

Gradient descent

Gradient descent je jednou z nejpoužívanějších optimizačních funkcí. Existují jeho tři varianty, ty se liší tím, jakou část databáze jako celku použijeme pro výpočet výsledného gradientu. Mezi verzemi vybíráme podle toho, zda upřednostníme rychlost výpočtu, nebo jeho přesnost.

Batch gradient descent Batch gradient, také nazývaný Vanilla gradient, používá pro výpočet směru gradientu najednou celou databázi. Jelikož musíme pro výpočet jedné epochy propočítat veliké množství dat najednou, jedná se o velmi pomalý algoritmus, který může mít často problém s hardwarovými parametry výpočetního serveru, zejména s velikostí paměti ram. Na druhou stranu je u něj garantována konvergence do globálního minima v případě konvexního prostoru.

Stochastic gradient descent Stochastic gradient descent, dále sgd, se od předchozí verze liší tím, že pro každý vzorek vstupní databáze provádí výpočet gradientu zvlášť. Tím dochází ke snížení nároků na velikost paměti výpočetního stroje. Většinou je rychlejší, než verze Batch gradient descent, ale například u velkých databází počítá zbytečně znovu gradient pro velmi podobné vstupní záznamy a provádí mnoho nadbytečných výpočetních operací.
Typický průběh konvergence neuronové sítě při výpočtu gradientu pomocí optimizéru sgd je velmi fluktující, což na jednu stranu zpomaluje celý průběh konvergence, ale výhoda spočívá v tom, že algoritmus mnohem snáze nalezne globální minimum chyby. Celý průběh lze ovlinit správným nastavením parametru rychlost učení viz 2.1.3 a použitím momentu 2.1.3. Při pomalém zmenšování rychlosti učení však sgd dává velmi dobré výsledky podobně jako batch gradient descent a to při menším výpočetním čase a menších paměťových nárocích na výpočetní server.

Mini-batch gradient descent Tato verze v sobě spojuje výhody obou předchozích, gradient se počítá pro skupiny vstupních dat většinou ve velikosti od 50 do 256 vzorků. Tím dosahujeme snížení fluktuace konvergence průběhu učení. V praxi se ve vývojových prostředcích pro modelování neuronových sítí často setkáváme s optimizérem sgd, který ovšem je implementován jako mini-batch gradient descent.

[11]

Adagrad

Optimizér adagrad je navržen tak, aby adaptoval hodnotu parametru rychlost učení na množství výskytu daných parametrů. Pro parametry s menším výskytem jsou prováděné updaty větší, častější parametry upravuje méně. Proto se velmi dobře používá na data, která nemají velkou hustotu, vzorky v takových databázích jsou různé a moc se nepřekrývají. Výhodou adagrad je, že odpadá nutnost sledování a nastavování rychlosti učení v čase. Většinou se používá hodnota rychlosti učení 0,01.

Nevýhodou algoritmu je, že se často stává, že rychlost učení se postupně zmenší na velmi malou hodnotu. Při každé epoše si sít uchovává předchozí hodnoty velikosti úpravy rychlosti učení, které se nachází jako součet jejich druhých mocnin v děliteli vzorce pro výpočet parametru rychlosti učení. Tímto dochází ke snižování parametru na hodnotu blízkou nule. V takovém případě se již síť nic nenaučí a skončí v lokálním minimu. Následující optimizéry, vycházející z Adagrad, mají většinou za cíl se vypořádat s tímto problémem.[12]
Tato optimizační funkce byla s úspěchem použita v mnoha řešeních, například Google pomocí ní vytvořil úspěšnou síť na rozpoznávání přítomnosti koček v obsahu youtube videí. Rád bych u tohoto příkladu upozornil na jasnou spojitost mezi umělými neuronovými sítěmi a lidským mozkem, pro mnoho uživatelů youtube je hledání videí koček smyslem existence.

Adadelta

Optimizér adadelta vychází z adagrad, ale omezuje součty předchozích gradientů na určitou maximální hodnotu. Tím zajišťuje, aby hodnota parametru rychlost učení neklesala k nule a průběh učení se nezastavil předčasně. [13]

Adam

Adam, zkratka clého názvu adaptive moment estimation, je další velmi oblíbený a používaný optimizér. Vychází z adadelta a dalšího optimizéru RMSprop, který se často používá pro optimizaci rekurentních sítí. Kromě ukládání předchozích hodnot gradientu také používá moment vycházející z průměru předchozích gradientů.
V práci jsme ho používali s dobrou úspěšností, ovšem vyžaduje správné nastavení parametrů, jinak se vyskytly problémy s konvergencí sítě. [14]

Adamax

Adamax optimizér vychází z optimizéru adam. Určité parametry výpočtu gradientu nahrazuje limitou nekonečna a zjednodušuje a zrychluje konvergenci.
[14] [8]

Nadam

Jedná se o optimizér adam s použitím Nesterova momentu. [15] [8]

2.1.3. Hyperparametry

Kromě základních určujících prvků hlubokých konvolučních neuronových sítí, jako je například jejich architektura, musíme pracovat také s nastavováním jejich hyperparametrů. Mezi ně řadíme především nastavení optimizační funkce, rychlost učení, moment optimizační funkce a nastavení funkce dropout. Nastavení dropout v modelech knihovny Keras, použité v praktické části práce, je určeno použitím samostatné vrstvy o jednom koeficientu. V následující podkapitole proto následuje teoretický

popis ostatních hyperparametrů a několik přístupů k hledání jejich správných hodnot.

Rychlost učení

Rychlost učení je určení velikosti kroku, o který se mění nastavení hodnoty jednotlivých vah při jejich úpravě v rámci backpropagation algoritmu. Velikost tohoto hyperparametru významně určuje rychlost učení, fluktuaci konvergence průběhu učícího algoritmu a pravděpodobnost nalezení globálního minima chyby. V případě, že nastavíme rychlost učení malou, síť nám s velkou pravděpodobností uvázne v lokálním minimu. Při nastavení moc velké hodnoty zase riskujeme nekonvergenci sítě a síť bude chaoticky hledat minimum ve směru gradientu.
Abychom dosáhli konvergentní sítě a přitom nalezli globální minimum, v drtivé většině případů se používá proměnná, zmenšující se hodnota rychlosti učení v závisloti na několika parametrech. [7]

Krokové snižování Základní metoda, která jednoduše snižuje rychlost učení při každé epoše o předem danou konstatní hodnotu. Správné nastavení rychlosti učení na záčátku algoritmu a velikosti kroku je velmi závislé na daném problému a použité optimizační funkci.

Kontrola velikosti chyby Při použití této metody sledujeme vývoj výsledků validace na testovací databázi. V případě, že se po daném počtu epoch výsledky nezlepší, začneme postupně snižovat hodnotu learning rate a snažíme se docílit přesnějšího nalezení minima chybové funkce jemnějším nastavováním váhových koeficientů.

Moment optimizační funkce

Moment optimizační funkce zajišťuje korekci konvergence funkce k minimu. Cílem je eliminovat oscilaci okolo minima vzhledem k předchozím gradientům. Moment optimizační funkce zvyšuje gradient v případě, že předchozí gradient měl stejný směr. V opačném případě gradient redukuje.
Velmi používaný je moment Nesterovův. Ten po nalezení gradientu vypočítá směrový korekční vektor a výsledný gradient je součtem původního a korekčního vektoru. V dnešní době je použití tohoto momentu rozšířené díky jeho dobrým výsledkům. Sítě v praktické části práce jsou převážně trénovány pomocí Nesterovova momentu. [7]

Hyperparametrická optimalizace

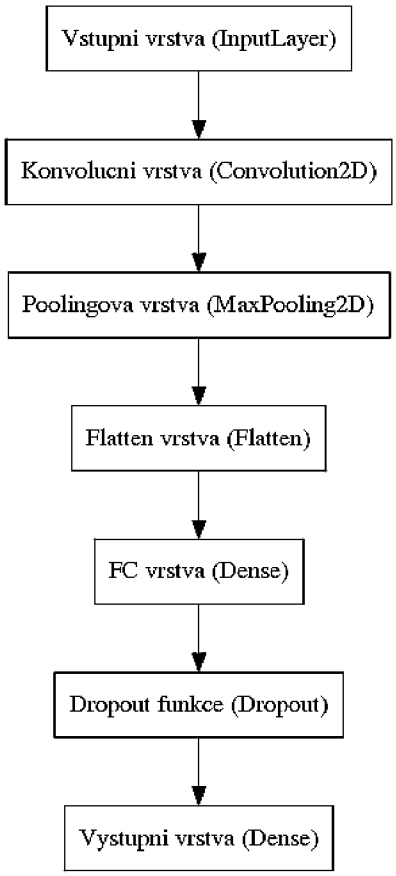
Nalezení správných hodnot hyperparametrů pro každou síť je náročným úkolem. Přestože existuje doporučené základní nastavení hyperparametrů, je nutné každou síť vyzkoušet s různými hodnotami. Jelikož se jedná o časově náročný proces, obsahují některé vývojové prostředky možnosti, jak daný cyklus automatizovat.

Hledání správných hyperparametrů, neboli jejich optimalizace pro danou síť je proces, kdy zkoušíme různé kombinace jejich nastavení na pár prvních epochách a testujeme, které hodnoty hyperparametrů přinesou nejlepší výsledky. Ty s lepšími výsledky poté použijeme pro další testování. V rámci práce již bohužel nezbyl čas na hyperparametrickou optimalizaci výsledné sítě. Jednalo by se o další logický krok pro zlepšení jejich výsledků. [7]

2.2. Architektury použitých neuronových sítí

Architektura neuronové sítě je základním prvkem, který ovlivňuje její funkci a výsledné vlastnosti. Od počátků neuronových sítí a využití samotného perceptronu bylo vytvořeno již mnoho architektur s cílem je aplikovat na široké spektrum problémů s různým procentem úspěšnosti. V této kapitole jsou probrány čtyři použité přístupy ke tvorbě architektur konvolučních neuronových sítí použitých v praktické části této práce.

2.2.1. Dopředná konvoluční neuronová síť



Obrázek 2.5: Základní model dopředné konvoluční neuronové sítě

Dopředná konvoluční neuronová síť je základním příkladem používané architektury. Síť neobsahuje větvící prvky a data na vstupu vrstvy jsou vždy výstupní data vrstvy předchozí. Poprvé danou architekturu popsal Yann LeCun v článku [16] v roce 1998 a od té doby se její varianty úspěšně používají na řešení různých problémů jako počítačové vidění a porozumnění obsahu obrazu, zpracování textů a mnohem více. LeCun se na vývoji konvolučních neuronových sítí dále podílí a je jednou z vedoucích osob v oboru.
Na obrázku 2.5 vidíte zjednodušenou dopřednou konvoluční architekturu. Celou síť si pro potřeby této práce můžeme rozdělit do třech částí.

Vstupní část Vstupní část zajišťuje konverzi a úpravu dat do formátu, který dokáží zpracovat následující konvoluční vrstvy. V knihovně Keras jsme používali vstupní vrstvu, která načetla data a poté vrstvu reshape, která data uvedla do požadovaného formátu. Pro zpracování textu se také na vstupu často používá word embedding. Při použití embeddingu se slova či slabiky vstupního textu zpracují do formy vektorů a teprve potom postupují dále do výpočetní části.

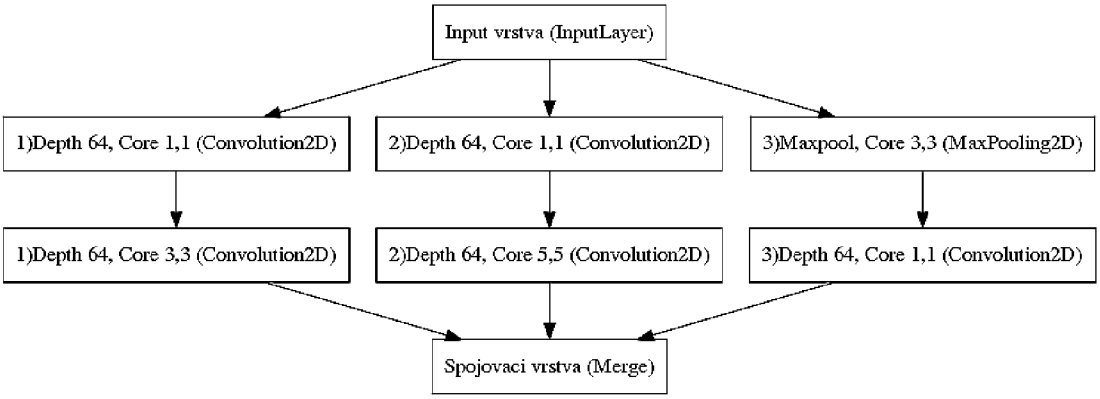
Výpočetní část Výpočetní část se většinou skládá z vrstev konvolučních a poolingových. Konvoluční vrstva zpracuje vstupní data na principu popsaném v 2.1.1 a v praxi často následuje vrstva poolingová. Základními prvky jsou hloubka a velikost jádra. Kombinace konvoluce a poolingu se v této části architektury často několikrát opakuje.
Vrstva poolingová zajišťuje odstranění přebytečných dat z výstupu konvolučních sítí a snižuje náročnost výpočtu.

Výstupní část Výstupní část se v tomto modelu skládá z vrstev flatten, plně propojené skryté vrstvy, dropout funkce a výstupní plně propojené vrstvy o dvou neuronech s aktivační funkcí softmax. Na začátku výstupní vrstvy je v daném modelu vrstva flatten, která zpracuje výstupní data z výpočetní vrstvy do formátu, který vyžadují vrstvy plně propojené. Data dále postupují do plně propojených vrstev, které se v praxi opakují s funkcí dropout. Po požadovaném počtu plně propojených vrstev se data dostávají na vstup výstupní vrstvy, která dle aktivovaného neuronu určí výsledek.

[17]

2.2.2. Inception modul a jeho použití

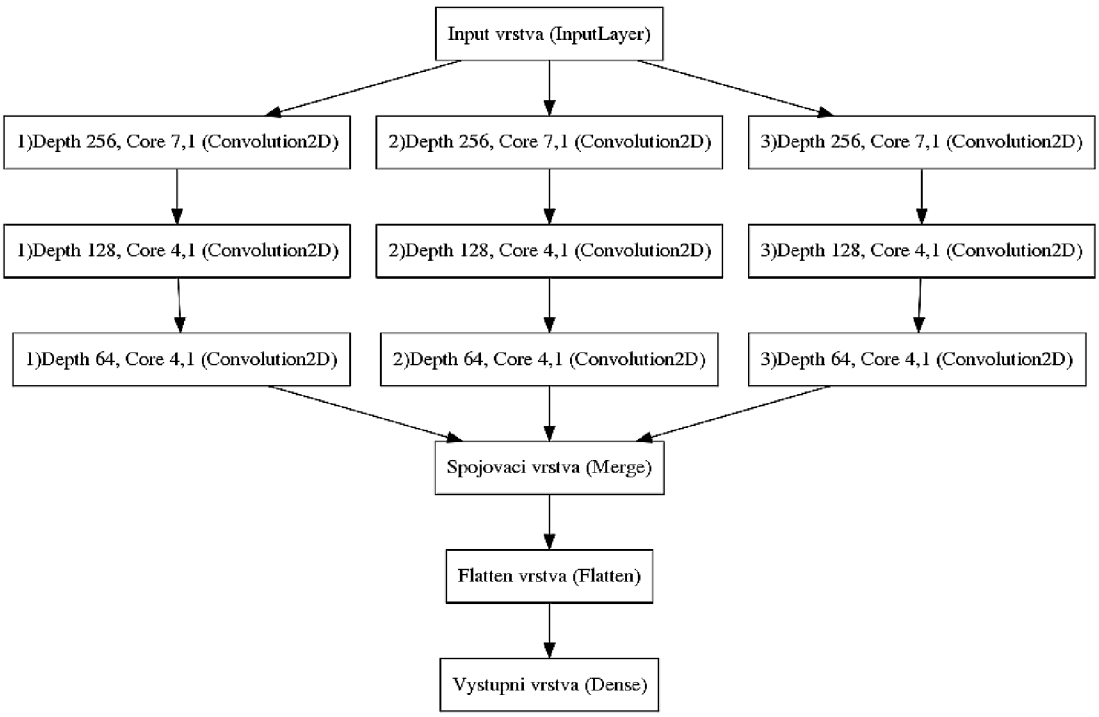
Inception modul, který svůj název získal dle slavného filmu Inception - odkazuje na průběh vnořování se do hlubších vrstev, byl vytvořen v rámci soutěže Imagenet v laboratořích Google. Síť Googlenet, která obsahuje použití inception modulů, v roce 2014 soutěž Imagenet vyhrála. [18]
Inception modul je založen na paralelním zpracování dat ve třech větvích s rozdílnými velikostmi jader konvolučních vrstev a kombinaci s maxpoolingem. Jeho model je znázorněn na obrázku 2.6.



Obrázek 2.6: Inception modul

Zajímavostí je použití konvoluce s jádrem o velikost [1,1]. V případě jednodimensionální konvoluce by jádro o této velikosti postrádalo smysl. V případě dvoudimensionální konvoluce ovšem tato vrstva propočítá konvoluci v tomto případě 64 prvků kvůli hloubce jádra a její využití přináší zajímavé výsledky.
Inception modul není navržen k použití jako samostatná síť, ale používá se v různých úpravách jako vložená součást výpočetní vrstvy s cílem jeho paralelizací snížit nároky výpočtu a zlepšit výsledky sítě. [18]

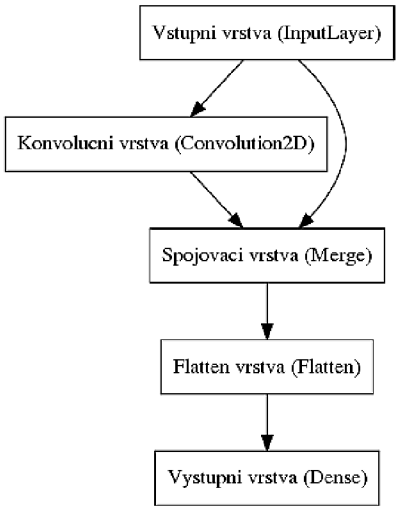
2.2.3. Multi-column architektura



Obrázek 2.7: Zjednodušený model multi-column architektury

Tento typ architektury vychází z článku [19] kde byl s úspěchem použit na rozpoznávání dopravních značek v obrazu. Jedná se o architekturu s paralelní výpočetní částí, která se skládá z několika podobných či stejných sloupců výpočetních vrstev. Architektura měla ve výchozím článku dobrou úspěšnost, proto jsme ji aplikovali v rácmi této práce. Pro lepší výsledky by bylo pravděpodobně potřeba její širší testování.
[19]

2.2.4. Residuální sítě



Obrázek 2.8: Zjednodušený model residuální sítě

Myšlenka residuálních sítí byla představena v roce 2015 článkem [20]. Síť založená na tomto principu byla vítěznou sítí v soutěži Imagenet roku 2015.
Princip residuálních sítí je založen na předávání nezpracovaných dat paralelně do vyšších vrstev. Její základní model můžete vidět na obrázku 2.8. V závislosti na velikosti sítě můžeme použít více residuí a kombinovat předávání různě zpracovaných dat do hlubších vrstev sítě přímo. Výhodou těchto sítí je na prvním místě jejich úspěšnost a menší výpočetní náročnost, protože pro dosažení stejného výsledku potřebují méně výpočetních vrstev.
V praktické části jsme myšlenku použití residuí úspěšně aplikovali na několik sítí. Residuální prinicp obsahuje i nejlepší síť, která je vybrána jako výstup této práce.
[20]

2.3. Vývojové prostředky

V rámci této kapitoly jsou probrány dnes používané vývojové prostředky pro modelování a práci s neuronovými sítěmi. Jelikož se jedná o dynamicky rozvíjející se obor, můžeme si vybrat mezi několika knihovnami a výpočetními frameworky. Pro účely

této práce byla jako výpočetní základ vybrána knihovna Theano pro její dobrou integraci s knihovnou Numpy, jednoduchou implementaci výpočtů na grafickou kartu a rozsáhlou uživatelskou komunitu. Pro modelování sítí byla použita knihovna Keras, která má rozsáhlou uživatelskou podporu a orientaci na jednoduchou tvorbu mnoho různých modelů neuronových sítí.

2.3.1. Přehled dnes používaných vývojových prostředků

Podkapitola obsahuje přehled v současnosti používaných řešení pro modelování a práci s neuronovými sítěmi. Vynechány jsou akorát frameworky Theano a Keras, které jsou blíže vysvětleny v následující podkapitole.

Lasagne Jedná se o knihovnu, která pracuje nad výpočetním frameworkem Theano. Jelikož Theano je poměrně nízkoúrovňový framework, nabízí knihovna Lasagne vyšší míru abstrakce a zjednodušuje práci s neuronovými sítěmi. Nabízí práci s vrstvami podobně jako Keras. Je okolo něj ovšem menší komunita vývojářů a proto je méně zdokumentované.

Blocks Podobně jako Lasagne nabízí knihovna Blocks vyšší level abstrakce nad frameworkem Theano. Jedná se o velmi dobře zdokumentovaný projekt, který je vyvíjen pod University of Montreal. Tato knihovna je vhodná pro práci s rekurentními neuronovými sítěmi, které jsou v ní dobře zdokumentovány a podporovány.

Caffee Jedná se pravděpodobně o nejrozšířenější výpočetní framework pro tvorbu neuronových sítí, který je vyvíjen pod společností Berkley vision and learning center. Je psané v C++ a schopno pracovat s kódem v jazycích Python a Matlab. Jedná se o velmi rychlý framework, ale testování a optimalizace parametrů sítí v něm není nejjednodušší. Je okolo něj vytvořena velká komunita lidí a je používán v průmyslu i výzkumu.

Tensorflow Tensorflow je opensource výpočetní knihovna vyvíjená pod záštitou Googlu. Jedná se o kombinaci nízkoúrovňového výpočetního frameworku typu Theano s možností použití vyšší míry abstrakce jako např. v knihovně Keras. Přestože není na použití tak komfortní jako jiné, obrovská uživatelská základna a podpora od společnosti Goole z ní dělají jeden z největších a nejlépe zdokumentovaných projektů pro práci s neuronovými sítěmi.

MXnet MXnet je knihovna podporovaná společností Amazon, která v poslední době zašťiťuje jedny z nejlepších cloudových služeb pro poskytování výpočetního výkonu pro machine learning operace profesionálům i amatérům. Nabízí možnost pracovat v nízké i vyšší úrovni abstrakce a je přenositelná mezi platformami. Podporuje Python. Má ovšem poměrně malou uživatelskou základnu.

PyTorch Nová knihovna pro práci s neuronovými sítěmi od společnosti Facebook. Vyšší i nižší míra abstrakce, ovšem velmi omezená dokumentace. Oproti ostatním možnostem je zatím málo ozkoušená s malou uživatelskou základnou.

Nolearn Novější knihovna, kompatibilní s numpy, Theanem i frameworkem Lasagne. Podporuje Python

NeuPy NeuPy je Pythonová knihovna pro práci s neuronovými sítěmi. Nabízí vysokou míru abstrakce a podporuje mnoho modelů sítí.

PyLearn2 Jedná se o pythonovou knihovnu, která běží nad frameworkem Theano se zaměřením na machine learning.

[21]

2.3.2. Použité vývojové prostředky

Na provádění výpočtů se používala knihovna numpy ve verzi 1.11.2. Pro izolaci výpočetního frameworku na serveru, ke kterému má přístup více uživatelů, byl použit program virtualenvwrapper 4.7.2 pro vytvoření virtuálního linuxového prostředí. Další výhodou izolovaného prostředí je fakt, že výpočetní knihovny se nebudou automaticky aktualizovat. Jelikož se pro výpočty používá několik knihoven zároveň, zaručíme tímto jejich stálou kompatibilitu. Skripty byly psány v jazyce Python ve verzi 2.7.6.
Simulování hlubokých neuronových sítí je velmi výpočetně náročná operace. Všechny praktické výsledky v této práci byly spočteny na výpočetním serveru s parametry gpu - Geforce GTX 980, RAM - 62 gB, os - linux Ubuntu 14.04.1.

Theano

Theano je výpočetní knihovna psaná v jazyce Python. Syntaxe výpočetních operací se velmi podobá té, kterou využívá NumPy. Je optimalizovaná pro výpočet přes CPU i GPU. Jedná se o open source projekt vyvíjený pod Université de Montréal. Theano samotné podporuje modelování i výpočty neuronových sítí, ale jedná se velmi nízkoúrovňový framework. Proto je často používán jako základ dalších knihoven, které nabízejí vyšší míru abstrakce a jednodušší modelování potřebných modelů neuronových sítí. Theano samotné je ovšem velmi flexibilní pro různé typy výpočtů a rozsáhlá komunita pomáhá udržovat Theano stabilním vývojovým prostředím. V práci jsme pracovali s verzí theana 0.9.0. [22]

Keras

Keras je knihovna pro modelování neuronových sítí s vysokou úrovní abstrakce. Je vyvíjena s ohledem na jednoduché používání a vysoký výpočetní výkon. Má kvalitně zpracovanou dokumentaci a poměrně rozsáhlou uživatelskou komunitu. Podporuje práci na frameworku Tensorflow nebo Theano. Dokáže provádět výpočty přes CPU i GPU. Pro použití byl vybrán kvůli možnosti práce nad frameworkem Theano a uživatelské přívětivosti. Jelikož cílem práce nebylo navrhovat nejrychlejší sítě, ale testovat různé přístupy, Keras se jevil jako velmi vhodný, jelikož podporuje širokou škálu různých architektur neuronových sítí. V práci jsme používali Keras ve verzi 1.1.2. [8]

CUDA

CUDA je výpočetní platforma vyvinutá společností Nvidia pro umožnění provádění výpočtů přes GPU. Možnost paralelizace výpočtů do velkého množství jader na grafické kartě výrazně urychluje simulování neuronových sítí oproti výpočtu přes CPU. V praktické části jsme používali verzi CUDA 7.0.27. [23]

2.4. Databáze textů

V práci jsme používali trénovací a testovací databázi textů krátkých amatérských filmových recenzí v češtině a angličtině. Rozsah databáze byl 5000 vzorků. Recenze byly získány automaticky, označeny na pozitivní a negativní dle procentuálního hodnocení. Formát vstupních dat pro soubor chunks vypadá následovně - příklad krátké recenze filmu Frajer Luke v češtině:

id,lang,positive,training,hash,text
1644334,cz2,1,0,0469ab7431a3e697f24ef623a93aaa450b021a51,"Výbornej film, možná neni na maximální hodnocení už dneska, ale já nevim, je to klasika a Paul Newman je asi nejsympatičtější herec všech dob. Skvělej film."

Struktura obsahuje nejdřív pořadové číslo recenze, jazyková databáze, positivní/negativní, tréninková/testovací, hash, samotný text. Z těchto dat skript chunks předpřipraví matici o rozměrech 512x86 kde 512 je maximální délka textu (v případě kratšího textu se doplní nulami na požadovanou délku) a 86 je sledované množství symbolů. Vzniklé matice mají na řádku daného znaku hodnotu jedna ve sloupci, který určuje daný znak. Ostatní hodnoty na řádku mají hodnotu nula.
Takto upravená data poté slouží jako vstup neuronové sítě.

3. Výsledky

V rámci kapitoly je probrán přehled všech testovaných architektur, poté následuje jejich srovnání v rámci různých parametrů. Sítě byly navrhovány z hlediska vyzkoušení různých architektur a jejich modifikací. Srovnání výsledků sítí probíhalo dle parametru velikosti jádra konvolučních sítí, hloubce konvolučních sítí, množství výstupních dense vrstev, použitých optimizačních funkcí, využití inception modulu a použití residuální architektury. Přesné parametry sítí, jejich průběhy učení a vizualizace všech architektur lze nalézt buď v příloze priloha.zip na vloženém cd. Vybrané vizualizace jsou přiloženy na konci práce v rámci příloh.
V podkapitole 3.3.1 je detailně popsána síť NN25, která je výsledkem této práce. Tato síť dosáhla nejlepšího výsledku 79,94 procent úspěšnosti na testovacích datech.

3.1. Přehled testovaných sítí

V rámci práce jsme otestovali celkem 43 architektur hlubokých konvolučních neuronových sítí s různou mírou úspěšnosti. Pokud není uvedeno jinak, byly sítě učeny pomocí optimizéru sgd s parametry - learning rate 0,001, decay 1e-8, momentum=0.9 a s použitím nesterovova momentu.

NN1 Úspěšnost 78,6 procent
Jedná se o základní architekturu vycházející z článku [17] na které jsme testovali množství parametrů a jejich vliv na úspěšnost sítě. Síť má dopřednou konvoluční arcchitekturu, používá jádra o velikost [7,86] a [4,1] v celkem třech konvolučních vrstvách proložených vrstvami maxpoolingovými s jádry [2,1]. Výstupní část zajišťují dvě plně propojené dopředné vrstvy o velikosti 512 neuronů s droupout funkcemi s koeficientem 0,125 a aktivačními funkcemi Relu. Na výstupu je vrstva o dvou neuronech s aktivační funkcí softmax. Visualizovanou architekturu sítě naleznete v příloze.

NN2 Úspěšnost 75,7 procent
Síť rozšiřuje architekturu NN1 o další konvoluční vrstvu, jsou použity konvoluční jádra [7,86] a [3,1] a větší dense vrstvy ve výstupní části, každá o počtu 1024 neuronů. Přestože síť by měla mít teoreticky větší výpočetní kapacitu, nedosáhla takových výsledků jako NN1. Visualizovanou architekturu sítě naleznete v příloze.

NN3 Úspěšnost 72,2 procent
Síť modelována na základě první části sítě Googlenet z článku [18] bez modulu inception. Na tuto síť navazuje NN4. Visualizovanou architekturu sítě naleznete v příloze.

NN4 Úspěšnost 74,1 procent
Síť vychází z NN3, ale je v ní použit inception modul. Výsledky těchto dvou sítí jsou porovnány v podkapitole 3.2.5. Visualizovanou architekturu sítě naleznete v příloze.

NN5 Úspěšnost 72,7 procent
Síť vychází architekturou z NN1, ovšem přidává mnohem větší hloubku vrstev - konvoluční vrstvy mají hloubky 1024 a 2048, plně propojené vrstvy na konci 2048 a 1024 neuronů. Přestože je síť mnohem robustnější, nedosáhla zdaleka tak dobrých výsledků jako NN1.

NN6 Úspěšnost 77,5 procent
Síť vychází z NN5 a testuje závislost výsledku na velikosti konvolučního jádra. Síť pracuje s jádry [30,86] a [30,1]. Srovnání výsledků s NN5 je uvedeno v podkapitole 3.2.1.

NN7 Úspěšnost 76,2 procent
Síť vychází hloubkou i architekturou z NN1 ovšem používá větší hodnoty jádra - [30,86][20,1][10,1]. Srovnání výsledků s NN1 je uvedeno v podkapitole 3.2.1.

NN8 Úspěšnost 79,0 procent
Síť vychází z NN5 a testuje závislost výsledku na velikosti konvolučního jádra. Síť pracuje s jádry [15,86] a [15,1]. Srovnání výsledků s NN5 je uvedeno v kapitole 3.2.1. Sít měla překvapivě dobrý výsledek a proto je použití jádra o velikosti 15 časté v následujících testovaných sítích. Visualizaci sítě naleznete v příloze

NN9 Úspěšnost 75,3 procent
Síť vychází z NN1 a byla navržena na test velmi velkého konvolučního jádra [75,86] a [70,1]. Síť měla průměrný výsledek. Srovnání výsledků s NN1 je uvedeno v podkapitole 3.2.1

NN10 Úspěšnost 75,3 procent
Síť tvoří kompromis mezi NN1 a NN5 v rámci hloubky, použité konvoluční vrstvy měly hloubku 1024 a používaly konvoluční jádra [75,86] a [70,1]. Plně propojené vrstvy ve výstupní části byly složeny z 1024 neuronů. Cílem bylo zjistit, zda větší hloubka nepřinese lepší výsledek. Síť měla pouze průměrný výsledek. Srovnání výsledků se sítí NN9 naleznete v podkapitole 3.2.2

NN11 Úspěšnost 64,9 procent
Síť vychází z NN1 ale její architekturu jsme upravili tak, abychom mohli zkoušet vliv vložení jednoho a poté dalšího inception modulu v sítí NN12. Výpočetní část sítě je zdvojena do série, hodnoty hloubky a konvolučních jader jsou stejné. Do výpočetní části byl vložen jeden inception modul. Na výstupu má síť plně propojené vrstvy o velikost 2048 a 1024 neuronů. Srovnání se sítí NN12 naleznete v podkapitole 3.2.5. Visualizovanou architekturu sítě naleznete v příloze.

NN12 Úspěšnost 69,2 procent
Síť navazuje na NN11 a přidává do její výpočetní části další inception modul. Visualizovanou architekturu sítě naleznete v příloze.

NN13 Úspěšnost 71,9 procent
Síť vychází z NN8 a přidává residuální prvek přenesení vstupních dat na konec výpočetní části. Síť neměla moc dobrý výsledek, srovnání s NN8 naleznete v 3.2.6. Visualizovanou architekturu sítě naleznete v příloze.

NN14 Nekonvergovala, bez výsledku
Síť vychází architekturou z NN8 a testovala nastavení optimizéru Adam. Při nastavené hodnotě hyperparametrů nekonvergovala.

NN15 Úspěšnost 78,4 procent
Síť vychází z NN8 a má poloviční hloubku konvolučních vrstev. Srovnání s NN8 naleznete v 3.2.2.

NN16 Úspěšnost 78,9 procent
Síť vychází z NN8 a má čtvrtinovou hloubku konvolučních vrstev. Srovnání s NN8 naleznete v 3.2.2.

NN17 Úspěšnost 77,1 procent
Síť vychází z NN8, cílem bylo otestovat vliv množství plně propojených vrstev na výsledek sítě. Síť má výstupní vrstvy o velikosti 512 neuronů. Srovnání s NN8 naleznete v 3.2.3.

NN18 Úspěšnost 77,4 procent
Síť vychází z NN16 a ve výstupní části má každá plně propojená vrstva pouze 512 neuronů. Srovnání s NN16 naleznete v 3.2.3.

NN19 Úspěšnost 75,9 procent
Síť vychází z NN15 a má oproti ní ve výstupní části vrtsvy o velikosti 512 neuronů. Srovnání s NN15 naleznete v 3.2.3.

NN20 Úspěšnost 79,1 procent
Síť vychází z NN8, liší se velikostí vrstev ve výstupní části. Obě plně propojené vrstvy jsou o velikosti 2048 neuronů. Cílem bylo otestovat co největší velikost výstupní části, ale jednalo se o maximální velikost, která nepřekročila velikost paměti RAM na grafické kartě. Srovnání s NN8 naleznete v 3.2.3. Síť měla lehce lepší výsledek než NN8.

NN21 Úspěšnost 77,5 procent
Síť architektonicky vychází z NN8 a obsahuje inception modul. Srovnání s NN8 naleznete v 3.2.5. Visualizovanou architekturu sítě naleznete v příloze.

NN22 Úspěšnost 73,7 procent
Síť architektonicky vychází z NN8 a obsahuje dva inception moduly. Srovnání s NN8 a NN21 naleznete v 3.2.5. Visualizovanou architekturu sítě naleznete v příloze.

NN23 Úspěšnost 79,9 procent
Síť byla prvním pokusem shrnout dosavadní poznatky k vytvoření úspěšné

architektury (sítě jsou řazeny chronologicky dle vývoje testování). Jedná se o síť, která vychází z inception modulu, ale používá větší konvoluční vrstvy o hloubce 512, úspěšnou velikost jádra [15,1] a plně propojené vrstvy ve výstupní části o velikosti 2048 a 1024 neuronů. Síť dosáhla výborného výsledku. Visualizovanou architekturu sítě naleznete v příloze.

NN24 Úspěšnost 78,9 procent
Síť vychází z NN23 a testuje hloubku konvolučních vrstev 1024. Srovnání s NN23 naleznete v 3.2.2

NN25 Úspěšnost 79,9 procent
Síť vychází z NN23 a testuje použití residuální architektury. Jedná se o síť s nejlepším výsledkem. Síť je blíže popsána v kapitole 3.3.1 ke naleznete také visualizaci její architektury. Srovnání s NN23 naleznete v 3.2.6.

NN26 Úspěšnost 79,8 procent
Síť vychází z NN23 a používá hloubku konvolučních vrstev ve větvené části 256. Srovnání s NN23 naleznete v 3.2.2.

NN27 Nekonvergovala, bez výsledku
Síť vychází z NN25 a sériově zdvojuje její výpočetní část. Síť je již poměrně komplexní a nepodařila se zkonvergovat při použití optimizéru sgd. Síť se později podařilo zkonvergovat s optimizérem Adam pod označením NN37.

NN28 Úspěšnost 77,9 procent, v pozdějších epochách nekonvergovala
Jedná se o síť NN27 s optimizační funkcí Adam bez použití parametru decay. Síť v polovině učení přestala konvergovat.

NN29 Nekonvergovala, bez výsledku
Síť vychází z NN28 bez použití residua. Byl použit optimizér Adam s nulovou hodnotou decay. Síť nekonvergovala.

NN30 Úspěšnost 73,3 procent, v pozdějších epochách nekonvergovala
Síť architektonicky vychází z NN27, ale hloubky sítí ve výpočetní části snižuje na 64. Pro učení byl použit optimizér Adam s nulovou hodnotou decay. Síť v polovině učení přestala konvergovat.

NN31 Nekonvergovala, bez výsledku
Síť vychází z NN28 do jejíž výpočetní části byly vloženy další výpočetní vrstvy. Síť nekonvergovala. Visualizaci architektury sítě naleznete v příloze.

NN32 Úspěšnost 78,5 procent
Síť architektonicky vychází z NN1 a byla trénována s použitím optimizéru Adam s hyperparametry (lr=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08, decay=0.00005). Síť dosáhla dobrého úspěchu. Srovnání s dalšími sítěmi na testování optimizérů naleznete v 3.2.4.

NN33 Nekonvergovala, bez výsledku
Síť architektonicky vychází z NN1 a byla trénována s použitím optimizéru Adam s hyperparametry adam = Adam(lr=0.01, beta1=0.9, beta2=0.999, epsilon=1e-08, decay=0.00005). Síť nekonvergovala pravděpodobně kvůli vysokému parametru rychlosti učení.

NN34 Úspěšnost 75,0 procent
Síť architektonicky vychází z NN1 a byla trénována s použitím optimizéru Adamax s hyperparametry (lr=0.002, beta1=0.9, beta2=0.999, epsilon=1e-08, decay=0.0). Srovnání s dalšími sítěmi na testování optimizérů naleznete v 3.2.4.

NN35 Úspěšnost 74,6 procent
Síť architektonicky vychází z NN1 a byla trénována s použitím optimizéru Adadelta s hyperparametry (lr=1.0, rho=0.95, epsilon=1e-08, decay=0.0). Srovnání s dalšími sítěmi na testování optimizérů naleznete v 3.2.4.

NN36 Úspěšnost 76,7 procent
Síť architektonicky vychází z NN1 a byla trénována s použitím optimizéru Adagrad s hyperparametry (lr=0.01, epsilon=1e-08, decay=0.0). Srovnání s dalšími sítěmi na testování optimizérů naleznete v 3.2.4.

NN37 Úspěšnost 77,0 procent
Síť vychází z architektury NN27 a pro trénování byla použita optimizační funkce Adam s následujícími parametry (lr=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08, decay=0.00005).

NN38 Nekonvergovala, bez výsledku
Síť vychází z architektury NN27 a pro trénování byla použita optimizační funkce Adagrad. Síť nekonvergovala při nastavení hyperparametrů (lr=0.01, epsilon=1e-08, decay=0.0)

NN39 Nekonvergovala, bez výsledku
Síť vychází z architektury NN27 a pro trénování byla použita optimizační funkce Adam s následujícími parametry (lr=0.002, beta1=0.9, beta2=0.999, epsilon=1e-08, decay=0.00005). Síť nekonvergovala.

NN40 Úspěšnost 79,5 procent
Síť vychází z architektury NN25 a pro trénování byla použita optimizační funkce Adam s následujícími parametry (lr=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08, decay=0.000005).

NN41 Úspěšnost 76,3 procent
Síť vychází z NN1 a používá danou síť pro modelování architektury multi-column. Síť byla učena pomocí optimizéru Adam s hyperparametry (lr=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08, decay=0.00005). Visualizaci architektury sítě naleznete v příloze.

NN42 Úspěšnost 75,7 procent
Síť vychází z architektury NN41 ale používá větší hodnoty konvolučního jádra [3,86], [5,1], [15,1]. Síť dosáhla průměrných výsledků.

NN43 Úspěšnost 77,3 procent
Síť architektonicky vychází z NN1 a byla trénována s použitím optimizéru Nadam s hyperparametry (lr=0.002, beta1=0.9, beta2=0.999, epsilon=1e-08, scheduledecay=0.004). Srovnání s dalšími sítěmi na testování optimizérů naleznete v 3.2.4.

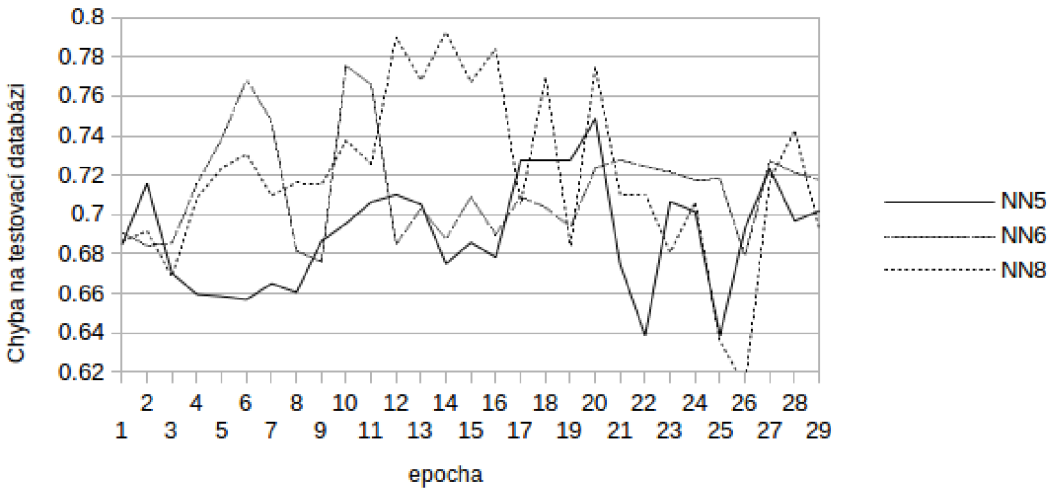
3.2. Porovnání testovaných sítí dle parametrů

V této kapitole jsme se zaměřili na porovnání sítí, které jsme modelovali záměrně na stejných architekturách pouze se změnou některých parametrů. Cílem jejich srovnání je nalezení optimálního nastavení architektury a hyperparametrů výsledné sítě. Postupně jsou srovnávány sítě na základě velikosti konvolučních jader, hloubky konvolučních vrstev, velikosti plně propojených vrstev ve výstupní části, optimizační funkce, použití inception modulu a residuální architektury. Porovnávali jsme pouze vybrané sítě.
Některé výsledky porovnání dle určitých parametrů nepřinesly jednoznačný výsledek, jelikož tvorba hlubokých konvolučních sítí je při testování nových architektur často možná pouze metodou pokus-omyl. Jasné výsledky ovšem přineslo porovnání dle rozměru plně propojených vrstev na konci sítě a podle velikosti konvolučního jádra.

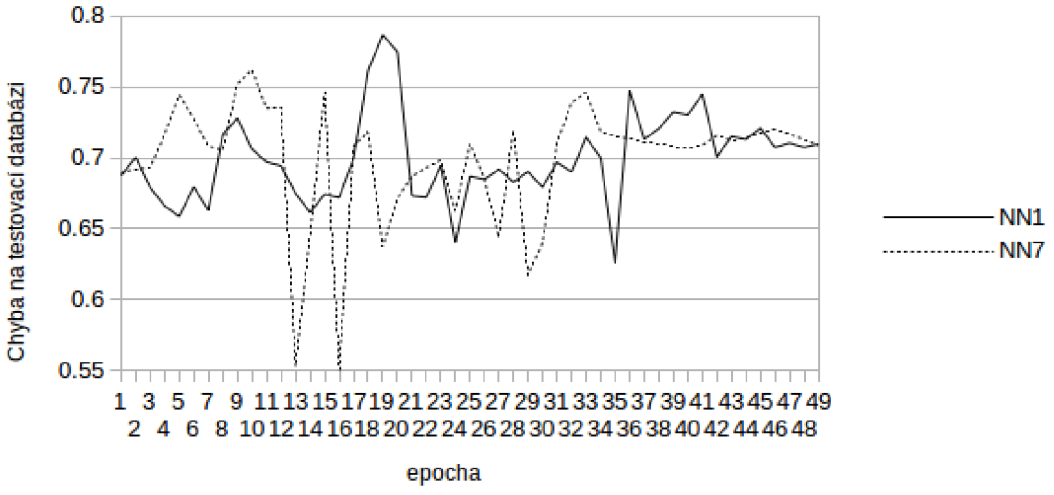
3.2.1. Porovnání dle velikosti konvolučního jádra

V této podkapitole jsme zkoumali vliv velikosti jádra na výsledky daných sítí. V rámci práce nám pro použití na textová data vyšlo nejlépe jádro o velikosti [15,1]. Na grafu 3.1 vidíme srovnání sítí NN5(jádro[7,1] a [4,1]), NN6 (jádro[30,1]) a NN8 (jádro[15,1]). Z grafu je jasně vidět, že síť NN8 má po většinu výpočetních epoch lepší výsledky, než ostatní dvě sítě.

Na grafu 3.2 jsou srovnány sítě NN1 s jádrem o velikosti [7,86] a [4,1] se sítí NN7, na které byly testovány jádra o velikosti [30,86], [20,1] a [10,1]. Z grafu je vidět, že síť NN1 má lepší výsledky.
Z těchto poznatků jsme vyvodili, že konvoluční sítě pro tento problém zpracování textu lépe pracují s menšími a středními velikostmi konvolučního jádra.



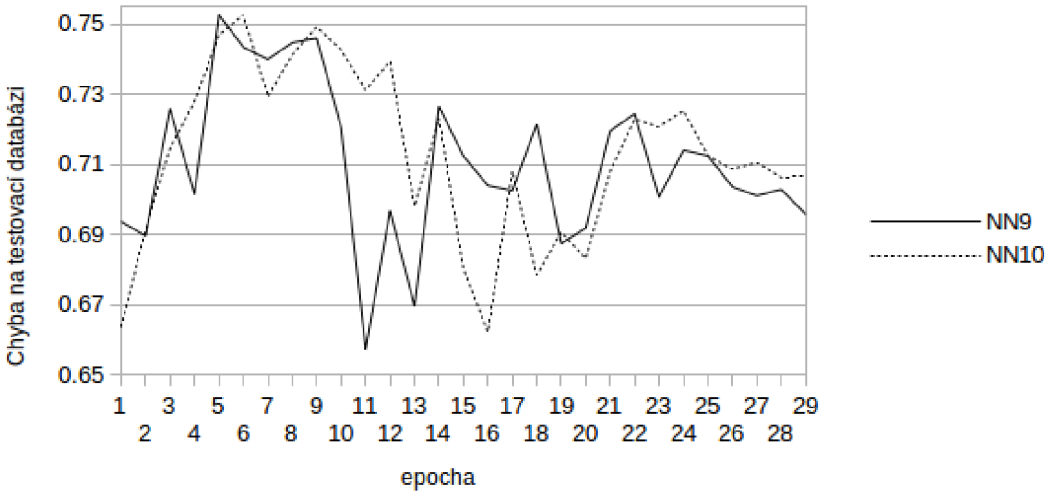
Obrázek 3.1: Graf průběhu testování sítí NN5, NN6, NN8



Obrázek 3.2: Graf průběhu testování sítí NN1 a NN7

3.2.2. Porovnání dle hloubky konvolučních vrstev

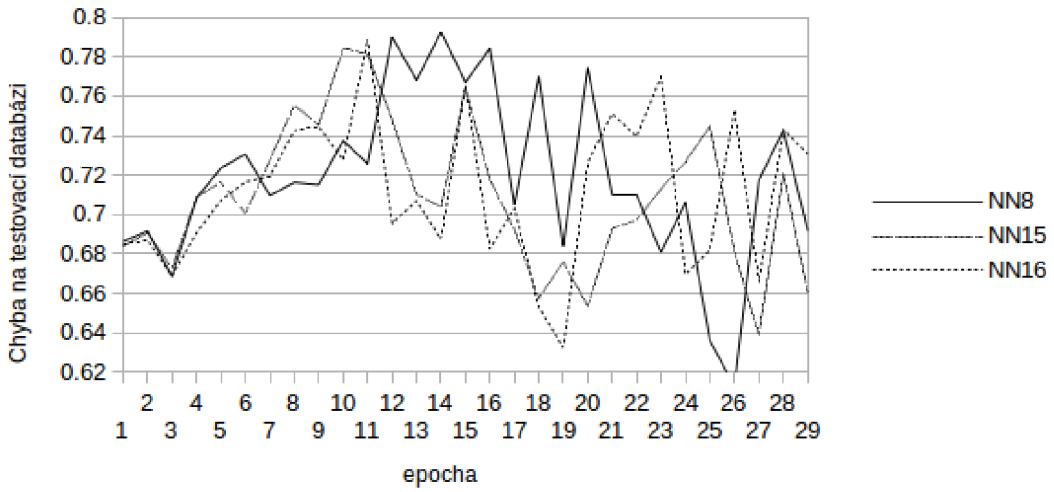
Testovali jsme několik architektur konvolučních sítí. Logicky jsme mysleli, že hlubší síť bude mít víc parametrů a tím pádem bude mít lepší schopnost vyhodnocovat daný problém na úkor výpočetní náročnosti.
Toto porovnání však jasné výsledky nepřineslo. Často měly hlubší sítě menší úspěšnost, než sítě o menší hloubce konvolučních vrstev. Z výsledků vyplývá, že nelze vyvodit jednoznačné pravidlo pro nastavení hloubky konvolučních vrstev.
Graf 3.3 zobrazuje porovnání výsledků sítí NN9, která používala hloubku konvolučních vrstev 512, a sítě NN10 s hloubkami 1024. Z grafu nelze jednoznačně určit, která síť má lepší výsledek. Obě sítě měly v nejlepší epoše úspěšnost 75.3 procent.



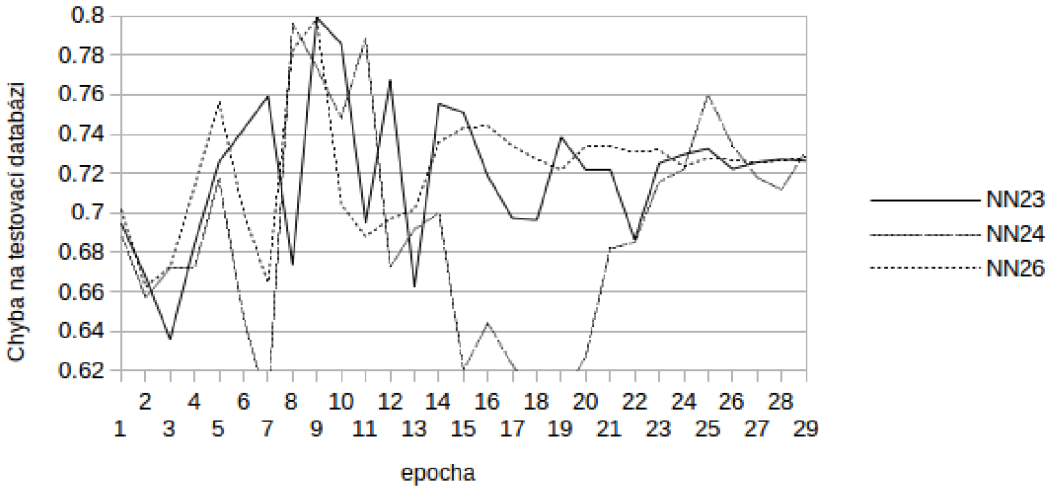
Obrázek 3.3: Graf průběhu testování sítí NN9 a NN10

Graf 3.4 zobrazuje průběh testování NN8, která měla hloubky 1024 a 2048, NN15, jejíž hloubky konvolučních vrstev byly poloviční a NN16, která byla modelována s hloubkami pouze 256 a 512, tedy čtvrtinovými. Přestože NN8 měla nejlepší výsledky, čekali jsme větší rozdíl při takové změně parametrů.

Graf 3.5 srovnává sítě vycházející architekturou z NN23, která má hloubky 512, síť NN24 používá hloubky 1024 a NN26 hloubky o hodnotě 256. Nejlepší výsledek dosáhla síť NN23, proto jsme dále při práci s touto architekturou používali hloubku 512. S touto hloubkou pracuje i nejúspěšnější síť NN25.



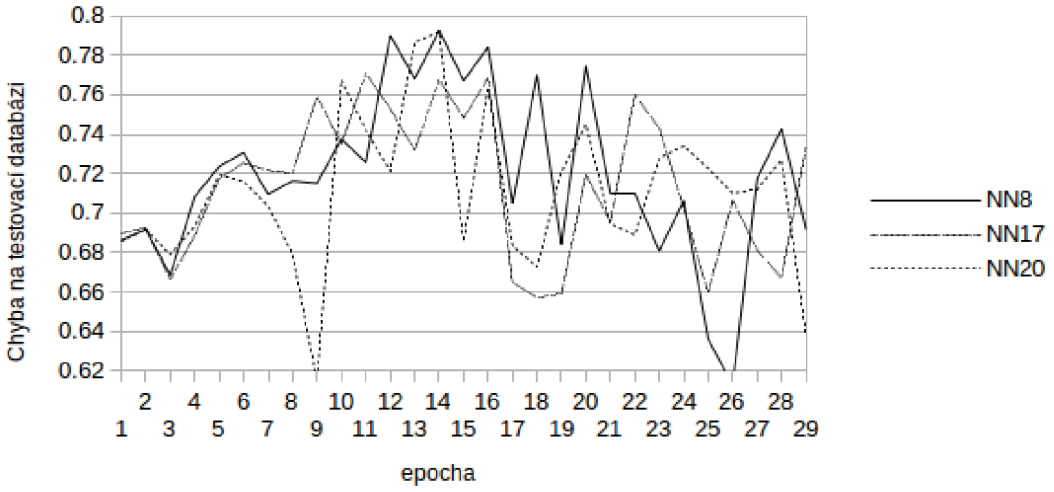
Obrázek 3.4: Graf průběhu testování sítí NN8, NN15 a NN16



Obrázek 3.5: Graf průběhu testování sítí NN23, NN24 a NN26

3.2.3. Porovnání dle rozměru plně propojených vrstev výstupní části

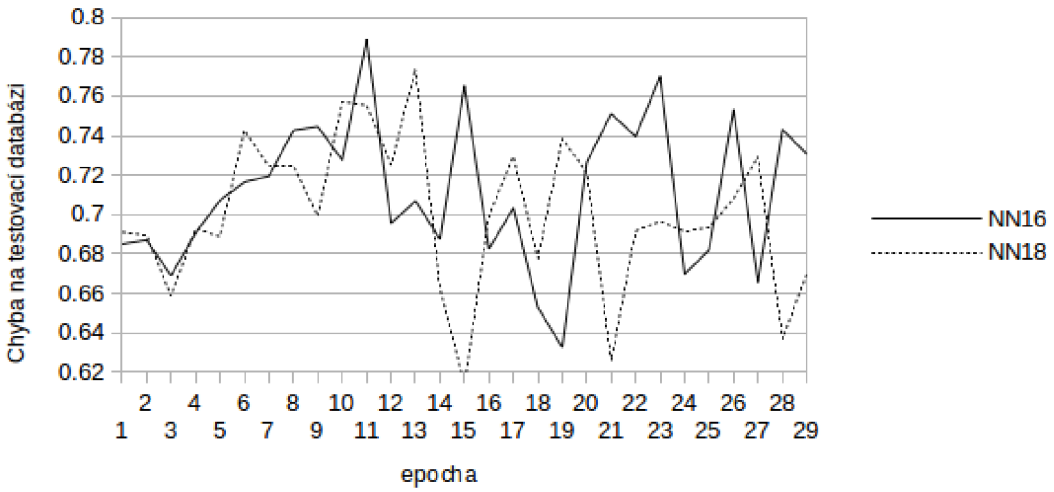
Velikost výstupních plně propojených vrstev se ukázala jako důležitý parametr ovlivňující výsledek sítě. Jako nejlepší se pro naše účely ukázala hodnota 2048 a 1024 neuronů ve výstupních vrstvách.
Graf 3.6 ukazuje srovnání sítí NN8, která má vrstvy o velikostech 2048 a 1024, poté NN17, která má vrstvy poloviční a NN20, jejíž plně propojené vrstvy ve výstupní části mají velikost obě shodně 2048 neuronů. NN8 měla nejlepší výsledek a nevykazovala tolik fluktuací jako NN20.



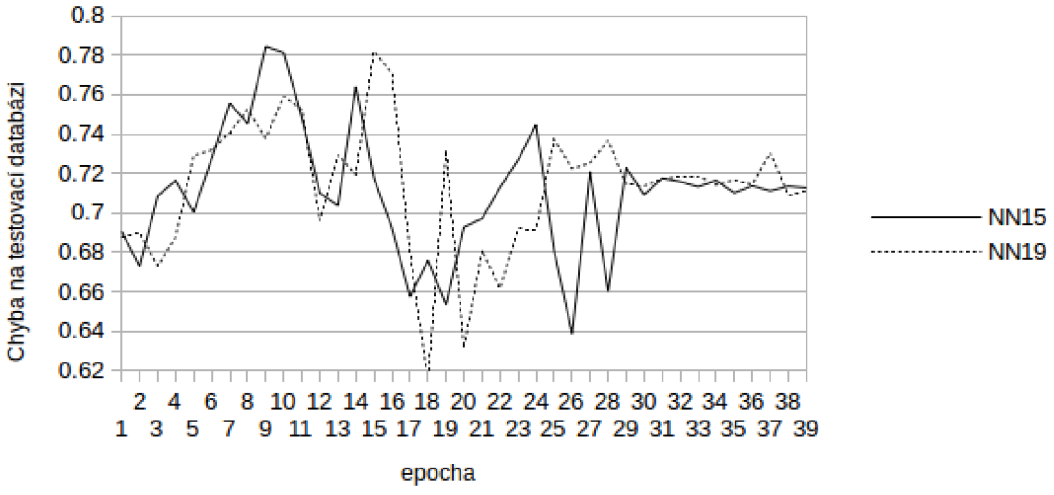
Obrázek 3.6: Graf průběhu testování sítí NN8, NN17 a NN20

Graf 3.7 srovnává NN16 se sítí NN18, která má oproti NN16 poloviční velikost výstupních vrstev. Z grafu je patrné, že výrazně lepší výsledek měla NN16.

Graf 3.8 srovnává NN15 se sítí NN19, která má oproti NN18 podobně jako v minulém případě poloviční velikost výstupních vrstev. Lepší výsledek zase přinesla síť NN15 s větším počtem nenuronů v plně propojených výstupních vrstvách.



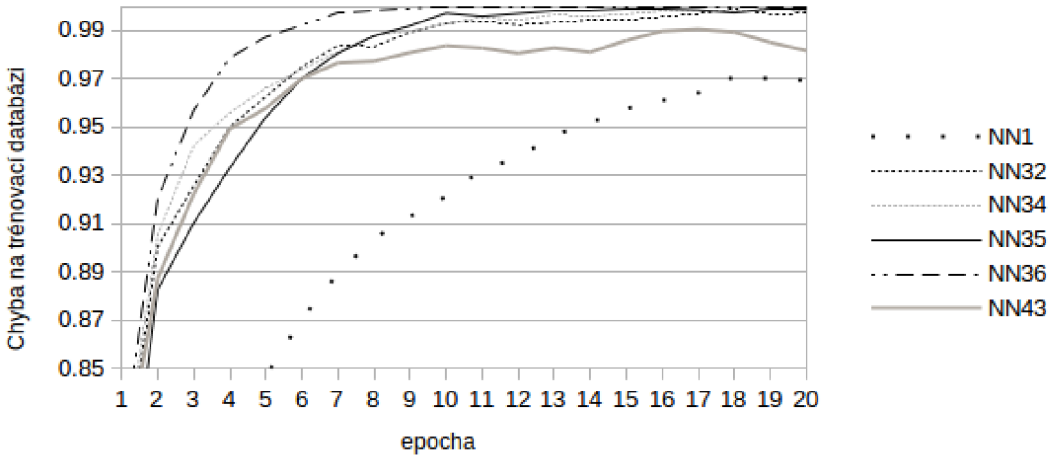
Obrázek 3.7: Graf průběhu testování sítí NN16 a NN18



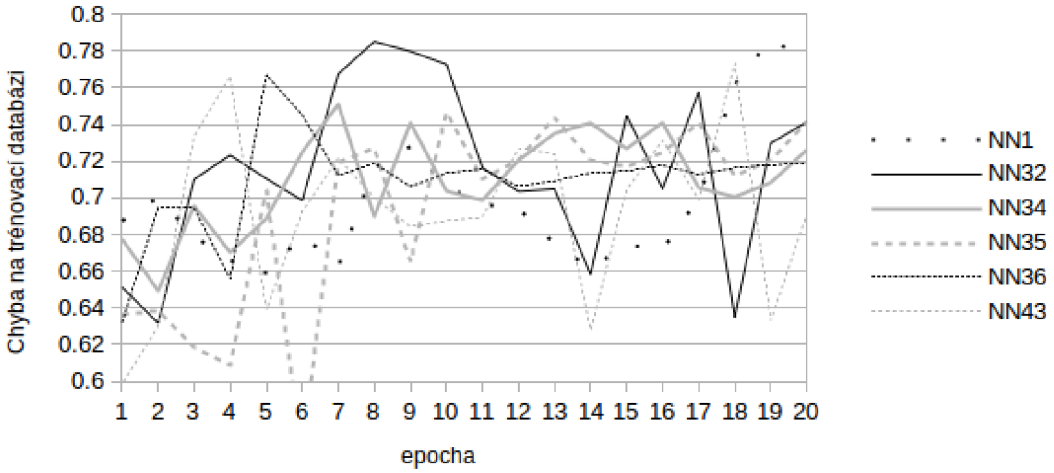
Obrázek 3.8: Graf průběhu testování sítí NN15 a NN19

3.2.4. Porovnání dle optimizační funkce

V této části jsme porovnávali použití různých optimizačních funkcí. Jelikož se sítě lišily nejen výsledky, ale celkem markantně také rychlostí konvergence, zařadili jsme do práce také graf 3.9 který ukazuje průběh učení pomocí jednotlivých optimizérů. Testované sítě používaly následující optimizéry: NN1-sgd, NN32-Adam, NN34-Adamax, NN35-Adadelta, NN36-Adagrad a NN43-Nadam. Nejrychleji konvergovala síť NN36, ovšem moc rychle snížila svoji hodnotu learning rate a nedosáhla dobrých výsledků. Nejpomaleji konvergovala síť NN1.Nejlepších výsledků dosáhly sítě NN1 a NN32, proto jsme v dalších sítích používali tyto dva optimizéry.



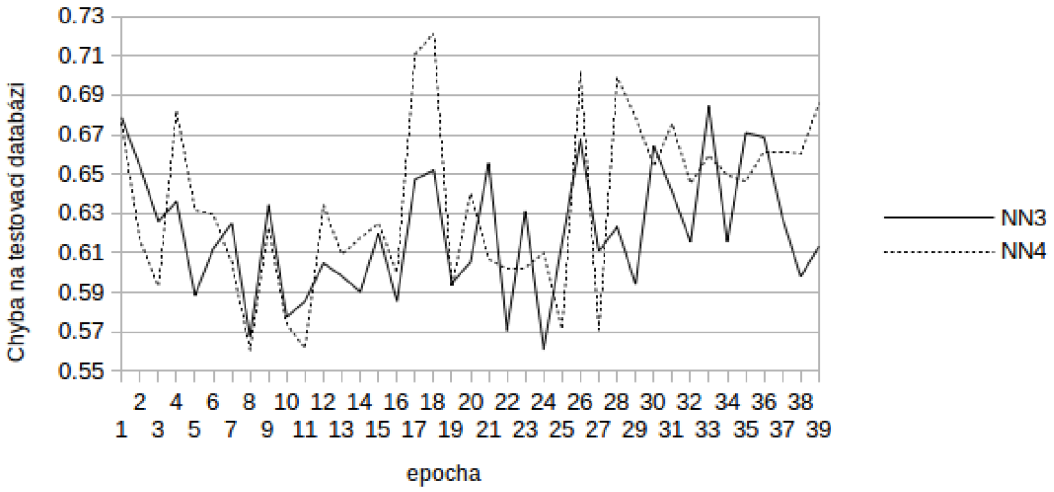
Obrázek 3.9: Graf průběhu trénování sítí NN1, NN32, NN34, NN35, NN36 a NN43



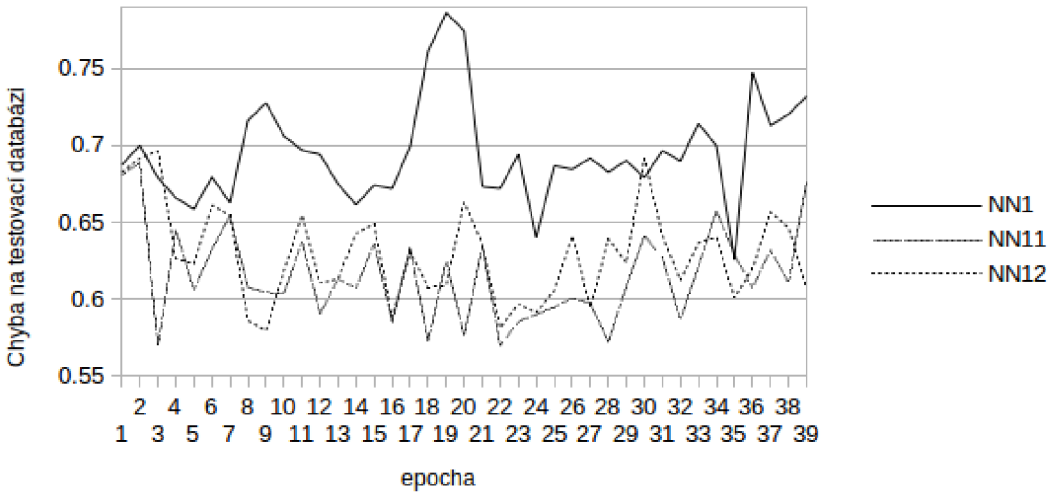
Obrázek 3.10: Graf průběhu testování sítí NN1, NN32, NN34, NN35, NN36 a NN43

3.2.5. Porovnání dle použití inception modulu

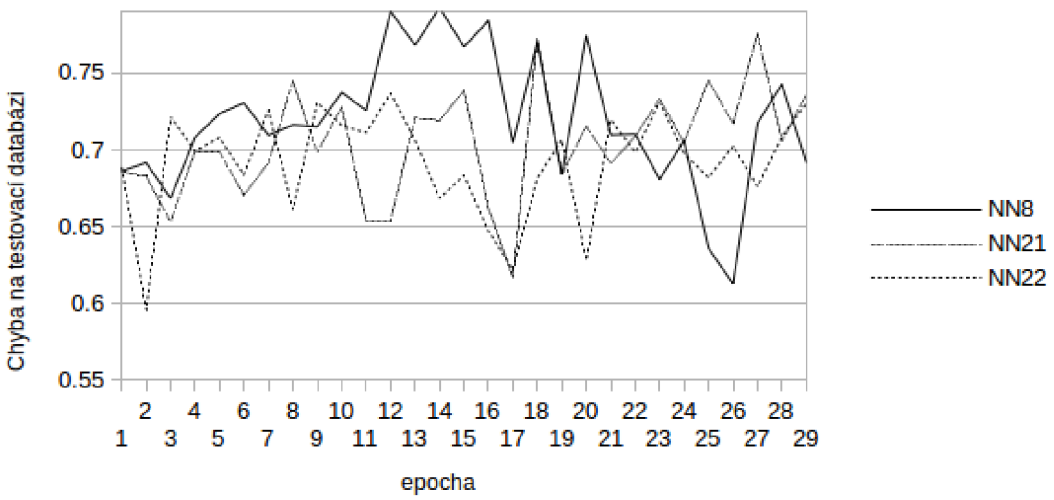
Inception modul jsme úspěšně použili jenom v jednom případě. Ve všech ostatních aplikacích jeho použití zhoršilo výsledek sítě. Zlepšení je jasně viditelné na grafu 3.11 kde NN3 inception modul neobsahuje a NN4 se od ní liší práve jeho přítomností v architektuře. V dalších dvou případech 3.12 a 3.13 vždy nejlepšího výsledku dosáhla síť, která inception modul neobsahovala. Hledání správného použití inception modulu je nad rámec této práce. Z architektury samotného modulu jsme však vycházeli při konstrukci sítě NN25, která měla nejlepší výsledek.



Obrázek 3.11: Graf průběhu testování sítí NN3 a NN4



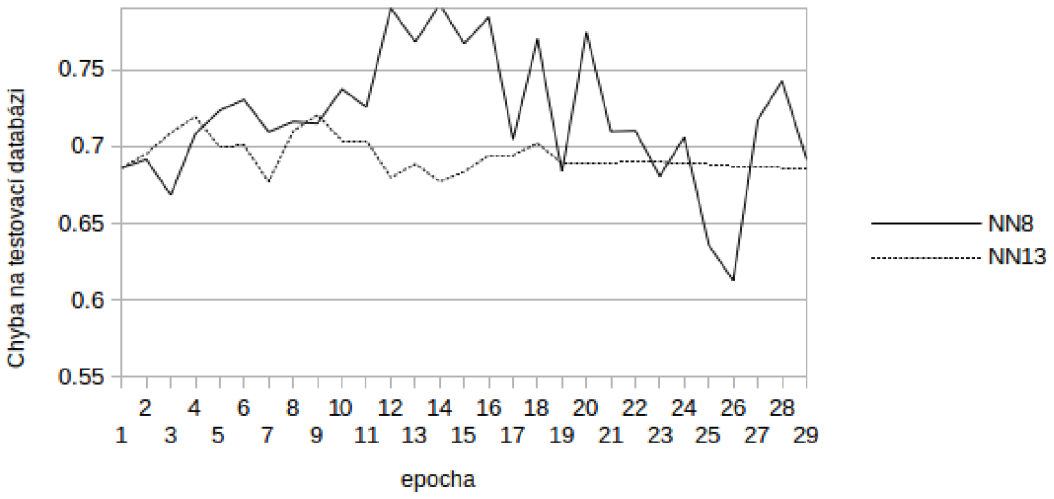
Obrázek 3.12: Graf průběhu testování sítí NN1, NN11 a NN12



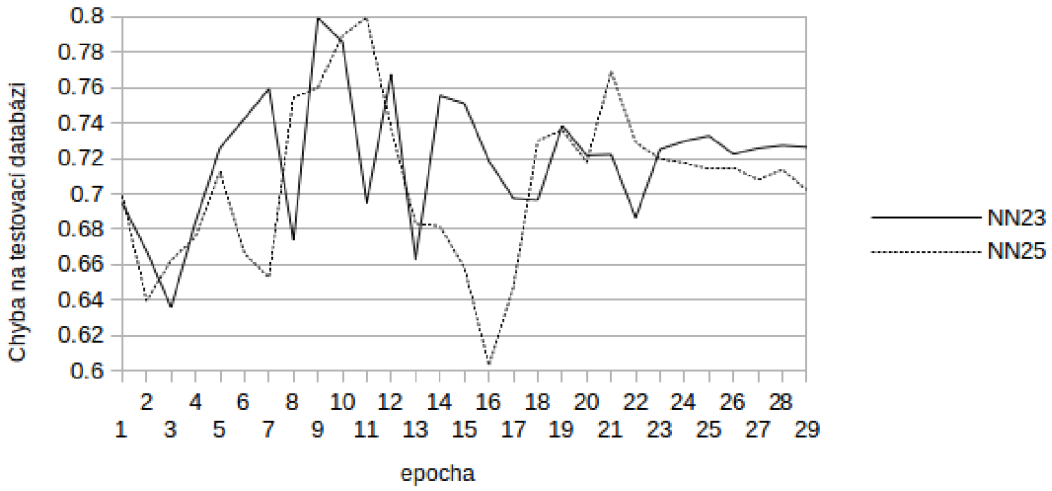
Obrázek 3.13: Graf průběhu testování sítí NN8, NN21 a NN22

3.2.6. Porovnání dle použití residuální architektury

Při testování residuálních sítí jsme zkoušeli aplikovat residuum, přenos dat z nižších vrstev, na některé již existující, úspěšné sítě.
Na prvním příkladu 3.14 je vidět, že lepší výsledek má síť NN8, než síť NN13, která má stejnou architekturu, akorát s residuem.
Naopak v případě 3.15 měla residuální architektura pozitivní vliv na výsledek, zde NN23 bez residuí, NN25 s residuální architekturou.



Obrázek 3.14: Graf průběhu testování sítí NN13 a NN18



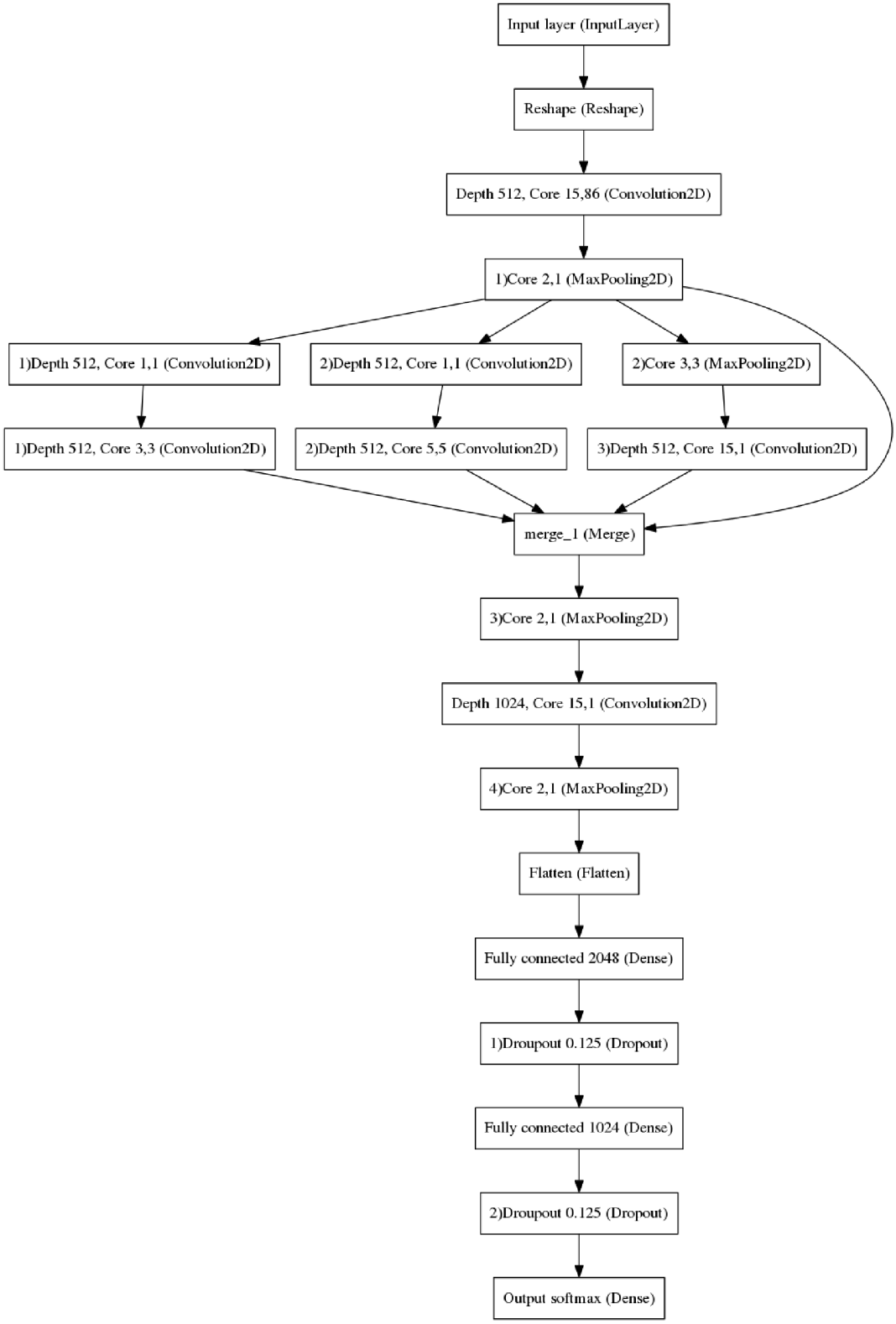
Obrázek 3.15: Graf průběhu testování sítí NN23 a NN25

3.3. Výsledná architektura

Výsledkem této práce je architektura hluboké konvoluční neuronové sítě, v práci označovaná NN25, která byla navržena s ohledem na získané poznatky ohledně velikosti konvolučního jádra, hloubce konvolučních vrstev, velikosti plně propojených vrstev výstupní části, použité optimizační funkce a použití residuální architektury. Její model je znázorněn na obrázku 3.16.
Síť dosáhla nejlepšího výsledku 79,94 procent.

3.3.1. Rozbor architektury

Výpočetní část architektury vychází z myšlenky inception modulu, ale uzpůsobuje ho k použití pro zpracování textu. Hloubka konvolučních vrstev v paralelní části je zvolena na hodnotu 512 a kromě jader o velikost [1,1,] jsou použita jádra o velikostech [3,3], [5,5] a [15,1]. Původní myšlenka použití těchto velikostí jader vychází z průměrné délky slova v českém textu, která je 5,5 znaku. V prvním sloupci bychom jádrem o velikosti [3,3] zpracovali trigramy, dalším sloupce bychom zpracovali průměrně téměř jedno slovo jádrem [5,5] a posledním sloupcem bychom zpracovali paralelně téměř tři průměrná slova konvolučním jádrem [15,1], která by se nacházela v textu vedle sebe. Jelikož paralelní výpočetní část architektury leží za další konvoluční vrstvou, data do ní jdou již zpracovaná. Použití tohoto výpočetního modulu by bylo zajímavé vyzkoušet v budoucnu přímo na vstupu sítě a zjistit, zda se předchozí teorie ukáže jako pravdivá. Data z paralelní části jdou na vstup další konvoluční vrstvy, abychom snížili počet parametrů na vstupu plně propojených vrstev a tím zkrátili dobru trénování. Ve výstupní části jsou použity vrstvy o velikostech 2048 a 1024 neuronů. Tato velikost se při testování ukázala jako nejúspěšnější. Síť byla učena pomocí optimizéru sgd. V architektuře sítě NN25 jsme také s úspěchem

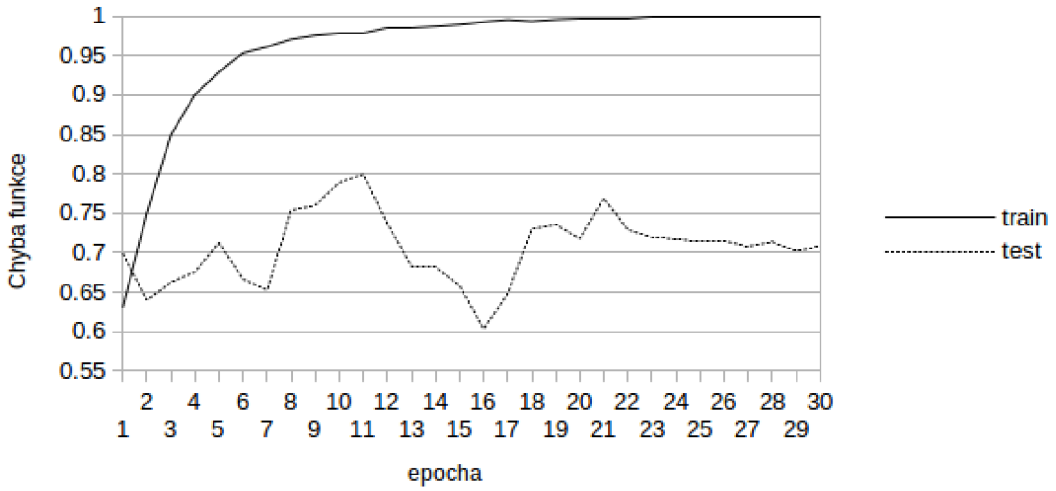


Obrázek 3.16: Model architektury sítě NN25

použili residuum, které nám přemostí paralelní výpočetní část a data ze vstupu sítě spojí s daty po průchodu paralelní částí architektury. V sítí jsou použity maxpoolingové vrstvy s jádry [2,1] a [3,3]. Jako aktivační funkce všech vrstev je použita rectified linear unit. Ve výstupní části jsou použity vrstvy Dropout s koeficientem 0,125 pro zabránění přeučení sítě a pro větší pravděpodobnost nalezení globálního minima chybové funkce.

3.3.2. Naměřené výsledky

Síť měla v nejlepší výpočetní epoše chybovost na validačních datech 79,94 procent. Pro lepší výsledky by bylo dobré provést výpočet většího množství epoch a následně hyperparametrickou optimalizaci, na kterou bohužel před odevzdáním práce nezbyl čas.



Obrázek 3.17: Graf průběhu trénování a testování NN25

4. Závěr

V rámci diplomové práce jsme nejdřív probrali teoretický základ neuronových sítí se zaměřením na jejich konvoluční variantu, její vlastnosti a různé přístupy ke tvorbě architektur. Poté jsme vyzkoušeli 43 testovaných sítí a vyhodnocením jejich parametrů jsme vytvořili a popsali výslednou síť v práci označovanou jako NN25 s úspěšností 79,94 procent na validační databázi.
Hlavním přínosem práce je vytvoření již zmiňované specializované architektury sítě NN25 pro řešení problému emoční klasifikace textových dat a v otestování různých architektur na tuto problematiku. Mezi další přínosy patří také vytvoření základního přehledu vlastností a nastavení konvolučních neuronových sítí v českém jazyce. Všechny důležité materiály, ze kterých bylo v rámci práce čerpáno, byly v angličtině, a věřím, že práce bude v tomto ohledu někomu užitečná.
Další pokračování práce by bylo možné v rámci hyperparametrické optimalizace sítě NN25 a její hlubší testování. Dále také ve zkoušení dalších architektur na daný problém a postupném rozšiřování teoretického základu v rámci práce.

Literatura

[1] ŠÍMA, Jiří a Roman NERUDA.: *Teoretické otázky neuronových sítí*. Praha: MATFYZPRESS. 1996 ISBN 80-858-6318-9.

[2] SCHMIDHUBER, Jürgen.: *Deep learning in neural networks: An overview*. Neural Networks. 61, 85-117. DOI: 10.1016/j.neunet.2014.09.003. ISSN 08936080.

[3] GATYS, Leon A., Alexander S. ECKER a Matthias BETHGE.: *A Neural Algorithm of Artistic Style*. 2015. CoRR [online]. (abs/1508.06576), 12 [cit. 2016-12-14]. Dostupné z: http://arxiv.org/abs/1508.06576

[4] JONES, Karen Sparck.: *Natural Language Processing: A Historical Review*. Current Issues in Computational Linguistics: In Honour of Don Walker. Dordrecht: Springer Netherlands, , 3. DOI: 10.1007/978-0-585-35958-8 1. ISBN 978-0-7923-2998-5. Dostupné také z: http://link.springer.com/10.1007/978-0-585-35958-8 1

[5] KAMINSKY, Jermain C.: *Nowcasting the Bitcoin Market with Twitter Signals.*. 2016. https://arxiv.org [online]. [cit. 2017-21-5]. Dostupné z: https://arxiv.org/pdf/1406.7577.pdf

[6] POVODA, Lukáš, Radim BURGET, Jan MAŠEK a Lubomír CVRK.: *Automatické rozpoznávanie emócií z českého textu pomocou umelej inteligencie*. Elektrorevue [online]. 2015(17), 5 [cit. 2016-12-13]. Dostupné z: http://www.elektrorevue.cz/cz/clanky/zpracovani-signalu/0/automaticke-rozpoznavanie-emocii-z-ceskeho-textu-pomocou-umelej-inteligencie–emotion-recognition-system-based-on-artificial-inteligence-for-czech-texts-/

[7] KARPATHY, Andrej.: *Convolutional networks for visual recognition*. In: Github [online]. [cit. 2016-12-13]. Dostupné z: http://cs231n.github.io/convolutional-networks

[8] CHOLLET, Francois.: *Keras.*. 2015. In: GitHub [online]. [cit. 2016-12-13]. Dostupné z: https://github.com/fchollet/keras

[9] RUDER, Sebastian.: *An overview of gradient descent optimization algorithms*. 2016. In: https://arxiv.org [online]. [cit. 2017-05-22]. Dostupné z: https://arxiv.org/pdf/1609.04747.pdf

[10] KARPATHY, Andrej.: *Convolutional networks for visual recognition, Optimization*. In: Github [online]. [cit. 2017-05-22]. Dostupné z: http://cs231n.github.io/optimization-1/

[11] BOTTOU, L´eon.: *Stochastic Gradient Learning in Neural Networks*. [online]. Bell Laboratories [cit. 2017-05-22]. Dostupné z: http://leon.bottou.org/publications/pdf/nimes-1991.pdf

[12] DUCHI, John, Elad HAZAN a Yoram SINGER.: *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*. In: The Journal of Machine Learning Research. 2011(12), 2121-2159.

[13] ZEILER, Matthew D.: *ADADELTA: An Adaptive Learning Rate Method.*. CoRR [online]. (abs/1212.5701) [cit. 2017-05-22]. Dostupné z: http://dblp.uni-trier.de/rec/bib/journals/corr/abs-1212-5701

[14] KINGMA, Diederik P., Jimmy LEI BA.: *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*. In: CoRR [online]. [cit. 2017-05-22]. Dostupné z: https://arxiv.org/pdf/1412.6980v8.pdf

[15] DOZAT, Timothy.: *INCORPORATING NESTEROV MOMENTUM INTO ADAM.*. [online].[cit. 2017-05-22]. Dostupné z: https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ

[16] LECUN, Y., L. BOTTOU, Y. BENGIO a P. HAFFNER.: *Gradient-based learning applied to document recognition.*. 1998. Proceedings of the IEEE. 86(11), 2278-2324. DOI: 10.1109/5.726791. ISSN 00189219. Dostupné také z: http://ieeexplore.ieee.org/document/726791/

[17] ZHANG, Xiang a Yann LECUN.: *Text Understanding from Scratch.*. 2015. NCoRR [online]. abs/1502.01710, 10 [cit. 2016-12-13]. Dostupné z: http://arxiv.org/abs/1502.01710

[18] SZEGEDY, Christian, WEI LIU, YANGQING JIA, et al.: *Going deeper with convolutions.*. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, , 1-9 [cit. 2016-12-13]. DOI: 10.1109/CVPR.2015.7298594. ISBN 978-1-4673-6964-0. Dostupné z: http://ieeexplore.ieee.org/document/7298594/

[19] CIREŞAN, Dan, Ueli MEIER, Jonathan MASCI a Jürgen SCHMIDHUBER.: *Multi-column deep neural network for traffic sign classification.*. Neural Networks [online]. 32, 333-338 [cit. 2016-12-13]. DOI: 10.1016/j.neunet.2012.02.023. ISSN 08936080. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0893608012000524

[20] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN.: *Deep Residual Learning for Image Recognition.*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, 770-778 [cit. 2017-05-23]. DOI: 10.1109/CVPR.2016.90. ISBN 978-1-4673-8851-1. Dostupné z: http://ieeexplore.ieee.org/document/7780459/

[21] MAY, Madison.: *An Overview of Python Deep Learning Frameworks.*. Http://www.kdnuggets.com/ [online]. [cit. 2017-05-23]. Dostupné z: http://www.kdnuggets.com/2017/02/python-deep-learning-frameworks-overview.html

[22] RAMI Al-Rfou, Guillaume Alain, Amjad Almahairi a et al.: *Theano: Python framework for fast computation of mathematical expressions.*. [online]. 2016. , 19 [cit. 2016-12-13]. Dostupné z: https://arxiv.org/abs/1605.02688

[23] NICKOLLS, John, Ian BUCK, Michael GARLAND a Kevin SKADRON.: *Scalable parallel programming with CUDA*.. 2008. Queue [online]. 6(2), 40- [cit. 2016-12-13]. DOI: 10.1145/1365490.1365500. ISSN 15427730. Dostupné z: http://portal.acm.org/citation.cfm?doid=1365490.1365500

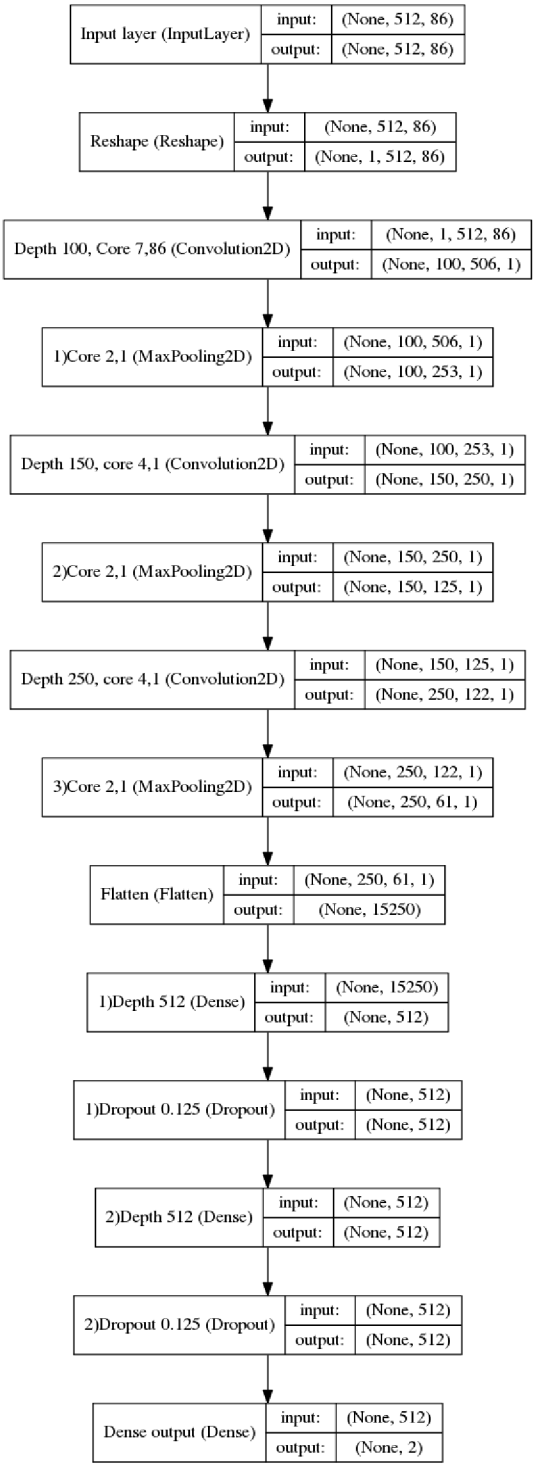
Seznam použitých zkratek a symbolů

x_i	vstup perceptronu s indexem i
w_i	váha vstupu perceptronu s indexem i
y	výstup perceptronu
w_0	práh perceptronu
ξ	vnitřní potenciál perceptronu
η	koeficient učení perceptronu
XOR	označení funkce exkluzivní disjunkce
λ	koeficient rychlosti učení backpropagation
SGD	stochastic gradient descent
NN	Neural network, česky neuronová síť
DNN	Deep neural network, česky hluboká neuronová síť
CDNN	Convolutional deep neural network, česky konvoluční hluboká neuronová síť
SVM	Support vector machine
LSTM	Long short-term memory, rekurentní architektura neuronových sítí
CPU	central processing unit, procesor
GPU	graphic processing unit, grafická karta
RAM	random access memory, počítačová paměť

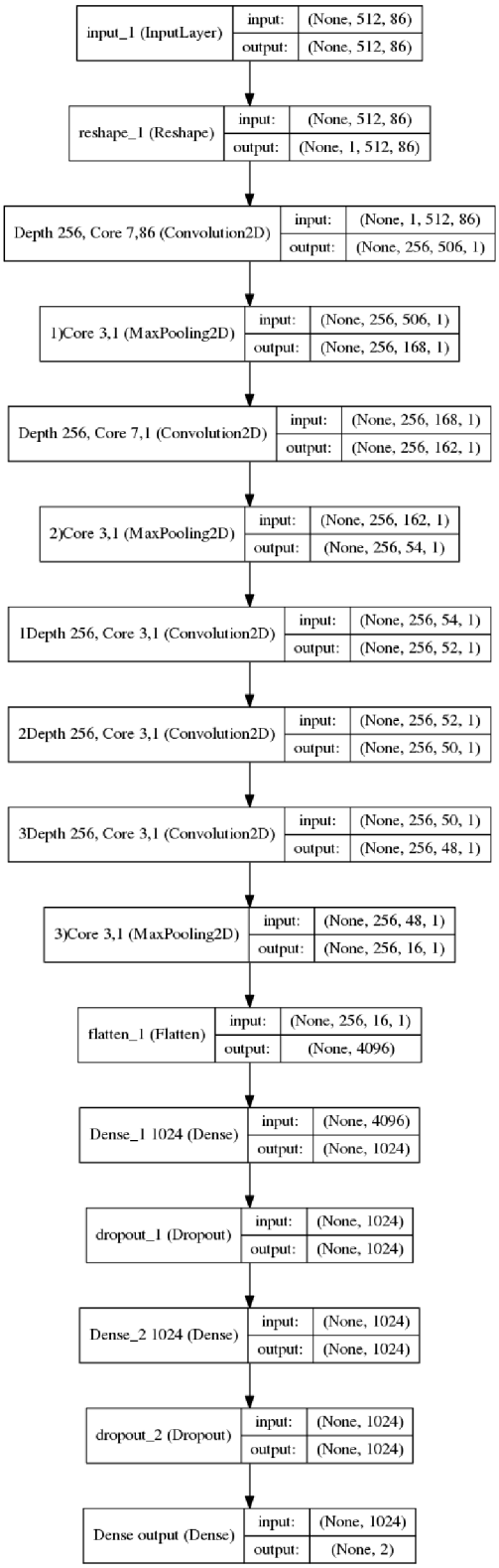
Seznam příloh

skripty.zip - Soubor obsahuje kompletní zdrojové kódy k práci se vstupní databází (chunks1.py), kódy jednotlivých neuronových sítí použitých v praktické části (modely/NNx.py), soubory obsahující výsledky jednotlivých neuronových sítí (modely/NNxlog.txt), soubory obsahující visualizace architektur včech sítí (visualizace/NNx.png), konfigurační soubory pro theano (.theanorc) a keras (keras.json).

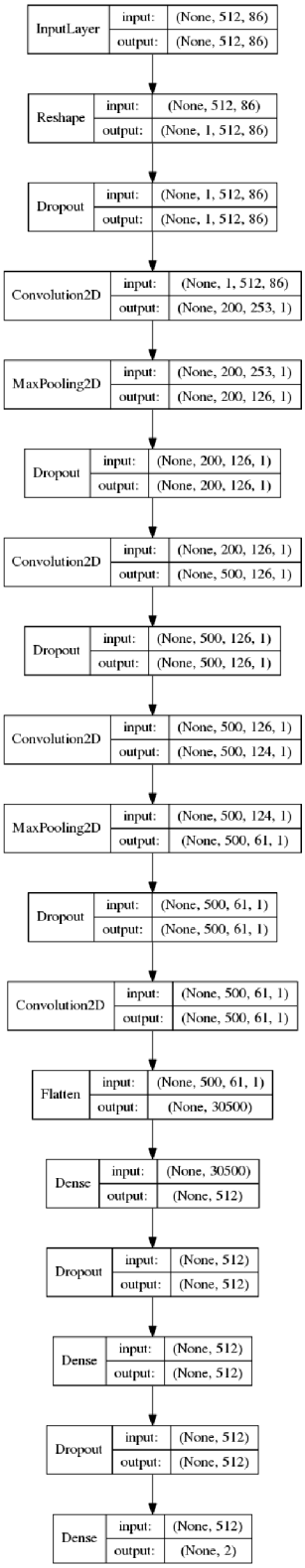
Přílohy



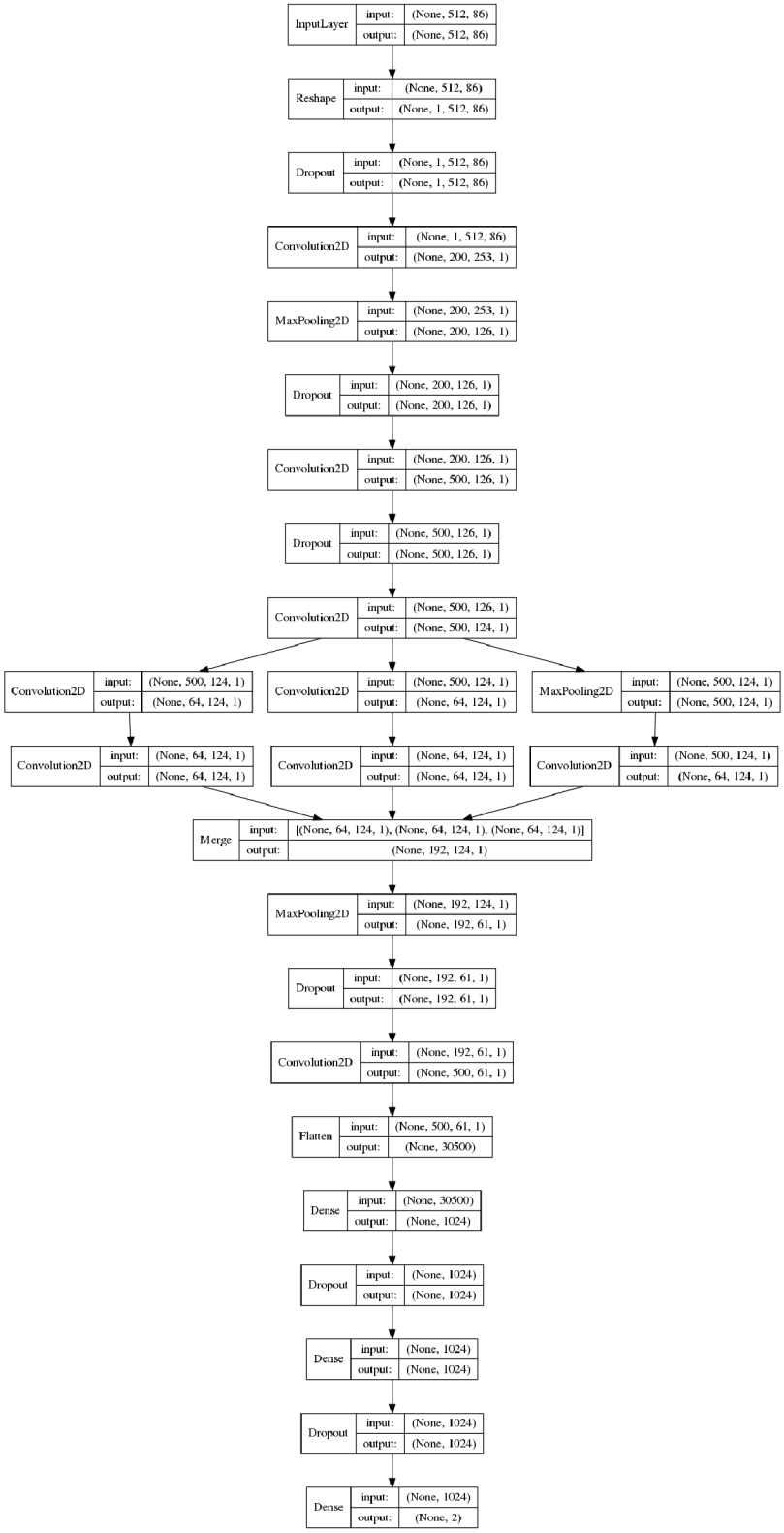
Obrázek 4.1: Architektura sítě NN1



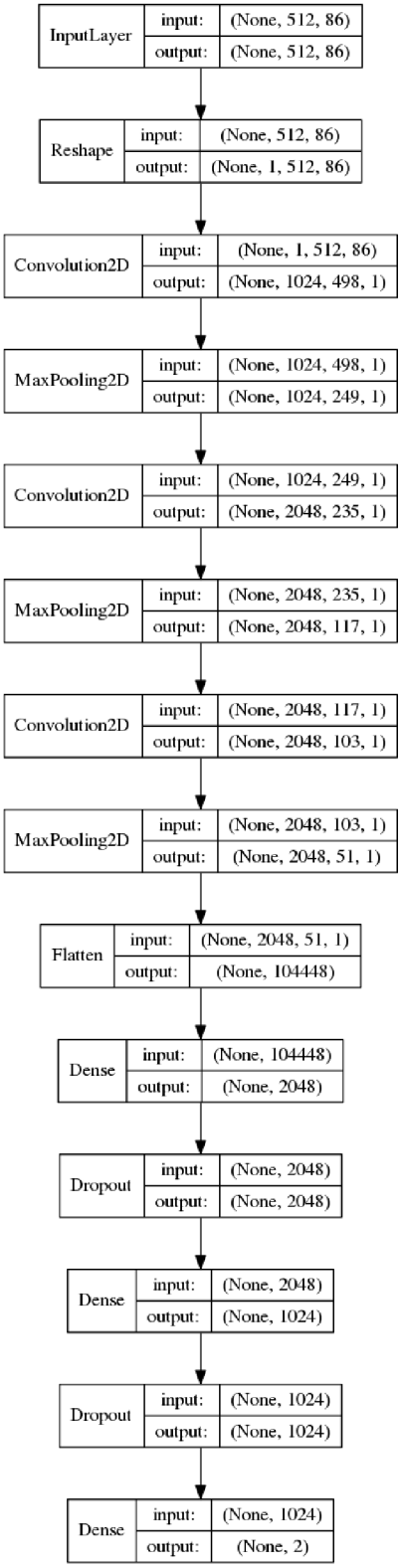
Obrázek 4.2: Architektura sítě NN2



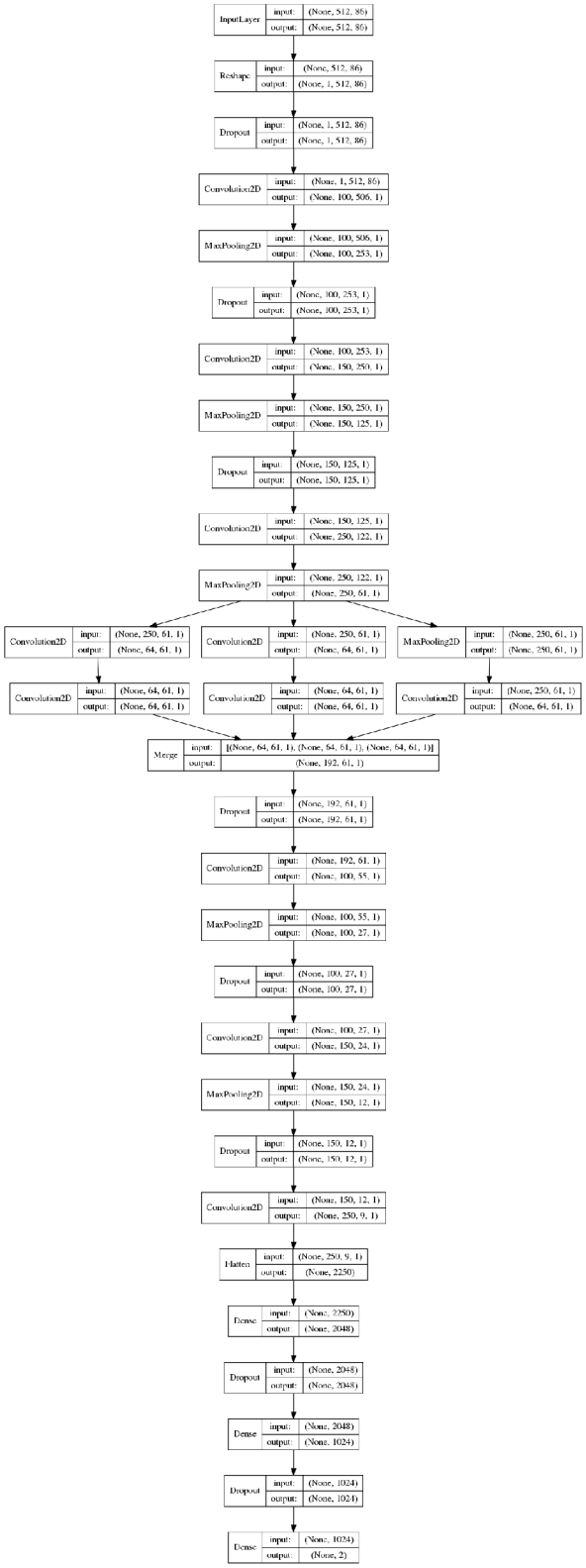
Obrázek 4.3: Architektura sítě NN3



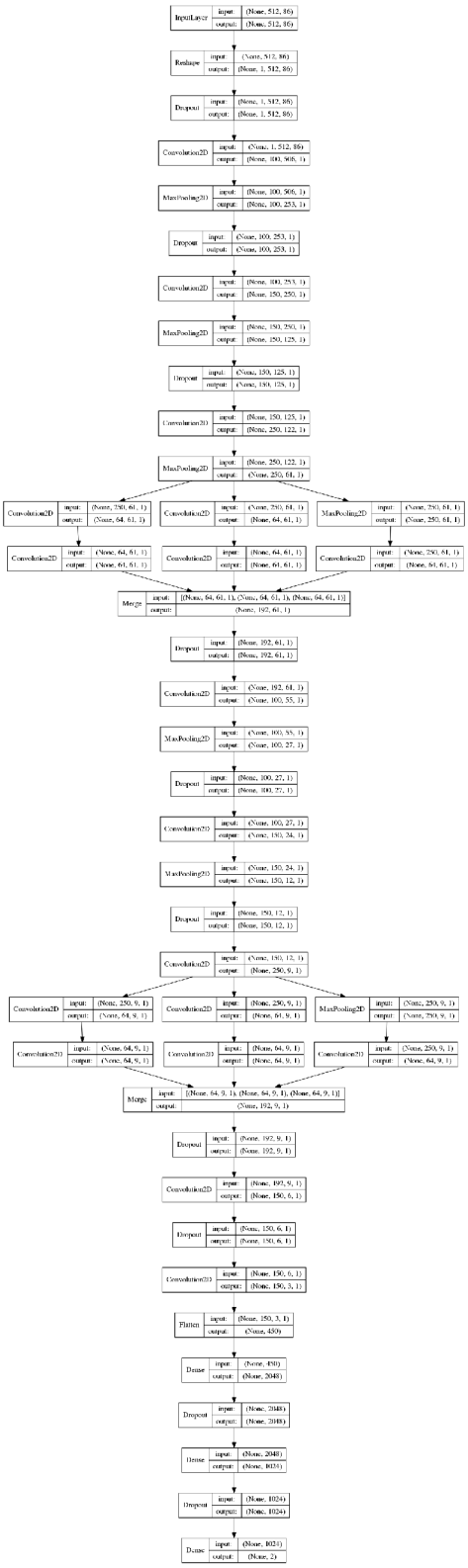
Obrázek 4.4: Architektura sítě NN4



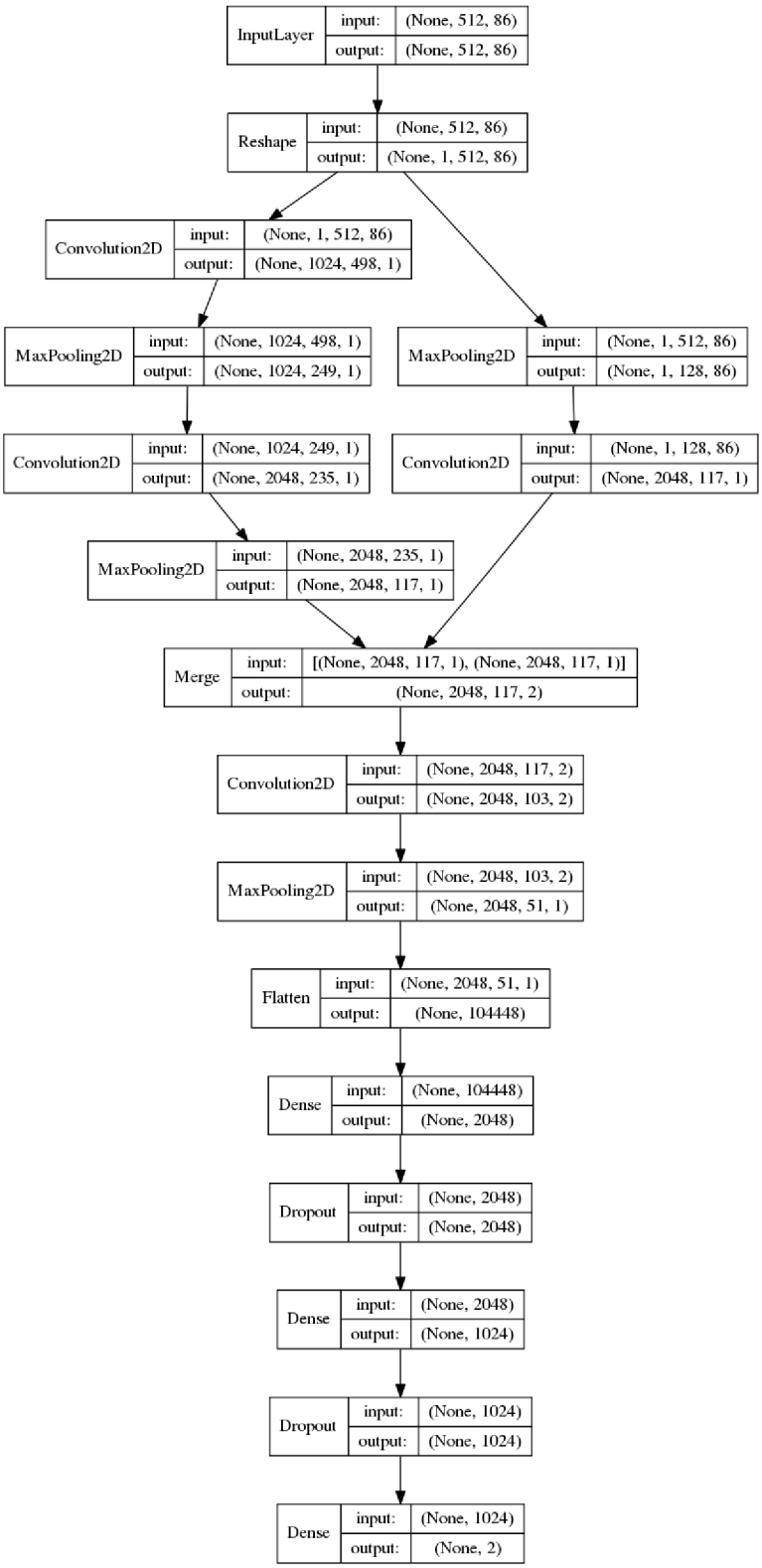
Obrázek 4.5: Architektura sítě NN8



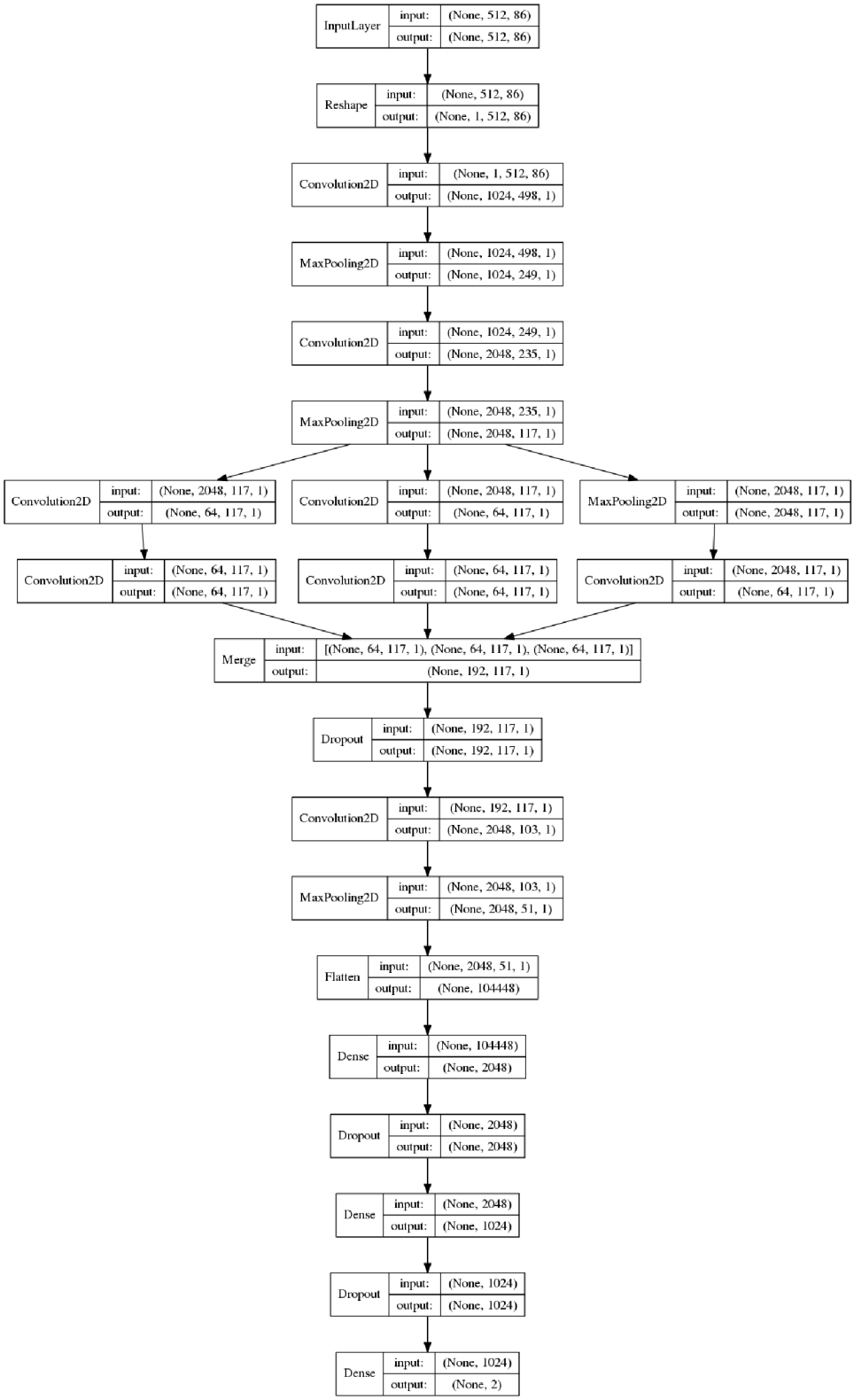
Obrázek 4.6: Architektura sítě NN11



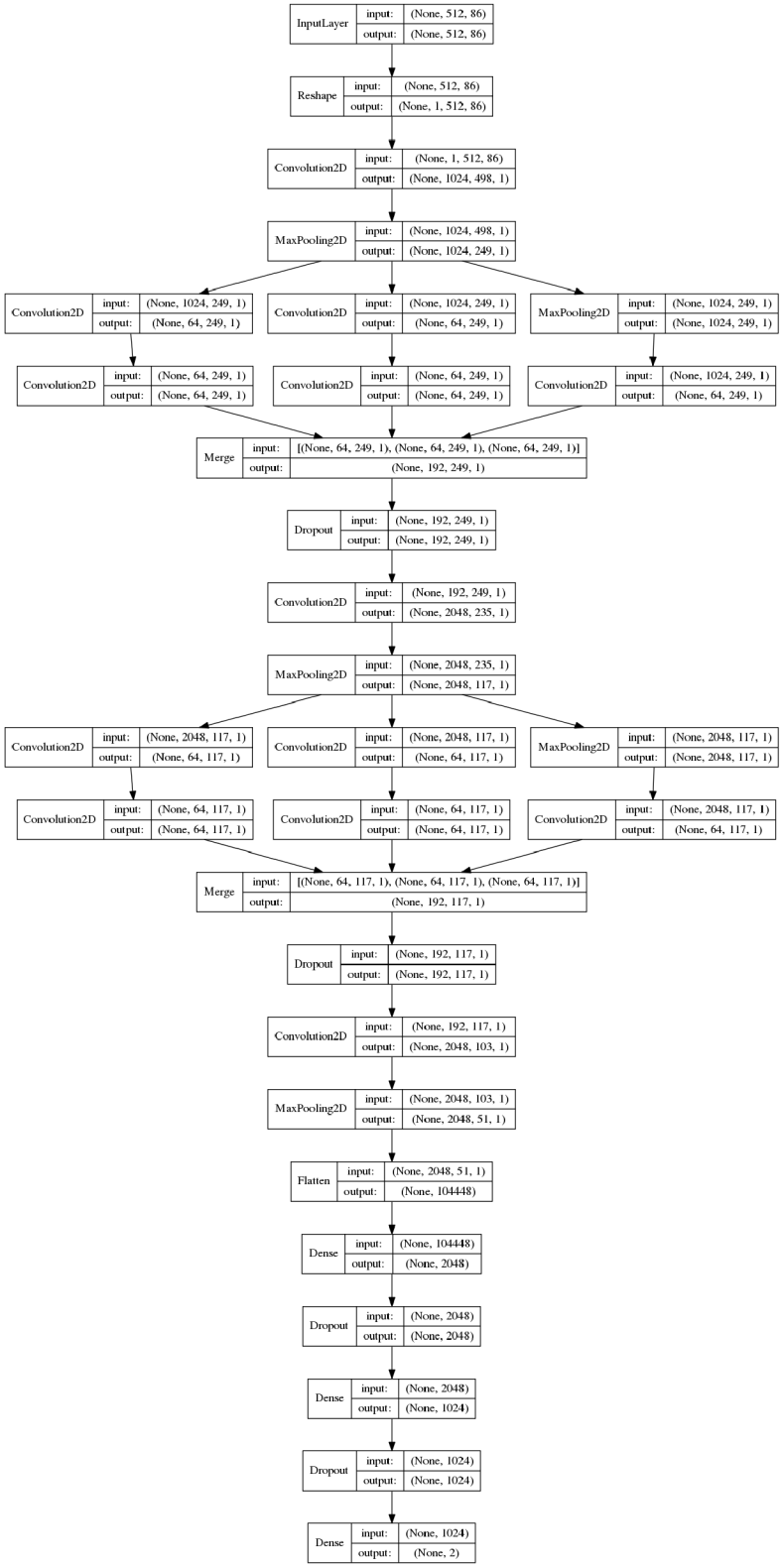
Obrázek 4.7: Architektura sítě NN12



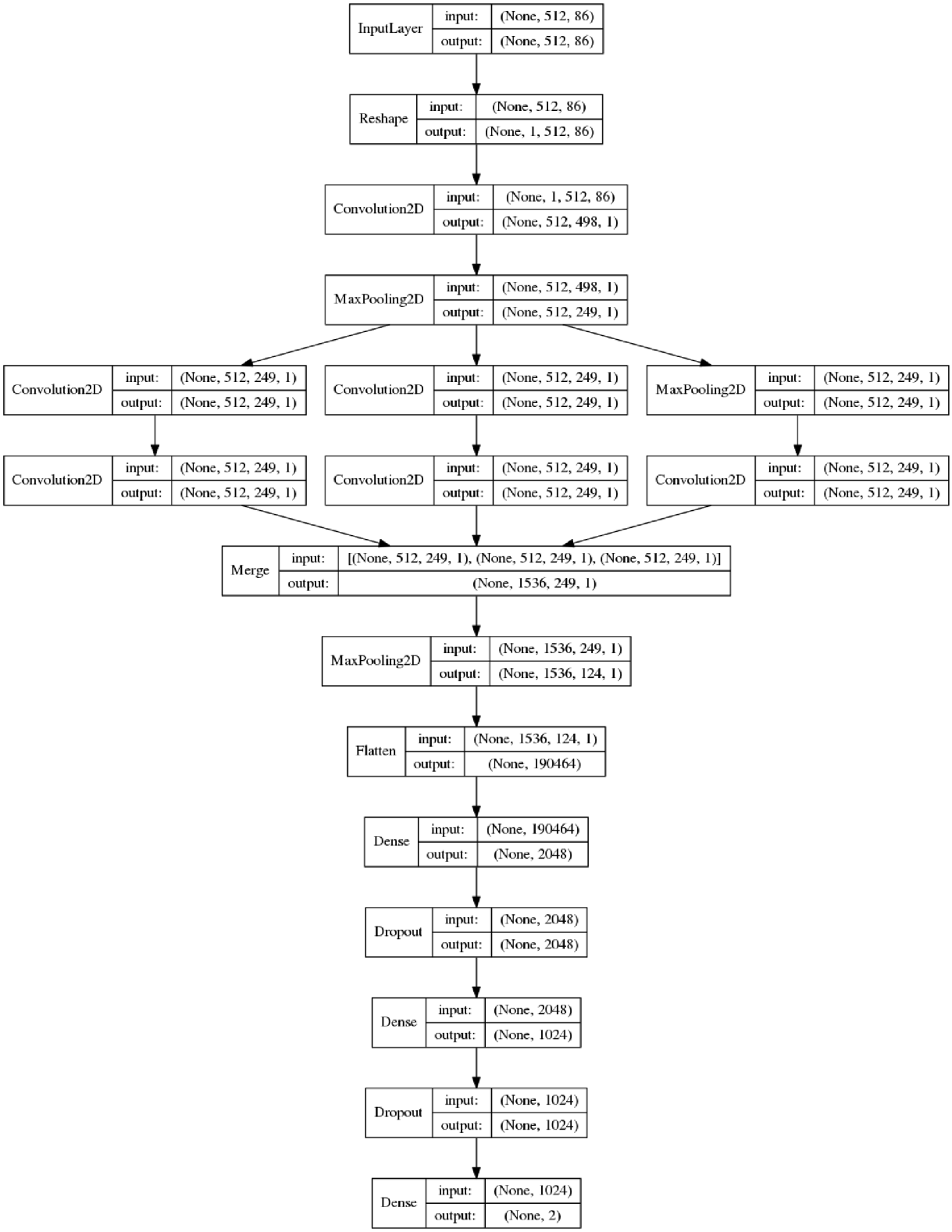
Obrázek 4.8: Architektura sítě NN13



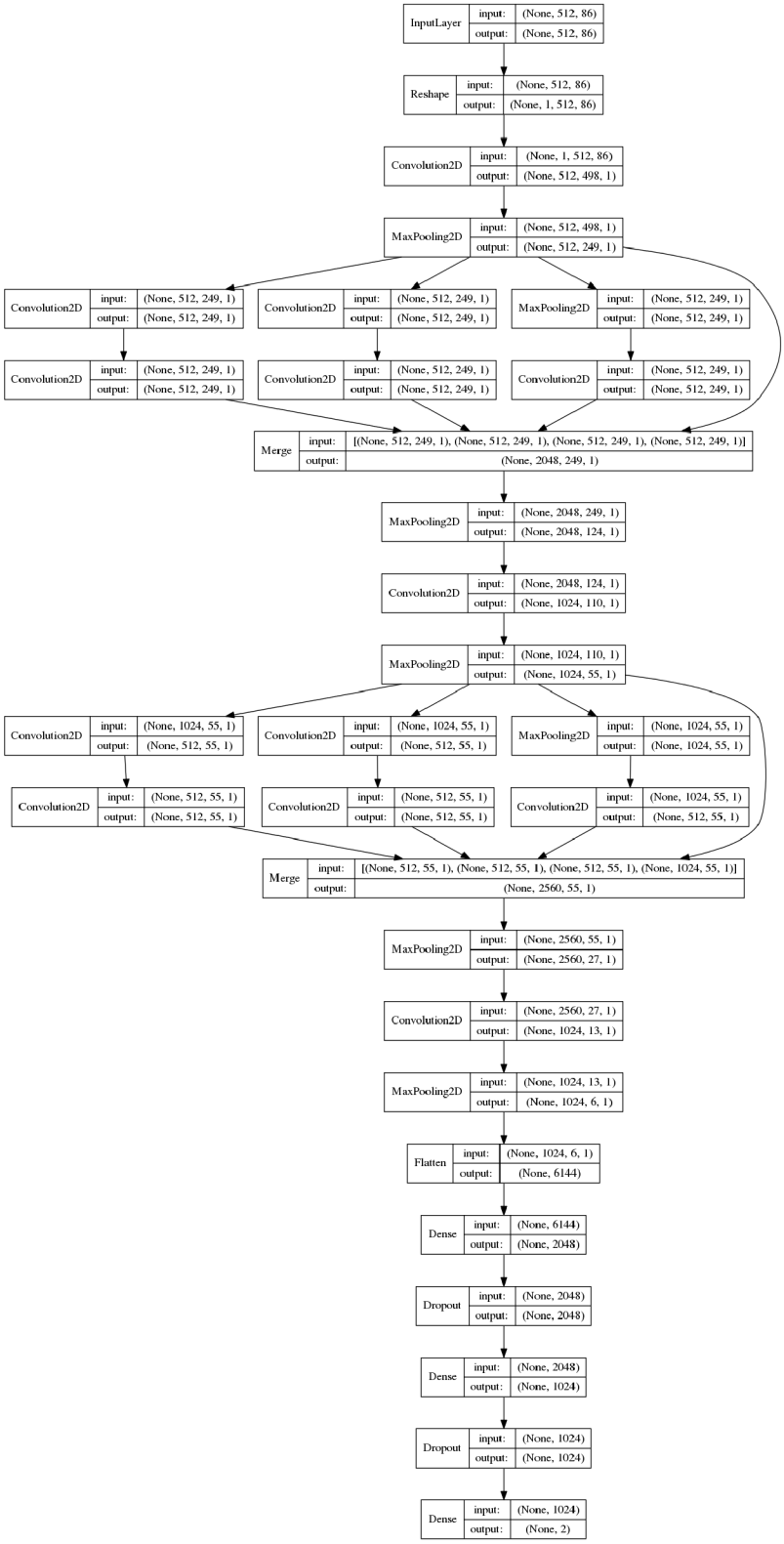
Obrázek 4.9: Architektura sítě NN21



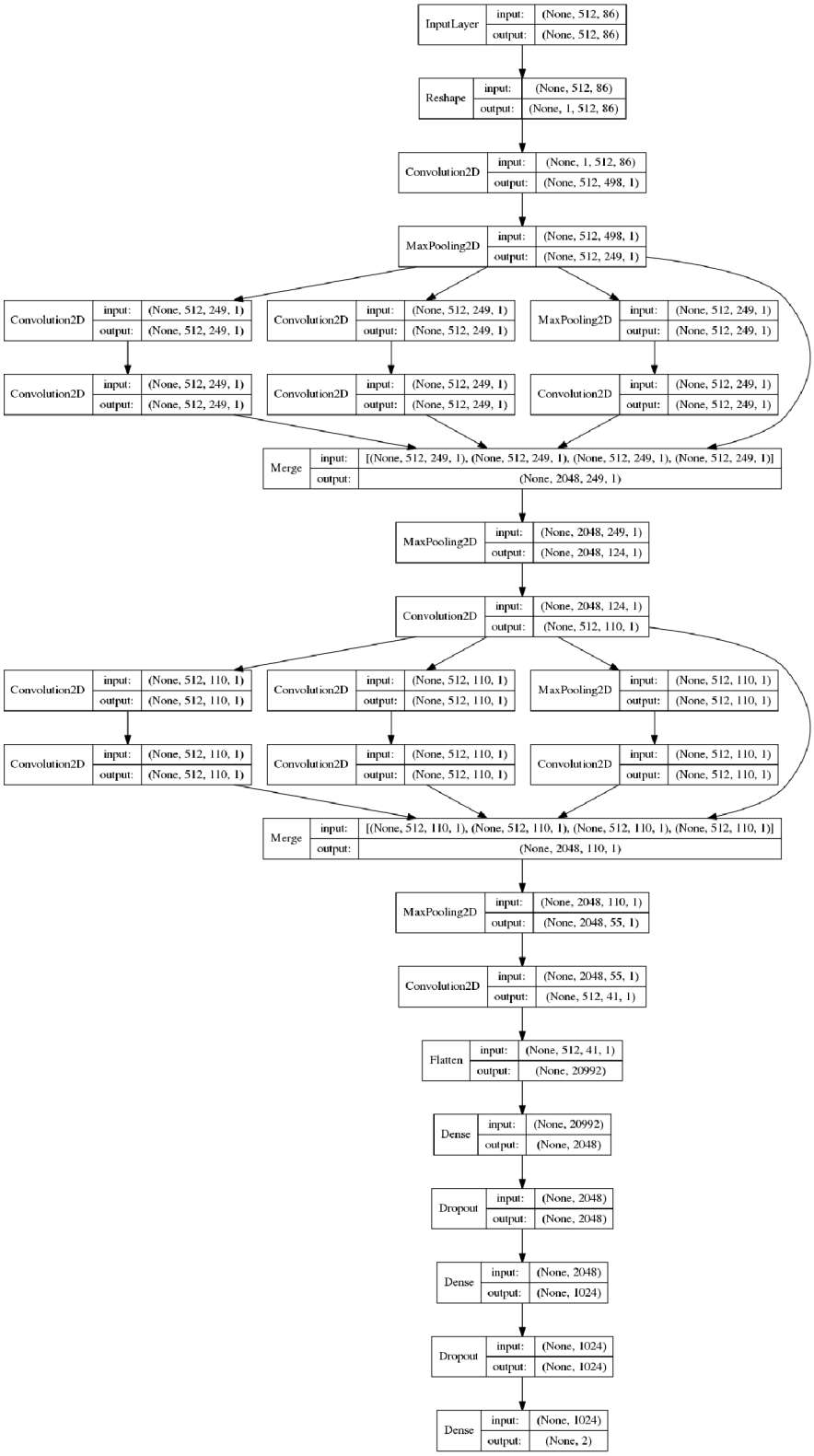
Obrázek 4.10: Architektura sítě NN22



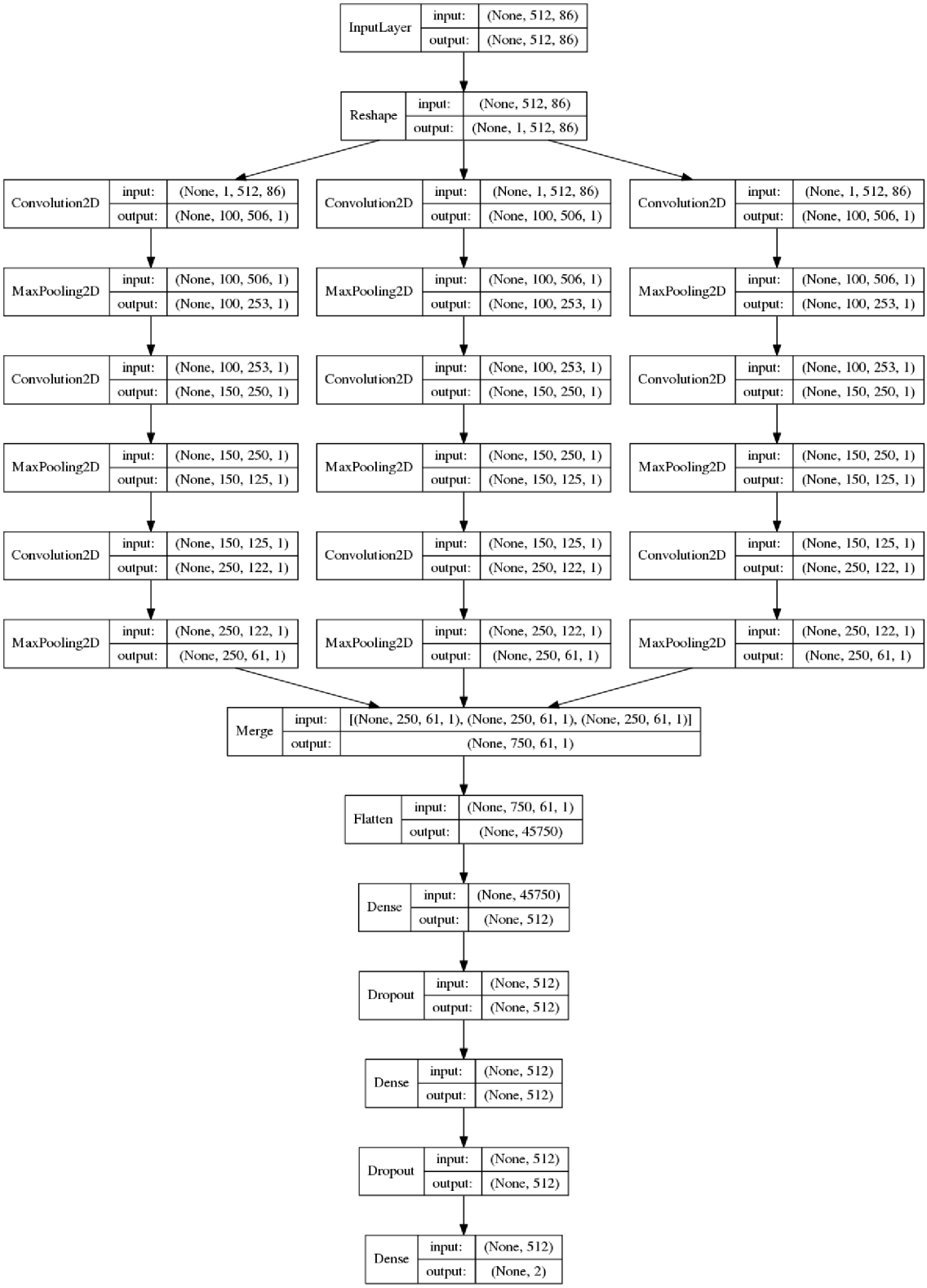
Obrázek 4.11: Architektura sítě NN23



Obrázek 4.12: Architektura sítě NN27



Obrázek 4.13: Architektura sítě NN31



Obrázek 4.14: Architektura sítě NN41