Vysoka škola ekonomická v Praze

Fakulta informatiky a statistiky

Obor: informační a znalostní inženýrství

Diplomová práce

Rozpoznávání znaků z reálných scén pomocí neuronových sítí

Diplomant: **Bc. Petr Fiala**

Vedoucí práce: **Ing. Lukáš Neumann**

Praha 2014

Poděkování

Na tomto místě bych chtěl velmi poděkovat vedoucímu diplomové práce Ing. Lukáši Neumannovi za jeho podporu, mnoho cenných rad a připomínek při vedení diplomové práce. Dále bych rád poděkoval mé mámě za podporu a pomoc s korekturou textu.

Prohlášení

Prohlašuji, že jsem vypracoval samostatně diplomovou práci na téma *Rozpoznávání znaků z reálných scén pomocí neuronových sítí*. Použitou literaturu a další podkladové materiály uvádím v přiloženém seznamu literatury.

V Praze dne 7. května 2014 . . . . . .

podpis diplomanta

Abstrakt

Tato práce se zabývá úlohou rozpoznávání znaků z reálných scén, které je věnována značná pozornost s rozvojem moderních technologií. Cílem studie je k rozpoznávání použít algoritmus, který dosahuje aktuálně nejlepších výsledků na standardních datových sadách. Vybraným modelem je konvoluční síť s *deep* architekturou, jejíž aplikace na zadanou úlohu nebyla dosud publikována. Implementované řešení navazuje na teoretickou část, která poskytuje ucelený přehled dané problematiky. V praktické části se vyskytují dva typy neuronových sítí: vícevrstvý perceptron a zmíněný model. Z porovnání výsledků těchto dvou typů architektur na první datové sadě vychází výrazně lépe použití komplexní struktury konvoluční sítě. Tento model byl dále ověřen na dvou veřejných datových sadách, které korespondují se zadáním úlohy. Zároveň bylo vyzkoušeno několik modifikací sítě a použití různých úprav vstupních dat s cílem získat optimální řešení v závislosti na struktuře dat. Prezentované řešení dokázalo poskytnout srovnatelnou úspěšnost predikce v porovnání s nejlepšími dosaženými výsledky, při použití syntetických učících vzorů a ověřilo možnost využití této architektury pro danou úlohu. V závěru studie jsou zmíněny možné rozšíření a vylepšení modelu, která by mohla vést k dalšímu snížení klasifikační chyby.

Abstract

This thesis focuses on a problem of character recognition from real scenes, which has earned significant amount of attention with the development of modern technology. The aim of the paper is to use an algorithm that has state-of-art performance on standard data sets and apply it for the recognition task. The chosen algorithm is a convolution network with deep structure where the application of the specified model has not yet been published. The implemented solution is built on theoretical parts which are provided in comprehensive overview. Two types of neural network are used in the practical part: a multilayer perceptron and the convolution model. But as the complex structure of the convolution networks gives much better performance compare with the classification error of the MLP on the first data set, only the convolution structure is used in the further experiments. The model is validated on two public data sets that correspond with the specification of the task. In order to obtain an optimal solution based on the data structure several tests had been made on the modificated network and with various adjustments on the input data. Presented solution provided comparable prediction rate compare to the best results of the other studies while using artificially generated learning pattern. In conclusion, the thesis describes possible extensions and improvements of the model, which should lead to the decrease of the classification error.

Obsah

Seznam ilustrací

1 Úvod

Počítačové vidění je díky nástupu moderních technologií jedním z velmi rychle rozvíjejících se oblastí aplikované informatiky. Společně s navýšením výpočetní kapacity zařízení, rozšířením chytrých telefonů a možností pořízení fotky či videa kdykoliv a kdekoliv jsou úlohy z této oblasti aktuální a je jim věnována vysoká pozornost. Problém rozpoznávání znaků z reálných scén je jednou z těchto úloh.

Aplikace řešení je možné naleznout na mnoha místech, nejčastěji se objevují ve spojení s internetovými vyhledávači, které disponují v podstatě neomezeným zdrojem dat, nebo v aplikacích pro mobilní telefony.

Nejvýznamnější skupinou algoritmů, které řeší úlohy počítačového rozpoznávání objektů obecně, jsou algoritmy založené na strojovém učení. V tomto případě je předem vytvořená množina učících vzorů, na kterých se metoda trénuje. Cílem postupu je vytvoření si obecné představy o struktuře dat, aby byla metoda schopna klasifikovat nový, dosud neznámý příklad. K dosažení tohoto cíle je potřeba dvou faktorů: získat obsáhlou a rozmanitou datovou sadu plnou pozitivních a negativních příkladů a dostatečnou výpočetní kapacitu pro možnost naučení metody na obsáhlé množině dat v přiměřeném čase.

Jednou skupinou metod, implementujících strojové učení, jsou neuronové sítě. Tyto metody založené na poznání biologické nervové soustavy, více než ostatní metody, vyžadují naplnění dvou výše zmíněných faktorů, nicméně při zvolení vhodného modelu a poskytnutí dostatečného množství kvalitních trénovacích dat dosahují algoritmy z této skupiny velmi dobrých výsledků. Neuronová síť je složena z neuronů, které jsou skládány do vrstev, mezi nimiž dochází k interakcím a síla předaného impulzu následující vrstvě rozhoduje o výsledku na výstupu sítě. Způsob uspořádání neuronů, vrstev a jejich propojení se odlišuje dle zvoleného modelu sítě.

Po delší dobu byla snaha o vytvoření a naučení struktury, která bude obsahovat větší počet skrytých vrstev, tzv. *deep networks*. Taková síť by byla schopna při rozpoznávání objektů postupného rozeznání základních rysů, které by byly posléze skládány do větších celků. Tento princip navazuje na představy o fungování vizuální nervové soustavy. Nicméně až do roku 2006 se nepodařilo modely správně naučit tak, aby vykazovaly lepší výsledky než sítě *shallow*[1]. V tomto roce byly publikovány 3 studie v čele s prací Hinton a kol. [1], ve které byl prezentován postup, jak takové sítě naučit. Spolu s tímto posunem došlo i k vývoji hardwarových prostředků, hlavně pak programů upravených pro běh na grafických procesorech, které dokáží učení až několika set násobně zrychlit (viz [2]). Od té doby jsou metody s *deep* architekturou nejlepšími modely pro mnoho úloh s komplexním řešením.

Rozpoznávání textu z reálných scén lze rozdělit na 3 pod-úlohy: lokalizace textu v obrázku, jeho extrakce a rozpoznání znaků. Tato studie se zaměřuje pouze na třetí část, ve které právě *deep network* dosahují aktuálně nejlepších výsledků. Speciálně se pak jedná o konvoluční sítě, které jsou výsledkem zjištěných poznatků o fungování zpracovaní informací ze zraku.

Cílem práce je vytvoření modelu neuronové sítě, která bude dosahovat co nejlepších výsledků při aplikaci na rozpoznávání znaků z reálných scén. Zvoleným postupem je nejprve nalezení a implementace modelu s nejnižší klasifikační chybou na standardních datových sadách. Na základě výsledků ze zdrojové studie bude možné výsledky vlastní implementace porovnat. Po dosažení akceptovatelné úrovně predikce na zvolené množině dat, bude model dále modifikován a upravován pro dosažení lepší predikce na příkladech, které korespondují se zadáním práce. Model, který bude

1 Neuronové sítě jsou nazývány *shallow* (mělké), pokud mají maximálně 2 skryté vrstvy.

dosahovat nejlepších výsledků, bude otestován a výsledky budou porovnány s jinými studiemi.

Teoretická část je rozdělena do několika kapitol. Dvěma hlavními oddíly jsou části prezentující poznatky o neuronových sítí a *deep learning*[2]. Část věnovaná neuronovým sítím je zaměřena na model vícevrstvého perceptronu, který má typickou architekturu sítě. Důraz je kladen na popsání struktury této sítě, možnosti změny jednotlivých parametrů a popsání učícího procesu. Sepsané teoretické znalosti jsou nezbytné pro pochopení a schopnost vytvoření sítí s více skrytými vrstvami.

Druhý oddíl teoretické části zabývající se *deep learning* začíná motivační kapitolou, která má za úkol objasnit, k čemu je dobré se zabývat těmito složitými modely. Na to navazuje kapitola o učení této architektury. Jak bylo zmíněno dříve, učení tohoto typu sítí je složitější a učící proces nebo struktura modelu musí být změněna pro úspěšné natrénovaní modelu. Je zde zmíněna důležitost regularizace a představení některých jejich technik. Na závěr jsou popsány dva nejběžnější modely s mnoha vrstvou architekturou: *Deep belief network* a konvoluční sítě.

Praktická část je rozdělena do třech oddílů, dle datové sady, na kterých jsou testy prováděny. První oddíl je zaměřen na vytvoření a implementaci vhodného modelu pro datovou sady MNIST. Tato sada je považována za jednu ze základních sad pro rozpoznávání. Studie, jenž na ní dosáhla nejlepších výsledků [2] je vodítkem pro vytvoření základního modelu této práce. Cílem této části je dosáhnout podobných výsledků, jako bylo dosaženo ve zmiňované studii, a ověřit tak funkčnost vytvořeného modelu.

Ve druhém oddílu je použita datová sada ICDAR 2011, která se skládá z reálných obrázků s textem. Cílem této části je nelézt co nejlepší klasifikační model pro danou množinu číslic a písmen. Základní model je vytvořen na základech získaných znalostí z předchozích testů a je porovnáván s různými modifikacemi.

V poslední části se nacházejí testy provedené na datové sadě ICDAR 2003. V této části je cílem získání procentuální úspěšnosti modelu, který byl vybrán na základě předchozích testů jako nejúspěšnější. Výsledná úspěšnost klasifikace bude porovnána s výsledky ostatních modelů.

2 Oblast zabývající se učením sítí s *deep* architekturou.

2 Strojové učení

Strojové učení je kombinace několika disciplín: statistiky, teorie informace, teorií algoritmů, pravděpodobnosti a funkční analýzy [3], jejímž hlavním cílem je optimalizace, jak ji definoval Vapnik v [4]. Odpovídá na otázku, jak vytvořit počítačový program, který se dokáže zlepšit na základě zkušeností. Strojové učení se prokázalo jako velmi vhodné řešení pro úlohy z některých aplikačních oblastí. Jedná se například o oblast *data miningu*[3], o úlohy, ve kterých člověk nedokáže nalézt efektivní algoritmus[4] a kde je potřeba, aby se program přizpůsobil změnám prostředí.

Mitchell definuje v [5] strojové učení jako:

*O počítačovém programu lze tvrdit, že se **učí** ze zkušeností Z ve vztahu k úloze U měřeno výkonností V, pokud se zlepší jeho výkon v úloze U měřeno V při použití zkušeností Z.*

Pro zadanou úlohu této práce se za zkušenosti považuje množina trénovacích dat. Výkonnost implementovaného programu je měřena pomocí množiny testovacích dat, kde výstup programu je porovnán s požadovanou hodnotou. Odchylka je považována za chybu a cílem optimalizace, učení je její minimalizace.

3 Neuronové sítě

3.1 Úvod do neuronových sítí

Ve světě výpočetních technologií jsou umělé neuronové sítě matematickými modely, které se nechaly inspirovat fungováním a strukturou nervové soustavy. Hlavní myšlenka je založena na jednotce neuronu, který je schopen přijímat impulzy z blízkého okolí, složit je do výsledného vzruchu a ten pak předat následujícímu neuronu. Celý systém nervové soustavy tak umožňuje přijímat podměty a zajišťovat odpovídající reakce.

Umělé neuronové sítě (ANN) tento model přejímají, i když jen ve velmi omezené kapacitě v porovnání s biologickou soustavou, a na základě vstupu propočítávají výstup. Jedná se o množinu neuronů, které jsou orientovaně propojené. Celá síť se tak skládá z vrstev neuronů, umělých neuronů a jejich propojení. Síť lze definovat jako orientovaný graf s dynamicky ohodnocenými vrcholy a hranami, uspořádanou pětici [*V, E, ε, y, w*], kde je:

- V množina vrcholů (neuronů)
- E množina hran (synapsí)
- ε zobrazení incidence hran s vrcholy ($\epsilon : E \rightarrow V \times V$)
- y dynamické ohodnocení vrcholů ($y : V \times \boldsymbol{t} \rightarrow \mathbb{R}$
- w dynamické ohodnocení hran ($w : \epsilon(E) \times \boldsymbol{T} \rightarrow \mathbb{R}$)

[6]

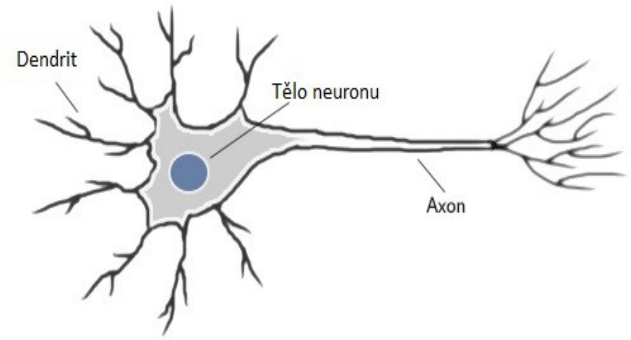
3.2 Biologická inspirace a historie

Neuron (nervová buňka) je speciální biologickou buňkou, která předává informace (viz Obr. 1). Je složena z těla, nebo též soma, a dvou typů spojení s okolím: axon a dendrity. Buněčné tělo obsahuje jádro a plasmu pro vytváření nezbytných substancí [7]. Neuron dostává signály od okolních neuronů přes dendritová vlákna (receptory) a vysílá impulzy z těla neuronu pomocí axonu přes terminální zakončení. Těmto zakončením se říká synapse, které navzájem spojují axon vysílajícího

3 „Dolování dat“: Získávání netriviálních, předem neznámých znalostí z dat.
4 Úlohy s potřebou generalizování získaných znalostí (rozpoznávání obrázků, textu, zvuku, atd.).

neuronu a dendrit příjemce. Pokud má dojít k přenosu vzruchu přes synapse je uvolněna některá chemická látka ze skupiny neurotransmiterů, která přenese signál přes mezeru mezi synapsemi. Synaptická spojení fungují u přijímacího neuronu jako místo příjmu informace. Tyto spojení jsou schopná se adaptovat na probíhající informace a tím se učit znalostem na základě přenosů, které jsou reprezentovány sílou spojení.

Soma přijímá signály z dendritů a dále je zpracovává: agregace, prahování a nelineární transformace. Pokud potenciál neuronu (agregace vstupu) přesáhne práh, pak je aktivován a vysílá výstupní signál.



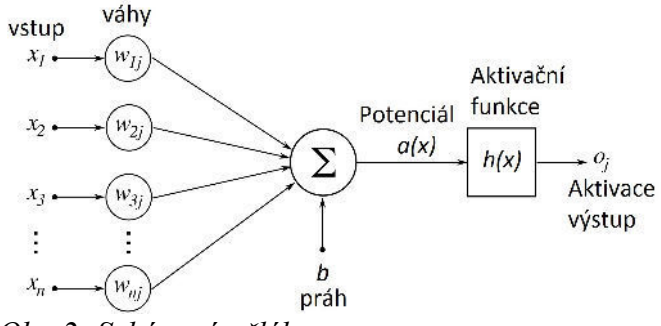
Obr. 1: Schéma biologického neuronu [66]

Kůra mozková obsahuje přibližně 10^{11} neuronů [8], které jsou vzájemně velmi hustě propojené. Každý neuron je spojen s něco přes 10^3 dalšími uzly a dohromady vytvářejí síť obsahující přibližně 10^{14} až 10^{18} propojení.

Neurony komunikují vysíláním sekvence impulzů, které můžou nabývat frekvence okolo stovek Hertz. Ovšem k některým reakcím dochází téměř okamžitě v několika milisekundách. Zároveň předávaný signál je velmi malý, jen pár bitů. To implikuje, že informace je rozdělena na více kratších úseků, distribuována přes více cest, aby byla na konci opět složena a vyhodnocena v co nejkratším čase [9].

3.3 Neuron

Základním stavebním kamenem ANN je neuron, který se vyskytuje v mnoha různých podobách, nejčastěji však jak ho popsali McCulloch a Pitt v [10]. Struktura umělého neuronu vychází ze skutečností zjištěných o neuronu biologickém. Skládá se z vazeb na předešlou vrstvu (dendrity), prahu, aktivační funkce (tělo neuronu) a vazeb na vrstvu následující (axon) (znázorněno na Obr. 2).



Obr. 2: Schéma umělého neuronu.

Cílem každé početní jednotky sítě je na základě vstupu vypočítat hodnotu aktivace na výstupu. Informace proudí do těla neuronu přes dynamicky ohodnocené hrany (dendrity), kde každá vstupní hodnota je násobena individuální váhou spojení (váha synaptického spojení). Vstupy neuronu se sečtou (agregace) společně s prahem (prahování). Pro výsledek sumace je posléze vypočtena

funkční hodnota (nelineární transformace), která je šířena do následující vrstvy.

Potenciál i-tého neuronu se spočítá jako

$$a(\boldsymbol{x}) = b + \sum_i w_i x_i = b + \boldsymbol{w}^T \boldsymbol{x} \tag{1}$$

a aktivace (výstup) neuronu

$$h(\boldsymbol{x}) = g(a(\boldsymbol{x})) = g\left(b + \sum_i w_i x_i\right) \tag{2}$$

, kde

x je vstup, vektor, do neuronu,

w značí, vektor, váhy spojení,

b je bias (práh aktivace neuronu),

g(.) je aktivační funkce.

Neuron je zajímavý svojí kapacitou, tzn. komplexitou výpočtu, kterou může provést. Pomocí aktivační funkce dochází k určení pravděpodobnosti příslušnosti případu k jedné ze dvou kategorii, tento postup se nazýváme *logistickou regresí.* Výsledkem tak je reálná hodnota z intervalu [-1,1] (nebo [0,1]). Někdy se též používají pouze jednoduché binární neurony, které mohou nabývat hodnot {-1, 1} (nebo {0,1}) [11]. Vše záleží na zvolené aktivační funkci.

V případě sigmoidy a dvou-rozměrného vstupu (viz Obr. 4) může aktivační hodnota nabývat libovolné hodnoty z otevřeného intervalu (0,1). Natočení spojité hrany mezi funkční hodnotou 0 a 1 je způsobeno vahami vstupních hran. Zvolený bias, neboli práh, zase hranu posouvá směrem do záporných či kladných hodnot. Důležité je ovšem poznamenat, že jeden neuron má rozlišovací schopnost velmi omezenou a stačí pouze na jednoduché příklady, jako je namodelování logické konjunkce či disjunkce, tedy příklady, které jsou lineárně separovatelné[5]. Pro složitější operace, jako je exkluzivní disjunkce (XOR), je potřeba neurony skládat do vrstev a sítí.

3.4 Aktivační funkce

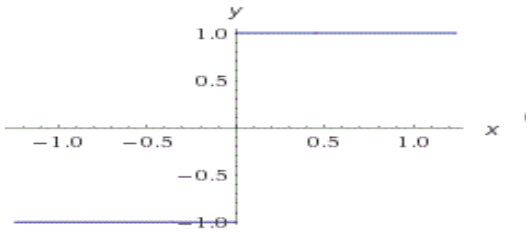
Jak je vidět z rovnice (2) parametrem, který nejvíce ovlivňuje chování neuronu je aktivační funkce, nazývaná též funkcí přenosovou. Zvolit je možné jakoukoliv matematickou funkci, nicméně používání aktivační funkce navazuje na nelineární transformace uvnitř jádra neuronu a proto se nejčastěji používají funkce nelineární.

5 Říkáme, že množina bodů je lineárně separovatelná, pokud existuje přímka (nadrovina v prostoru s více jak 2 dimenzemi), která rozdělí prostor tak, že body jedné třídy budou na jedné straně a body druhé třídy budou na straně druhé.

3.4.1 *Krokové*

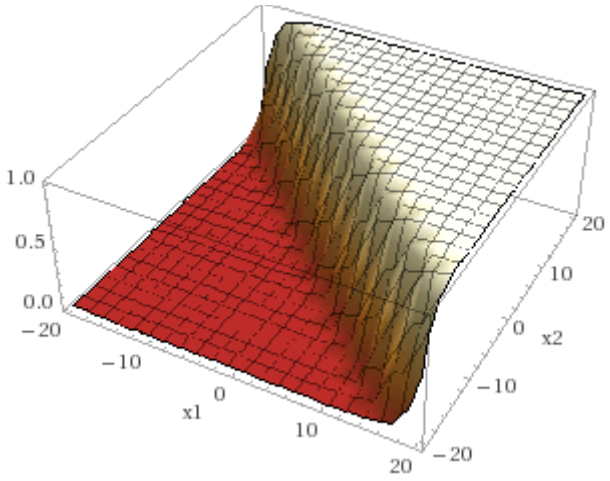
Jeden z prvních učících se modelů, popsán v [12] , použil funkci ***signum*** (sign). Ta přiřadí libovolnému číslu jednu z hodnot {1, -1, 0}, podle toho zda je číslo > 0, < 0, = 0. *Sign* popisuje klasifikační problém následují rovnicí:

$$h(\boldsymbol{x}) = \begin{matrix} 1 \; pro \; \boldsymbol{w}^T \boldsymbol{x} \geq b \\ 0 \; pro \; \boldsymbol{w}^T \boldsymbol{x} < b \end{matrix} \tag{3}$$



Obr. 3: Graf funkce signum

Signum se používá hlavně při binární klasifikaci, protože přesně definuje, kdy případ do dané kategorie patří či nikoliv. Použitím více takovýchto neuronů může být detekováno více rysů najednou a řešit tak velmi komplikované shlukovací a klasifikační úlohy.

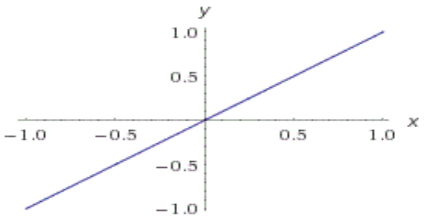


Obr. 4: 3D graf funkce sigmoid

3.4.2 Lineární

Identita, ***lineární*** funkce, je další, která se používá. Nedochází zde k žádné transformaci vstupu. Ta samá hodnota je poslána na výstup. Nemá proto žádnou horní a dolní mez a nezajišťuje ne-linearitu, která je důležitou vlastností pro řešení složitějších úloh.

$$g(a) = a \tag{4}$$



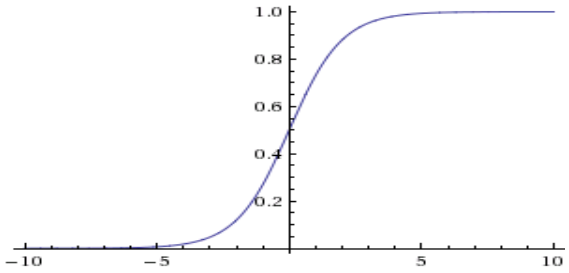
Obr. 5: Graf lineární funkce

3.4.3 *Logistické*

Nejužívanější skupinou aktivačních funkcí jsou funkce logistické, mezi které patří ***sigmoida*** a ***hyperbolický tangens***. Mají několik vlastností, které jsou při řešení úloh neuronovými sítěmi zásadní. Funkční hodnoty jsou zdola i z hora omezené (0, 1), resp. (-1, 1) a vstup je „vmáčknut“ do příslušného oboru hodnot. Lze tak přímo vyjádřit pravděpodobnost příslušnosti do dané třídy. Zároveň se jedná o funkce striktně rostoucí.

Sigmoid:

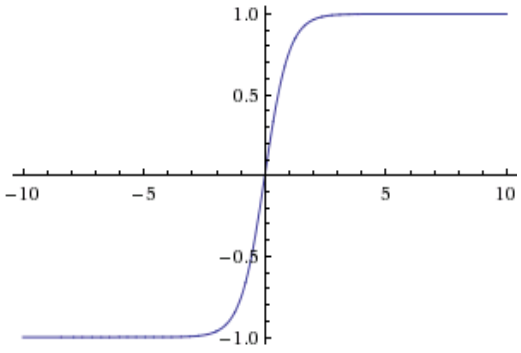
$$g(a)=sigm(a)=\frac{1}{1+\exp(-a)} \tag{5}$$



Obr. 6: Graf funkce sigmoid

Hyperbolický tangent:

$$g(a)=\tanh(a)=\frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)}=\frac{\exp(2a)-1}{\exp(2a)+1} \tag{6}$$



Obr. 7: Graf funkce tanh

Na výstupní vrstvě sítě se používá ještě logistická funkce ***softmax***. Je to z toho důvodu, že dokáže

interpretovat výstupy sítě jako posteriorní pravděpodobnosti rozpoznávaných tříd. Jinak řečeno, výstupem sítě jsou hodnoty z intervalu (0,1), které určují pravděpodobnosti příslušnosti k dané třídě [13]. Pokud se jedná pouze o binární klasifikaci, stačí na výstupní vrstvě použití funkce sigmoid, jejímž výstupem je přímo pravděpodobnost příslušnosti případu do jedné třídy, tzn. $P(y=1|\boldsymbol{x})$.

Nicméně v případě, že je více tříd, více neuronů ve výstupní vrstvě, je nutné určit pravděpodobnost pro každou třídu zvlášť. Hledanou hodnotou je pak podmíněná pravděpodobnost $P(y=c|\boldsymbol{x})$, která vyjadřuje pravděpodobnost, že případ x patří do třídy c, která je shodná s požadovanou klasifikací. K tomuto určení se používá funkce ***softmax*** na výstupní vrstvě:

$$\boldsymbol{o}(\boldsymbol{a})=softmax(\boldsymbol{a})=[\frac{\exp(a1)}{\sum_c \exp(a_c)}\dots\frac{\exp(a_c)}{\sum_c \exp(a_c)}]^T \tag{7}$$

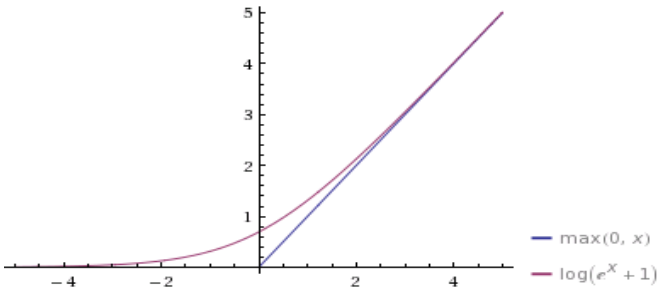
Funkce je striktně rostoucí a sumace aktivací případu přes všechny třídy c se rovná jedné. Výsledkem je předpověď příslušnosti k třídě, která má nejvyšší pravděpodobnost. V neposlední řadě je výhodou též snadná interpretace výsledků.

3.4.4 Usměrňovací funkce

Usměrňovací (rectified) lineární funkce (dále jen *relu*) se v mnoha různých podobách začala vyskytovat jako další typ aktivace. ***Relu*** má výhodu oproti binomickému určení příslušnosti, například pomocí *sign*, že předává část informace dál. Na druhou stranu je oproti logistickým funkcím, kde téměř pokaždé dochází k aktivaci neuronu (pouze *tanh(0) = 0*), aktivace při použití *relu* více řídká (pouze pro *x>0*). Této vlastnosti se vyžívá v rámci *deep learning* [14].

Jedná se o funkci zdola omezenou, nikoliv však zhora a je neklesající. Používá se ve více variantách, nejčastěji však jako:

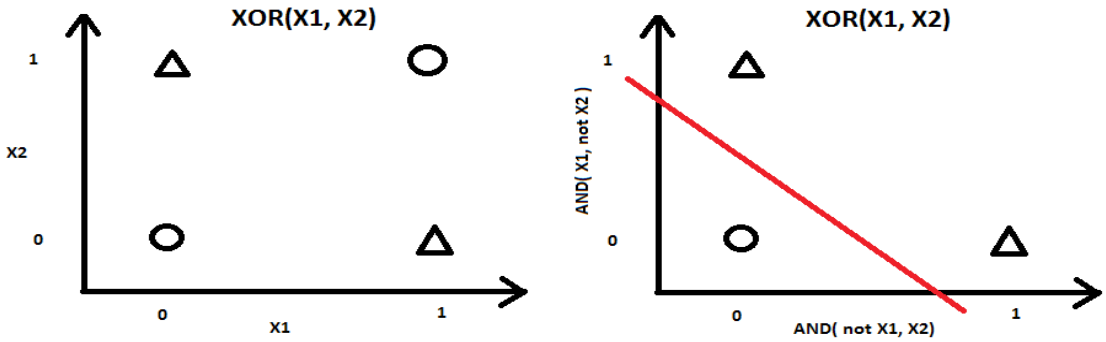
$$g(a)=max(0,a) \tag{8}$$



Obr. 8: Graf funkce relu

3.5 Neuronová síť

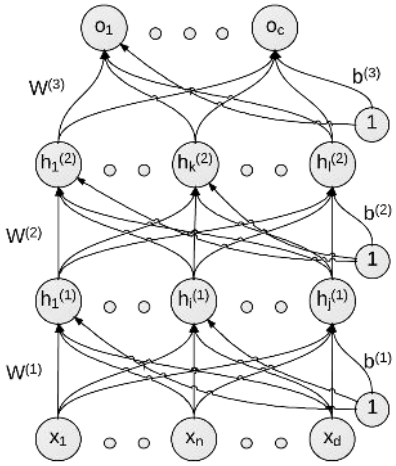
Jak bylo zmíněno dříve, klasifikační schopnost jednoho neuronu je značně omezená. Proto se skládají do vrstev, které tvoří celou síť.



Obr. 9: Graf klasifikační schopnosti ANN

Na obrázku vlevo je znázorněna logická operace XOR, která není lineárně separovatelná. Ovšem pokud se podaří vstup do neuronu modifikovat předchozí vrstvou[6] tak, jak je to znázorněno na schématu vpravo, není již problém třídy oddělit. Toto je motivace pro vytváření vícevrstvých sítí, které dokáží řešit složitější úlohy.

Sítě se rozdělují do několika skupin dle typologie. Tato práce se výhradně zabývá dopřednými sítěmi[7] (*feed-forward*, FNN), které odpovídají jedno, či vícevrstvému perceptronu (*Multi-layer perceptron, MLP*). Jejich graf je acyklický a informace plyne pouze jedním směrem od vstupní k výstupní vrstvě [15] v průběhu aktivní dynamiky[8] (viz Obr. 10).



Obr. 10: Schéma vícevrstvého perceptonu se dvěma skrytými vrstvami.

Základní struktura modelu je poskládaná ze tří částí. První vrstva je vstupní. Její počet jednotek se rovná vždy dimenzi informace na vstupu. Nedochází zde k žádné aktivaci, data můžou být pouze normalizovaná či jinak upravena.

Druhá část, která se skládá ze skrytých vrstev, není povinnou součástí modelu. Příkladem sítě, která nemá skrytou vrstvu, může být lineární asociativní paměť [6]. V ostatních případech se v modelu nachází jedna až několik skrytých vrstev, kdy každá z nich přijme hodnoty ze vstupu, transformuje

6 Konjunkce je lineárně separovatelná i v případě, že obsahuje negaci.
7 Pokud nebude v dokumentu řečeno jinak, „neuronovou sítí“, nebo jen „sítí“, je vždy myšlena síť dopředná (FNN).
8 Rozlišením dvou časů *aktivní* a *adaptivní dynamiky* je vyjádřena skutečnost, že neuronová síť pracuje ve dvou časově nezávislých režimech. V adaptivním čase dochází k učení sítě. V aktivním režimu probíhá realizace sítě naučené v adaptivní dynamice [6].

je a pošle na výstupní vrstvu. Každá z vrstev se skládá z libovolného počtu neuronů, jejichž vzájemné vazby určují charakter sítě.

Poslední částí je vrstva výstupní. Ta předává předpověď modelu uživateli. Použitá aktivační funkce v této vrstvě kopíruje potřebu tvaru poskytovaných informací.

Síť s jednou skrytou vrstvou lze matematicky vyjádřit jako:

potenciál skryté vrstvy:

$$a(\boldsymbol{x}) = \boldsymbol{b}^{(1)} + \boldsymbol{W}^{(1)}\boldsymbol{x} \quad , \quad a(\boldsymbol{x})_i = b_i^{(1)} + \sum_j W_{i,j}^{(1)} x_j \tag{9}$$

aktivace skryté vrstvy:

$$h(x) = g(a(x)) \tag{10}$$

aktivace výstupní vrstvy:

$$f(\boldsymbol{x}) = o(b^{(2)} + \boldsymbol{w}^{(2)T}\boldsymbol{h}^{(1)}\boldsymbol{x}) \tag{11}$$

, kde *o* značí aktivační výstupní funkci.

Univerzální aproximační věta, jak ji definoval Hornik v [16], říká, že neuronová síť s

- jednou skrytou vrstvou,
- libovolnou omezenou, nekonstantní aktivační funkcí
- a lineárním výstupním neuronem

je univerzálním aproximátorem libovolné spojité funkce do požadované přesnosti. A to pouze za podmínky, že obsahuje dostatečné množství neuronů ve skryté vrstvě.

Teorém zajišťuje existenci řešení pro velké množství úloh, nicméně neříká nic o tom, jestli existuje učící algoritmus, který by našel takové parametry modelu pro požadovanou přesnost aproximace.

4 Učení sítě

Existují základní metody učení: učení s učitelem, bez učitele a jejich kombinace. Používají se dle toho, zda jsou k dispozici požadované výstupy, kategorie zařazení, u jednotlivých prvků trénovacích dat. Pro každý z těchto učících procesů existuje mnoho trénovacích algoritmů.

4.1 Učení s učitelem

Cílem učení s učitelem je nastavit parametry sítě tak, aby správně predikovala. V případě klasifikace do předem definovaných tříd, tzn. aby mapovala $\boldsymbol{X}$ do $\boldsymbol{y}$, při dané trénovací množině prvků ($\boldsymbol{x}_i, y_i$). $\boldsymbol{y}$ Zde reprezentuje možné třídy zařazení, též nazýváno cílem, štítkem informace ze sady $\boldsymbol{X}$. Výsledkem by měla být shoda výstupu modelu s přiřazenou správnou hodnotou.

Pokud by se $\boldsymbol{y} = \mathbb{R}$ nebo $\boldsymbol{y} = \mathbb{R}^n$, pak se jedná o regresní úlohu a cílem je co nejvíce přiblížit předpověď k požadované hodnotě. Úlohy s učitelem jsou velmi dobře definovatelné, protože při

mapování lze přesně určit úspěšnost předpovědí [17], kapitola 4.5.

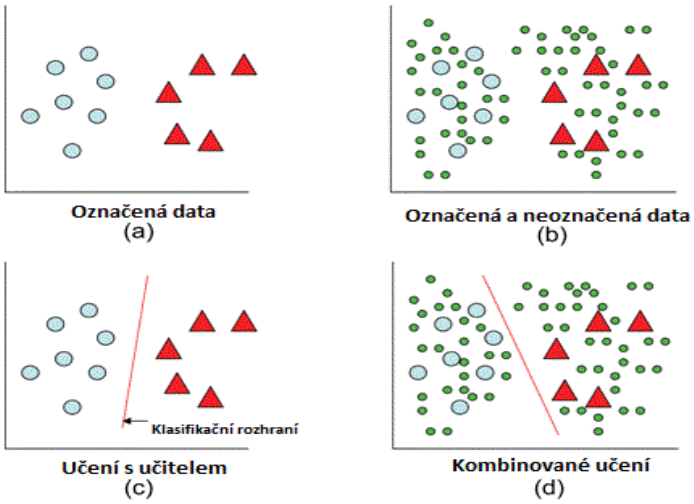
4.2 Učení bez učitele

Učení bez učitele se liší v tom, že spolu s daty nejsou poskytnuty požadované výstupy. Trénovací informace se tak skládají pouze ze vstupní množiny dat $\boldsymbol{X}=(\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_n)$ o velikosti n. Cílem je nalezení charakteristických struktur v datech [17]. Je tedy čistě na modelu, aby rozřadil prvky dle společných rysů do skupin. Rozdílem oproti učení s učitelem je, že v tomto případě jsou třídy utvářené samotnými případy, který byly do skupiny přiřazeny a vytvářejí charakteristické rysy těchto tříd.

Hlavními případy užívání učení bez učitele je ve shlukovacích úlohách a úlohách na snížení dimensionality [18].

4.3 Kombinované učení

Tento druh učení kombinuje obě zmíněné formy. Metoda má k dispozici nejen neoznačená data, bez požadovaného výsledku, ale i s. Vstupní informace se tak rozdělují na dvě podskupiny: část, pro kterou jsou štítky dodány $\boldsymbol{X}=(\boldsymbol{x}_1,..,\boldsymbol{x}_l)$, $\boldsymbol{y}=(y_1,\ldots,y_l)$ a část pro kterou jsou označení neznámé $\boldsymbol{X}=(\boldsymbol{x}_{(l+1)},\ldots,\boldsymbol{x}_{(l+u)})$.



Obr. 11: Schéma kombinovaného učení [67]

4.4 Princip učení s učitelem

Hlavním principem učení neuronových sítí je empirická minimalizace chyby. Snahou je minimalizace hodnot chybové funkce, které se docílí úpravou parametrů sítě. Těmito parametry jsou váhy hran a prahy neuronů, označeny v matematickém modelu jako ***w***, resp. ***b***.

Empirickou minimalizaci chyby lze vyjádřit jako:

$$\arg\min_{\theta}\frac{1}{T}\sum_{t} l(f(\boldsymbol{x}_t;\theta),y_t)\ \ (+\lambda\Omega(\theta)) \tag{12}$$

, kde

θ reprezentuje množinu všech zvolených parametrů pro daný model (matice vah a vektor prahů),

$\boldsymbol{x}_t$ značí vstup t,

y_t je požadovaný výstup pro případ t,

$l(f(\boldsymbol{x}_t;\boldsymbol{\theta}), y_t)$ je chybová funkce, která porovnává shodu mezi výstupem sítě a očekávanou (správnou) hodnotou,

$\Omega(\boldsymbol{\theta})$ je regularizátor[9] a

λ , která slouží k vybalancování poměru optimalizace chybové funkce a regularizátoru.

V rámci klasifikačních úloh učící algoritmus hledá optimální množinu parametrů tak, aby byla chyba na datech minimální. Ovšem chybovost klasifikace, dle které se modely posuzují, není hladká, ale skoková funkce, proto je při trénování nahrazena funkcí chybovou *l*.

Stochastický gradientní sestup je jedním z nejjednodušších principů pro spojitou optimalizaci [19]. Jedná se o iterativní postup, který na základě faktu, že chybová funkce nejrychleji klesá ve směru negativního gradientu, upravuje hodnoty parametrů modelu.

Princip gradientního sestupu:

1. Inicializace $\boldsymbol{\theta}$ ($\boldsymbol{\theta} = \{\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \ldots, \boldsymbol{W}^{(L+1)}, \boldsymbol{b}^{(L+1)}\}$), kde L značí počet skrytých vrstev.
2. Pro *N* iterací:

   Pro každý tréninovací případ ($\boldsymbol{x}_t, y_t$)

$$\Delta = -\nabla_{\theta} l(f(\boldsymbol{x}_t;\boldsymbol{\theta}), y_t) \quad (-\lambda \nabla_{\theta} \Omega(\boldsymbol{\theta})) \tag{13}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \Delta \tag{14}$$

[20]

4.5 Chybová funkce

Existuje celá řada chybových funkcí, které se používají dle jejich charakteristik. Pro regresní úlohy je výsledkem podmíněná distribuce výstupních proměnných[10] a proto je vhodný čtverec chyby [21].

Čtverec chyby:

$$l(f(\boldsymbol{x}), y) = \frac{1}{2} \sum_c (f(\boldsymbol{x})_c - y_c)^2 \tag{15}$$

, kde

$f(\boldsymbol{x})_c$ je aktivace výstupního neuronu c,

y_c je požadovaný výstup pro neuron c.

Ovšem, jak již bylo zmíněno, není tato funkce vyhovující pro klasifikační úlohy. Z tohoto důvodu se používají funkce pracující s entropií.

Cross-entropy error, nejdříve případ se dvěma třídami. Nechť je ve výstupní vrstvě pouze jeden neuron. Jeho aktivace nabývá hodnoty $P(c_1|\boldsymbol{x}) = f(\boldsymbol{x})$ vyjadřující pravděpodobnost příslušnosti prvku do první skupiny, respektive do druhé skupiny rovnající se $P(c_2|\boldsymbol{x}) = 1 - f(\boldsymbol{x})$ [21]. Pak funkce, u které je hledáno minimum má tvar:

9 Regularizátor slouží k penalizaci vysoké složitosti sítě, tzn. pokud jsou použity určité hodnoty pro parametry z množiny θ, především při použití extrémních hodnot. Tím se částečně zamezuje přeučení sítě. Teto část rovnice není mandatorní a přeučení se dá zamezit i jinými technikami, diskutováno v kapitole 5.

10 Podmíněná vstupními daty.

$$l(f(\boldsymbol{x}), y) = -t \log q - (1-t) \log(1-q) \tag{16}$$

, kde

q je vypočtená předpověď a

t je požadovaná hodnota.

Cross-entropy error (více-dimenzionální), pokud máme více uzlů na výstupní vrstvě je třeba funkci upravit. Nechť $f(\boldsymbol{x})_c = p(y=c \mid \boldsymbol{x})$ je výsledná podmíněná pravděpodobnost, že prvek $\boldsymbol{x}$ patří do skupiny c, která je ta správná, tzn. že se jedná o požadovanou třídu y. Pak je cílem maximalizovat tuto hodnotu, která je predikcí přiřazení prvku $\boldsymbol{x}_t$ do požadované třídy y_t .

$$l(f(\boldsymbol{x}), y) = -\sum_c 1_{(y=c)} \log f(\boldsymbol{x})_c = -\log f(\boldsymbol{x})_y \tag{17}$$

[22] je chybová funkce, kde

$1_{(y=c)}$ je identická funkce nabývající hodnoty 1 jen pokud se $y=c$, jinak se rovná 0,

$f(\boldsymbol{x})_y$ je hodnota aktivace na cílovém neuronu.

I když je cílem maximalizace správné hodnoty, učící algoritmus pracuje s minimalizací funkce. Proto se ve funkci vyskytuje mínus, které obrací maximalizaci monotónně rostoucí funkce na minimalizace funkce.

4.6 Zpětné šíření chyby

Gradientně založená učící procedura se nazývá ***backpropagation***[11]. Základní ideou je, že lze gradient efektivně vypočítat pro každou vrstvu zpětným šířením od výstupu ke vstupu [23].

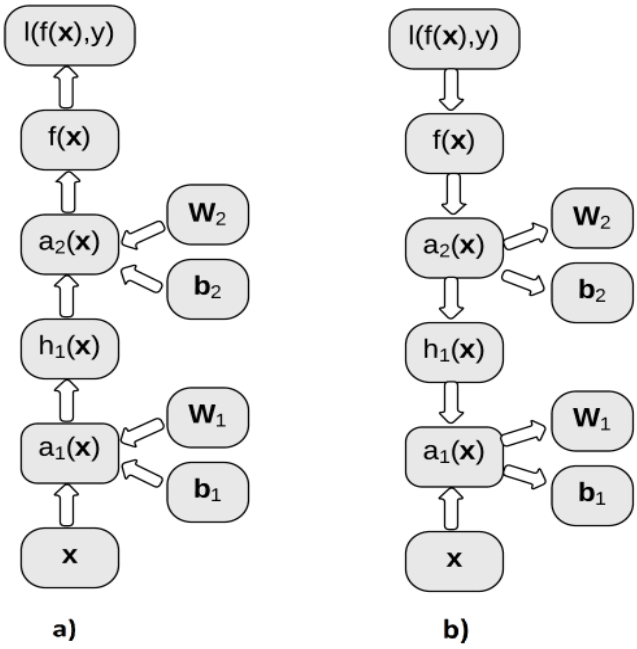
Cílem učení je empirická minimalizace chyby, respektive globální chybové funkce. Celková chyba vyjadřuje pravděpodobnost špatné klasifikace modelem a proto je snaha o její minimalizaci ve vztahu ke všem parametrům modelu (matice vah ***W***, vektor biasů ***b***) [23].

Obr. 12 schéma *a)* znázorňuje grafickou reprezentaci modulární implementace dopředného šíření (feed-forward) v podobě acyklického grafu[12]. Každý uzel je objektem s metodou šíření, která vypočítá hodnotu pro svého potomka. Provedení dopředného šíření znamená, proběhnutí informace všemi uzly ve správném pořadí.

Každý uzel, nazývaný též modulem, dopředného šíření je zároveň modulem zpětného šíření viz schéma *b)* Obr. 12. Aby bylo možné využití stochastického gradientního sestupu, je nutná existence derivace chybové funkce ve vztahu ke všem modulům. To je splněno pokud: existuje derivace chybové funkce, funkce implementované jednotlivými moduly jsou spojité a mají derivaci „téměř všude“ ve vztahu k interním parametrům modulu a vstupům do modelu [23]. Pokud je tento předpoklad splněn, lze jednoduše použít *backpropagation* k propočtu gradientu chybové funkce ve vztahu ke všem parametrům systému [24].

11 V překladu znamená „zpětné šíření“.

12 Znázorňuje model vícevrstvého perceptronu se dvěma skrytými vrstvami.



Obr. 12: Schéma znázorňujíci modularní reprezentace a) feed-forward, b) backpropagation

4.7 Učící proces

Algoritmus procesu učení pomocí zpětného šíření se skládá z několika kroků:

i. *Feed-forward* výpočet

ii. *Backpropagation* výstupní vrstvy

iii. *Backpropagation* skrytých vrstev

iv. Přičtení gradientů k vahám

Vhodné je definovat deltu, která referuje hodnotu chyby v neuronu *j* ve vrstvě *k*.

$$\delta_j = \frac{\partial l}{\partial a_{(k)}(\boldsymbol{x})_j} \tag{18}$$

Prvním krokem je provedení aktivní dynamiky pro vstupní vektor $\boldsymbol{x}$. Tento krok je shodný se samotným používání modelu, kdy je do sítě vložena informace a výsledkem je výstup, predikce modelu.

Druhým krokem je výpočet gradientu pro výstupní vrstvu. Aby mohl být vypočten v závislosti na vahách spojujících poslední skrytou a výstupní vrstvu $\frac{\partial l}{\partial w_{ij}}$, je třeba parciální derivaci rozšířit pomocí řetězového pravidla[13] na

13 Věta: Předpokládejme, že funkce *f* je diferencovatelná na otevřeném množině *G* a že funkce *g* je diferencovatelná na množině *f*[G]. Pak je složená funkce $g \circ f = g(f)$ diferencovatelná na *G* a $[g(f)]' = g'(f) * f'$ je řetězové pravidlo.

$$\frac{\partial l}{\partial w_{ij}} = \frac{\partial l}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}} \tag{19}$$

Nejdříve se vypočte chyba pro každý neuron ve výstupní vrstvě ve vztahu k potenciálu neuronu, tzn. hodnota před použitím aktivační funkce.

$$\delta_j = \frac{\partial l}{\partial a_{(L+1)}(\boldsymbol{x})_j} = f'(a_{(L+1)}(\boldsymbol{x})_j) \frac{\partial l}{\partial f(\boldsymbol{x})_j} \tag{20}$$

, kde

$a_{(L+1)}(\boldsymbol{x})_j$ je potenciál neuronu *j* ve vrstvě *L+1*,

$\frac{\partial l}{\partial f(\boldsymbol{x})_j}$ značí parciální derivace chybové funkce pro aktivaci *j* neuronu,

$f'(a_{(L+1)}(\boldsymbol{x})_j)$ je derivace aktivační funkce neuronu.

Druhá část rozšířené parciální derivace (19) odpovídá

$$\frac{\partial a_j}{\partial w_{ij}} = h_i(\boldsymbol{x}) \tag{21}$$

Substitucí v rovnici (19) za (20) a (21) vznikne vztah:

$$\frac{\partial l}{\partial w_{ij}} = \delta_j h_i(\boldsymbol{x}) \tag{22}$$

Třetím krokem je výpočet gradientu pro zbylé, skryté vrstvy. Obsahem kroku je opět výpočet parciální derivace chybové funkce dle parametrů sítě, který je třeba nahradit pomocí řetězového pravidla.

$$\frac{\partial l}{\partial w_{ij}} = \frac{\partial l}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}} \tag{23}$$

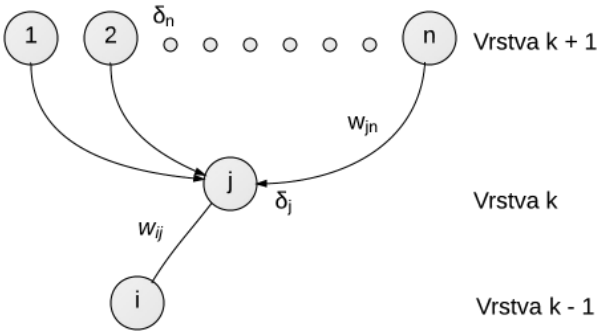
Ovšem při výpočtu delty neuronu skryté vrstvy *k* je třeba zohlednit všechny uzly *n* z následující vrstvy *k+1* se kterými má neuron vazby, znázorněno na Obr. 13.

$$\delta_j = \frac{\partial l}{\partial a_{(k)}(\boldsymbol{x})_j} = \sum_n \frac{\partial l}{\partial a_{(k+1)}(\boldsymbol{x})_n} \frac{\partial a_{(k+1)}(\boldsymbol{x})_n}{\partial a_{(k)}(\boldsymbol{x})_n} \tag{24}$$

Substitucí definice δ (18) a použitím rovnic (9), (10) z popisu neuronové sítě, lze deltu vyjádřit jako:

$$\delta_j = g'(a_{(k)}(\boldsymbol{x}_j)) \sum_n w_{jn} \delta_n \tag{25}$$

[21]



Obr. 13: Schéma propočtu gradientu ve skryté vrstvě

Tvar rovnice říká, že konkrétní deltu můžeme získat zpětným šířením chyby z vyšších vrstev (viz Obr. 13). Jelikož tyto hodnoty jsou známé z druhého kroku, stačí rekurzivně použít krok tři pro každou předešlou vrstvu (znázorněno na Obr. 12 b)).

Změna váhy, gradient, se získá shodným postupem jako ve druhém kroku:

Druhá část rozšířené parciální derivace (23) odpovídá

$$\frac{\partial a_j}{\partial w_{ij}} = h_i(\boldsymbol{x}) \tag{26}$$

Substitucí rovnice (23) za (25) a (26) vznikne vztah:

$$\frac{\partial l}{\partial w_{ij}} = \delta_j h_i(\boldsymbol{x}) \tag{27}$$

Ve **čtvrtém krok** pak dochází jen k upravení vah a prahů. Tento krok nemusí být proveden po každém trénovacím příkladu viz kapitola 4.10.

$$\Delta w_{ij} = -\alpha \delta_j h(\boldsymbol{x})_i \tag{28}$$

$$w_{ij} = \Delta w_{ij} \quad (+\beta \Delta w_{ij}^{(t-1)}) \tag{29}$$

, kde

α je učící parametr.

$\Delta w_{ij}^{(t-1)}$ je změna váhy v předchozím kroku,

β značí moment, sílu gradientu ve směru předešlé změny.

Práh *b* z rovnice (9) lze vyjádřit jako

$b_j = w_{(i+1)j} \cdot 1$, proto změna práhu se rovná:

$\Delta b_j = \Delta w_{(i+1)j} = -\alpha \delta_j$

4.8 Derivace chybových a aktivačních funkcí

V předchozí kapitole je použita derivace chybových a aktivační funkcí pro výpočet odchylky ve vztahu k potenciálu neuronu v (20) a (25). Práce proto uvádí derivace některých základních funkci.

Pro ***čtverec chyby***

$$\frac{1}{2}\sum_c (f(\boldsymbol{x})_c - y_c)^2 \tag{30}$$

, je derivace podle aktivační funkce

$$\frac{\partial l}{\partial f(\boldsymbol{x})_c} = o_c - y_c \tag{31}$$

, kde

$o_c = f(\boldsymbol{x})_c$ je výstup neuronu *j* a

y_c je požadovaná hodnota.

Cross-entropy:

$$l(f(\boldsymbol{x}), y) = -\sum_c 1_{(y=c)} \log f(\boldsymbol{x})_c = -\log f(\boldsymbol{x})_y \tag{32}$$

, má derivaci podle aktivační funkce ve výstupním neuronu *c*:

$$\frac{\partial l}{\partial f(\boldsymbol{x})_c} = -\frac{y_c}{o_y} = -\frac{1_{(y=c)}}{o_y} \tag{33}$$

V případě ***lineární funkce*** je výpočet triviální, pro:

$g(a)=a$ se gradient rovná

$$g'(a)=1 \tag{34}$$

Gradient se proto nemusí vůbec brát v úvahu při výpočtu poteciálu neuronu.

Mnohem častěji používaná aktivační funkce ***sigmoid***

$$g(a)=sigm(a)=\frac{1}{1+\exp(-a)} \tag{35}$$

, má gradien

$$g'(a)=g(a)(1-g(a)) \tag{36}$$

Dále pak ***hyperbolický tangens*** ve tvaru

$$g(a)=\tanh(a)=\frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)}=\frac{\exp(2a)-1}{\exp(2a)+1} \tag{37}$$

, má gradient

$$g'(a)=1-g(a)^2 \tag{38}$$

V obou případech logistických funkcí je vidět, že pokud budou hodnoty potenciálu nabývat extrémních hodnot, bude se derivace limitně blížit nule[14]. To může mít za následek obtížné učení modelu, pokud budou hodnoty vah příliš vysoké, respektive nízké.

Výpočet gradientu funkce ***softmax*** je mírně obtížnější. Funkce má tvar:

$$\boldsymbol{o}(\boldsymbol{a})=softmax(\boldsymbol{a})=[\frac{\exp(a_1)}{\sum_c \exp(a_c)}\dots\frac{\exp(a_c)}{\sum_c \exp(a_c)}]^T \tag{39}$$

Tvar derivace aktivační funkce z (20) je třeba nahradit derivací parciální a to podle aktivace konkretního neuronu.

$$\frac{\partial f(\boldsymbol{x})_y}{\partial a_{(L+1)}(\boldsymbol{x})_c}=1_{(y=c)}f(\boldsymbol{x})_y-f(\boldsymbol{x})_y f(\boldsymbol{x})_c=1_{(y=c)}o_y-o_y o_c \tag{40}$$

U klasifikačních úloh je nejčastějším případem, že výstupní vrstva má za aktivační funkci právě *softmax* a jako chybová funkce se využívá *cross-entropy*. V tomto případě, pokud v rovnici (20) je

14 Patrné je to i z grafů obou funkcí.

provedena substituce rovnicemi (33) a (40) je delta vypočítána jako:

$$\delta_c = \frac{\partial l}{\partial f(\boldsymbol{x_c})} \frac{\partial f(\boldsymbol{x_y})}{\partial a_{(L+1)}(\boldsymbol{x})_c} = -\frac{1}{f(\boldsymbol{x})_y}(1_{y=c} f(\boldsymbol{x})_y - f(\boldsymbol{x})_y f(\boldsymbol{x})_c) = -(1_{(y=c)} - f(\boldsymbol{x})_c) \tag{41}$$

, tzn že

$\delta_c = f(\boldsymbol{c})_c = o_c$ když $c \neq y$ a

$\delta_c = f(\boldsymbol{x})_c - 1 = o_c - 1$ když $c = y$.

[21]

4.9 Resilient propagation

Zpětné šíření chyby není jedinou učící metodou. Studie prezentuje ještě jeden učící algoritmus, používaným hlavně v rámci *Encog*[15].

Vícevrstvé sítě často používají *logistické* funkce jako aktivační. Tito funkce jsou nazývány „smršťovacími[16]“, jelikož vytvářejí kompresi hodnot z neomezeného intervalu do intervalu o délce jedna, respektive dva. Je to dáno tím, že jejich derivace se velmi rychle blíží nule, pokud jsou hodnoty vychýlené od nuly. Z tohoto důvodu je občas velmi obtížné naučit síť pokud potenciály neuronů nabývají extrémních hodnot. V takovémto okamžiku jsou vypočtené gradienty pomocí zpětného šíření velmi malé a tedy i změny vah jsou nepatrné [25].

Tuto nevýhodu odstraňuje odolné šíření. O změnách parametrů rozhoduje pouze *signum* hodnota derivace. Její velikost nemá na úpravu žádný vliv. Každému parametru sítě je přiřazena individuální delta, které je upravována[17]:

$$\Delta_{ij}^{(t)} = \eta^{+} \Delta_{ij}^{(t-1)}, \; pokud \; \frac{\partial l^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial l^{(t)}}{\partial w_{ij}} > 0 \tag{42}$$

$$\Delta_{ij}^{(t)} = \eta^{-} \Delta_{ij}^{(t-1)}, \; pokud \; \frac{\partial l^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial l^{(t)}}{\partial w_{ij}} < 0 \tag{43}$$

$$\Delta_{ij}^{(t)} = \Delta_{ij}^{(t-1)}, \; jinak \tag{44}$$

Parametr je pak změněn následovně:

$$\Delta w_{ij}^{(t)} = -\Delta_{ij}^{(t)}, \; pokud \; \frac{\partial l^{(t)}}{\partial w_{ij}} > 0 \tag{45}$$

15 Jedna z nejobsáhlejších knihoven pro implementaci neuronových sítí.
16 Z anglického „squashing“.
17 Algoritmus zvaný RPORP [24].

$$\Delta w_{ij}^{(t)} = +\Delta_{ij}^{(t)},\ pokud\ \frac{\partial l^{(t)}}{\partial w_{ij}} < 0 \tag{46}$$

$$\Delta w_{ij}^{(t)} = 0,\ jinak \tag{47}$$

4.10 Způsoby dávkování

Ve čtvrtém kroku učícího procesu (viz 4.7) bylo zmíněno, že k úpravám parametrů nemusí docházet okamžitě, vše záleží na zvoleném způsobu. Existují tři možnosti: změna vah po iteraci jednoho případu, po iteraci množiny případů nebo po proběhnutí kompletní trénovací sady. Každý z těchto postupů má svá úskalí a výhody.

On-line učení, nebo též nazývané „stochastickým", je případem, kdy učící algoritmus je proveden ve všech čtyřech krocích pro každý příklad zvlášť. Hodnota gradientu je spočítaná jen pro tento jeden vzor a parametry jsou pak hned upraveny před proběhnutím další iterace.

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) - \alpha \frac{\partial l^{(t)}}{\partial \boldsymbol{W}} \tag{48}$$

Protože jsou v tomto případě hodnoty gradientů závislé pouze na jednom trénovacím příkladu, vykazují značný šum, často pak chyba kolísá kolem klesajícího trendu. [27] zmiňuje důvody proč je stochastického učení používáno:

1. On-line učení je většinou mnohem rychlejší než dávkové učení.
2. Výsledky on-line učení jsou často lepší než v případě dávkového.
3. V průběhu učení lze pozorovat změny.

Rozdíl v rychlosti je hlavně patrný při učení na velkých redundantních datech. Například pokud by data obsahovala každý prvek desetkrát, dávkové učení bude propočítávat ty samé hodnoty desetkrát avšak parametry upraví pouze jednou. Zatímco pro stochastické učení to nic nezmění, po kompletním proběhnutí jedné epochy na těchto datech, je výsledek stejný jako po proběhnutí 10 epoch na datech neredundantních. V praxi bývají data redundantní zřídka, nicméně případy jsou si často natolik podobné, například ve shlukovacích úlohách, že se za totožná dají téměř považovat.

Druhá výhoda plyne z negativní vlastnosti tohoto typu učení. Nelineární sítě mají většinou několik lokálních minim s různou hloubkou. To, že on-line učení vykazuje silný šum, způsobuje, že se může z těchto lokálních minim dostat a nalézt nějaké jiné, lepší řešení. Naopak batch učení spíše nalezne minimum blízké prvotní inicializaci, které může ale nemusí být optimální. Tento jev byl prezentován na příkladě v [28].

Batch učení je typem, kdy ke změně parametrů dojde až po proběhnutí epochy, celé trénovací množiny. To znamená, že z učícího algoritmu nejprve proběhnou první tři kroky pro každý příklad z trénovací množiny. Gradienty se postupně sčítají nebo průměrují a čtvrtý krok úpravy vah proběhne až na závěr.

I přes zmíněné nevýhody oproti on-line učení existují důvody proč dát přednost batch učení, které zmiňuje [19].

1. Podmínky konvergence jsou zcela zřejmé.

2. Je zde mnoho urychlujících technik, praktikovaných pouze u *batch* učení.
3. Teoretická analýza dynamických vah a rychlost konvergence je jednodušší.

Důvodem těchto výhod je opět šum, kterým trpí stochastické učení. Tento šum umožňuje v případě *online* učení nalezení lepšího lokálního minima, ale důsledkem je zároveň neschopnost konvergovat do absolutního minima. Namísto nalezení přesného lokálního optima se konvergence zastaví z důvodu kolísání hodnot vah. Tuto fluktuaci lze omezit snižováním učícího parametru nebo adaptivní velikostí *batche*. V praxi se často využívá první způsob ať už plánovaným skokovým snižováním, snižováním závisejícím na počtu epoch nebo dynamicky se měnícím na základě změny velikosti chyby.

Mini-batch učení je třetí možností. Snahou tohoto postupu je využití výhod obou prezentovaných metod. V tomto případě je zde jeden parametr navíc, kterým je velikost dávky. Ten v poměru s množstvím případů v trénovací sadě určuje, zda se učení bude více chovat jako stochastické nebo *batch*. Zároveň nabízí možnost velikost *mini-batche* měnit v závislosti na učícím procesu.

4.11 Více-vláknové učení

V dnešním světě více-jádrových počítačů je velmi aktuální myšlenka více-vláknového učení. Ta se dá relativně snadno implementovat v případě *batch* učení. Přičemž nezáleží na velikosti dávky, pokud je větší než počet vláken, na kterých by měl program zároveň běžet.

Trénovací proces je rozdělen do čtyř etap, z nichž první tři mohou běžet současně. Důvodem je charakter *batch* učení, kdy množina dat je šířena modelem o stejných parametrech a gradienty jsou sčítány. Proto může jak dopředné, tak i zpětné šíření probíhat najednou. Pouze při úpravě vah je použito jen jedno vlákno, ovšem jedná se o etapu učení, která je časově zanedbatelná [29].

5 Přeučení

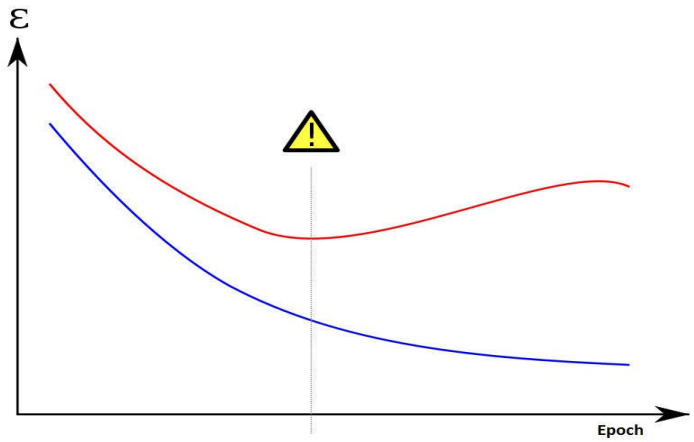
5.1 Efekt

Přeučení[18] je největším problémem, se kterým se neuronové sítě setkávají (znázorněný na Obr. 14). Z grafu je patrné, že zatímco se model více přizpůsobuje předloženým datům pro trénování, od určitého okamžiku testovací chyba začne růst. Je to dáno tím, že se model přizpůsobí specifickým případům a není dále schopen predikovat na nových datech, různých od trénovacích.

Jinak řečeno, správně naučený model dokáže generalizovat charakteristické rysy nalezené při trénování na veškerá data generovaná doménou. Tato schopnost se odvíjí od komplexity sítě, která muže být ovlivněna složitostí a typem modelu nebo regularizací [30]. Schopnost modelu generalizovat predikci je měřena právě pomocí testovací chyby.

Pokud existuje v předešlých etapách učení nastavení parametrů, které poskytovalo lepší generalizaci, hovoří se o přeučení. V případě, že je tomu naopak, jedná se o *nedoučení*. Tzn. pokud by se nechal model déle učit, aby lépe pokryl trénovací data, zlepšila by se předpověď i na testovacích datech. K nedoučení dochází i z důvodu nedostatečné velikosti sítě, která tak nemá dostatečnou kapacitu na aproximaci funkce, viz následující kapitola. Nedoučení lze poznat, pokud má chyba na trénovacích a testovacích datech podobnou hodnotu.

18 Známější pod anglickým názvem „overfitting“.

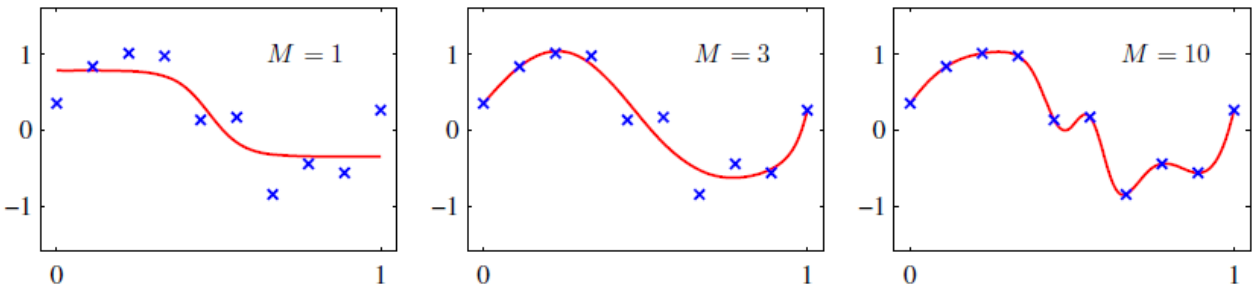


Obr. 14: Graf přeučení, Je zde znázorněna trénovací (modrá) a testovací(červená) chyba. Typycky trénovací chyba neustále klesá, zatímco testovací chyba se od určitého okamžiku přestane snižovat.[68]

5.2 Složitost modelu

Při vytváření architektury neuronové sítě je třeba rozhodnout mimo jiné o počtu a velikosti vrstev. Zatímco vstupní a výstupní vrstva jsou součástí modelu vždy, počet skrytých vrstev je možné měnit. Velikosti mandatorních vrstev jsou pevně dané strukturou vstupu, respektive požadovaného výstupu. Je zřejmé, že pokud se informace skládá z pěti hodnot, bude ve vstupní vrstvě pět neuronů. Ovšem počet neuronů ve skryté vrstvě je opět volitelný a úzce souvisí s výskytem přeučení. Počtem vrstev se práce zabývá v kapitole 6.

Takto [13] popisuje danou problematiku. Dané množství skrytých neuronů M, kde M je volným parametrem, kontroluje počet trénovacích parametrů[19]. Lze tedy předpokládat, že při dosažení maximální hodnoty úspěšnosti předpovědí modelu, bude M nastaveno na optimální hodnotu, která poskytuje nejlepší generalizaci znalostí. Ta odpovídá optimální rovnováze mezi nedoučením a přeučením modelu. Viz následující schéma.



Obr. 15: Schéma závislosti generalizace na počtu neuronů ve skryté vrstvě. [13]

Nicméně problém generalizace není prostou funkcí M z důvodu lokálních minim chybové funkce. Je tudíž téměř nemožné v případě složitějších modelů získat přesné M. Ve většině implementací se proto používá spíše větší hodnota, která zajistí, že nedojde k nedoučení. Zároveň je přeučení velmi dobře rozpoznatelné a lze použít některou z technik regularizace k jejímu odstranění.

19 Váhy a prahy.

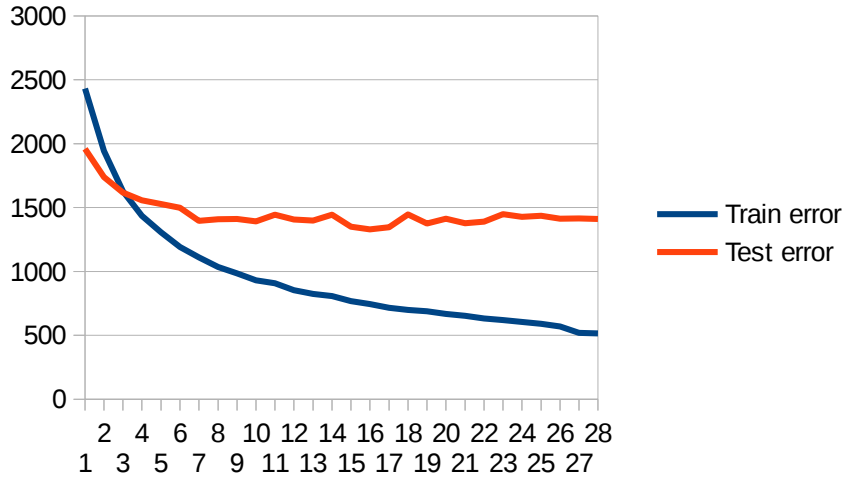
5.3 Brzké zastavení

Základní metodou pro zamezení přeučení je brzké zastavení[20]. Jedná se o široce využívanou metodu z důvodu snadnosti implementace, která je zároveň prezentována jako velmi efektivní [31].

Jak je popsáno v [32], testovací chyba nabývá svého minima (viz Obr. 14), které je velmi snadno detekovatelné. [32] dodává, že pokud by průběh testovací chyby byl takto hladký, stačil by následující postup pro získání optimálně naučené sítě:

- Rozděl data na trénovací a testovací množinu[21].
- Nauč síť pouze na trénovacích datech a testuj jednou za několik epoch na testovacích.
- Zastav učení ve chvíli, kde testovací chyba vzroste.
- Vezmi parametry modelu z předposlední epochy, jako výsledek učení.

Nicméně reálně se testovací chyba nevyvíjí hladce (znázorňuje Obr. 14), ale nepravidelně kolísá (viz Obr. 16). Nelze tedy definovat třetí krok jako pevné pravidlo, které zajistí nejlepší možný model. Pro včasné zastavení se proto používají jiná pravidla.



Obr. 16: Graf reálného znázornění vývoje testovací chyby v průběhu epoch.

Pokud je cílem dosáhnutí jen určité úrovně úspěšnosti predikce, pak se dá tato hodnota považovat za hraniční a při jejím dosažení proces zastavit.

$$Pokud\ l(f(\boldsymbol{x}_{test}), y) < konst.$$

Další možností je v průběhu učení ukládat nejlepší model. Tím je zajištěno získání modelu s nejmenší testovací chybou na proběhnuvších iteracích. Pak už stačí učení pouze přerušit ve chvíli, kde je zřejmé, že úspěšnost modelu nebude dále klesat a dochází pouze k přeučení.

Více typů brzkého zastavení je uvedeno v [32].

5.4 Regularizátor

Další technikou regularizace je často použití regularizátoru. Tradičně je regularizátor

20 Z anglického „Early stopping“.
21 Poměr rozdělení se různí, [32] uvádí 2:1. V případě datové sady MNIST to je 6:1.

implementován jako dodatečná část chybové funkce učícího algoritmu (viz (12) a (13)). Hlavní nevýhodou této metody je nutnost předem definovat hodnotu parametru, který řídí sílu regularizátoru ve vztahu k učícímu procesu[33].

Cílem této metody je omezit komplexitu sítě pomocí penalizace vysoké složitosti struktury sítě během učení. Stává se součástí chybové funkce, ke které je regularizátor přičten (12). V důsledku snahy o minimalizaci chyby působí sčítanec jako negativní faktor[22], který závisí na struktuře modelu.

Častými typy regularizátoru jsou funkce L1, L2:

L1 má podobu:

$$\Omega(\boldsymbol{\theta}) = \sum_k \sum_i \sum_j \left| W_{i,j}^{(k)} \right| \tag{49}$$

Pro výpočet gradientu (13) je nutná parciální derivace ve vztahu k vahám:

$$\nabla_{\boldsymbol{W}^{(k)}} \Omega(\boldsymbol{\theta}) = sign(\boldsymbol{W}^{(k)}) \tag{50}$$

, kde

$sign(\boldsymbol{W}^{(k)}) = 1_{\boldsymbol{W}_{i,j}^{(k)} > 0} \quad -1_{\boldsymbol{W}_{i,j}^{(k)} < 0}$.

L2 má podobu:

$$\Omega(\boldsymbol{\theta}) = \sum_k \sum_i \sum_j (W_{i,j}^{(k)})^2 = \sum_k \| \boldsymbol{W}^{(k)} \|_F^2 \tag{51}$$

Parciální derivace ve vztahu k vahám:

$$\nabla_{\boldsymbol{W}^{(k)}} \Omega(\boldsymbol{\theta}) = 2\, \boldsymbol{W}^{(k)} \tag{52}$$

, kde

F značí Frobeniusovu normu.

Obě uvedené funkce se zaměřují pouze na váhy a nepenalizují prahy. Rozdíl v použití je, že úprava L2 vede k malým hodnotám vah, zatímco L1 vede k jejím úplným vynulováním a vytváří tak řidší prostor propojení vrstev.

Další metody pro zamezení přeučení lze nalézt v [21].

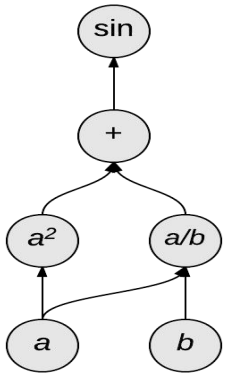
22 Regulátor je vždy kladný a tedy zvyšuje hodnotu chyby.

6 Deep learning

6.1 Hloubka sítě

Výpočet hodnot předpovědi pro daný vstup lze reprezentovat vývojovým diagramem, kde každý uzel znázorňuje elementární výpočet přijaté hodnoty, jejíž funkční hodnota je poslána na další vrstvu potomkovi.

Vývojový diagram pro výraz $\sin(a^2+b/a)$ může vypadat následovně Obr. 17:



Obr. 17: Vývojový diagram sin(a^2 + a/b)

V takovém případě se ***hloubkou*** grafu myslí délka nejdelší cesty ze vstupní do výstupní vrstvy. U tradičních dopředných sítí takto definovaná hloubka odpovídá počtu vrstev, přesněji počtu skrytých vrstev + 1 pro vrstvu výstupní, viz Obr. 10 s hloubkou 3.

Přívlastek *deep* značí fakt, že se jedná o struktury s mnoha vrstvami. Doposud byl diskutován pouze model dopředné sítě: vícevrstvý perceptron. Ten obvykle obsahuje jednu až dvě skryté vrstvy. Avšak *deep learning* se zabývá modely s několika skrytými vrstvami. Tento počet se různí dle typu úlohy, ale jedná se zpravidla o více než dvě skryté vrstvy, u kterých nastává problém s jejich učením.

6.2 Motivace

V mnohých situacích jedna či dvě skryté vrstvy stačí pro aproximaci libovolné funkce s danou přesností, jak definuje Hornik v [16], ovšem často za cenu značného nárůstu počtu neuronů ve vrstvách. Teoretické výsledky publikovány Hastadem v [34] ukazují na určitou množinu funkcí, jenž mohou být efektivně reprezentovány s $O(n)$ uzly (kde *n* značí počet případů) při hloubce sítě *d*, avšak pro aproximaci sítí o hloubce $d-1$ je již třeba $O(2^n)$ uzlů.

Bengio v [35] diskutuje, že mnohé libovolně zvolené funkce nelze *efektivně* modelovat ať už s *deep* či *shallow* architekturou sítě. Nicméně v některých případech je možná efektivní aproximace *deep* strukturou, ne však *shallow*. Například v případě naučení se komplikovaných funkcí, které reprezentují vysokou míru abstrakce[23], je nutné použití *deep* architektury.

„*Deep learning*“ je oblastí strojového učení, která se právě touto architekturou zabývá. Glorot a Bengio v [36] popisují cíl metod *deep learning*, jako naučení se hierarchie rysů od těch vyšší úrovně po nižší úrovně, ze kterých jsou složeny. Každá vrstva koresponduje s nějakou úrovní distributivní reprezentace rysů. To znamená, že uzly v jedné vrstvě nejsou vzájemně se vylučující a tedy dva neurony v jedné vrstvě mohou být současně aktivovány. Tato myšlenka se vylučuje například s rozdělováním do shluků, kde jeden případ patří do právě jednoho shluku.

23 Jako je například počítačové vidění, jazykové rozpoznávání a další úlohy na úrovni umělé inteligence.

Příkladem může být interpretace obrázku (například Obr. 18). Když se pokouší o vyřešení podobné úlohy[24] člověk často dochází k intuitivnímu rozložení na pod-problémy a více úrovňovou reprezentaci [37]. Zná tedy charakteristické rysy jednotlivých objektů vyskytujících se na obrázku. Je možné, že zobrazenou věc nikdy v životě neviděl, ale je schopen ji na základě elementárních charakteristických rysů klasifikovat jako auto.

Obr. 18: Příklad extrakce rysů pro interpretaci obrázku

Distributivní reprezentantce by se dala demonstrovat v tomto případě na rozdělení obrázku na objekt a pozadí. Zvlášť by se klasifikoval objekt jako auto a pozadí jako předměstí. Výsledkem je pak popis obrázku, jako auto na předměstí, místo aby existoval přímo shluk auto na předměstí, do kterého by obrázek spadal.

Struktura mozku savce by se dala charakterizovat jako *deep* architektura [38]. Vizuální vjem je v mozku reprezentován mnohými úrovněmi abstrakce, jenž každá odpovídá jiné části mozkové kůry.

Například vizuální kortex je ta část mozkové kůry, která zpracovává podměty vysílané zrakovým smyslovým orgánem. Rozděluje se na mnoho částí, kde každá má za úkol zpracování jiných vzorů a podnětů. Sítnice posílá do kortexu jednotlivé části obrazu. V1 nebo-li primární kortex tyto tyto signály přijímá. Má za úkol rozpoznání jednoduchých vizuálních obrysů a hran. Další, zajímavou částí, je oblast pojmenovaná V4, kam proudí signál z primární části přes V2. V4 skládá jednotlivé rysy do skupin a vytváří tak části scény. Poté signál putuje přes PIT do AIT, kde se tyto skupiny rysů skládají do jednotlivých objektů [39].

Nejen struktura mozku, ale i způsob poznání se zdá být typu „*deep*“ architektury. Mozek používá přetvořenou představu objektu k obrazu svému. Do takovéto reprezentace vkládá svoje zkušenosti a prožitky. Tím si upravuje vzpomínky a přetváří obrazy objektů.

K učení využívá převážně učení bez učitele. Toho je schopen nesmírně složitou a komplexní strukturou nervové soustavy. Velmi často učící vzor nepotřebuje nebo ho zapomene. Nicméně si stačí vytvořit obecnou představu a tu si zapamatovat.

Mozek se naučí snazší úlohy první a poté nabyté znalosti využije k řešení úloh složitějších a komplexnějších. Tento postup se objevuje snad ve všech oblastech lidského vědění. Lidé organizují svoje myšlenky a koncepty hierarchicky.

24 Na mysli je úloha pro umělou inteligenci.

6.3 Trénování

Přestože koncepce *deep* architektury modelu spadá do 80. let 20 století[25], nedošlo v této oblasti strojového učení k výraznému posunu až do poloviny prvního desetiletí 21 století. Pokusy o naučení *deep network* končily nezdarem a jejich výsledky byly horší než v případě mělkých sítí s jednou či dvěma skrytými vrstvami.

Průlom přišel v roce 2006, kdy byly publikovány tři studie [1] [41] [42] v čele s Hintonovou revoluční prací ohledně „*Deep Belief Network (DBN)"*. V těchto dokumentech byly prezentovány klíčové principy pro naučení *deep network*:

- Učení bez učitele je použito v fázi před samotným trénováním na každé vrstvě.
- Učení bez učitele na jedné vrstvě následuje po naučení předešlé vrstvy. Reprezentace na každé vrstvě je použita jako vstup pro vrstvu následující.
- Učení s učitelem následuje po fázi bez učitele. Používá se pro jemné doladění parametrů, vytvoření klasifikačních tříd a případně přidání jedné či více vrstev pro vytváření predikcí.

Druhým podstatným faktorem pro rozvoj *deep networks* bylo umožnění použití výpočetní kapacity grafických karet. Hlavně v případech operací na velké matici dat, jsou techniky využívající GPU[26] schopné výpočet velmi urychlit.

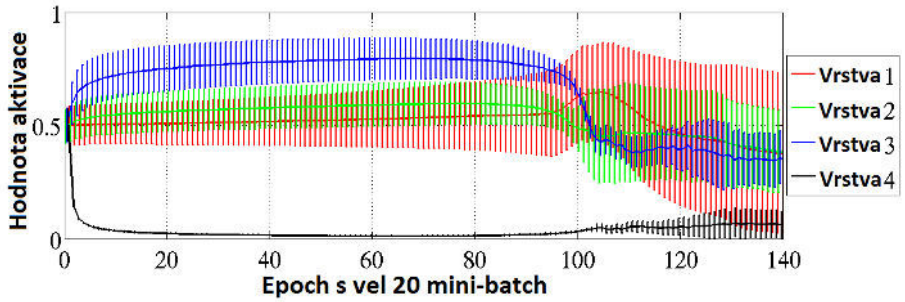
Učící proces používající standardní gradientní sestup se potýká při aplikaci na *deep* struktuře s dvěma závažnými problémy: *nedoučení* a *přeučení*. Oba nevyhovující stavy modelu již byly diskutovány v kapitole 5.2, avšak v tomto případě je situace komplikovanější.

K ***nedoučení*** dochází z důvodu, že optimalizační problém učení sítě je mnohem složitější v případě, kdy se model skládá z mnoha vrstev. Znamená to, že pokud by byl zvolen lepší optimalizační postup a síť by byla lépe naučena na trénovacích datech, dosáhlo by se úspěšnější predikce na datech testovacích. Efekt, který to způsobuje, se nazývá „*problém mizejícího gradientu (vanishing gradient problem)*".

Při zpětné propagaci se gradient chyby násobí parciální derivací aktivační funkce ve vztahu k potenciálu neuronu. Pokud je ale skrytý uzel saturován, tzn. že hodnota aktivace je blízka hraniční funkční hodnotě[27], pak se derivace logistické křivky blíží nule. A tedy i gradient potenciálu neuronu bude velmi blízký nule. Gradient chyby s touto nízkou hodnotou je pak dále šířen do dalších vrstev, kde dochází ke stejnému efektu. Hodnota chyby postupně mizí a u nižších vrstev pak již vůbec nedochází k učení. Tento jev se též vyskytuje u rekurentních sítí.

Situaci demonstruje Glorot a Bengio v [34] na grafu (viz Obr. 19). Ten ukazuje hodnoty aktivací ve čtyřvrstvé síti při použití sigmoidy jako aktivační funkce. Je zde vidět, jak jsou hodnoty poslední vrstvy od začátku téměř nulové a až po 100 epochách se odlepí od nuly. V tu chvíli začne docházet i k učení sítě, jelikož poslední vrstva už nešíří jen nulové gradienty.

25 Kunihiko Fukushima popsal model *neocognitron* a publikoval v roce 1980 v [34].
26 *Graphic processing unit* (grafický procesor).
27 Nastává v situaci, kdy potenciál neuronu dosahuje extrémních hodnot.



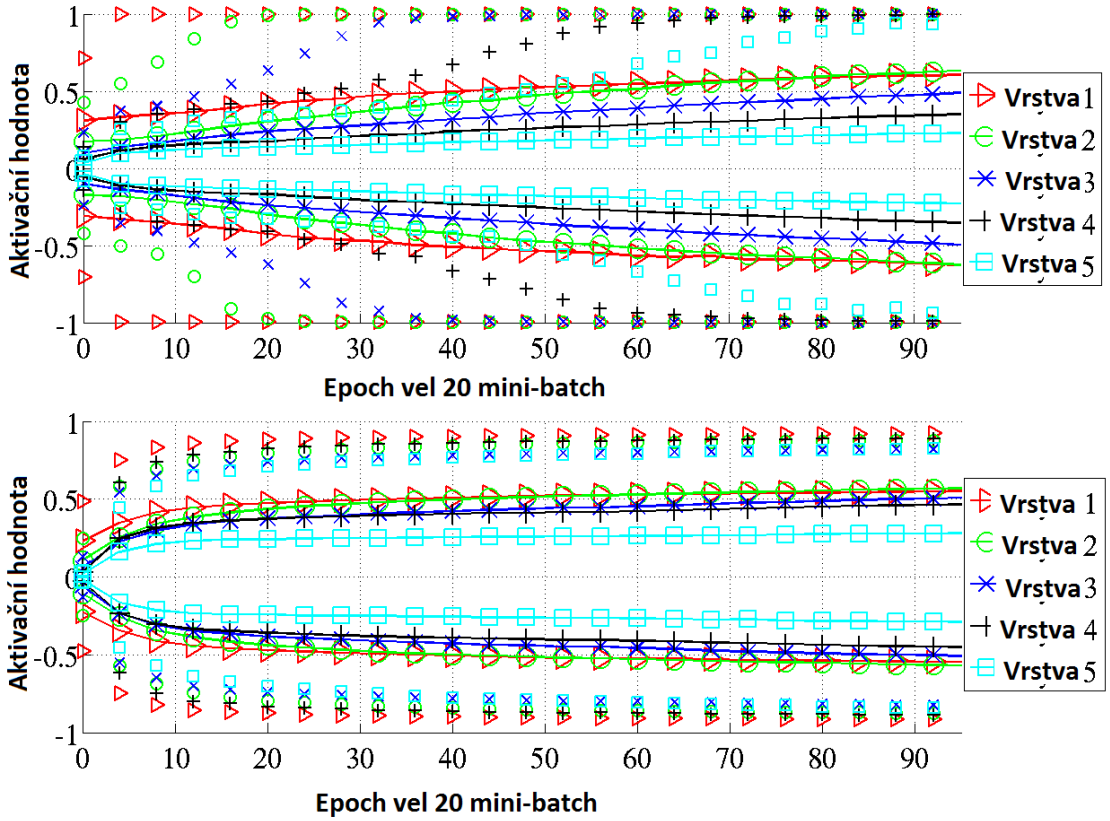
Obr. 19: Střední hodnota a směrodatná odchylka aktivačních hodnot při učení s učitelem. Je zde vidět, jak se síť s deep architekturou začne učit až po proběhnutí 100 epoch. [36]

Druhá problémová situace, která nastává pří učení *deep* modelů je ***přeučení***. Jak bylo zmíněno v kapitole 5, hlavní roli zde hraje komplexnost struktury sítě.

Důsledkem více vrstev je značné množství trénovacích parametrů a vysoká komplexita modelovaných funkcí, které jsou složitější s přidáním každé vrstvy. Snadno pak dochází k optimalizaci predikce pro trénovací data ale bez zachování generalizace. Výsledkem je nízká trénovací ale vysoká testovací chyba. Problém lze též interpretovat tak, že struktura sítě má velkou variabilitu řešení na různých podmnožinách trénovacích dat.

Řešení problému *nedoučení* se nachází ve fázi výzkumu a neexistuje tedy jednoznačné řešení. Literatura navrhuje změnu optimalizačního procesu. To znamená buďto změnu učícího procesu, nebo použití jiné aktivační funkce. Druhou možností se zabývá [36]. Zkoumá vztah tří typů aktivace a saturace neuronů. Nejhorším případem je použití *sigmoidy* (viz Obr. 19), u vrchní vrstvy vůbec nedochází k aktivaci. Lépe je na tom *hyperbolický tangens*, u něhož dochází k postupné saturaci od první po poslední vrstvu. Nejlépe z testu vychází aktivační funkce ***softsign*** $x/(1+|x|)$, u které se hodnoty aktivace jen přiblížují krajním mezím (viz Obr. 20).

Pokud se během učení sítě vyskytne *přeučení*, je nutné použití některé z regularizačních technik. Nejčastěji se implementuje učení bez učitele nebo stochastické trénování zvané „*dropout*“. Obě možnosti jsou detailně popsány v následujících kapitolách.



Obr. 20: Porovnání aktivačních hodnot tanh (horní) a softsign (spodní) [36]

6.4 Učení bez učitele

Myšlenkou tohoto postupu je změnit metodu inicializace vah a to pomocí učení bez učitele v etapě *pre-training*[28] [43]. Je doplněn požadavek, aby síť nejen rozpoznala charakteristické rysy nacházející se v datové sadě, ale zároveň určila, čím se tyto příklady odlišují od náhodných dat. Jinak řečeno, přinutit síť rozpoznat skryté struktury v distribuci informace v učících datech.

Obr. 21: Ilustrace rozdíly mezi náhodným uspořádáním pixelů a trénovacím příkladem

Model by pak měl umět rozeznat, který z příkladů vlevo nebo vpravo na Obr. 21 je trénovacím příkladem. Vizualizace hodnota vah v případě, že jsou nastaveny náhodně, odpovídají obrázku vlevo. Ale pokud je použit *pre-training,* jsou váhy nastaveny na specifické hodnoty, aby odpovídali příkladům na vstupu. Síť by pak měla být schopna rozpoznat, že vpravo se nachází písmeno.

Přidáním dodatečné podmínky, že kromě klasifikace musí síť ještě umět rozpoznat data, by mělo zmenšit prostor možných řešení pro nastavení parametrů.

28 Běžný postup pří získávání modelu je: vytvoření modelu, inicializace parametrů a učení v tomto pořadí. V případě *pre-trainingu* bez učitele je postup následovný: vytvoření modelu, inicializace vah, učení bez učitele a trénování.

Existuje více způsobů před učení bez učitele odvíjejících se od typu sítě. Práce popisuje jeden z nich: hladovou, vrstveně orientovanou proceduru.

Greedy, layer-wise precedure [41]:

- Trénuj vždy jen jednu skrytou[29] vrstvu od první po poslední.
- Zafixuj parametry předešlé naučené vrstvy.
- Aktivaci naučené vrstvy považuj za vstup do právě učící se vrstvy.

Výsledkem *pre-trainingu* je, že první vrstva rozpozná základní rysy, které jsou v trénovacích datech častěji než v libovolné distribuci. Druhá skrytá vrstva rozpozná kombinace těchto rysů, které jsou častější něž náhodné základní rysy skryté vrstvy, atd.

Ve chvíli, kdy je model naučen bez učitele, je použito učení s učitelem. Přidá se výstupní vrstva a nechá se proběhnout klasický učící proces *zpětné šíření chyby* všemi vrstvami. S modelem se tak v této fázi pracuje jako s obyčejnou dopřednou sítí. Nicméně tím, že parametry jsou již nastaveny na hodnoty blízké lokálnímu minimu chybové funkce, dochází jen k jemným úpravám.

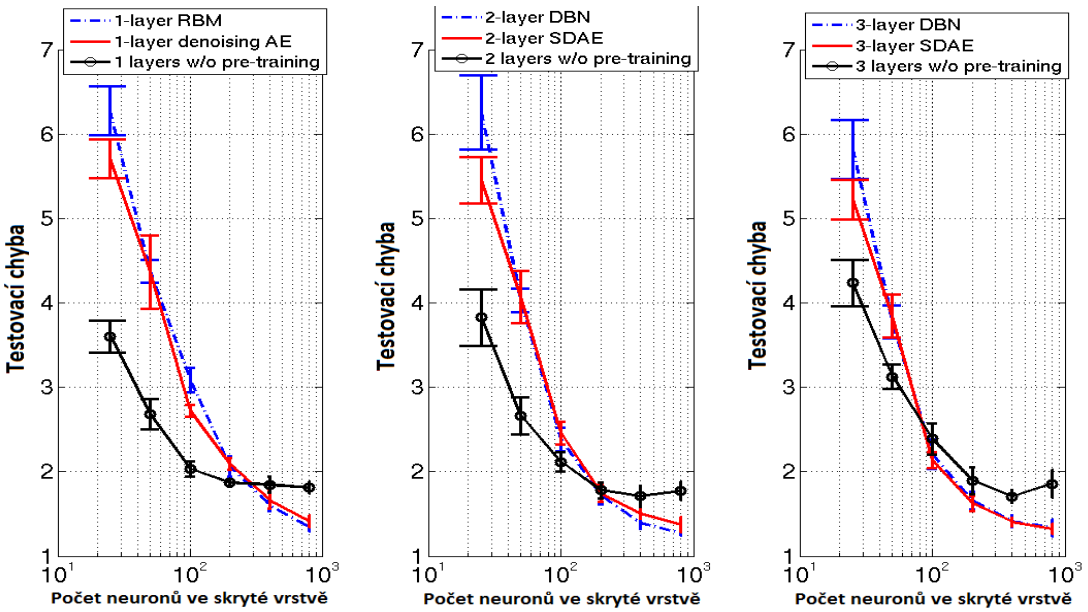
Algoritmus učení bez učitele *deep network* [36]:

- Pro $l=1$ po L – počet skrytých vrstev
  - Vytvoř trénovací množinu bez cílů $(h^{(0)}(\boldsymbol{x})=\boldsymbol{x})$:
    $D=\{h^{(l-1)}(\boldsymbol{x}^{(t)})\}_{(t=1)}^{T}$
  - Nauč vrstvu na datech D
- Použij naučené parametry jako inicializaci parametrů modelu $\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}$.
- Inicializuj náhodně parametry výstupní vrstvy $\boldsymbol{W}^{(L+1)}, \boldsymbol{b}^{(L+1)}$.
- Trénuj celou síť s učitelem a použitím stochastického gradientního sestupu.

Bengio a spol. dále v [41] zmiňují důvod znatelného zlepšení učení, ke kterému dochází u *Deep belief Netowrk* v případě použití hladového algoritmu bez učitele: *pre-training* je významné pro inicializaci parametrů do oblasti, kde je optimalizace jednodušší a kde lepší lokální optimum lze nalézt pro zadaná kritéria.

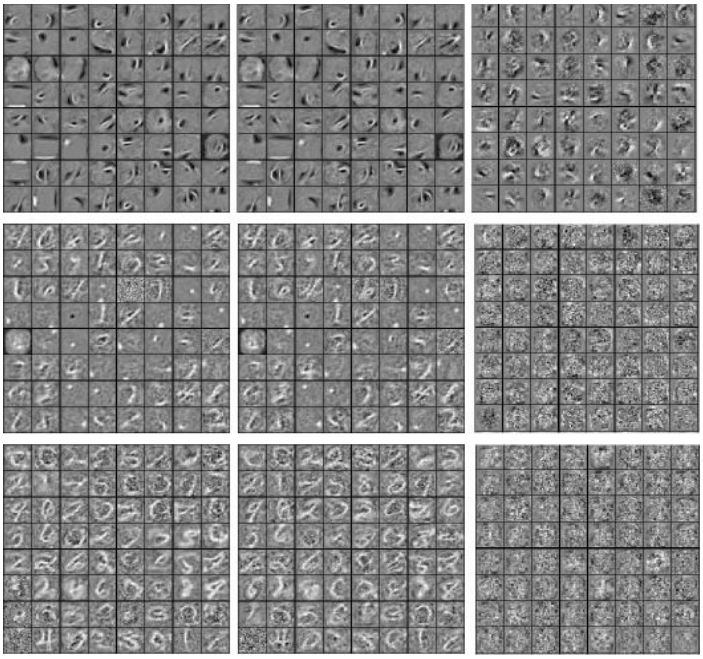
Výsledky nalezené v [43] poukazují na to, že *pre-training* se chová jako určitý druh regularizačního mechanizmu, kdy významně zmenšuje prostor možných řešení a tím snižuje jejich variabilitu (znázorněno na schématu Obr. 22).

29 Vstupní vrstva nemá žádné parametry a výstupní vrstva se učí až klasifikovat případy v rámci učení s učitelem.



Obr. 22: Ukázka funkce učení před tréninkem jako regularizační metoda pro různé modely s jednou, dvěma a třemi vrstvami. Modrá a červená čára znázorňuje chybu pro modely, kde bylo použito pre-trainingu, a černá vyznačuje chybu, kde učení bez učitele chybělo. S nárůstem počtu neuronů, tzn. s růstem komplexity sítě, se funkce protnou a tam kde bylo použito učení bez učitele, modely dosahují lepších výsledků. To je příčinou omezení přeučení. [42]

Další významnou vlastností tohoto postupu je, že u kombinovaného učení nedochází při učení s učitelem k anulování nalezených rysů během učení bez učitele, jak demonstruje následující ukázka.



Obr. 23: Vizualizace výberu naučených filtrů z 3 vrstvého (řádky) DBN modelu. Levý sloupec je po použití pouze pre-trainingu. Střední sloupec ukazuje filtry po kombinovaném učení a pravý znázorňuje situaci po učení jen s učitelem.[43]

6.5 Dropout

Přeučení lze redukovat pomocí metody *dropout*. Tento způsob regularizace předchází komplexní koadaptaci na trénovací data [44], ke které často dochází. Ideou je narušit strukturu neuronové sítě stochastickým vypnutím několika neuronů během učení, tzn. přiřazení nulové aktivační hodnoty náhodným neuronům ve vrstvě, na kterou je *dropout* aplikován. Jiný pohled na proceduru, zmiňovanou v [44], je, že dochází k průměrování modelů. Ověřenou praxí pro snížení testovací chyby je zprůměrování výsledků velkého množství různých modelů. Standardní metodou, jak toho dosáhnout, je zvlášť naučit mnoho sítí, a vytvářet predikce na testovacích datech pro každou zvlášť a výsledky zprůměrovat. Náhodné odpadnutí dává možnost trénovat obrovské množství různých sítí v přiměřeném čase. Je téměř jistě, že se během učení použije pro každý případ jiná struktura modelu [44].

Každému skrytému neuronu je přiřazena pravděpodobnost *p,* s jakou se při dopředném průchodu dat vynuluje. To znamená, že aktivace přibližně *p* procent neuronů, se bude rovnat nule. Důsledkem metody je, že uzly nemůžou kooperovat při hledání rysů, protože libovolné z nich mohou být „vypnuty“. Musí se tedy hledat rysy, které budou užitečné obecně, ne jenom v kombinaci s ostatními aktivacemi neuronů.

V učícím procesu je třeba vynásobit aktivace skrytých vrstev binární maticí masky, jenž určuje, které uzly budou aktivní a které ne. Nutno dodat, že maska se mění pro každý učící příklad. J, jinak nedochází k dostatečné regularizaci modelu [2].

S použitím náhodné binární masky se změní aktivace neuronu z (2) na:

$$h^{(k)}(\boldsymbol{x}) = g(a^{(k)}(\boldsymbol{x})) \circ \boldsymbol{m}^{(k)} \tag{53}$$

a u zpětného šíření chyby se definice delty (18) nahradí:

$$\delta_j = \frac{\partial l}{\partial a_{(k)}(\boldsymbol{x})_j} \circ \boldsymbol{m}^{(k)} \tag{54}$$

Během testování je nezbytné zaměnit masku za očekávanou střední hodnotu. Teoreticky by se dala vypočítat předpověď testovacího případu na všech možných kombinacích struktury modelu a výsledek zprůměrovat

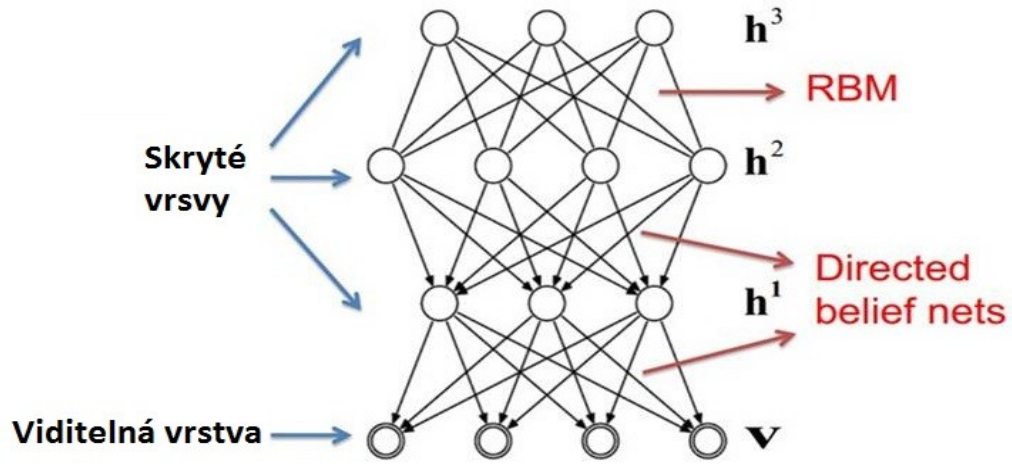
$$1/|M| \sum_{M} a((M * W)\boldsymbol{x}) \tag{55}$$

Avšak těchto různých struktur je $2^{|M|}$, což je reálně nepoužitelné [2]. Proto se jen zamění maska za očekávanou hodnotu, která se rovná pravděpodobnosti odpadnutí. Aktivace skrytého neuronu pak bude:

$$h^{(k)}(\boldsymbol{x}) = g(a^{(k)}(\boldsymbol{x}))\, p \tag{56}$$

6.6 Deep belief network

Základním modelem uváděným v souvislosti s *deep learning* je *deep belief network* (dále jen DBN). Respektive se jedná o model, na kterém Hinton a spol. demonstrovali možnosti využití *pre-trainingu* bez učitele v [1], zásadní objev pro trénování *deep* modelů viz 6.3.



Obr. 24: Struktura DBN. Zdroj [1]

DBNs jsou pravděpodobnostní generativní modely složené z několika vrstev náhodných a skrytých proměnných [45]. Skryté neurony nabývají typicky binárních hodnot, jenž odpovídají Booleovým hodnotám pro výskyt dané vlastnosti v případu. Struktura sítě se skládá ze dvou vrstev, které mají neorientované, symetrické hrany mezi sebou a ze dvou spodních vrstev s orientovanými hranami od vrstvy nad nimi (viz Obr. 24). Poslední spodní vrstva reprezentuje vektor dat.

Síť tedy můžeme rozdělit na dvě části: horní dvě vrstvy, které jsou svojí charakteristikou *Restricted Bolzman machine* (dále jen RBM), a spodní dvě vrstvy, které jsou *sigmoid belief netowrk* (dále jen SBN). Přestože jsou obě části mělké struktury, vyskytují se v dokumentu pouze v rámci DBN. Budou proto stručně definovány v následujících kapitolách.

6.6.1 Restricted Bolzman machine

RBM je dvouvrstvým modelem, který se umí naučit pravděpodobnostní distribuci množiny vstupních případů. Stochastické binární uzly jedné (spodní) vrstvy jsou symetricky, neorientovaně propojeny se stochastickými binárními uzly druhé (horní) vrstvy. Spodní vrstva je vrstvou viditelnou a slouží jako vstup do sítě. Horní vrstva je skrytá a její uzly reprezentuji distribuci rysů nalezených v datech. Spojení neuronů ze dvou vrstev má energii rovnající se:

$$E(\boldsymbol{v},\boldsymbol{h}) = -\sum_{i \in vstup} a_i v_i - \sum_{j \in skrytá} b_j h_j - \sum_{i,j} v_i h_j w_{i,j} \tag{57}$$

, kde

v_i, h_j jsou binární stavy neuronů *i* ve spodní a *j* ve skryté vrstvě,

a_i, b_j jsou prahy těchto neuronů a

$w_{i,j}$ je váha spojení.

Síť přiřazuje pravděpodobnost každému možnému spojení viditelné a skryté vrstvy s použitím energické funkce:

$$p(\boldsymbol{v},\boldsymbol{h})=\frac{1}{Z}e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{58}$$

, kde funkce *Z* je partiční funkce:

$$Z=\sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{59}$$

Pravděpodobnost, kterou síť přiřazuje viditelné vrstvě ***v***, je rovna:

$$p(\boldsymbol{v})=\frac{1}{Z}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{60}$$

Pravděpodobnost, se kterou síť přiřadí trénovací příklad, může být měněna váhami a prahy. To sníží energii případu a zvýší energii ostatním příkladům. Derivace logaritmické pravděpodobností funkce trénovacího vektoru ve vztahu k vahám je:

$$\frac{\partial \log p(\boldsymbol{v})}{\partial w_{i,j}}=\langle v_i, h_j\rangle_{data}-\langle v_i, h_j\rangle_{model} \tag{61}$$

, kde zalomené závorky jsou použity pro očekávanou hodnotu distribuce viz rovnice (63), (64).

Výsledkem je jednoduché učící pravidlo pro stochastický vzestup pravděpodobnostní logaritmické funkce na trénovacích datech:

$$\Delta w_{i,j}=\alpha\left(\langle v_i h_j\rangle_{data}-\langle v_i h_j\rangle_{model}\right) \tag{62}$$

, kde α je učící parametr.

Protože ve skryté vrstvě nejsou žádně vzájemné spojení, je snadné získat očekávanou hodnotu při dané distribuci bez prahu $\langle v_i, h_j\rangle_{data}$. Při náhodném vstupním vektoru ***v***, se stav skrytého binárního neuronu *j* rovná 1 s pravděpodobností:

$$p(h_j=1\,|\,\boldsymbol{v})=\sigma\left(b_j+\sum_i v_i w_{i,j}\right) \tag{63}$$

, kde

$\sigma(x)=1/(1+\exp(-x))$ *sigmoida.*

Protože ani ve viditelné vrstvě nejsou žádná vzájemná spojení, tak očekávaná hodnota bez prahu pro daný skrytý vektor se rovná:

$$p(v_i=1\,|\,\boldsymbol{v})=\sigma\left(a_i+\sum_j h_j w_{i,j}\right) \tag{64}$$

Ovšem získání $\langle v_i h_j\rangle_{model}$ je výpočetně příliš náročné. Proto Hinton v [46] navrhl pracovat s

„rekonstrukční“ očekávanou hodnotou. Ta vloží do vstupní vrstvy vektor trénovacích dat. Poté jsou spočítány binární hodnoty skryté vrstvy na základě rovnice (63). Následuje zpětná rekonstrukce na viditelné vrstvě, kde jsou uzly ex-citovány na hodnotu 1 na základě pravděpodobnosti dané rovnicí (64). Změna vah se pak rovná:

$$\Delta w_{i,j} = \alpha\left(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{rekons}\right) \tag{65}$$

[47]

RBM nachází aplikaci hlavně jako generativní model při učení bez učitele: redukce dimensionality [48], klasifikační úlohy popsané Larochellem a Bengiem v [49], učení se rysům použitý Coatesem a spol. v [50], nebo modelování témat popsaný Hintonem v [51].

6.6.2 Sigmoid belief network

Je druhem ***belief networks*** (též uváděno jako ***Bayesian network***), které patří do rodiny pravděpodobnostních grafických modelů. Tyto struktury se používají pro reprezentaci znalostí ohledně určité domény. Přesněji, každý uzel grafu znázorňuje náhodnou veličinu, zatímco hrany mezi uzly reprezentují pravděpodobnostní závislosti mezi korespondujícími uzly. Pravděpodobnosti jsou často odhadované pomocí statistických a výpočetních metod [52].

V případě *deep belief network*, kde jsou spodní dvě vrstvy SBN, reprezentují podmíněnou distribuci vrstvy rodiče. Tato distribuce je počítána z výstupu rodiče, jako funkční hodnota *sigmoidy*[30] pro potenciál neuronu:

$$p(h_j^{(1)} | \boldsymbol{h}^{(2)}) = sigm\left(b^{(1)} + \boldsymbol{W}^{(2)T} \boldsymbol{h}^{(2)}\right) \tag{66}$$

$$p(v_i | \boldsymbol{h}^{(1)}) = sigm\left(b^{(0)} + \boldsymbol{W}^{(1)T} \boldsymbol{h}^{(1)}\right) \tag{67}$$

6.6.3 Model DBN

Z důvodu charakteristiky modelu RBM, který se skládá z neorientovaných hran, nelze o DBN mluvit jako o dopředné *(feed-forward)* síti.

Celková distribuce model se spočítá jako:

$$p(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}) = p(\boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)})\, p(\boldsymbol{h}^{(1)} | \boldsymbol{h}^{(2)})\, p(\boldsymbol{x} | \boldsymbol{h}^{(1)}) \tag{68}$$

, kde

$$p(\boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}) = \exp\left(\boldsymbol{h}^{(2)T} \boldsymbol{W}^{(3)} \boldsymbol{h}^{(3)} + \boldsymbol{b}^{(2)T} \boldsymbol{h}^{(2)} + \boldsymbol{b}^{(3)T} \boldsymbol{h}^{(3)}\right) / Z \tag{69}$$

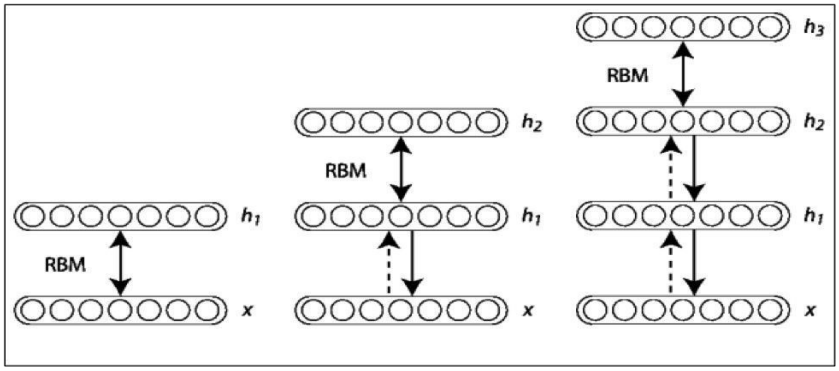
$$p(\boldsymbol{h}^{(1)} | \boldsymbol{h}^{(2)}) = \prod_j p(h_j^{(1)} | \boldsymbol{h}^{(2)}) \tag{70}$$

30 Proto se jedná o ***Sigmoid*** *belief network.*

$$p(\boldsymbol{x}|\boldsymbol{h}^{(1)}) = \prod_i p(x_i|\boldsymbol{h}^{(1)}) \tag{71}$$

6.6.4 Učení modelu

Tento model bylo velice obtížně efektivně naučit až do doby, než byla publikována metoda hladového algoritmu *(greedy, layer-wise procedure)*. Tak, jak je to popsáno v kapitole 4.2, postupuje učení po vrstvách (viz Obr. 25). Nastaví se parametry jednotlivých vrstev na hodnoty jenž jsou použity při inicializaci modelu. *Pre-training* může být následován nějakou další učící procedurou, nejčastěji to bývá *backpropagation*, která jemně doladí váhy a prahy pro lepší generalizaci či rozlišovací schopnost sítě.



Obr. 25: Aplikace hladového učení na DBN po vrstvách.[35]

6.7 Konvoluční neuronové sítě

Konvoluční sítě *(Convolutional neural network, dále jen CNN)* jsou variantou vícevrstvého perceptronu, inspirované biologií. Z práce Hubela a Wiesela [53] je známo, že vizuální kortex má komplexně uspořádané buňky. Každá z těchto buněk přijímá signály pouze z podoblasti, které se říká pole vnímání, celkového zrakového vjemu. Receptivní pole jsou buňkám přiřazeny tak, aby pokrývaly celou oblast vidění.

Dva typy buněk vizuálního kortexu byly rozpoznány: jednoduchá *(simple)* a komplexní *(complex)* buňka. Jednoduchá buňka reaguje na maximálně specifickou hranu, která se vyskytuje v poli jejího vnímání. Komplexní buňka má větší pole vnímaní a je invariantní vůči poloze stimulu [40].

Biologické vizuální rozpoznávání je velmi komplexní a výkonný systém, proto se modely pro počítačové vidění snaží na tento systém navázat a v omezené míře ho napodobit: *NeoCognitron* [40], *HMAX* [54] a *LeNet-5* [23]. Jedná se různé typy konvolučních sítí, které velmi dobře kopírují strukturu vizuálního kortexu, tak jak byla popsána v kapitole 6.2.

Tento druh neuronových sítí je založen na třech základních principech:

1. Lokální propojení
2. Sdílení parametrů
3. Sdružování skrytých uzlů

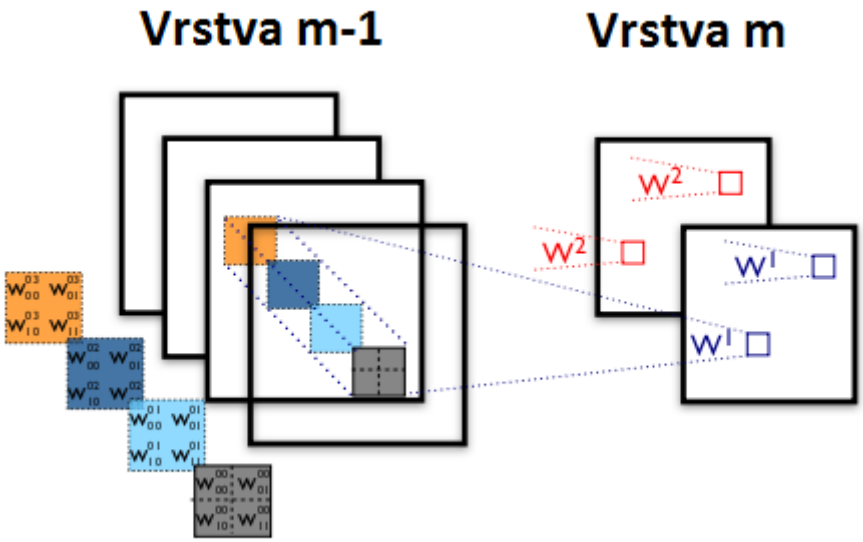
, které zajistí do určité míry odolnost vůči posunům, změně velikosti a zkreslení dat [23].

CNN se využívají převážně v oblasti vizuálního rozpoznávání, kde dosahují znatelně lepších

výsledků, než ostatní modely. S přidáním regularizační metody *drop-connect*[31] dosáhli Wan a spol nejlepšího výsledku 0,23% na datech MNIST [2].

6.7.1 Lokální propojení

Řešení vizuálních úloh je doplněno několika stěžejními částmi. Jednou z nich je velikost vstupních dat. Například černo-bílý obrázek 120 na 120 pixelů obsahuje celkem 144 000 hodnot. A vezme-li se v úvahu barevný obrázek, je zde hodnot třikrát tolik. V případě, že by byla vrstva plně propojena s daty na vstupu, je počet parametrů nezvladatelný. Zároveň prostý výpočet potenciálu neuronu by byl výpočetně velmi složitý. Charakteristické rysy a hrany jsou tedy hledány pouze v malém okolí, receptivním poli přiřazeném neuronu. Stejně jako v biologickém případě i umělá síť těmito neurony pokrývá celou plochu předešlé vrstvy (vstupní, či skryté). Vrstva *m-1* může být buďto vstupní nebo skrytá vrstva (znázorněno na Obr. 26). V případě vstupní vrstvy více panelů reprezentuje jednotlivé barevné kanály. Pokud by na vstupu byl jednobarevný obraz, pak by se ve vrstvě *m-1* nacházel pouze jeden panel, též nazýváno mapou. Potenciál ve vrstvě *m* se počítá z polí vnímání neuronu, které se nacházejí na všech mapách předchozí vrstvy na stejném místě.



Obr. 26: Lokální propojení vrstev CNN, kde potenciál neuronu se rovná sumě přes všechny oblasti vnímání, se kterými má spojení.[69]

Pro lepší ilustraci následuje jednoduchý příklad. Nechť je na vstupu barevný obrázek (RGB) o rozměrech 32x32 pixelů. První skrytá vrstva obsahuje panel o rozměru 28x28 a pole vnímání má velikost 5x5. Pak potenciál uzlu na pozici (1;1) ve vrstvě *m* se spočítá z hodnot v polích {(1;1), (1;2) … (1;5)(2;1)...(5;1) … (5;5)} pro všechny tři kanály.

6.7.2 Sdílení parametrů

Druhou charakteristikou CNN je sdílení parametrů. Jak bylo zmíněno v předchozí kapitole, konvoluční vrstvy jsou rozděleny na mapy. Každá mapa má matice vah o velikosti pole vnímání, které jsou sdíleny všemi neurony v dané mapě. To znamená, že každý uzel v jednom panelu má přiřazenou jinou oblast v předešlé vrstvě o stejné velikosti a stejných vahách. Pokud je vzdálenost mezi neurony v konvoluční vrstvě menší než polovina velikosti pole vnímání, jejich vstupní oblasti se překrývají. Zároveň jsou posunuty právě o vzdálenost, která je mezi uzly.

Počet matic vah, které jsou přiřazené jedné mapě, se rovná počtu kanálů, respektive panelů, se kterou má mapa spojení. Pokud například je vstup barevný, bude mít každá mapa v první

31 Velmi podobné metodě *drop-out*, ale namísto vypínání neuronů se vypínají pouze hrany s pravděpodobností *p*.

konvoluční vrstvě tři matice vah.

Tento způsob sdílení má dvě výhody. Zaprvé dále snižuje počet parametrů sítě a to velmi významným způsobem. Zadruhé každá mapa hledá specifický rys v předešlé vrstvě definovaný maticí vah. Tím, že jsou váhy společné pro každý neuron v panelu, je ten samý rys hledán ve všech oblastech předešlé vrstvy. Pokud je do sítě implementován i práh, pak je jeho hodnota společná pro celou mapu.

Konvoluční vrstva se skládá z několika takových map, kde každá by se měla změřit na hledání jiných obrysů, protože mají různé matice vah. Sekvenční implementace map rysů prohlédne vstup lokálním polem vnímání neuronu a stav přiřadí příslušnému uzlu v mapě. Tento postup je schodný s konvolucí[32]. K hodnotě je případně přidán práh a je vypočtena aktivační hodnota [23].

Hodnota potenciálu neuronu v konvoluční vrstvě se rovná [55]:

$$y_j = b_j + \sum_i k_{ij} * x_i \tag{72}$$

, kde

k_{ij} trénovatelný filtr

a operace * značí 2D diskrétní konvoluci.

Pokud se tyto znalosti zakomponují do příkladu z předcházející kapitoly, tak první vrstva může mít například 6 panelů. Tudíž síť rozezná 6 základních rysů, ze kterých jsou posléze v dalších vrstvách skládány další kombinace, rysy vyšších úrovní. První mapa tedy může hledat svislou přímku, druhá například vodorovnou čáru, atd. Takto nadefinovaná první vrstva bude mít :

- $6 * 28 * 28 = 4704$ neuronů
- $(6 * 28 * 28) * (5 * 5) * 3 = 352\,800$ spojení
- $6 * 3 * (5 * 5) = 450$ vah
- 6 prahů
- $450 + 6 = 456$ trénovatelných parametrů

6.7.3 Sdružování skrytých uzlů

V momentě, kdy je charakteristický rys nalezen, není už tak důležitá jeho pozice. Nutné je pouze uchování relativní pozice vůči ostatním rysům. Lze tedy provést „*pooling / sub-sampling*“ na konvoluční vrstvě. V podstatě se jedná o snížení rozměrů map, včetně proběhnuvších různých výpočetních operací, aby měla transformace určité vlastnosti. Hlavním cílem je snížení velikosti map, aby se snížila výpočetní složitost.

Vrstva provádějící metodu sdružení skrytých uzlů (dále jen *pooling*) vždy následuje za konvoluční vrstvou. Obsahuje stejné množství map, které mají n^2 méně neuronů. Každá pooling mapa je propojena právě s jedním konvolučním panelem. Každý pooling neuron je propojen s lokálním okolím o velikosti *n* na *n* v konvoluční vrstvě. Tyto okolí jsou navzájem disjunktní, tudíž zde nedochází k překryvům.

Pro výpočet hodnoty potenciálu neuronu se používají především dvě základní metody:

Max pooling je velmi často volenou možností. V tomto případě se potenciál neuronu v mapě *i,*

32 Proto název konvoluční.

řádku *j* a sloupci *k* rovná:

$$y_{ijk} = \max_{p,q} x_{i,j+p,k+q} \quad , kde\ p,q = \{0,..,n-1\} \tag{73}$$

Tato úprava má výhodu, že je silně odolná vůči lokálním změnám. Tzn. pokud se změní nějaká hodnota jiná než maximum pro danou lokální oblast, tak se hodnota v pooling vrstvě nemění.

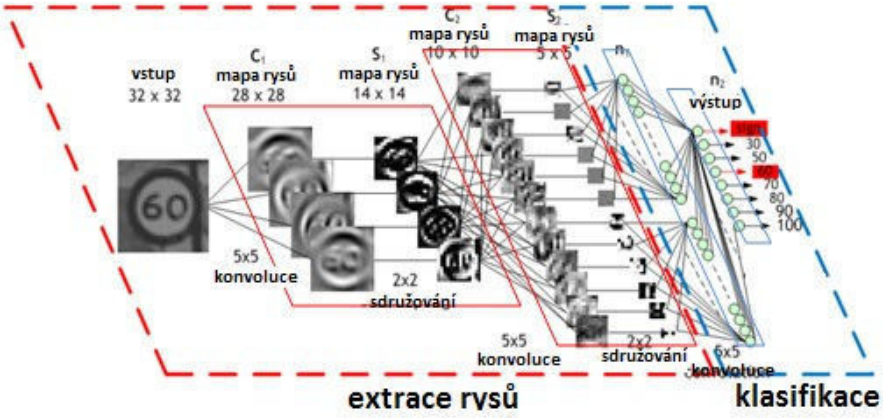
Pooling průměr, jak název napovídá, počítá aritmetický průměr přes všechny hodnoty v okolí. Potenciál neuronu se proto rovná:

$$y_{ijk} = \frac{1}{n^2} \sum_{p,q} x_{i,j+p,k+q} \tag{74}$$

Další možností může být například *stochastický pooling* popsán v [56]. „Stochastický“ v názvu znamená, že náhodně vybírá hodnotu z oblasti vnímání. Metoda tudíž funguje jako jeden z typů regularizace modelu.

6.7.4 Model

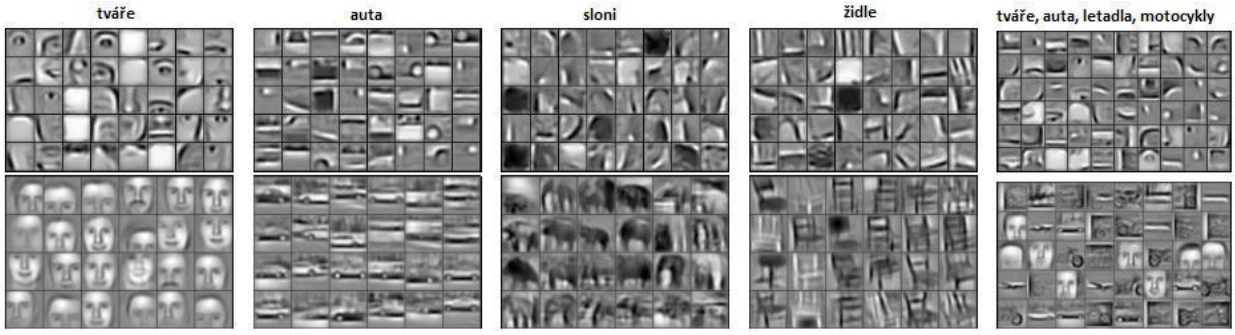
Model *deep* konvoluční sítě se skládá z několika vrstev. Pokud jsou použity *pooling* vrstvy[33], pak vždy následují po konvoluční vrstvě[34] a navzájem se tak střídají (viz Obr. 27).



Obr. 27: CNN model na kterém je znázorněné rozdělení vrstev na dvě skupiny. První část extrahuje rysy z dat a druhá část na základě přijatých impulsů vytváří kalsifikaci.[70]

Extrakce rysů často probíhá na dvou konvolučních vrstvách, kde každá z nich je doplněna vrstvou sdružující. Ovšem u složitějších obrazců může být těchto dvojic více, například u rozpoznávání čínských znaků v [57] V takovém případě první vrstva detekuje základní obrysy, jako jsou přímky, jejich konce, části kruhů, atd. Tyto základní rysy jsou dalšími konvolučními vrstvami skládány do částí objektů a objektů (viz Obr. 28).

33 Pooling vrstvou (sdružující vrstvou) se myslí vrstva, která provádí sdružování skrytých neuronů.
34 Kovoluční vrstvou se myslí vrstva, která provádí konvoluci na předchozí vrstvě.



Obr. 28: Naučené rysy vyšší úrovně pro různé kategorie. Horní řada je pro 2 konvoluční vrstvu, spodní řada pro třetí. [71]

Klasifikace se skládá z jedné či více vrstev. Vždy je obsažena plně propojená výstupní vrstva, která ke klasifikaci nejčastěji používá aktivační funkci *softmax*. Mezi ní a poslední vrstvou extrakce může být vloženo libovolné množství jiných vrstev. Obvykle se vkládá ještě alespoň jedna plně propojená vrstva. Pokud mapy poslední vrstvy extrakce mají rozměry 1x1, pak je vložená vrstva klasickou vrstvou z vícevrstvého perceptronu. V opačném případě zde dochází k ještě jedné konvoluci, kde jádro má velikost dané mapy. Vložená vrstva tak zredukuje velikost na jednu hodnotu [58].

6.7.5 Učení

CNN je učena pomocí metody stochastického gradientního sestupu. Zpětné šíření chyby je podobné jako v případě vícevrstvého perceptronu. Akorát je třeba myslet na specifikace jednotlivých vrstev.

Gradient konvoluční vrstvy pro konvoluční operaci $y_j = x_i * k_{ij}$ gradient x_i je :

$$\nabla_{x_i} l = \sum_j (\nabla_{y_j} l) \underline{*} (\boldsymbol{W_{ij}}) \tag{75}$$

,kde

$\underline{*}$ symbolizuje konvoluční rozšíření s ošetřeným přidáním nulových hodnot.

$\boldsymbol{W_{ij}}$ je matice vah mapy *i* z předešlé vrstvy a mapy *j*.

A gradient pro $\boldsymbol{W_{ij}}$ se rovná:

$$\nabla_{W_{ij}} l = (\nabla_{y_j} l) * \boldsymbol{x_i^T} \tag{76}$$

Gradient pooling vrstvy se rozlišuje dle typu aktivace. Pro dva základní postupy se hodnota chybové funkce *l* šíří:

Pro *max pooling operaci* $y_{ijk} = \max_{p,q} x_{i,j+p,k+q}$ se gradient rovná:

$$\nabla x_{ijk} l = 0 \quad , kromě\ \nabla x_{i,j+\acute{p},k+\acute{q}} l = \nabla y_{ijk} l \tag{77}$$

, kde

$\acute{p}, \acute{q} = argmax\ x_{i,j+p,k+q}$.

V případě použití *aritmetického průměru* $y_{ijk}=\frac{1}{m^2}\sum_{p,q} x_{i,j+p,k+q}$ se gradient x_{ijk} rovná:

$$\nabla_x l=\frac{1}{m^2} UP(\nabla_y l) \tag{78}$$

, kde

UP je opakem sdružování. [59]

Jinak řečeno: Delta neuronů ve vrstvě předcházející konvoluční vrstvě se spočítá jako součet delt neuronů vynásobených váhou, se kterými má spojení. V případě *max pooling* vrstvy se všechny delty rovnají nule pokud se nejedná o neuron, který měl největší hodnotu.

7 Inicializace

Posledním krokem před aplikací trénování neuronové sítě je inicializace parametrů. Tzn. nastavení všech vah a prahů na počáteční hodnoty. Neexistuje žádný přesný algoritmus, který by zaručoval optimální výchozí bod, je zde jen několik pravidel, dle kterých by mělo postupovat:

- Pro bias:
  - většinou se nastavuje na nulovou hodnotu
  - práh nastavený na jinou hodnotu posouvá hranu aktivační funkce, čehož lze využít například pro řídkou aktivaci při zvolení vysoké negativní hodnoty
- Pro váhy:
  - při nastavení vah na nulové hodnoty bude velikost gradientu při zpětném šíření chyby též nulová, tzn. že nebude docházet k učení
  - při nastavení vah na stejnou hodnotu se budou všechny skryté uzly chovat stejně, síť bude symetrická
  - je tedy vhodná inicializace pomocí náhodného rozdělení

7.1 Náhodná inicializace vah

Hodnoty náhodných veličin pro inicializaci by se měly pohybovat v případě logistických aktivačních funkcí v jejich lineární části. Pokud by byly zvoleny příliš velké hodnoty, pak by neurony snadno saturovaly a výsledkem by bylo pomalé učení [23].

Dalším požadavkem je nepřímá úměrnost k velikosti vrstev, které hrany propojují. Jelikož výstup jedné vrstvy je vstupem do další, je snahou zachování podobného rozptylu aktivací a zpětně šířených gradientů.

$$\boldsymbol{W} \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}\right] \tag{79}$$

, kde

U značí rovnoměrné rozdělení,

n_j je počet uzlů vrstvy *j*.

8 Praktická část

Ve všech experimentech pro výstupní vrstvu platí, že:

velikost vrstvy koresponduje s počtem zadaných klasifikačních tříd.

- pro určení předpovědi je použita aktivační funkce *softmax,* jejíž funkční hodnotou je pravděpodobnost příslušnosti případu do odpovídající klasifikační třídy.
- chybovou funkcí je zvolena *cross-entropy error (více-dimenzionální)*.

Časová náročnost jednotlivých výpočtů při zvoleném jazyku a způsobu implementace je vysoká, proto jsou pokusy většinou ukončeny dříve, než proběhne plánovaný počet iterací. K tomuto ukončení je přistoupeno až ve chvíli, kdy se nedá předpokládat další zlepšení testovací chyby a dále dochází jen k přeučení. Tento postup se schoduje s regularizační metodou brzkého zastavení a je tedy považován za korektní. Výsledkem trénování je model, který dosáhl během učení nejlepší hodnotu klasifikace na testovacích datech.

8.1 Vyhodnocení učení

Během učení sítě po každé iteraci jsou zaznamenány tři údaje.

1. ***Trénovací entropie***, jenž je průměrnou hodnotou negativního logaritmu pravděpodobnosti předpovědi správné klasifikace. Entropie se tudíž blíží k nule, čím přesnější je předpověď. Naopak, pokud model špatně klasifikuje trénovací příklad[35], blíží se entropie $+\infty$.
2. ***Testovací entropie*** stejně počítaná jako *trénovací* ovšem na testovacích datech.
3. ***Testovací klasifikační chyba*** je počet špatně klasifikovaných testovacích příkladů. Poměr této hodnoty k velikosti testovací množiny vyjadřuje chybovost modelu.

8.2 Normalizace dat

Proces normalizace je úpravou parametrů dat do vhodnější podoby. Obecně jsou normalizační techniky používány pro snížení variability dat patřící do stejné skupiny. V případě rozpoznávání znaků je předzpracovaní vstupu velmi důležitou součástí, často významnější než samotná optimalizace sítě. Například u obrázků se skládá kompletní normalizace z mnoho postupných kroků: úprava jasu, transformace perspektivy, rotace, změna velikosti.

8.2.1 Rozměr

Vstupní vrstva neuronové sítě má pevnou velikost. Většinou se pro úlohy extrakce rysů a rozeznávání znaků používá standardní čtvercový panel o rozměrech N x N [60]. Všechny originální obrázky, které mají jiné rozměry, musí být upraveny na požadovanou velikost. Většinou se tak děje se zachováním poměru stran, tzn. že poměry stran

R_1 originálního obrázku pro šířku W_1 a výšku H_1 , který je roven

$$R_1 = \frac{min(W_1, H_1)}{max(W_1, H_1)}$$

a R_2 normalizovaného obrázku pro šířku W_2 a výšku H_2 , který je roven

35 Tzn., že hodnota výstupního neuronu cílové třídy je téměř nula. Pak hodnota negativního logaritmu stoupá k nekonečnu.

$$R_2 = \frac{min(W_{2,} H_2)}{max(W_{2,} H_2)}$$

si jsou rovny.

Zde jsou prezentovány dva způsoby mapování originálního obrázku do normalizovaného. Při použití *moment-based* normalizace se mohou některé nové souřadnice pixelů dostat mimo cílovou oblast. Pro $\acute{x} = \acute{x}(x, y)$, $\acute{y} = \acute{y}(x, y)$ a $x = x(\acute{x}, \acute{y})$, $y = y(\acute{x}, \acute{y})$:

	Mapování	**Zpětné mapování**
Lineární	$\acute{x} = \alpha x$ $\acute{y} = \beta y$	$x = \acute{x}/\alpha$ $y = \acute{y}/\beta$
Moment	$\acute{x} = \alpha(x - x_c) + \acute{x}_c$ $\acute{y} = \beta(y - y_c) + \acute{y}_c$	$x = (\acute{x} - \acute{x}_c)/\beta + x_c$ $y = (\acute{y} - \acute{y}_c)/\beta + y_c$

Tabulka 1: Výpočet nových souřadnic normalizovaného obrázku

, kde

$\alpha = W_2/W_1$, $\beta = H_2/H_1$,

těžiště obrázku je dáno

$x_c = m_{10}/m_{00}$, $y_c = m_{01}/m_{00}$,

m_{pq} značí geometrický moment:

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y) \quad ,$$

a $\acute{x}_c \acute{y}_c$ značí geometrické těžiště normalizovaného plátna:

$\acute{x}_c = W_2/2$, $\acute{y}_c = H_2/2$.

9 Datová sada MNIST

Datová sada MNIST[36] je databáze ručně psaných číslic 0-9 (10 klasifikačních tříd) dostupných na http://yann.lecun.com/exdb/mnist/. Je zde k dispozici 60 000 učících vzorů a 10 000 testovacích případů. Jedná se o vybranou podmnožinu z dostupných dat NIST. Černo-bílé příklady z NIST byly velikostně normalizovány, aby se vešli do čtverce o rozměrech 20x20 pixelů se zachováním poměru stran. Zatímco pixely původních dat nabývají pouze binárních hodnot, pixely po normalizaci reprezentují odstín šedé barvy. Je to výsledkem vyhlazení hran (*anti-aliasing technique*). Číslo je poté zasazeno do obrázku o velikosti 28x28, kde těžiště je umístěno doprostřed.

Obr. 29: Ukázka datové sady MNIST. [72]

Sada je velmi dobrým odrazovým můstkem pro otestování různých technik klasifikace a implementace díky množství případů a jednoduché použitelnosti.

9.1 MLP model

Cílem prvního experimentu je ověřit funkčnost implementace základních principů neuronových sítí a vyhodnotit použitelnost odlišných aktivačních funkcí a regularizace „*drop-out*“. Ve všech případech je použita stejná architektura sítě, se stejnými parametry zadání.

Vstupní data jsou normalizována na velikost 20x20 se zachováním poměru stran. Zároveň jsou hodnoty pixelů „vmáčknuty“ do intervalu [0,1]. Pro trénování je použito všech 60 000 trénovacích vzorů a 10 000 odlišných příkladů pro testování. Zadané hodnoty a použité metody učení vycházejí z testů provedených v [2], podrobněji se jimi studie zabývá v následujících kapitolách.

Architektura modelu odpovídá vícevrstvému perceptronu se čtyřmi vrstvami: vstupní, dvou skrytých a výstupní. Velikost vstupní vrstvy koresponduje s velikostí vstupních dat, tzn. 400 vstupních uzlů. Skryté vrstvy mají schodně velikost 400 neuronů z důvodu, že se jedná o nejmenší velikost, kde dle [2] již dochází k přeučení modelu bez použití regularizační metody. Aktivační funkcí skrytých vrstev jsou postupně voleny funkce *sigmoid, hyperbolický tangens a relu.* Velikost výstupní vrstvy kopíruje počet klasifikačních tříd, v tomto případě to je 10 tříd (číslice 0 – 9).

Model je učen klasickou metodou učení s učitelem při použití *mini-batch* o velikosti 100 (viz kapitola 4.10). Inicializační hodnota učícího parametru α z (28) je 0,1 a učící moment (29) má pevně nastavenou hodnotu β na 0,9. Testy jsou naplánovány na 1000 iterací s postupnou změnou učícího koeficientu α po 300 iteracích následovně (0,1; 0,05; 0,01; 0,005).

Ovšem pro ověření cílů tohoto experimentu, jak vyplývá z dosažených průběžných výsledků, stačí provést pouze několik desítek iterací. Z tohoto důvodu jsou jednotlivé učící procesy zastaveny dříve.

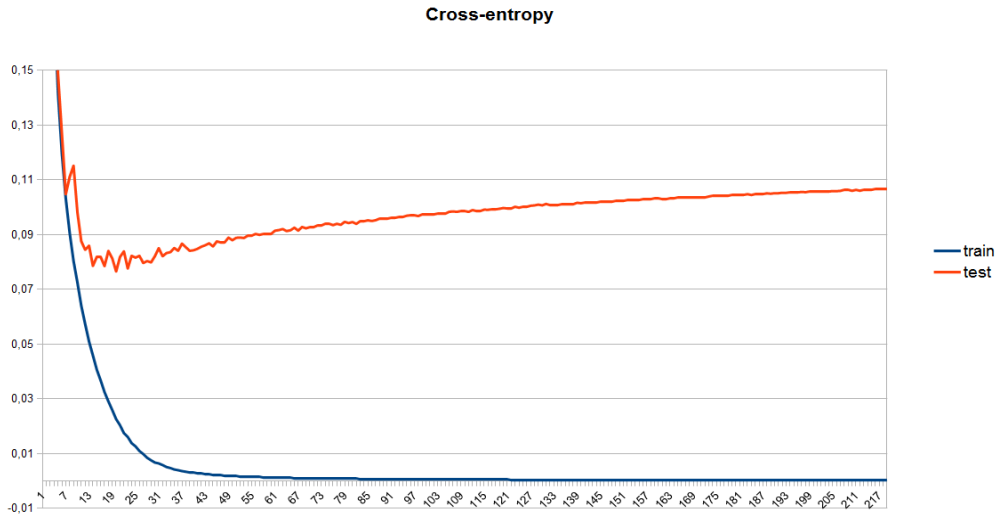
36 Modified NIST

Klasifikační chyba	No-Drop	Drop-Out
Sigmoid	2,01%	1,89%
Tanh	1,96%	1,62%
Relu	1,91%	1,90%

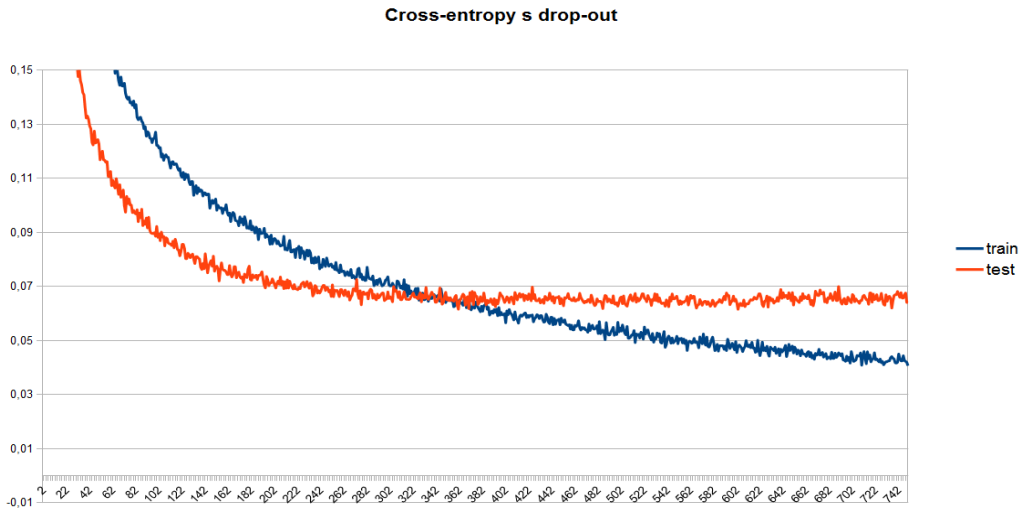
Tabulka 2: Přehled úspěšnosti predikce sítí MLP na datech MNIST. Levý sloupec je pro modely bez použití regularizační metody, v pravém sloupci jsou výsledky při použití "drop-out" na obou skrytých vrstvách. V řádcích se modely liší v použití aktivační funkce skrytých vrstev.

Klasifikační chyba	No-Drop	Drop-Out
Relu	1,55%	1,40%

Tabulka 3: Prezentuje hodnoty pro srovnání dosažených výsledků. Zanesené procentuální chyby jsou hodnoty, kterých dosáhla studie [2] při použití stejného počtu skrytých neuronů a aktivační funkce relu. Hodnoty jsou průměrem predikcí 5 sítí.



Graf 45: Vývoj trénovací a testovací chyby behem 220 epoch modelu s aktivační funkcí tanh. Během prvních 30 iterací dochází k přeučení a klasifikační chyba již dále neklesá.



Graf 46: Vývoj trénovací a testovací chyby během 750 epoch modelu s aktivační funkcí tanh a regularizací drop-out. Je zde velmi dobře patrné, že nedochází k následnému přeučení. Nevýhodou je výrazně pomalejší konvergence do minima.

V rámci implementace modelů MLP pro data MNIST bylo vyzkoušeno i učení ***resilient propagation***. Nicméně výsledky při jeho použití nebyly lepší než v případě klasického *back propagation* učení (3,71% v porovnání s 2,70%) a proto není dále užívána.

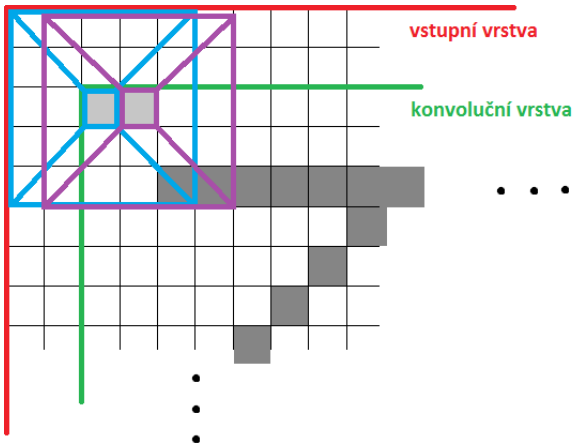
9.2 CNN model

Jak naznačuje mnoho studií [23], [2], [57] při použití konvoluční sítě v modelu by mělo dojít k výraznému zlepšení předpovědi modelem. K ověření této teze slouží následující úloha. Cílem experimentu je vytvoření modelu konvoluční sítě, která bude výrazně lépe klasifikovat testovací příklady datové sady MNIST. Návodem pro dosažení tohoto cíle je výše zmíněná literatura.

Architektura sítě (28x28 – 16C1 – MP1 – 32C2 – MP2 – 150N(C3) – 10N) se skládá ze 7 vrstev.

1. Rozhodnutí o velikosti vstupní vrstvy v případě konvolučních sítí je trochu složitější než u MLP. Jelikož při výpočtu předpovědi dochází ke snižování rozlišení, je třeba zvážit velikost na vstupu tak, aby nedocházelo k dělení se zbytkem. Při použití této konkrétní architektury, zvolené velikosti vnímání 5x5 a sdružovací oblasti 2x2, dochází při průchodu konvoluční a sdružovací vrstvou ke snížení rozměru na $(n-4)/2$. K tomuto snížení dojde dvakrát, je tedy nutné určit rozměr vstupu jako násobek čtyř. Uvažovány jsou dvě možnosti 32x32, nebo 28x28. Zatímco první možnost je uváděna spíše jen v [23], druhá velikost[37] je používáná mnohem častěji, zřejmě i z důvodu nižší výpočetní náročnosti.

   Na Obr. 30 je znázorněno, proč je nutné zachování minimální velikosti vstupních dat (28x28 s číslem o velikosti 20x20). Aby bylo možné zachycení nejkrajnějších charakteristických rysů, pokud je velikost oblasti vnímání 5x5, je třeba mít u každého čísla alespoň 4 bodový okraj.



Obr. 30: Znázornění oblastí vnímání neuronů z první konvoluční vrstvy na vstupní vrstvě.

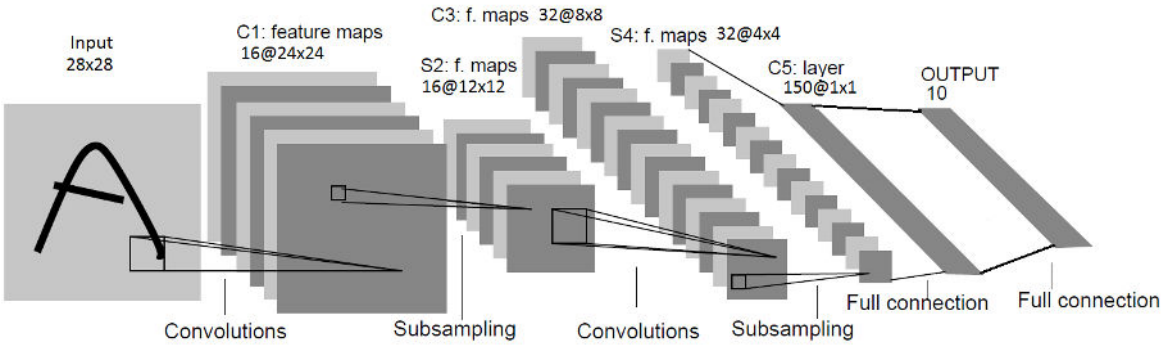
2. 16C1 - U konvoluční vrstvy je třeba rozhodnout pouze o počtu map, jejich velikost je daná rozměry vstupu a oblastí vnímaní neuronů. Jak již bylo zmíněno, velikost jádra je 5x5 a tedy rozměry map jsou 24x24 $(n-4)$. Panely by měli korespondovat s očekávaným výskytem základních rysů. Pro tento experiment jich bylo zvoleno 16.
3. MP1 značí první sdružovací vrstvu typu *Max-Pooling*. Počet panelů se musí rovnat počtu v konvoluční vrstvě. Rozměry map se opět vypočítají na základě velikosti sdružování v tomto případě 12x12 (24/2).
4. 32C2 Druhá konvoluční vrstva má opět velikost panelu danou předchozí vrstvou 12x12 a

37 Možností je též použití vstupu 29x29 s rozměry jádra první konvoluční sítě 4x4 [57].

velikostí jádra konvoluce 5x5, tzn. že mapy jsou o rozměrech 8x8 (12 – 4). Počet panelů je stanoven na 32. Způsob napojení na předešlou vrstvu je předmětem diskuze [23]. Pro tento experiment bude použito plné propojení [57].

5. MP2 je stejného typu jako MP1 a dále snižuje rozměry dat na 4x4 (8/2).
6. 150N(C3) je vrstvou přechodovou mezi částí pro extrakci rysů a klasifikaci. Na vstupu je propojena s MP2. Rozměry oblasti vnímání 4x4 udělají ze vstupních hodnot jednorozměrnou složku, která je dále šířena. Obsahuje tedy mapy o rozměrech 1x1. Počet map je stanoven na 150. Vrstva je plně propojena na vstupní i výstupní straně. Proto se vrstva částečně chová jako konvoluční, částečně jako vrstva MLP.
7. 10N výstupní vrstva odpovídá popisu v úvodu praktické části s 10 uzly.

Aktivační funkcí pro skryté vrstvy je *tanh*. Stejně jako v předchozím případu je učící parametr nastaven na 0,1 a učení probíhá s použitím momentum 0,5.



Obr. 31: Ukázka struktury konvoluční sítě použité na datové sadě MNIST [23]

Experiment

Minimální klasifikační chyba na testovacích datech: 0,74% (74 chyb z 10 000) po průběhu 14. epochy.

9.2.1 Shrnutí

Vzorová studie se stejným modelem konvoluční sítě a použitou *relu* jako aktivační funkce konvolučních vrstev dosáhla klasifikační chyby 0,77%. Tímto je cíl pro experimenty na datové sadě MNIST považován za splněný.

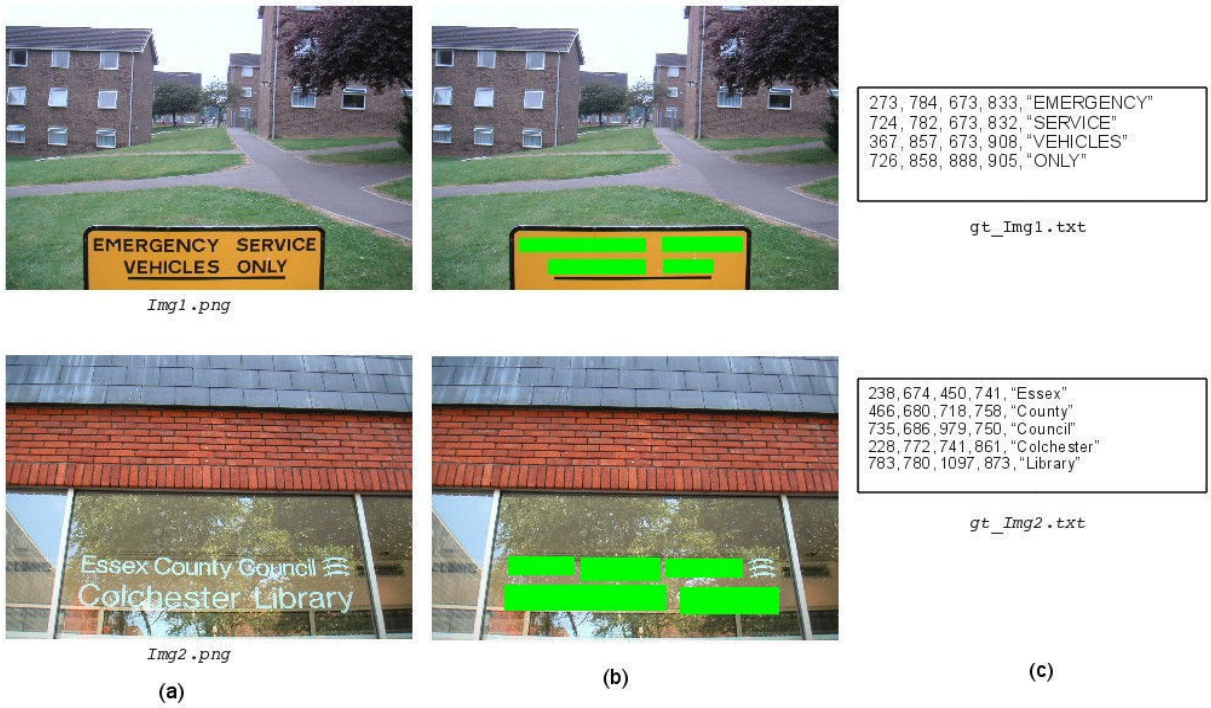
10 Datová sada ICDAR2011

ICDAR (International Conference on Document Analysis and Recognition) je konference pořádaná pro výzkumníky, kteří se zabývají se lokalizací a rozpoznáváním textu. V rámci konference probíhá soutěž pro objektivní porovnání algoritmů, jenž byly vytvořeny pro řešení těchto úloh. Kompletní rozpoznání datové sady ICDAR 2011[38] se skládá se tří částí: lokalizace textu, segmentace textu a rozpoznání textu.

10.1 Rozponávání textu z reálných scén

10.1.1 Lokalizace textu

Tato část je zaměřena na rozpoznání textu v obrázcích. Cílem je najít pozice obdélníků, které opíší jednotlivá slova.



Obr. 32: Ilustrace trénovacího příkladu pro lokalizaci textu. V sloupci (a) je originální obrázek, (b) značí požadovanou lokalizaci textu a (c) prezentuje požadovaný výstup kompletního rozpoznání. [74]

K lokalizaci je možné přistoupit dvěma způsoby. Jedná se o metody jenž přesouvají po obrazu pole, které text detekuje, a metody založené na spojování znaků. Postupem prvního typu je použití oblasti, která je postupně přesouvána přes obraz a o přítomnosti písmene rozhodují lokální rysy obrazu. Zatímco je tato metoda velmi odolná vůči šumu, její negativní stránkou je náročnost celého procesu, kdy je potřeba detekci opakovat pro různé rozměry a natočení oblasti detekce.

Používanějším postupem je druhá možnost. Jednotlivé algoritmy se odlišují v přístupu pro samotnou detekci znaků. Jsou založené na detekcích hran, výpočtu energie znaku nebo hledání množiny extermálních oblastí. Více informací je možné naleznout v [61].

10.1.2 Segmentace textu

Po nalezení umístění textu, je nutná jeho extrakce z obrázku. Cílem tohoto procesu je získání

38 Dostupné na http://dag.cvc.uab.es/icdar2011competition/?com=tasks.

binární masky pro text vyskytující se na obrázku. Oba tyto procesy nejsou předmětem práce a nebudou tedy dále rozebírány.

10.1.3 Rozpoznávání textu

Cílem etapy rozpoznávání je správná klasifikace jednotlivých znaků, které byly vy-segmentovány z obrázku. Protože se práce zabývá pouze touto třetí pod úlohou rozpoznávání, jsou znaky předem vyextrahovány pomocí algoritmu *TextSpotter*[39] z datové sady ICDAR 2011.

10.2 Datová sada

Data jsou rozdělena na dvě skupiny. Testovací data jsou získána ze zmiňované trénovací datové sady *ICDAR 2011 Robust Reading Competition Challenge.* Jedná se většinou o fotografie reálných scén, kde se nachází tištěný text. Vyskytující se znaková sada obsahuje malá a velká písmena latinské abecedy a číslice 0-9, tzn. 62 klasifikačních tříd. Pomocí algoritmu *TextSpotter* jsou vy-segmentovány jednotlivé znaky, které jsou použity jako testovací sada pro ověření předpovědí sítě. Jelikož jsou znaky extrahovány z reálných scén, jsou jejich proporce, fonty a zkreslení značně různorodé.

Druhou množinou jsou data trénovací. Pro zajištění vysoké generalizace modelu, jsou učící data automaticky generována z dostupných fontů na stroji. Na množství, typu a úpravách trénovacích dat silně závisí úspěšnost modelu.

10.3 Model

Cílem experimentů na této sadě je získání modelu s nejnižší klasifikační chybou na testovacích datech. Základem pro vytvoření modelu jsou znalosti získané při práci na MNIST datech. Ovšem použitý konvoluční model je možné modifikovat mnohými způsoby. Každý z parametrů je proto teoreticky rozebrán a vyvozené důsledky jsou prakticky ověřeny na ICDAR datech. K poskytnutí porovnatelných výsledků je vždy použit stejný základní model, u kterého je změněn jeden parametr. Predikce upravených modelů jsou navzájem porovnány.

10.4 Výchozí model

Výchozím modelem je konvoluční síť: 32x32 – 16C1 – SP1 – 32C2 – SP2 – 120N – 62N.

Struktura kopíruje architekturu modelu použitého na datech MNIST s několika odlišnostmi:

- Vstup sítě je o rozměrech 32x32, protože znaky jsou normalizovány na velikost 28x28 při zachování poměru stran. Rozšíření je dáno potřebou, aby krajní rysy byly nalezeny uprostřed pole vnímání první konvoluční vrstvy.
- Konvoluční a sdružené vrstvy tedy musejí mít jiné velikosti map, vypočtené na základě vstupu a jádra konvolucí a sdružení viz Tabulka 4.
- Sdružující vrstvy „SP“ jsou typy *average pooling.*
- Poslední skrytá vrstva je o velikosti 120.

Rozměry	**16C1**	**SP1**	**32C2**	**SP2**	**120N**
Pole vnímání	5x5	2x2	5x5	2x2	5x5
Mapa	28x28	14x14	10x10	5x5	1x1

Tabulka 4: Rozměry pole vnímání a map jednotlivých vrstev základního modelu konvoluční sítě

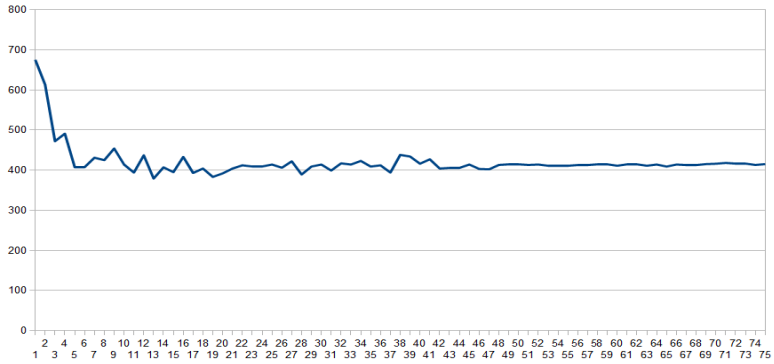
39 http://www.textspotter.org/

Základní parametry učení jsou nastaveny takto:

- učící parametr $\alpha = 0{,}1$
- *momentum* $\beta = 0{,}5$
- *mini-barch* = 100

Experiment

Minimální klasifikační chyba na testovacích datech: 18,6% (379 chyb z 2038) po průběhu 16. epochy.



Graf 47: Testovací klasifikační chyba v průběhu 75 epoch

10.5 Parametry modelu

V této kapitole jsou otestovány možné modifikace konvolučního modelu, které by mohly přinést snížení klasifikační chyby na testovacích datech. V závěru této části jsou výsledky shrnuty a zhodnoceny.

10.5.1 Velikost trénovací množiny

Intuitivní představou je, že čím je větší množství trénovacích dat, tím se model lépe naučí. Ke stejným závěrům došel i Ellis v [62]. Nicméně, větší objem dat zvyšuje časovou náročnost trénování modelu, zvláště pokud je zvolenou metodou učení *batch učení*. V případě testů na datech ICDAR je podstatné, aby trénovací data obsahovala dostatečně obsáhlou a pokud možno co nejvíce různorodou množinu jednotlivých znaků. Oba tyto požadavky budou splněny, pokud budou vzory generovány z dostatečného počtu odlišných fontů.

Experiment

Test je proveden na téměř poloviční množině trénovacích dat (2397 namísto 4266). Trénovací vzory jsou rovnoměrně rozložené mezi jednotlivé znaky.

Minimální klasifikační chyba na testovacích datech: 20,8% (425 chyb z 2038) po průběhu 16. epochy.

10.5.2 Velikost *mini-batche*

V kapitole 4.10 bylo zmíněno, že *mini-batch* učení kombinuje výhody obou způsobů trénování: stochastické a *batch*. Je ovšem otázkou, jak velká by měla být dávka. Většinou se zmiňuje velikost v řádech jednotek, či desítek. Větší dávka již zbytečně zvyšuje nutný počet vykonaných epoch [63].

Experiment

Cílem je zjištění, zda snížení parametru *mini-batch* bude mít vliv na klasifikační chybu. Pokud by se její hodnota snížila nebo zůstala stejná, neznamenalo by to jen zlepšení předpovědí, ale i zrychlení učícího procesu.

Minimální klasifikační chyba na testovacích datech: 19,5% (398 chyb z 2038) po průběhu 6. epochy.

10.5.3 Aktivační funkce *relu*

Dle výsledků testů provedených na datech MNIST by mohlo při použití aktivační funkce *rectifier* mohlo dojít ke snížení chybovosti. Aktivační funkce *relu* je též používána Wanem a spol. v [2].

Experiment

Cílem je zjištění, zda změna aktivační funkce konvolučních vrstev na *relu* bude spojena se snížením klasifikační chyby. Zároveň je nutné změnit aktivační funkci vrstvám sdružujícím na lineární[40].

Minimální klasifikační chyba na testovacích datech: 23,7% (484 chyb z 2038) po průběhu 6. epochy.

10.5.4 Počet map konvolučních vrstev

Další možností úpravy modelu je změna počtu map. Jak z [57] a mnohých dalších prací vyplývá, s rostoucí složitostí roste i složitost a kapacita optimálních modelů pro generalizaci dat. Je zřejmé, že čínská abeceda má více charakteristických rysů, ze kterých se skládají jednotlivé znaky, než písmena latinské abecedy. Nicméně rozhodnutí o počtech těchto rysů je mnohem složitější. A právě tyto počty by měla kopírovat struktura modelu.

Experiment

Cílem testu je zjištění, zda zdvojnásobení počtu panelů prvních dvou konvolučních vrstev z (16,32) na (32,64) povede ke zlepšení klasifikace modelem. Zároveň je nutné zvýšit počty map vrstev sdružujících na stejnou hodnotu.

Minimální klasifikační chyba na testovacích datech: 19,1% (390 chyb z 2038) po průběhu 19. epochy.

10.5.5 Regularizace drop-out

V [64] a [2] je diskutována možnost použití techniky regularizace *drop-out*, která zabrání přeučení a zlepší klasifikační chyby modelu. Je akorát nutné rozhodnout o tom, kde bude „odpadávání“ umístěno.

Experiment

Zlepší se klasifikační chyba při přidání regularizační techniky *drop-out* do poslední konvoluční vrstvy na výstupu?

Minimální klasifikační chyba na testovacích datech: 22,6% (460 chyb z 2038) po průběhu 36. epochy.

40 V tomto případě je použití *relu* nebo *linearní* funkce zaměnitelné z důvodu, že sdružující vrstva vytváří prostý aritmetický průměr hodnot z prostoru sdružování.

10.5.6 Řídké propojení sdružující vrstvy

Konvoluční vrstva, která má za předka *pooling* vrstvu, respektive vrstvu s panely, může být připojena dvěma způsoby. První z nich, který se používá ve větší míře je plné propojení (základní model) [57], [58]. Druhý způsob je řídké propojení, tak jak ho naznačuje LeCunn v [23]. Náhodné nebo předem definované řídké propojení má výhodu ve snížení počtu parametrů a rozbití symetrie.

Experiment

Zjištění, zda řídké propojení vrstev s panely zlepší predikci modelu. Způsob propojení je definovaný předem dle klíče použitého v [21].

Minimální klasifikační chyba na testovacích datech: 19,4% (395 chyb z 2038) po průběhu 16. epochy.

10.5.7 Změna velikosti jádra

Velikost oblasti vnímání konvoluční vrstvy je především učena na základě velikosti plochy, ze které se rysy extrahují. Respektive záleží na velikosti charakteristických rysů. Ty samozřejmě rostou se zvětšující se plochou. Je proto nutné zvolit takové rozměry, aby jádro dokázalo pojmout daný rys, ale zároveň ne příliš veliké z důvodu snížené rozlišovací schopnosti. Pro první konvoluční vrstvu je tak možné uvažovat i o jádru o velikosti 7x7.

Experiment

Cílem je zjištění, zde zvětšení oblasti vnímání neuronů první konvoluční vrstvy povede ke snížení chybovosti predikce na testovacích datech.

Minimální klasifikační chyba na testovacích datech: 19,6% (401 chyb z 2038) po průběhu 28. epochy.

10.5.8 Typ pooling vrstvy

Aktivaci neuronu ve sdružující vrstvě je možné spočítat více způsoby viz kapitola 6.7. Základní model používá průměrování hodnot z oblasti vnímání. Druhou, často používanou, možností je výběr maximální hodnoty a upravení zpětného šíření gradientu. Tento typ byl použit v modelu na sadě MNIST.

Experiment

Dojde ke zlepšení klasifikace při použití *max-pooling* vrstev namísto základního typu *sub-sampling*?

Minimální klasifikační chyba na testovacích datech: 20,1% (410 chyb z 2038) po průběhu 13.

10.5.9 Shrnutí výsledků

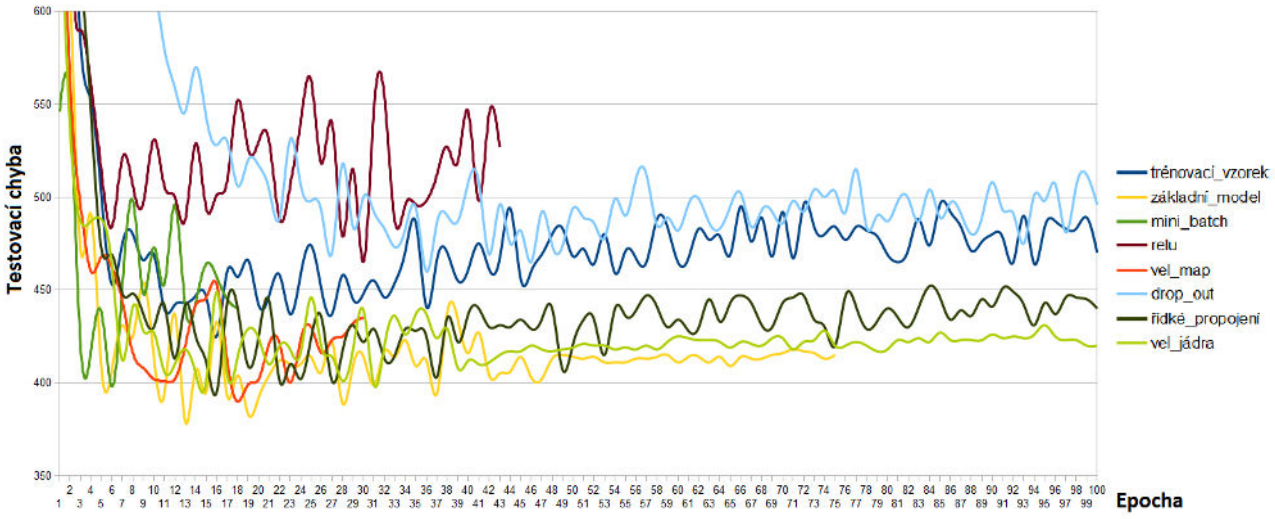
Z výsledků zanesených v Tabulka 5, lze pro další testy na datech ICDAR vyvodit následující úvahy:

- Základní model poskytuje nejlepší klasifikační výsledky.
- Při zvětšení trénovací množiny, respektive zvětšení počtu fontů, ze kterých jsou trénovací data generována, dochází ke zlepšení výsledků.
- Snížení dávky způsobuje značné zrychlení učícího procesu, ovšem zvyšuje chybovost modelu.
- Aktivační funkce relu nedosahuje lepších výsledků, tak jak to platilo na datech MNIST.

- Regularizace *drop-out* zatím není efektivní.
- Řídké napojení konvoluční vrstvy není potřeba.
- Zvětšení map nevede ke zlepšení. Při zachování velikosti vstupních dat je velikost jádra 5x5 dostačující.
- Při použití sdružujících vrstev typu *max-pooling* nedojde ke zlepšení predikce.

	Testovací chyba %	**Testovací chyba abs**	**V průběhu epochy**
Základní model	**18,6**	379	13
Trénovací vzorek (#4266 → #2397)	20,8	425	16
Mini-batch (vel 100 → 10)	19,5	398	6
Aktivace Relu	23,7	484	6
Zvětšení # map (16,32 → 32,64)	19,1	390	19
Drop-out	22,6	460	36
Řídké spojení	19,4	395	16
Velikost jádra (5x5 → 7x7)	19,6	401	28
Max-pooling	20,1	410	13

Tabulka 5: Srovnání modifikací sítě



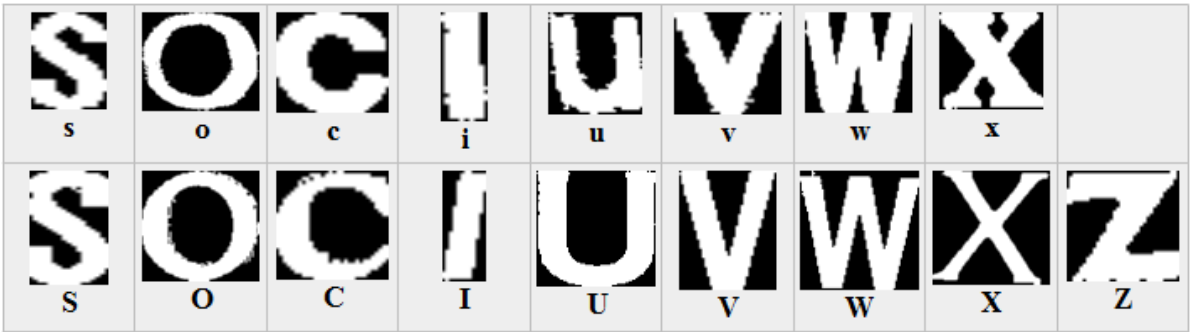
Graf 48: Vývoj klasifikační chyby na testovacím vzroku jednotlivých modifikací základního modelu s proměnlivou délkou experimentu. Je zde vidět, že vždy dochází k postupnému ustálení, či mírnému nárůstu hodnoty testovací chyby po 40. epoše.

10.6 Úprava dat

10.6.1 Podobné znaky

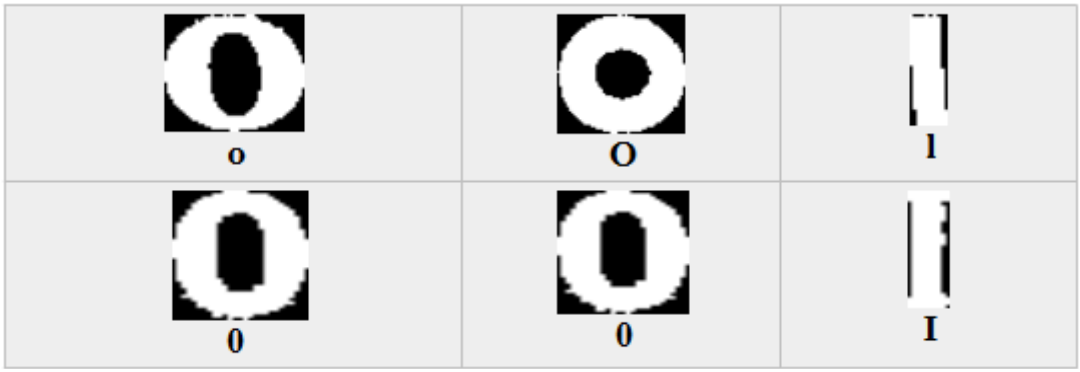
Jedním ze závěrů předešlého experimentu může být mínění, že při zvažovaných úpravách modelu je možnost snížit klasifikační chybu o jednotky procent. Ovšem snahou je tuto chybu minimalizovat, aby byly předpovědi přesnější. K tomuto je možné se dopracovat další analýzou dat a chybných klasifikací. Matice předpovědí poukazuje na několik znaků, které jsou od sebe velmi obtížně rozeznatelné, jak ukazuje následující tabulka. Tato písmena lze rozdělit do dvou kategorií:

- Podobné malé a velké písmeno (s, o, c, i, u, v, w, x, z)



Obr. 33: Ukázka podobnosti malých a velkých písmen v datové sadě. V horním řádku se nacházejí malá písmena a v dolním velká.

- Písmeno podobné jinému znaku (o, O, l)



Obr. 34: Ukázka podobnosti znaků o, O, l, I v datové sadě ICDAR 2011

Následující tabulka ukazuje úspěšnost správné klasifikace těchto znaků.

	TP	FN / FP
s	24	66/26
o	29	35/18
O	19	12/62
c	18	33/10
i	0	0/48
l	4	10/43
u	6	4/2
v	5	6/1
w	1	1/1
x	4	7/0
z	0	0/0

Tabulka 6: Přehled špatně rozpoznatelných znaků. TP - true positive, správně klasifikované (s ->s). FN - false negative, klasifikované do jiné třídy (s->y). FP - false positive, jiná třída špatně klasifikovaná (y->s).

Z několika následujících důvodů jsou tyto třídy sjednoceny a nejsou již dále rozlišována některá malá a velká písmena. Zároveň jsou sjednoceny i znaky druhé kategorie, ty které si jsou navzájem velmi podobné. Tímto se zmenšil počet klasifikačních tříd z 62 na 51 znaků.

- Pro metodu, která pracuje na základě rozeznávání pomocí charakteristických rysů, je nemožné rozeznat stejně vypadající znaky.
- Často vůbec nezáleží na tom, zda jsou písmena klasifikována jako velká či malá.
- Pokud je podstatné rozlišit sjednocené znaky, je třeba provést další analýzu zkoumaného textu na základě výstupu ze sítě. Tento postup je nutný vždy provést pro správnou klasifikaci odlišných znaků, které stejně vypadají. Správnou třídu lze zjistit v rámci dalšího zpracování, například analýzou okolních písmen a použitím slovníku.
- V průběhu učení se síť může snažit nalézt nepatrné rozdíly mezi podobně vypadajícími znaky. To se ovšem vylučuje s principem generalizace nalezeých rysů.

Experiment

Nejlepším modelem na upravených datech je opět model základní s rozšířením poslední vrstvy na 150 uzlů.

Minimální klasifikační chyba na testovacích datech: 5,2% (106 chyb z 2038) po průběhu 91. epochy.

10.6.2 Změna normalizace dat

Test provedený na datech, u kterých nebyl zachován poměr stran při změně velikosti, měl vyšší klasifikační chybu 6,5% (133 chyb z 2038) v průběhu 20 epochy. Zajímavým zjištěním ovšem je, že se základním modelem mají pouze 17 společných chyb.

Na základě této zkušenosti jsou vyzkoušeny další dvě možnosti úpravy struktury vstupních dat. První je *moment-based* normalizace. Při normalizaci množin trénovacích a testovacích dat kolem svého těžiště by teoreticky mělo dojít k přeskupení hodnot pixelů tak, aby se v klasifikačních

třídách nalézaly co nejmenší rozdíly.

Druhou možností je rozšíření trénovací množiny dat o různá zkreslení. Používají se úpravy jako například změna poměru stran, natočení, oříznutí obrázku, atd.

	Testovací chyba %	**Testovací chyba abs**	**V průběhu epochy**
Základní model	5,2	106	91
Bez zachování poměru stran	6,5	133	20
Moment-based normalizace	8,9	182	64
Rozšíření trénovacích dat[41]	7,9	161	22

Tabulka 7: Přehled klasifikačních chyb při různých modifikacích datové sady

10.6.3 Příklady klasifikací

Obr. 35: Ukázky typyckých chyb při klasifikaci. Odshora: 1)Chyby, které se objevují napříč modely. 2) Chyby, které se vyskytují pouze u modelu se zachováním poměru stran. 3) Chyby, které se vyskytují u modelu, který nezachovává poměr stran příkladů.

Na vybraných příkladech (uvedených na Obr. 35) je demonstrováno, kde jsou slabiny naučeného modelu. Typické chyby se objevují u znaků netradičních fontů, které se výrazně odlišují od těch, na kterých byla síť trénována a to i v případech, kdy je znak velmi dobře čitelný. Při zachování poměru stran se stávají problematické data s vysokým poměrem stran[42]. Změna poměru stran a modifikace obrázku do čtverce o rozměrech vstupních dat deformuje příklad tak, že není dále rozpoznatelný.

41 Dle výsledků předchozího testu jsou trénovací data rozšířena pouze o vzory s poměrem výška : šířka = 3:1.
42 Resp. poměr výšky : šířce.

Správné klasifikace	 B	 r	 D	 C
Správná klasifikace při zachování poměru stran	 T	 Z	 y	 c
Správná klasifikace při nezachování poměru stran	 Z	 D	 H	 8

Obr. 36: Ukázky správných klasifikací. Odshora: 1)Správné klasifikace, které se objevují napříč modely. 2) Správné predikce, které se vyskytují pouze u modelu se zachováním poměru stran. 3) Spravné predikce, které se vyskytují u modelu, který nezachovává poměr stran příkladů.

Ukázka správné klasifikace sítí (viz Obr. 36) prezentuje silné stránky zvoleného modelu. Metodě často stačí pouze části základních rysů proto, aby dokázala znak správně určit. Pokud nejsou zachovány poměry stran, model správně klasifikuje velkou část chyb, které vznikají při zachování poměru. Takto jsou převážně odstraněny chyby u znaků, které mají vysoký poměr stran, respektive nedochází k jejich deformaci při normování vstupu.

11 Datová sada ICDAR 2003

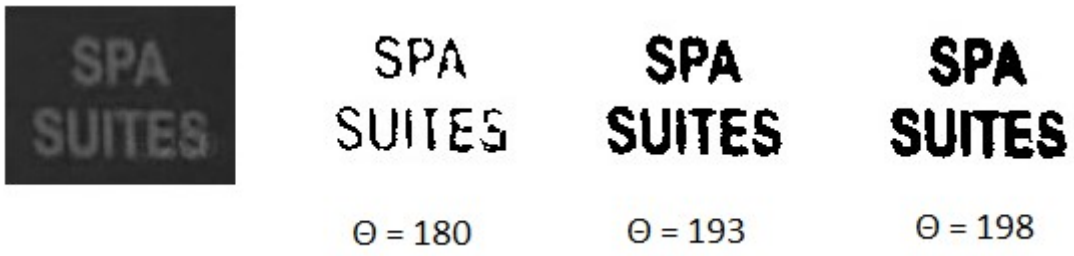
V roce 2003 v rámci konference *ICDAR* byla publikována datová sada[43] pro porovnání algoritmů pro rozpoznávání textu. Otestování modelů na těchto datech má dva důvody:

1. Sada je rozdělena na jednotlivá písmena oproti celým fotkám v případě 2011. Tím se odbourává závislost na úspěšnosti extrakce jednotlivých písmen z obrázků.
2. Možnost porovnat úspěšnost klasifikace s výsledky ostatních studií.

Obr. 37: Ukázka znaků z datové sady ICDAR 2003

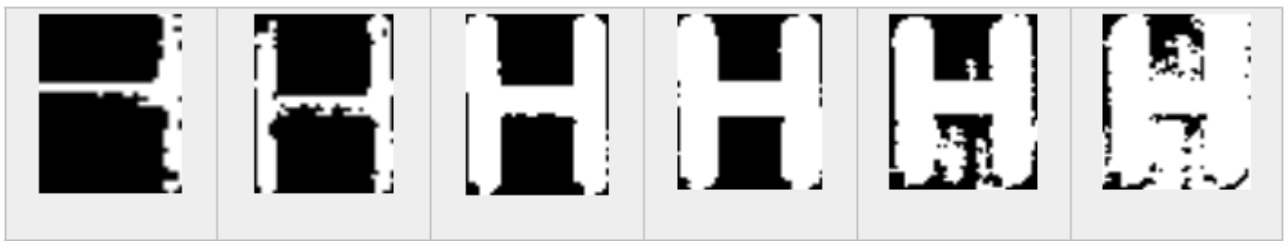
11.1 Normalizace dat

Znaky je třeba převést do černo-bílé podoby. K tomuto kroku je opět použit *Text-Spotter*. Při převodu je možné zvolit libovolný prah, jehož hodnota rozhodne, zde daný pixel připadne ke skupině extrémní oblasti[44], podrobný popis je možné nalézt v práci Neumanna a Matase [73].



Obr. 38: Ukázka možné segmentace znaků z obrázku při použití programu text-spotter s rozdílnou hodnotou Θ. Jelikož není hrana písmen jednoznažná, nelze přesně určit práh.

Důsledkem možnosti volby prahu je vytvořeno několik množin dat, které se liší volbou hrany pro bílou a černou barvu. Znaky jsou poté sjednoceny na velikost 28x28 pixelů.



Obr. 39: Ukázka převedení znaku do jednobarevné podoby při použití odlišných hodnot prahu

43 http://algoval.essex.ac.uk/icdar/Datasets.html

44 Nechť existuje mapování C jenž každému pixelu nabývající typicky hodnot z $\{0,\ldots,255\}^3$ přiřadí hodnotu z uspořádané množiny $\{0,\ldots,255\}$, pak pixel p patří do extrémní oblasti právě když $C(p)>\theta$.

11.2 Model

Architektura sítě se schoduje s základním modelem z předchozího experiment: 32x32 – 16C1 – SP1 – 32C2 – SP2 – 150N – 62N. Nicméně pro možnosti srovnání výsledků je třeba síť opět rozšířit na celou skupinu 62 klasifikačních tříd.

Experiment

Testovací chyba je počítána na kompletní množině normalizovaných obrázků. Tzn. že k jednomu případů existuje více černo-bílých reprezentací, které se liší použitým prahem při normalizaci. Za úspěšnou klasifikaci příkladu je považováno, pokud síť správně uhodne znak na alespoň jedné z jeho reprezentací. Testovací množina se skládá z 5430 případů, které jsou reprezentovány 67 408 obrázky [65].

Minimální klasifikační chyba na testovacích datech: 25,5% (1383 chyb z 5430) po průběhu 90. epochy.

Pro srovnání s byl test proveden i na neúplné klasifikační množině, bez rozlišení velkých a malých písmen, tzn. pouze 36 tříd.

Minimální klasifikační chyba na testovacích datech: 16,2% (883 chyb z 5430) po průběhu 10. epochy.

11.2.1 Příklady klasifikací

Typické chybné klasifikace lze rozdělit na několik množin.

- Navzájem si podobné znaky. Tato množina je v případě testu bez malých písmen výrazně nižší, ale stále tu jsou problémy s klasifikováním některých případů, například znak „l“.
- Obrázky, které nemohou být rozpoznány z důvodu špatné extrakce znaku z originální předlohy.
- Příklady napsány ve fontech, které trénovací množina neobsahuje.
- V této datové sadě se vyskytuje „*ampersand*“, který nebyl začleněn množiny možných klasifikací.

Naopak síť si velmi dobře poradila s jinými případy. Ukázky správné predikce korespondují s poznatky z předešlé datové sady. Je zde především znázorněno, že síť nepotřebuje celé písmeno, pokud vstupní data poskytují alespoň charakteristické rysy daného znaku.

Chybné	 S → {c, s}	 c → {P, m, W}	 a → {3, A, W}	 R → {n,G}
Správné	 S	 J	 R	 A

Obr. 40: Ukázka chybných a správných klasifikací základním modelem na datech ICDAR 2003

Chybné	 U → {N, 3, K}	 E → {A, N ,M}	 B → {T, 7, M}	 & → {A, M, V}
Správné	 M	 V	 B	 I

Obr. 41: Ukázka chybných a správných klasifikací základním modelem na datové sadě ICDAR 2003 bez malých písmen, tzn. 36 tříd.

12 Implementace

Všechny popsané modely v praktické části jsou implementovány v programovacím jazyku Java SE 7 ve vývojovém prostředí NetBeans IDE 7.4. Tento jazyk byl zvolen z několika důvodů:

- Praktická zkušenost a znalost jazyka
- Přehledná struktura objektově orientovaného jazyka (dále jen OOP)
- Obsáhlost diskuzí a příkladů

Velkou nevýhodou při používání Javy je rychlost a paměťová náročnost viz následující tabulka:

Model	**Počet parametrů**	**Počet hran**	**Velikost trénovací + testovací množiny**	**Doba proběhnutí jedné epochy (min)**
MLP (400-400-400-10)	3,24 * 10^5	3,24 * 10^5	60 000 + 10 000	1,6
CNN (784–16C1–P1–32C2–P2 –150–10)	7,95*10^4	1,96*10^6	60 000 + 10 000	46
CNN (1024–16C1–P1–32C2–P2 –150–10)	1,1 * 10^5	1,7 * 10^6	4514 + 2038	5,2
CNN (1024–32C1–P1–64C2–P2 –120–10)	2,5*10^5	6,0*10^6	4514 + 2038	32,5

Tabulka 8: Porovnání časové náročnosti[45] provedení jedné epochy na různých modelech neuronových sítí a datových množinách. Testy byly prováděny na procesoru Intel Core i7-3520M CPU @ 2.90GHz s operační pamětí 4GB.

Vysoký nárůst časové náročnosti v případě konvolučních sítí lze přičíst hlavně objektově orientované struktuře programu. Zatímco síť MLP se skládá pouze z několika více-dimenzionálních polí, v případě CNN je využito všech principů OOP.

45 Časové údaje jsou orientační. Hodnoty silně fluktuovali v závislosti na aktuálních parametrech sítě a celkové vytíženosti výpočetních zdrojů.

13 Závěr

Cílem práce bylo vytvoření modelu neuronové sítě pro rozpoznávání znaků z reálných obrázků. V teoretické části studie prezentuje veškeré teoretické poznatky nutné k pochopení dané problematiky. Zaměření této části je ovlivněno výběrem modelu konvoluční sítě s *deep* architekturou. Kapitola 3 o neuronových sítí je proto převážně zaměřena na vícevrstvý perceptron a kapitola 6 *deep learning* je zaměřena na detailní popis architektury konvoluční sítě. Na těchto teoretických základech je pak postavena praktická část, kde jsou jednotlivé modely otestovány na třech různých datových sadách.

13.1 Dosažené výsledky

V rámci diplomové práce se podařilo implementovat metodu popsanou ve studii [2], která dosahuje nejlepších výsledků na standardní datové sadě MNIST, a získat srovnatelné výsledky. Model sítě byl poté upraven pro použití na datech, které korespondují se zadáním úlohy. Touto datovou sadou se stala množina znaků ICDAR 2011, ze které byly jednotlivé znaky (číslice, velká a malá písmena latinské abecedy) vyextrahovány pomocí programu *text-spotter*.

Na datech bylo provedeno mnoho testů s různými parametry sítě pro získání optimální kombinace nastavení. Nejlepších výsledků (klasifikační chyba **18,6%**) dosáhl základní model s architekturou konvoluční sítě [32x32 – 16C1 – SP1 – 32C2 – SP2 – 120N – 62N] a aktivační funkcí *hyperbolický tangens* ve všech skrytých vrstvách. Tento model byl dále otestován na upravených množinách dat. Analýza výsledků experimentů ukázala na hlavní problém, který nastává při rozlišování některých navzájem si velmi podobných písmen. Sjednocením problematických skupin byla vytvořena nová datová sada s 52 třídami, na které model dosáhl výrazně lepších výsledků (**5,2%**).

Třetí sadou, na které byl model otestován, byla množina obrázků ICDAR 2003. Zde model dosáhl mírně horších výsledků v porovnání s prací [65] jak na celé množině tříd (**25,5%** v porovnání s 18,3%) tak i na množině bez rozlišení velkých a malých písmen (**16,2%** v porovnání s 14,5%). Nicméně model ze studie [65] byl trénován přímo přiřazené trénovací sadě, která se výrazně více podobá testovacím datům, zatímco navrhované řešení pracuje se syntetickými vzory.

13.2 Přínos a aplikace řešení

Práce ověřila použitelnost konvoluční sítě s *deep* architekturou na obrázcích znaků s vysokou variabilitou tvaru, velikostí a kvalitou. Tyto prvotní testy dosáhly velmi dobré úspěšnosti a jsou příslibem pro vytvoření algoritmů, které budou dosahovat nejlepších výsledků i na těchto datových sadách.

Při učení sítě byly použity syntetické učící vzory. Důsledkem může být mírné zhoršení výsledků na specifické sadě dat. Tento postup ovšem přináší značné výhody. Nabízí se možnost vygenerování neomezeného počtu trénovacích vzorů odlišujících se fonty a znakovou sadou, které jsou pro natrénování neuronových sítí nezbytné. Druhou výhodou je větší generalizace rysů, které se síť naučí. V tomto případě se klasifikace nezaměřuje na specifické rysy datové sady, ale na rysy jednotlivých znaků přes celé spektrum tvarů a podob.

Navržený model byl otestován na číslicích a písmenech latinské abecedy, ale teoreticky zde není omezení týkající se použité abecedy. Při aplikaci modelu na jinou sadu znaků je jen potřeba upravit velikosti vrstev, aby například nedocházelo k nedoučení sítě. Správně modifikovaná a naučená síť by pak měla mít velmi rychlou a kvalitní rozpoznávací schopnost, použitelnou například pro okamžité rozpoznání textu v mobilních aplikací.

13.3 Náměty na budoucí rozšíření

Experimenty ukázaly, že použitý algoritmus je značně závislý na kvalitě a obsáhlosti trénovacích dat. Ke zlepšení predikce by mohlo dojít po rozšíření trénovací množiny. Různé natočení, ořezávání a změny poměrů stran trénovacích vzorů vedou k větší odolnosti sítě vůči těmto odchylkám. Druhou volbou je rozšíření seznamu fontů, ze kterých jsou trénovací data generována. Tento postup by měl zajistit lepší rozlišovací schopnost nestandardních tvarů písmen.

Výrazný rozdíl v klasifikační chybě modelu u dat ICDAR 2011 a 2003 lze vysvětlit kvalitou testovacích dat. Důkladnější analýza závislosti správné predikce sítě na předzpracování dat by mohla přinést zvýšení úspěšnosti modelu.

Pro aplikaci algoritmu a jeho rozšíření by měl být použit jiný programovací jazyk. Vysoká časová a paměťová náročnost pří použití jazyka *Java* téměř znemožňují další navyšování struktury sítě nebo vstupních dat. Jako nejvhodnější způsob implementace se jeví programy napsané pro spuštění na GPU. Tento postup je popsán Wanem a kol. v [2].

Posledním popsaným námětem je důkladnější zpracování hodnot predikovaných sítí. V aktuálním algoritmu dochází k určení predikce pouze na základě nejvyšší hodnoty pravděpodobnosti přiřazení případu k dané třídě. Následná analýza výstupních dat by mohla predikci upravit dle vložených externích znalostí. Pokud například dojde ke sjednocení tříd, jejichž znaky jsou si velmi podobné a pro síť tedy těžko rozpoznatelné, s pomocí vloženého slovníku lze jednotlivé možné tvary slov přijmout či zavrhnout.

Literatura

[1] HINTON, Geoffrey E., Simon OSINDERO a Yee-Whye TEH. A fast learning algorithm for deep belief nets. *Neural computation*. 2006, roč. 18, č. 7, s. 1527–1554.

[2] WAN, Li, Matthew ZEILER, Sixin ZHANG, Yann L. CUN a Rob FERGUS. Regularization of neural networks using dropconnect. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* [online]. 2013, s. 1058–1066 [vid. 17. duben 2014]. Dostupné z: http://machinelearning.wustl.edu/mlpapers/papers/icml2013_wan13

[3] MUNOZ, Andres. Machine Learning and Optimization [online]. nedatováno [vid. 13. duben 2014]. Dostupné z: http://www.cims.nyu.edu/~munoz/files/ml_optimization.pdf

[4] VAPNIK, Vladimir N. *Statistical learning theory (adaptive and learning systems for signal processing, communications and control series)* [online]. B.m.: John Wiley & Sons, New York. A Wiley-Interscience Publication, 1998 [vid. 13. duben 2014]. Dostupné z: http://jwpdf.yombabook.com/vladim_statistical_learning_theory_pdf_2784284.pdf

[5] MITCHELL, Tom M. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*. 1997, roč. 45.

[6] KŘIVAN, M. *Úvod do umělých neuronových sítí. Vysoká škola ekonomická v Praze, Nakladatelství Oeconomica. 2008*. B.m.: ISBN 978-80-245-1321-8. 2008.

[7] MUDR. RADIM JANČÁLEK, Ph D. a Masaryk UNIVERSITY. *Základy neurověd v zubním lékařství* [online]. 16. prosinec 2010 [vid. 14. duben 2014]. Dostupné z: http://portal.med.muni.cz/clanek-560-zaklady-neuroved-v-zubnim-lekarstvi.html

[8] BRUNAK, Søren a Benny LAUTRUP. *Neural networks: computers with intuition*. B.m.: World Scientific, 1990.

[9] JAIN, Anil K., Jianchang MAO a K. Moidin MOHIUDDIN. Artificial neural networks: A tutorial. *IEEE computer*. 1996, roč. 29, č. 3, s. 31–44.

[10] MCCULLOCH, Warren S. a Walter PITTS. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*. 1943, roč. 5, č. 4, s. 115–133.

[11] PETERSON, Carsten a T. S. RÖGNVALDSSON. *An introduction to artificial neural networks* [online]. 1991 [vid. 6. duben 2014]. Dostupné z: http://cds.cern.ch/record/231036

[12] ROSENBLATT, Frank. *Two theorems of statistical separability in the perceptron* [online]. B.m.: United States Department of Commerce, 1958 [vid. 6. duben 2014]. Dostupné z: http://aitopics.org/sites/default/files/classic/TeddingtonConference/Teddington-3.4-Rosenblatt-TwoTheorems.pdf

[13] BISHOP, Christopher M. *Pattern recognition and machine learning* [online]. B.m.: springer New York, 2006 [vid. 6. duben 2014]. Dostupné z: http://soic.iupui.edu/syllabi/semesters/4142/INFO_B529_Liu_s.pdf

[14] NAIR, Vinod a Geoffrey E. HINTON. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* [online]. 2010, s. 807–814 [vid. 6. duben 2014]. Dostupné z: http://machinelearning.wustl.edu/mlpapers/paper_files/icml2010_NairH10.pdf

[15] KRENKER, Andrej, J. BESTER a Andrej KOS. Introduction to the artificial neural networks. *Artificial neural networks: methodological advances and biomedical applications. InTech, Rijeka.*

ISBN. 2011, s. 978–953.

[16] HORNIK, Kurt. Approximation capabilities of multilayer feedforward networks. *Neural networks*. 1991, roč. 4, č. 2, s. 251–257.

[17] CHAPELLE, Olivier, Bernhard SCHÖLKOPF a Alexander ZIEN. *Semi-supervised learning* [online]. B.m.: MIT press Cambridge, 2006 [vid. 9. duben 2014]. Dostupné z: ftp://icksie.no-ip.org/EBooks/Computers/Artificial%20Intelligence/Semi-supervised%20learning.pdf

[18] GHAHRAMANI, Zoubin. Unsupervised learning. In: *Advanced Lectures on Machine Learning* [online]. B.m.: Springer, 2004 [vid. 9. duben 2014], s. 72–112. Dostupné z: http://link.springer.com/chapter/10.1007/978-3-540-28650-9_5

[19] CULIOLI, J. C. Introduction à l'Optimisation, 1994. *Ellipses, Paris*. nedatováno.

[20] Florent BRUNET. *Contributions to Parametric Image Registration and 3D Surface Reconstruction* [online]. B.m.: Universite d'Auvergne. 30. listopad 2010. Dostupné z: http://www.brnt.eu/publications/brunet2010phd.pdf

[21] BISHOP, Christopher M. *Neural networks for pattern recognition*. B.m.: Oxford university press, 1995.

[22] MARTENS, James a Ilya SUTSKEVER. Training deep and recurrent networks with hessian-free optimization. In: *Neural Networks: Tricks of the Trade* [online]. B.m.: Springer, 2012 [vid. 9. duben 2014], s. 479–535. Dostupné z: http://link.springer.com/chapter/10.1007/978-3-642-35289-8_27

[23] LECUN, Yann, Léon BOTTOU, Yoshua BENGIO a Patrick HAFFNER. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, roč. 86, č. 11, s. 2278–2324.

[24] BOTTOU, Léon a Patrick GALLINARI. *A framework for the cooperation of learning algorithms* [online]. B.m.: Université Paris-Sud, Centre d'Orsay, Laboratoire de recherche en Informatique, 1991 [vid. 10. duben 2014]. Dostupné z: http://pdf.aminer.org/000/514/999/a_framework_for_the_cooperation_of_learning_algorithms.pdf

[25] KIŞI, O. a E. UNCUOGLU. Comparison of three back-propagation training algorithms for two case studies. *Indian Journal of Engineering and Materials Sciences*. 2005, roč. 12, č. 5, s. 434–442.

[26] RIEDMILLER, Martin a Heinrich BRAUN. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: *Neural Networks, 1993., IEEE International Conference on* [online]. B.m.: IEEE, 1993, s. 586–591 [vid. 11. duben 2014]. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=298623

[27] LECUN, Yann A., Léon BOTTOU, Genevieve B. ORR a Klaus-Robert MÜLLER. Efficient backprop. In: *Neural networks: Tricks of the trade* [online]. B.m.: Springer, 2012 [vid. 12. duben 2014], s. 9–48. Dostupné z: http://link.springer.com/chapter/10.1007/978-3-642-35289-8_3

[28] HESKES, Tom M. a Bert KAPPEN. On-line learning processes in artificial neural networks. *North-Holland Mathematical Library*. 1993, roč. 51, s. 199–233.

[29] HEATON, Jeff. *Programming neural networks with encog 2 in java.* St. Louis, MO: Heaton Research, Inc., 2010. ISBN 9781604390070 1604390077.

[30] SARLE, Warren S. Stopped training and other remedies for overfitting. In: *Proceedings of the 27th Symposium on the Interface of Computing Science and Statisfi (.'. _$\backslash$‘. pp. 352-360. Interface Foundation of North America, Fairfax Station. VA, USA* [online]. 1995 [vid. 12. duben 2014]. Dostupné z: http://www.lib.ctgu.edu.cn:8080/wxcd/qw/1329.pdf

[31] FINNOFF, William, Ferdinand HERGERT a Hans Georg ZIMMERMANN. Improving model selection by nonconvergent methods. *Neural Networks*. 1993, roč. 6, č. 6, s. 771–783.

[32] PRECHELT, Lutz. Early stopping-but when? In: *Neural Networks: Tricks of the trade* [online]. B.m.: Springer, 1998 [vid. 13. duben 2014], s. 55–69. Dostupné z: http://link.springer.com/chapter/10.1007/3-540-49430-8_3

[33] JIN, Yaochu, Tatsuya OKABE a Bernhard SENDHOFF. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In: *Evolutionary Computation, 2004. CEC2004. Congress on* [online]. B.m.: IEEE, 2004, s. 1–8 [vid. 14. duben 2014]. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1330830

[34] H\A ASTAD, Johan a Mikael GOLDMANN. On the power of small-depth threshold circuits. *Computational Complexity*. 1991, roč. 1, č. 2, s. 113–129.

[35] BENGIO, Yoshua. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning* [online]. 2009, roč. 2, č. 1, s. 1–127 [vid. 16. duben 2014]. ISSN 1935-8237, 1935-8245. Dostupné z: doi:10.1561/2200000006

[36] GLOROT, Xavier a Yoshua BENGIO. Understanding the difficulty of training deep feedforward neural networks. In: *International Conference on Artificial Intelligence and Statistics* [online]. 2010, s. 249–256 [vid. 15. duben 2014]. Dostupné z: http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS2010_GlorotB10.pdf

[37] WEBER, Markus, Max WELLING a Pietro PERONA. *Unsupervised learning of models for recognition* [online]. B.m.: Springer, 2000 [vid. 15. duben 2014]. Dostupné z: http://link.springer.com/chapter/10.1007/3-540-45054-8_2

[38] SERRE, Thomas, Gabriel KREIMAN, Minjoon KOUH, Charles CADIEU, Ulf KNOBLICH a Tomaso POGGIO. A quantitative theory of immediate visual recognition. *Progress in brain research*. 2007, roč. 165, s. 33–56.

[39] ROLLS, Edmund T. Invariant visual object and face recognition: neural and computational bases, and a model, VisNet. *Frontiers in Computational Neuroscience* [online]. 2012, roč. 6, s. 35 [vid. 15. duben 2014]. Dostupné z: doi:10.3389/fncom.2012.00035

[40] FUKUSHIMA, Kunihiko. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* [online]. 1980, roč. 36, č. 4, s. 193–202 [vid. 16. duben 2014]. ISSN 0340-1200, 1432-0770. Dostupné z: doi:10.1007/BF00344251

[41] BENGIO, Yoshua, Pascal LAMBLIN, Dan POPOVICI a Hugo LAROCHELLE. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*. 2007, roč. 19, s. 153.

[42] POULTNEY, Christopher, Sumit CHOPRA a Yann L. CUN. Efficient learning of sparse representations with an energy-based model. In: *Advances in neural information processing systems* [online]. 2006, s. 1137–1144 [vid. 17. duben 2014]. Dostupné z: http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2006_804.pdf

[43] ERHAN, Dumitru, Yoshua BENGIO, Aaron COURVILLE, Pierre-Antoine MANZAGOL, Pascal VINCENT a Samy BENGIO. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*. 2010, roč. 11, s. 625–660.

[44] HINTON, Geoffrey E., Nitish SRIVASTAVA, Alex KRIZHEVSKY, Ilya SUTSKEVER a Ruslan R.

SALAKHUTDINOV. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* [online]. 2012 [vid. 17. duben 2014]. Dostupné z: http://arxiv.org/abs/1207.0580

[45] HINTON, Geoffrey. Deep belief networks. *Scholarpedia* [online]. 2009, roč. 4, č. 5, s. 5947 [vid. 18. duben 2014]. ISSN 1941-6016. Dostupné z: doi:10.4249/scholarpedia.5947

[46] HINTON, Geoffrey E. Training products of experts by minimizing contrastive divergence. *Neural computation*. 2002, roč. 14, č. 8, s. 1771–1800.

[47] HINTON, Geoffrey. A practical guide to training restricted Boltzmann machines. *Momentum*. 2010, roč. 9, č. 1, s. 926.

[48] HINTON, Geoffrey E. a Ruslan R. SALAKHUTDINOV. Reducing the dimensionality of data with neural networks. *Science*. 2006, roč. 313, č. 5786, s. 504–507.

[49] LAROCHELLE, Hugo a Yoshua BENGIO. Classification using discriminative restricted Boltzmann machines. In: *Proceedings of the 25th international conference on Machine learning* [online]. B.m.: ACM, 2008, s. 536–543 [vid. 22. duben 2014]. Dostupné z: http://dl.acm.org/citation.cfm?id=1390224

[50] COATES, Adam, Andrew Y. NG a Honglak LEE. An analysis of single-layer networks in unsupervised feature learning. In: *International Conference on Artificial Intelligence and Statistics* [online]. 2011, s. 215–223 [vid. 22. duben 2014]. Dostupné z: http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS2011_CoatesNL11.pdf

[51] SALAKHUTDINOV, Ruslan a Geoffrey E. HINTON. Replicated Softmax: an Undirected Topic Model. In: *NIPS* [online]. 2009, s. 1607–1614 [vid. 22. duben 2014]. Dostupné z: https://papers.nips.cc/paper/3856-replicated-softmax-an-undirected-topic-model.pdf

[52] BEN-GAL, Irad. Bayesian networks. *Encyclopedia of statistics in quality and reliability* [online]. 2007 [vid. 18. duben 2014]. Dostupné z: http://onlinelibrary.wiley.com/doi/10.1002/9780470061572.eqr089/full

[53] HUBEL, D. H. a T. N. WIESEL. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*. 1968, roč. 195, č. 1, s. 215–243. ISSN 0022-3751, 1469-7793.

[54] SERRE, Thomas, Lior WOLF, Stanley BILESCHI, Maximilian RIESENHUBER a Tomaso POGGIO. Robust object recognition with cortex-like mechanisms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 2007, roč. 29, č. 3, s. 411–426.

[55] LECUN, Yann, Koray KAVUKCUOGLU a Clément FARABET. Convolutional networks and applications in vision. In: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on* [online]. B.m.: IEEE, 2010, s. 253–256 [vid. 21. duben 2014]. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5537907

[56] ZEILER, Matthew D. a Rob FERGUS. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557* [online]. 2013 [vid. 21. duben 2014]. Dostupné z: http://arxiv.org/abs/1301.3557

[57] CIRESAN, Dan, Ueli MEIER a Jürgen SCHMIDHUBER. Multi-column deep neural networks for image classification. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* [online]. B.m.: IEEE, 2012, s. 3642–3649 [vid. 21. duben 2014]. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6248110

[58] CIREŞAN, Dan C., Ueli MEIER, Jonathan MASCI, Luca M. GAMBARDELLA a Jürgen

SCHMIDHUBER. Flexible, high performance convolutional neural networks for image classification. In: *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two* [online]. B.m.: AAAI Press, 2011, s. 1237–1242 [vid. 21. duben 2014]. Dostupné z: http://dl.acm.org/citation.cfm?id=2283603

[59] BOUVRIE, Jake. Notes on convolutional neural networks [online]. 2006 [vid. 21. duben 2014]. Dostupné z: http://cogprints.org/5869/

[60] CHMIELNICKI, WIES\LAW a K. STAPOR. Investigation of normalization techniques and their impact on a recognition rate in handwritten numeral recognition. *Schedae Informaticae*. 2010, roč. 19, s. 53–78.

[61] NEUMANN, Lukas a Jiri MATAS. Text localization in real-world images using efficiently pruned exhaustive search. In: *Document Analysis and Recognition (ICDAR), 2011 International Conference on* [online]. B.m.: IEEE, 2011, s. 687–691 [vid. 25. duben 2014]. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6065399

[62] ELLIS, Dan a Nelson MORGAN. Size matters: An empirical study of neural network training for large vocabulary continuous speech recognition. In: *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on* [online]. B.m.: IEEE, 1999, s. 1013–1016 [vid. 26. duben 2014]. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=759875

[63] WILSON, D. Randall a Tony R. MARTINEZ. The general inefficiency of batch training for gradient descent learning. *Neural Networks*. 2003, roč. 16, č. 10, s. 1429–1451.

[64] SRIVASTAVA, Nitish. *Improving neural networks with dropout* [online]. B.m., 2013 [vid. 27. duben 2014]. University of Toronto. Dostupné z: http://www.cs.toronto.edu/~nitish/msc_thesis.pdf

[65] COATES, Adam, Blake CARPENTER, Carl CASE, Sanjeev SATHEESH, Bipin SURESH, Tao WANG, David J. WU a Andrew Y. NG. Text detection and character recognition in scene images with unsupervised feature learning. In: *Document Analysis and Recognition (ICDAR), 2011 International Conference on* [online]. B.m.: IEEE, 2011, s. 440–445 [vid. 5. květen 2014]. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6065350

[66] *Introduction to Artificial Neural Networks - Part 1* [online]. [vid. 6. květen 2014]. Dostupné z: http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7

[67] SHAH, Anuj R., Christopher S. OEHMEN a Bobbie-Jo WEBB-ROBERTSON. *SVM-HUSTLE—an iterative semi-supervised machine learning approach for pairwise protein remote homology detection* [online]. [vid. 6. květen 2014]. Dostupné z: http://bioinformatics.oxfordjournals.org

[68] *Overfitting* [online]. 2014 [vid. 10.dubna 2014]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Overfitting&oldid=605016809

[69] *Convolutional Neural Networks (LeNet)* [online]. 2014 [vid 28.dubna 2014]. Dostupné z: *http://deeplearning.net/tutorial/lenet.html*.

[70] *Neural network simulation: the recognition application* [online]. 2014 [vid 30.dubna 2014]. Dostupné z: http://parse.ele.tue.nl/education/cluster0

[71] *Tea Time With: CDBN* [online]. 2014 [vid 30.dubna 2014]. Dostupné z: http://www.chioka.in/tea-time-with-convolutional-deep-belief-networks-for-scalable-unsupervised-learning-of-hierarchical-representations/

[72] *Sample images in MNIST database* [online]. 2014 [vid 2.května 2014]. Dostupné z:

http://www.cad.zju.edu.cn/home/dengcai/Data/MNIST/images.html

[73] NEUMANN, Luka a Jiri MATAS. On Combining Multiple Segmentations in Scene Text Recognition. In: *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on* [online]. B.m.: IEEE, 2013, s. 523–527 [vid. 6. květen 2014]. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6628675

[74] *Robust reading competition challenge 2: Reading text in scene images* [online]. 2014 [vid 2.května 2014]. Dostupné z: http://robustreading.opendfki.de/wiki/SceneText

Příloha A - Seznam použitých zkratek

ANN	Artificial neural network
CNN	Convolution neural network
DBN	Deep belief network
FNN	Feed-forward neural network
ICDAR	International conference on document analysis and recognition
IDE	Integrated development environment
MLP	Multilayer perceptron
MP	Max pooling layer
OOP	Object-oriented programming
RBN	Restricted Boltzmann machine
SBN	Sigmoid belief nets

Příloha B - Obsah přiloženého CD

Přiložené CD obsahuje následující adresáře:

Datasets Datové sady použité v rámci experimentů

Results Dosažené výsledky navrhovaného řešení na uvedených datových sadách

Source Zdrojový kód projektu z IDE Netbeans

Thesis Tato diplomová práce ve formátu PDF