Disertační práce

Optimalizace systémů pro rozpoznávání ručně psaného textu pomocí metod umělé inteligence

Optimization of systems for handwritten text recognition using artificial intelligence methods

Ing. Jan Pálka

Školitel: doc. Mgr. Roman Jašek, Ph.D.

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Zlín, 2013

Poděkování:

Děkuji svému školiteli doc. Mgr. Romanu Jaškovi, Ph.D. za jeho podnětné rady, podporu a odborné vedení po celou dobu mého studia, výzkumu i řešení této práce.

Dále bych chtěl poděkovat svým rodičům za slušné vychování v dětství, všeobecné vzdělání v mládí, a podporu v dospělosti, čímž mi umožnili dosáhnout mnoha cílů v mém osobním i profesním životě.

ABSTRAKT

Tato disertační práce se zabývá využitím samoučících funkcí v oblasti rozpoznávání ručně psaného písma se zaměřením na rozpoznávání textu v českém jazyce s diakritikou. Teorie navrženého řešení vychází z oblasti neuronových sítí, konkrétně konvoluční neuronové sítě Neocognitron.

V úvodní části je zpracována problematika volby vhodné neuronové sítě pro účel daný tématem disertační práce. V dalších krocích se disertační práce zabývá analýzou konvoluční neuronové sítě Neocognitron, která je předpokladem pro její úspěšnou implementaci do algoritmů v navrženém systému. Výstupy z provedené analýzy slouží také pro proces optimalizace, který spočívá v nalezení řešení pro problematickou oblast rozpoznávání textu v českém jazyce, kterou je bezesporu diakritika.

V experimentální části disertační práce byly nejdříve vytvořeny algoritmy ve skriptovacím prostředí Matlab, pracující s neuronovými sítěmi, za účelem jejich snadného testování a prezentace výsledků. Po otestování a výběru neuronové sítě pro další práci na sadě znaků, získaných z databáze MNIST z národního institutu pro standardy a technologie v USA, bylo přistoupeno k vytvoření vlastní databáze znaků české abecedy, včetně diakritiky. Za tímto účelem byl proveden sběr za pomoci vytvořených standardizovaných formulářů. Dalším krokem byla implementace navržených algoritmů do výsledného systému, obsahujícím komponentu inteligentního preprocesingu pro eliminaci negativního vlivu diakritiky na rozpoznávání v jazyce C#, a otestování tohoto systému na základě, dílčích, k tomu vytvořených, testovacích aplikacích.

ABSTRACT

This thesis is focused on using self-learning functions in field of hand-written text recognition with focusing on recognition of text in czech language with diacritics. Principle of designed solution is based on neural networks field, specifically convolution neural network Neocognitron.

In the first part is analysis of the problem of choosing the appropriate neural network Neocognitron, which assumption for its successful implementation in the designed system. Outputs from performed analysis are also used for optimalization process, which is based on searching solution for problematic recognition of diacritics in text.

In the experimental part of this dissertation were created algorithms in the scripting environment Matlab working with neural networks for easy testing and presenting results. After testing and choosing suitable neural network for following work based on testing on the set of letters gathered from MNIST database from National Institute for Standards and Technologies in the USA was performed data collection using standardized forms to create own database of czech alphabet letters with diacritics. Next step was implementation of designed algorithms to the final system, containing component of intelligent preprocessing for elimination of negative effect caused by diacritics on recognition in C# language. This system was finally tested on testing apps created only for testing purposes.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

symbol	*význam*
BPMN	Business Process Model and Notation
CLR	Common Language Runtime
CR	Classification Rate
DCT	Discrete Cosinus Transform
DPI	Dots per Inch
HMM	Hidden Markov Model
HWAI	Hand Written Address Interpretation System
ICR	Inteligent Character Recognition
IWR	Intelligent Word Recognition
LDA	Linear Discrimininant Analysis
LGN	Lateral Geniculate Nucleus
LINQ	Language Integrated Query
MAT	Matlab File Format
MNIST	Mixed National Institute of Standards and Technology
NIST	National Institute for Standards and Technologies
OCR	Optical Character Recognition
PDF	Portable Document Format
RMSE	Root Mean Square Error
UML	Unified Modeling Language
VPN	Virtual Private Network
WCF	Windows Communication Foundation
XML	Extensible Markup Language

1 ÚVOD

Vzhledem k velkému rozmachu informačních technologií v novém tisíciletí došlo k přechodu mnoha oblastí lidské činnosti od tradiční formy do oblasti výpočetní techniky. Díky rozvoji těchto technologií je v dnešní době možné nahradit, automatizovat a zjednodušit mnoho těchto činností, zejména v oblasti zpracování dat, jejich ukládání a práci s nimi (Stair, 2003). Největší přínosy informačních technologií spočívají ve velmi snadném přenosu těchto dat mezi uživateli, možnosti jejich ukládání ve velmi velkém množství a také snadná opakovatelnost procesů s nimi prováděných. Vedle uvedených výhod se svět informačních systémů potýká také s řadou problémů, jako je ochrana těchto dat před krádeží, zajištění bezpečnosti jejich uložení pomocí zálohování, a dalších. Problematické však je již pouhé předání těchto dat informačním systémům z reálného světa tak, aby bylo možné využít všech výhod těchto systémů. Prakticky je možná tato data předat dvěmi způsoby, a to zadáním dat již přímo v digitální formě, či digitalizací těchto dat (Singh, 2012).

Tato disertační práce je zaměřena na digitalizaci ručně psaného textu tak, aby bylo možné tato data dále zpracovávat pomocí informačních systémů, k tomu určených. Je nutno podotknout, že již v současné době je velmi dobře zpracován proces digitalizace tištěného textu, nicméně digitalizace ručně psaného textu dnes není stále dokonale zvládnuta, a proto je stále v mnoha oborech lidské činnosti zapotřebí ručně zpracovávat data z psaných textů. Z tohoto důvodu je automatizace získávání dat z ručně psaných dokumentů v dnešní praxi využívána velmi málo, a to z důvodu nedostatečné úspěšnosti rozpoznávání současných OCR systémů, určených pro ručně psané písmo, a také z důvodu nedokonalosti, nebo rozmanitosti textů, se kterými se musejí potýkat. Tato rozmanitost je dána jazyky, ve kterých jsou tyto texty napsány. V případě češtiny, na jejíž rozpoznávání se tato disertační práce zaměřuje, je tento fakt ještě více umocněn použitím diakritických znamének.

Na základě výše uvedeného vyplývá, že úspěšnost OCR systémů v rozpoznávání ručně psaných textů, je velmi závislá na jejich vlastnostech a parametrech, které musí být, pro dosažení uspokojivých výsledků, uzpůsobeny pro konkrétní písmo (Cheriet, 2007). V případě OCR systémů, založených na neuronových sítích, musí být také vhodným způsobem proveden účící proces na správných vzorech znaků. Jedním z hlavních cílů této disertační práce je výzkum optimalizace těchto OCR systémů s příhlednutím ke specifikům českého jazyka.

Tato disertační práce je rozdělena na dvě hlavní části – teoretickou a praktickou. Teoretická část se zabývá oblastí neuronových sítí a dalších metod používaných pro řešení rozpoznávání ručně psaného textu. Detailně je zpracována rešerše neuronové sítě Neocognitron, včetně popisu její struktury a procesů, které v ní probíhají v učící a vybavovací fázi. V závěru teoretické části je také popsána důležitá součást sestémů pro rozpoznávání ručně psaného textu, kterými jsou detektory hran.

Praktická část je dělena systematicky dle vytyčených cílů této disertační práce. V kapitole 7 je proveden, na základě provedených rešerší a testů, výběr neuronové sítě vhodné pro účely této disertační práce, a zároveň jsou na vybrané neuronové síti pro další práci provedeny testy za účelem stanovení dalšího postupu v řešení hlavního cíle. Tím je optimalizace OCR systému pro rozpoznávání ručně psaného textu v češtině. V další části práce je popsán postup splnění vedlejšího, avšak pro další postup nezbytného, cíle této disertační práce, kterým byl sběr testovacích dat od respondentů, prostřednictvím, k tomu vytvořených, formulářů. V rámci této části bylo také provedeno srovnání vytvořené databáze znaků s databází znaků poskytovou americkým Národním Institutem Standardů a Technologií. Na základě poznatků a materiálů, získaných ve výše popsaných kapitolách, byl navržen optimalizovaný ICR systém a také otestován na testovacích datech. V závěru praktické části jsou vyjmenovány nástroje a další prostředky,

využité v rámci experimentů. Jsou zde rozebrány výsledky získané z měření a analýz a jsou zde také navrženy možnosti dalšího výzkumu.

2 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

V současné době stále probíhá rozmach informačních technologií, který s sebou přináší velké změny ve způsobu, jak člověk pracuje s informacemi. Informační technologie umožňují ukládat, třídit a zpracovávat data ve velkém množství a velkou rychlostí. Díky těmto vlastnostem se zpracování dat, pomocí informační techniky, provádí v téměř každém oboru lidské činnosti. V rámci tohoto pokroku se čím dál více dat vytváří a zpracovává přímo v digitální podobě. Nicméně v praxi je stále mnoho oblastí, kde se pracuje s informacemi mimo informační technologie. Většinou se takto děje z důvodů úspory nákladů, dále z důvodu obtížně zajištěné nepopiratelnosti, integrity, autenticity, což je dnes řešeno pomocí, čím dál více, rozšířeného elektronického podpisu, a velmi často se tak děje pouze z praktických důvodů. Nejčastější příčinou práce s daty mimo informační systémy, je kombinace všech výše uvedených důvodů. Jako příklad nám může sloužit uzavíraná smlouva mezi odběratelem a dodavatelem, kdy se velmi obtížně splní veškeré výše uvedené požadavky vzhledem k tomu, že ne každý klient vlastní potřebnou informační techniku, či elektronický podpis (Peterka, 2011). Z těchto důvodů se výzkum v posledních sto letech zabývá možnostmi vytvoření různých můstků v různých oborech mezi těmito dvěmi sférami: analogovou a digitální. Jednou z hlavních oblastí, řešenou v této problematice, je převod tištěného a ručně psaného textu do digitální podoby.

Počátky vývoje prvních systémů pro rozpoznávání textu v tištěné podobě se datují do roku 1912, kdy Emanuel Goldberg patentoval stroj pro čtení tištěného písma. Goldbergův stroj byl schopný konvertovat toto písmo do standardního telegrafního kódu. Hlavním důvodem k vytvoření tohoto přístroje bylo umožnění čtení textů nevidomým lidem (Buckland, 1992). V roce 1951 vyvinul David H. Shepard jeden z prvních funkčních strojů pod názvem Gismo, schopných číst tištěný text. Tento stroj využíval speciálně vyvinutý font, přímo přizpůsobený pro

účely Gisma, který se dodnes používá na platebních, kreditních kartách. Vzhledem ke specifickému stylu písma, který font používá, kdy každý znak je navržen tak, aby byl snadno odlišitelný a nebyl zaměnitelný s jiným znakem, dosahoval Gismo a další systémy, které jej využívají, mnohem vyšší úspěšnosti rozpoznávání, než kdyby se používalo písmo se standardním fontem. Emanuel Golgberg ve vývoji OCR systémů pokračoval a byl také prvním, kdo zavedl první komerční OCR systém a to v roce 1955. Je nutné podotknout, že se stále jednalo pouze o výzkum a vývoj metod rozpoznávání tištěného písma.

První komerční systémy pro rozpoznávání ručně psaného písma se objevují až v devadesátých letech. V roce 1997 byl zaveden systém HWAI (hand written address interpretation), v systémech poštovních služeb Spojených států amerických, pro rozpoznávání ručně psaných adres na doručovaných zásilkách. Tento systém však těží z omezeného okruhu zadávaných dat a je u něj velmi snadné nasazení slovníkové metody, pomocí které je uměle zvýšena úspěšnost rozpoznávání na přijatelnou mez. Vzhledem k nemožnosti použití těchto standardních slovníkových metod ve všech oblastech OCR, lze považovat rozpoznávání ručně psaného písma za stále problematické. Proto v této oblasti stále probíhá výzkum. Jako příklad můžeme jmenovat největší mezinárodní konferenci ICFHR, zabývající se výzkumem rozpoznávání ručně psaného písma, jejíž poslední ročník proběhl 20. 10. 2012. a je pořádána pravidelně ve dvouletých cyklech. Vzhledem k tomu, že velká část informačních systémů pracuje s anglickým a jinými světovými jazyky je také vývoj metod pro rozpoznávání zaměřen na tyto jazyky. Rozpoznávání ručně psaných textů v méně rozšířených jazycích, pomocí takto navržených systémů, je méně úspěšné.

Vzhledem k nákladnosti vývoje software a také omezenému trhu České republiky, je pro skenování, a s tím spojené rozpoznávání ručně psaného textu, v současnosti používáno nástrojů, určených pro rozpoznávání ručně psaného textu

ve světových jazycích. Proto skenování v textu v českém jazyce má diskutabilní výsledky. V případě institucí jsou implementována, sice komplexní nákladná řešení se stejnými omezeními, nebo jsou nasazena adhoc řešení dle dodavatelů, avšak komplexní vědecké analýzy možných řešení, metod a postupů rozpoznávacích systémy pro český jazyk jsou velmi omezené.

V praxi se dnes vývojem OCR systémů a nově také ICR systémů, což jsou pokročilé OCR systémy s možností účení na různé typy písem, zabývá mnoho společností. Vzhledem k vysoké hodnotě know-how, obsaženém v rozpoznávacích jádrech těchto systémů, lze jen obtížně zjistit reálné schopnosti těchto systémů a je potřeba usuzovat pouze na základě veřejných informací a praktických zkušenostech s těmito systémy. Vedle těchto OCR ss Autor práce má ze své praxe, díky níž se účastnil konzultací nasazení OCR systémů v jedné z předních českých poradenských společností, zkušenosti se společnostmi Abbyy, Kofax a IBM (využívá jádra rozpoznávání Kofax), které lze zařadit mezi jedny z největších společností zabývající se touto problematikou. První jmenovaná společnost Abbyy, kromě komplexních OCR řešení, poskytuje také menší aplikace, na kterých je možné si vyzkoušet schopnosti jádra rozpoznávání a potvrdit si tak výše uvedený předpoklad velmi nízké úspěšnosti rozpoznávání textu s českou diakritikou. Praktické zkušenosti, z konzultací implementace OCR systému Kofax, také potvrzují tuto tezi, neboť rozpoznávací systémy této společnosti nedosahovaly dostatečně uspokojivých výsledků rozpoznávání potřebných pro jejich možnost jejich implementace. Vzhledem k tomu, že výše uvedené společnosti lze bezesporu zařadit mezi přední světové dodavatele těchto systémů, jsou jejich neuspokojivé výsledky na poli rozpoznávání textu s českou diakritikou jednoznačným potvrzením, že tato problematika není ještě zcela vyřešena a je v této oblasti potřebný další výzkum.

3 CÍLE DISERTAČNÍ PRÁCE

Cílem disertační práce je navrhnout a realizovat optimalizovaný ICR systém pro rozpoznávání ručně psaného textu s českou diakritikou a jeho následné ověření na testovacích vzorcích českých znaků.

Hlavní cíle práce lze stručně shrnout do následujících bodů:

- Shrnutí aktuálního stavu na poli OCR systémů.
- Nalézt vhodný typ neuronové sítě, využitelné pro optimalizovaný systém ICR.
- Navrhnout optimalizovaný systém ICR pro rozpoznávání ručně psaného textu s českou diakritikou.
- Stanovit potřebné rozlišení a barevnou hloubku naskenovaného textu (optimalizace velikost vs. kvalita).
- Vytvořit kvalitní zdroj vzorů, vhodný pro učení neuronových sítí.
- Implementace navržených algoritmů do výsledného systému.
- Ověřit systém a aplikaci na testovacích vzorcích různých písem (od různých uživatelů).

TEORETICKÁ ČÁST

4 SOUČASNÉ METODY SYSTÉMŮ PRO ROZPOZNÁVÁNÍ TEXTU

Rozpoznávání ručně psaného písma se v současné době stále potýká s problémy. Dobrá úspěšnost rozpoznávání je dosažena pro znaky nebo číslice, kde je malý počet tříd. Se zvyšujícím se počtem těchto tříd, například při rozpoznávání izolovaných slov, tato úspěšnost znatelně klesá. V případě rozpoznávání obecných vět či řádků textu je situace ještě horší. Dle literatury je v těchto případech úspěšnost rozpoznávání mezi 50% a 80%, v závislosti na experimentálním odladění.

Jádrem většiny OCR systémů jsou algoritmy, založené na neuronových sítích, Skrytých Markovových modelech či klasifikátorech, na principu minimální vzdálenosti. Tyto metody mají dobrou úroveň úspěšnosti v rozpoznávání strojově psaných dat. Proces rozpoznávání ručně psaného textu je však více komplikovaný vzhledem k množství odlišných rukopisů. Nejlepšími metodami pro rozpoznávání ručně psaného textu se jeví metody, využívající neuronové sítě (NS). Za tímto účelem je možné použít různě strukturované NS (Haykin, 2009). Mezi nejznámější nástupce patří Vícevrstvý perceptron, Kohonenova síť, Neocognitron. Bez ohledu na použitou metodu rozpoznávání ručně psaného textu jsou často implementovány statistické modely jazyka, pro který je systém určen (Cheriet, 2007).

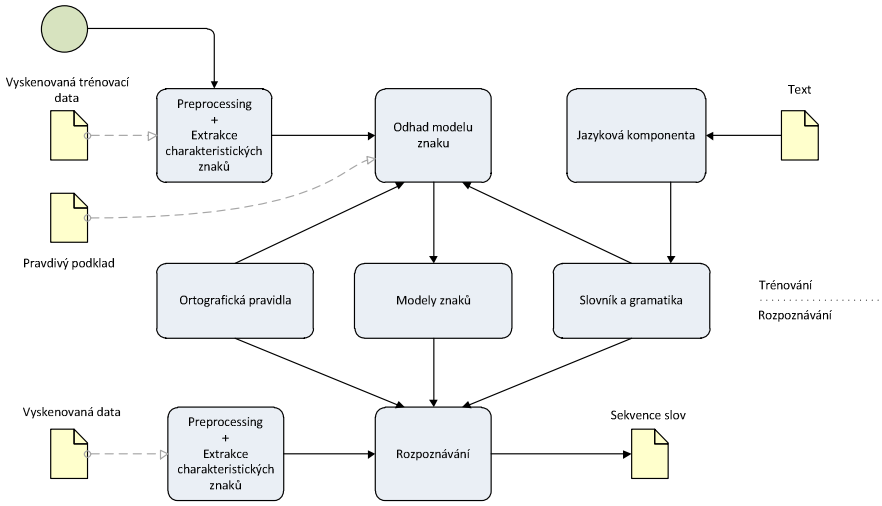
V rámci této kapitoly jsou detailně popsány principy nejčastěji používaných metod pro rozpoznávání ručně psaného textu.

4.1 Skryté Markovovy modely

Jedním typem rozpoznávacích systémů, využívaných v současné době, je OCR systém, založený na Skrytých Markovových modelech (dále HMM). Tyto systémy jsou velmi málo závislé na jazyku, písmu a jsou navíc schopny zpracovávat i poškozená data, jako je například fax a to pomocí technik adaptace

(Fink, 2007). Nejdříve byly metody Skrytých Markovových modelů používány pro rozpoznávání řeči, nicméně byla vyvinuta i varianta pro rozpoznávání textu (Plotz, 2011).

OCR systém založený na HMM (Hidden Markov Model) obsahuje dvě části: trénovací a rozpoznávací část, viz obr. 1.



Obr. 1 Blokový diagram OCR systému založeného na HMM (Lu, 2012)

Systém závisí na rozpoznávání znaků, slovníkové metodě a gramatice z trénovacích dat. Systém získává vstupní trénovací data jako naskenovaný obrázek znaku v páru s pravdivým podkladem, který je popisem daného znaku. V předzpracovávací fázi je potřeba upravit rotaci jednotlivých řádků textu a lokalizovat řádky textu a každý řádek rozdělit na velmi úzké, překrývající se vertikální okna, viz obr. 2.

Poté je získána sada jednoduchých znaků pro každé okno. Je potřeba spočítat vektor pro daný znak jako funkci jedné nezávislé proměnné. Prakticky se jedná o koordináty pera, které jsou funkcí času. Komponenta pro modelování znaku poté vezme vektor daného znaku a korespondující pravdivý základ

k odhadnutí modelu znaku. Trénovací proces bere v potaz také ortografické pravidla daného písma, které závisí na jazyce písma. V této vlastnosti spočívá závislost na jazyce. Mezi základní znaky oken, rámců patří:

- Intenzita.
- Vertikální derivace intenzity.
- Horizontální derivace intenzity.
- Lokální sklon a korelace přes dvě buňky v rámci.

Obr. 2 Rozdělení textu na rámce a buňky (Lu, 2012)

I když znak intenzity samotný reprezentuje celý obraz, jsou použity i další faktory, jako je vertikální derivace, lokální sklon, lokální korelace, aby bylo získáno více globálních informací. Na základě lineární analýzy diskriminantů (LDA) těchto základních znaků je vybrán malý počet z nich jako finální znaky pro systém. Výhodou této činnosti je vedle rychlosti také zlepšení přesnosti (Shi, 1997).

4.1.1 Kalkulace rámců

Je potřeba určit šířku a výšku jednotlivých segmentů a z toho vyplývající jejich počet. Tento počet je zásadní, neboť se od něj odvíjí počet stavů HMM. Na základě počtu segmentů je rozdělen text, nebo znak na určitý počet rámců

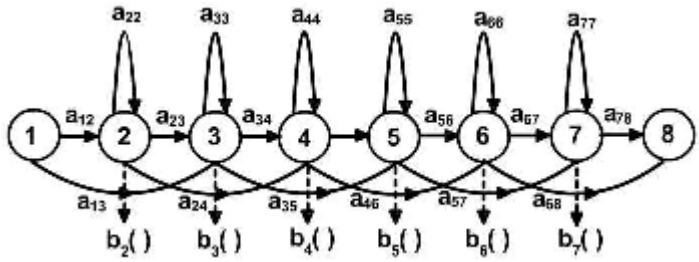
4.1.2 Kalkulace markantů

V této fázi je vzat odděleně každý segment v textu a na každém pixelu v tomto rámci je provedena diskrétní cosinova transformace. DCT transformace konvertuje spojitý signál na jednotlivé frekvenční komponenty. Pomocí DCT se

získají signifikantní markanty daného znaku. Těchto výhod DCT je často využito v metodách komprese obrazu.

4.1.3 Struktura systému

Základem OCR systému jsou Skryté Markovovy modely každého znaku. Pro každý model je potřeba specifikovat počet stavů a dostupné přechody mezi stavy. Pro OCR systémy se používá obvykle 14 ti stavového HMM. Na obr. 3 je zobrazen 7 mi stavový HMM (Plotz, 2011). Nejčastěji jsou všechny znaky řešeny HMM se stejným počtem stavů.



Obr. 3 7mi stavový Skrytý Markovův model (Plotz, 2011)

Je možné pro každý znak použít různý počet stavů, nicméně nejčastěji jsou všechny znaky řešeny Skrytými Markovovými modely se stejným počtem stavů. Aby bylo možné vytvořit model pravděpodobnosti struktury každého stavu, každého znaku HMM, je pro každý znak zjištěna hustota pravděpodobnosti 15 ti rozměrného vektoru znaků pomocí kombinace 128 Gaussových hustot. Parametry těchto hustot jsou nastaveny procesem sloučení podmnožin trénovacích dat a těchto dat po natrénování. Pro každý znak existuje 128 vah (každý pro jednu Gaussovu hustotu), která reprezentuje hustotu pravděpodobnosti pro daný znak.

4.1.4 Fáze trénování

Trénovací algoritmus je stejný jako u systémů pro rozpoznávání řeči. Řádek po řádku jsou získávány jednotlivé znaky z vertikálních rámců ve dvojicích s popisem skutečného základu a jsou použity jako vstup pro trénování znakového modelu trénovacích dat získaných.

4.1.4.1 Baumwelsch

Algoritmus vyvinutý v roce 1972 pojmenovaný po Leonardu E. Baumovi a Lloydu R. Welchovi. Je používaný k nalezení neznámých parametrů a využívá předně-zpětný algoritmus. Je využíván pro nalezení parametrů pro Skryté Markovovy modely.

4.1.4.2 EM algoritmus

Expectation – maximization algoritmus je iterační metoda pro hledání maximální pravděpodobnosti, kde model závisí na neznámých parametrech. Metoda je používána pro hledání skrytých parametrů ve Skrytých Markovových řetězcích.

4.1.5 Fáze rozpoznávání

Dopředně-zpětný trénovací algoritmus je použit k odhadnutí parametrů modelu a výsledné modely zvyšují pravděpodobnost trénovacích dat, daných pravdivými podklady. Rozpoznávací proces je založený na hledání nejpravděpodobnější sekvence znaků, daných vstupním vektorem vlastností (Plotz, 2011).

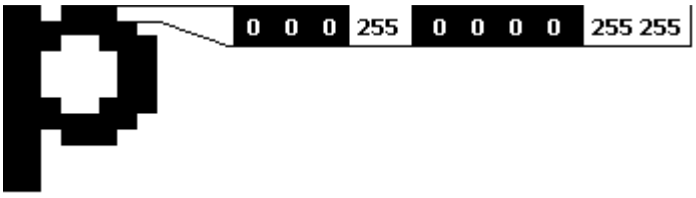
4.2 Klasifikátory dle minimální vzdálenosti

Klasifikací rozumíme metodu, díky které je systém schopen zařadit rozpoznávané vzory do příslušných tříd a to na základě určitých vlastností pro danou třídu charakteristických. Nejzásadnějším problémem, nutným vyřešit, je u klasifikačních metod výběr znaků pro zařazování. Vzory jsou, pomocí klasifikátorů

minimální vzdálenosti, zařazeny do tříd dle kritéria minimální vzdálenosti. Jedná se tedy o formu klasifikace bez učení.

4.2.1 Získání vektoru markantu znaku

Většina používaných metod rozpoznávání nedokáže klasifikovat tvary jako celek, ale je vždy potřeba vytvořit takzvaný markant, viz např. obr. 4, popisující daný tvar a je možné ho použít pro jeho klasifikaci.



Obr. 4 Příklad získání markantu znaku (vlastní zdroj)

4.2.2 Metoda nejbližšího a k-nejbližších sousedů

Jsou základními formami klasifikací minimální vzdálenosti. Každá jedna třída je charakterizována určitým typickým prvkem, tzn. etalonem. Klasifikovaný objekt je poté porovnán s etalony jednotlivých tříd a je vybrán etalon našemu obrazu nejpodobnější. To znamená, že objekt je zařazen do třídy, od jejíhož etalonu má nejmenší vzdálenost (Houdek, 2012). Jelikož klasifikovaný obraz i etalony jsou n-rozměrné vektory, definujeme podobnost obvykle jako Euklidovskou vzdálenost mezi nimi. Pro dvourozměrný prostor určujeme vzdálenost dvou bodů následujícím vztahem:

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 + y_2)^2} \quad (1)$$

Kde x, y jsou souřadnice, a d je vzdálenost.

pro n-rozměrný prostor může být vzdálenost zobecněna na:

$$d_{ij} = \sqrt{\sum_{k=1}^{n}(x_{ki} - x_{kj})^2} \quad (2)$$

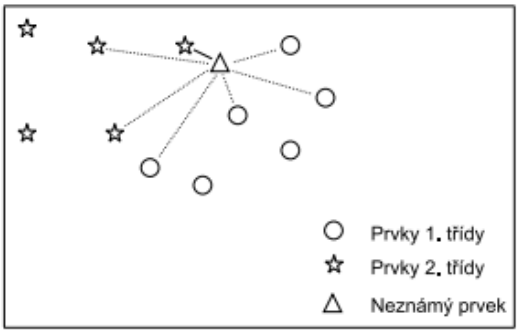
Kde x, y jsou souřadnice, a d je vzdálenost.

V případě potřeby zajistit větší odolnost vůči geometrickým informacím (posun, rotace) je možné použití jiných typů metrik, např. tangentové metriky. Vzhledem ke svým vlastnostem má tato metrika vysokou náročnost na výpočet. Z tohoto důvodu je v případě, kdy jsou třídy v nesymetrických shlucích, možné použít Mahalonobisovu vzdálenost.

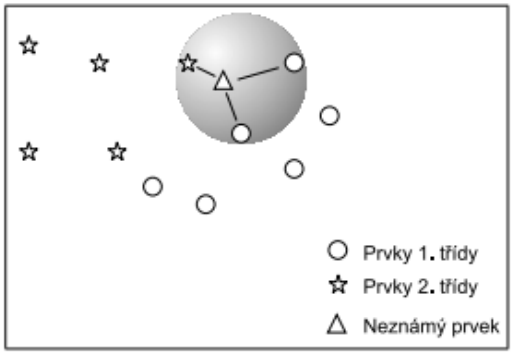
Klasifikátor je dán množinou etalonů μ, reprezentujících příslušnou třídu y. Vztah pro hledanou třídu prvku x lze poté zapsat rovnicí 3. Správná klasifikace je závislá na zvolení vhodného reprezentanta každé třídy. Etalon lze zvolit výpočtem průměrné hodnoty vzorů na vstupu v jednotlivých třídách. (Dasarathy, 1990)

$$f(x) = min\left\|x - \mu_y\right\|_{1,..,K} \quad (3)$$

Kde x je prvek hledané třídy a μ je množina etalonů



Obr. 5 Příklad klasifikace dle nejbližšího souseda (Houdek, 2012)



Obr. 6 Příklad klasifikace dle k-nejbližších sousedů (Houdek, 2012)

Výše uvedená metoda je vhodná díky své jednoduchosti a rychlosti své klasifikace. V případě, že lze skupiny prvků v jednotlivých skupinách triviálně oddělit, je možné vytvořit popis jednotlivých tříd příslušným etalonem, viz klasifikace dle nejbližšího a n-nejbližších sousedů na obr. 5 a 6. Ve výše popsaném případě klasifikace se dosahuje vysoké úspěšnosti rozpoznávání, nicméně při nerovnoměrném rozdělení prvků jednotlivých tříd nastává problém, kdy není možné snadno rozdělit skupiny tříd podle jediného etalonu. V těchto případech poté dochází ke značným chybám klasifikace.

Výše popsanou metodu je možné modifikovat vybráním většího počtu prvků, touto úpravou vznikne metoda k-nejbližších sousedů. Tyto prvky jsou vybrány tak, aby byly co nejvíce charakteristické pro danou třídu. Klasifikace je provedena vypočtem vzdálenosti předloženého vzoru od všech vzorků (Dasarathy, 1990). Následně je nalezen k-počet nejbližších prvků zastupujících předložený vzor a tento je zařazen do třídy nejvíce zastoupenou mezi těmito prvky.

V případě rovnosti počtu prvků zastoupených mezi prvky, je klasifikace neúspěšná. Řešením tohoto problému je modifikace této metody, kdy jsou postupně procházeny nejbližší prvky až do nalezení k-počtu prvků stejné třídy. Vzor je následně přiřazený k této třídě. Na velikosti parametru k je závislá nejen úspěšnost kvalifikace, ale také na náročnost celého algoritmu. Při použití výše popsané

modifikované verze metody lze dosáhnout vyšší úspěšnosti rozpoznání za současného narůstu množství informací a s ním spojeným zpomalením celého algoritmu.

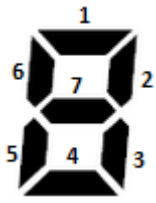
Výše popsané metody klasifikace minimální vzdálenosti nebo její modifikace mají velmi široké pole působnosti v různých oborech lidské činnosti. Jsou také úspěšně využívány v systémech pro rozpoznávání znaků. (Dasarathy, 1990)

4.2.3 Vlastnosti metod klasifikace minimální vzdálenosti

Metody klasifikace minimální vzdálenosti, především její modifikace, mají velmi široké pole působnosti v různých oborech lidské činnosti. Jedním z nich je i počítačové vidění, kde jsou využívány v systémech rozpoznávání znaků. Pro potřeby rozpoznání ručně psaných znaků však není tato metoda úplně ideální. Přesto v určitých konkrétních případech může dosahovat uspokojivých výsledků. Základem pro úspěšnou klasifikaci je kvalitní předzpracování vstupních znaků. Z důvodu toho, že tato metoda není příliš odolná proti jakýmkoliv tvarovým změnám, jako jsou posunutí, zkosení, a také zašumění jednotlivých znaků (Lin, 2008).

4.2.4 Princip

Klasifikační třídy jsou reprezentovány prvky, zvanými etalony. Na počátku jsou zvoleny markanty za účelem získání informací o oblastech odlišujících jednotlivé znaky. Pro zjednodušení bereme v potaz pouze varianty číslic 0 až 9 (Obr. 7).



Obr. 7 Rozdělení segmentů digitálních číslic (vlastní zdroj)

Z výše uvedeného obrázku je patrných 7 příznaků, které odpovídají jednotlivým segmentům digitální číslice. Stavu příznaku jednotlivých znaků je uložen v jednotlivých vektorech etalonů. V uvedeném příkladu se jedná pouze o binární hodnoty, zda je daný příznak aktivní. Vektory etalonů pro segmenty v digitální číslici jsou uvedeny v tab 1.

Číslice	Vektory
0	(1,1,1,1,1,1,0)
1	(0,1,1,0,0,0,0)
2	(1,1,0,1,1,0,1)
3	(1,1,1,1,0,0,1)
4	(0,1,1,0,0,1,1)
5	(1,0,1,1,0,1,1)
6	(1,0,3,4,1,1,1)
7	(1,1,1,0,0,0,0)
8	(1,1,1,1,1,1,1)
9	(1,1,1,1,0,1,1)

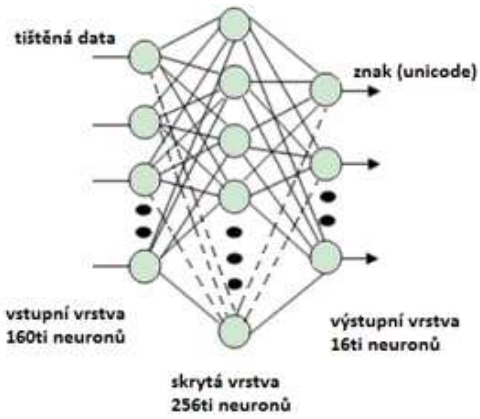
Tab 1 Vektory etalonů číslice 8

Při klasifikaci je pro předložený vzor zvolen vektor markantu a od tohoto jsou odečteny vektory etalonů jednotlivých tříd. Pro každou ze tříd je vypočtena absolutní hodnota rozdílu vektorů a z těchto absolutních hodnot je nalezeno minimum, na základě kterého je vzor zařazen do náležící třídy. V případě, že jsou znaky totožné, je rozdíl roven nule. Rozpoznávání pomocí této klasifikace je

v časté shodě minimálních vzdáleností etalonů a z toho vyplývající neschopnost systému určit vhodnou třídu pro daný znak.

4.3 Vícevrstvá neuronová síť perceptron

Vícevrstvá neuronová síť se skládá nejméně ze tří vrstev neuronů: vstupní, výstupní a nejméně jedné vnitřní vrstvy viz obr. 8. Neurony ve výstupní vrstvě a vnitřní vrstvě jsou definovány prahem, který odpovídá vahám, přiřazeným k jednotlivým spojením mezi neurony a fiktivními neurony, jejichž aktivace je vždy 1. Mezi dvěmi přilehlými vrstvami neuronů je plné propojení, kdy každý neuron v nižší vrstvě je připojen na všechny neurony ve vyšších vrstvách (Rosenblatt, 1962), (Shepherd, 1997).



Obr. 8 Ukázka neuronové sítě perceptron.(Singh, 2012)

4.3.1 Učení neuronové sítě Perceptron

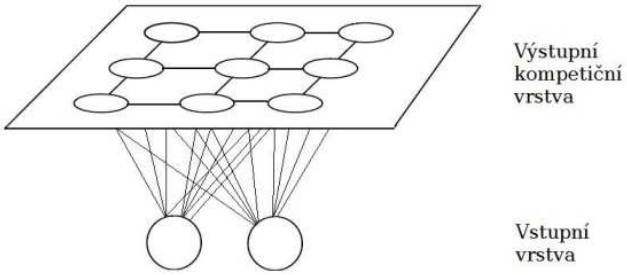
Pro učení této neuronové sítě je potřeba mít trénovací sadu vzorů, popisující daný problém k řešení a metodu k získání těchto vzorů v neuronové síti, pomocí hodnot synaptických vah. Každá trénovací sada vzorů popisuje, jak jsou nastaveny váhy ve vstupní a výstupní vrstvě. Nejobvyklejším adaptačním algoritmem pro

učení vícevrstvé neuronové sítě je metoda zpětného šíření chyby. Tato metoda umožňuje adaptovat neuronovou síť na předloženou sadu vzorů. Algoritmus se skládá ze tří částí: dopředné šíření vstupního signálu, zpětné šíření chyby a aktualizace hodnot vah na jednotlivých spojeních. V průběhu dopředného šíření je signál přijat každým neuronem ve vstupní vrstvě a tento dále zprostředkovává jeho přenos do všech dalších neuronů ve vnitřních vrstvách. Každý neuron ve vnitřní vrstvě spočítá jeho aktivační funkci a pošle signál všem dalším neuronům ve výstupní vrstvě. Každý neuron ve výstupní vrstvě spočítá svou aktivační funkci, která odpovídá aktuálnímu výstupu n-tého neuronu po předložení vstupního vzoru (Shepherd, 1997).

Experimentálně bylo ověřeno (Singh, 2012), že počet vrstev není ideální zvyšovat na hodnotu vyšší než 3. Nastavení počtu neuronů ve vnitřních vrstvách je klasický problém. Teze, zabývající se možností vyššího počtu neuronů za účelem vyšší, přesnosti jsou chybné, neboť se zvyšujícím se počtem neuronů ve skrytých vrstvách nebo počtem skrytých vrstev se nejen zvyšuje nelineární chování sítě, ale také stoupají nároky na učení (počet vzorů, čas učení). Příliš velká síť má také tendenci k přeučení (Liu, 2008). Přílišné zaměření na málo důležité detaily nemusí vést k lepším výsledkům, neboť tyto detaily nemusí mít vliv na správnou klasifikaci. Dosud není známo doporučení ke správnému nastavení počtu neuronů v jednotlivých vrstvách. Jedním z pravidel je nastavit počet neuronů ve vnitřní vrstvě na dvojnásobek neuronů ve vstupní vrstvě. (Haykin, 2009)

4.4 Kohonenova neuronová síť

Kohonenova neuronová síť, někdy je také nazývána Kohonenovy samoorganizační mapy, patří do skupiny samoorganizujících neuronových sítí, tedy s učením bez učitele. Svou schopností samoorganizace a shlukování objektů s podobnými vlastnostmi do skupin jsou Kohonenovy mapy předurčeny pro rozhodování, třídění a rozlišování dat (Kaňa, 2011).

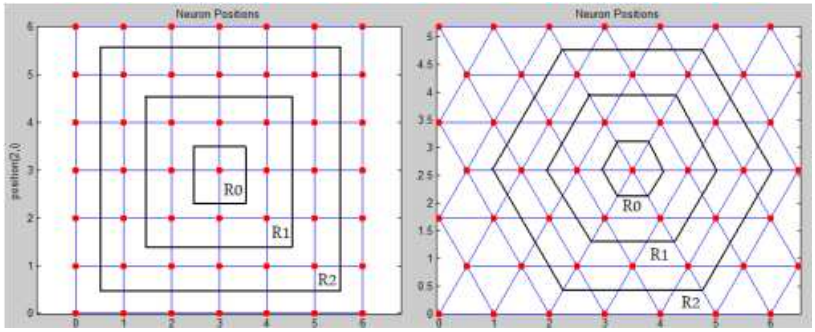


Obr. 9 Ukázka Kohonenovy neuronové sítě (Kaňa, 2011).

Principem vychází Kohonenova síť z Kohonenova učení. Jedná se tedy o jednovrstvou síť s úplným propojením jednotek mezi vstupy a kompetiční vrstvou, tedy každý neuron má informaci o hodnotě každého vstupu, viz obr 9. Neurony v kompetiční vrstvě jsou uspořádány do topologické struktury, většinou do dvojrozměrné obdélníkové nebo hexagonální dvojrozměrné oblasti. Struktura neuronů kompetiční vrstvy má vliv na tzv. okolí neuronu R, kterým jsou vymezené sousední, nejblíže postavené neurony. Velikost okolí je rovna počtu řad neuronů od centrálního neuronu. Poloha neuronu v prostoru je definována jeho vahami. Funkcí neuronů je výpočet vzdálenosti vstupního vektoru od vzoru ve vahách neuronu (Vojáček, 2012), tato funkce je popsána následujícím vztahem:

$$d_j = \sum_{i-0}^{N-1} (x_i(t) - \omega_{ij}(t))^2 \qquad (4)$$

váhy neuronu	ω
vstupní vektor	x
vzdálenost	d
vstup index	t



Obr. 10 Možné struktury uspořádání neuronů v okolí vítězného neuronu R. (Kaňa, 2011)

Kohonenova síť je dvouvrstvá síť s úplným propojením buněk mezi vrstvami viz obr 10. Vstupní vrstva je tvořena vstupními neurony, sloužícím k distribuci vstupů $x \in R^n$. Poloha ve vstupním prostoru je dána váhami, váhy náležející jednomu výstupnímu neuronu j viz vztah 5.

$$\omega_j = (\left(\omega_{j_1}\right), \dots, \omega_{j_n}) \tag{5}$$

váhy neuronu ω

V případě vstupů $x \in R^n$ se jedná o reálná čísla. Výstupy y_j nabývají hodnot nula a jedna. Jako správný neuron na výstupu je zvolen ten, který je aktivní. Výpočet vzdálenosti vítězného neuronu od vstupního prostoru $x^{(t)}$ je dán následujícím vztahem:

$$y_j = \{j = argmin_{l=1,..,h}\{\left|\left|x^{(t)} - \omega_l\right|\right|\} \tag{6}$$

Při průchodu celou tréninkovou množinou a po předložení jednoho tréninkového vzoru dochází mezi jednotkami v síti k porovnání. Vítězný neuron z porovnání změní své váhy podle následujícího vztahu:

$$\omega_{ji}^{t} = \{\omega_{ji}^{(t-1)} + \alpha\left(x_i^{(t-1)} - \omega_{ji}^{(t-1)}\right)\} \tag{6}$$

učící koeficient α

váhy neuronu ω

Parametr α představuje učící koecient určující míru změny vah. Vítězný neuron posune své váhy ω_c o poměrnou vzdálenost směrem k aktuálnímu vstupu, čímž ještě zlepší svou relativní pozici. (Kaňa, 2011)

5 HIERARCHICKÁ VÍCEVRSTVÁ NEURONOVÁ SÍŤ NEOCOGNITRON

5.1 Historie

Neocognitron je druh vícevrstvé hierarchické neuronové sítě, která se používá pro rozpoznávání ručně psaných znaků. Síť byla navržena profesorem Kunihiko Fukushimou z Japonska v roce 1979. Název je odvozen od předchozího návrhu samoorganizující se vícevrstvé neuronové sítě profesora Fukushimy, zvané cognitron z roku 1975. Cognitron byl schopen rozpoznávat vzory, ale jeho odezva byla velmi negativně ovlivněna posunutím podnětných vzorů. Tento nedostatek byl částečně vyřešen v Neocognitronu, nicméně i u něho je v případě posunutí vzoru ovlivněn výstup. Originální verze byla bez učitele, tzv. samoorganizační, ale pozdější verze byly již navrženy s učitelem, pro které je nutné vytvořit speciální sadu znaků pro učení (Fukushima, 1980). Vychází tak z biologických sítí, kde je využita zpětná vazba. Proces je popsán v kapitole 5.3.

5.1.1 Bez učitele

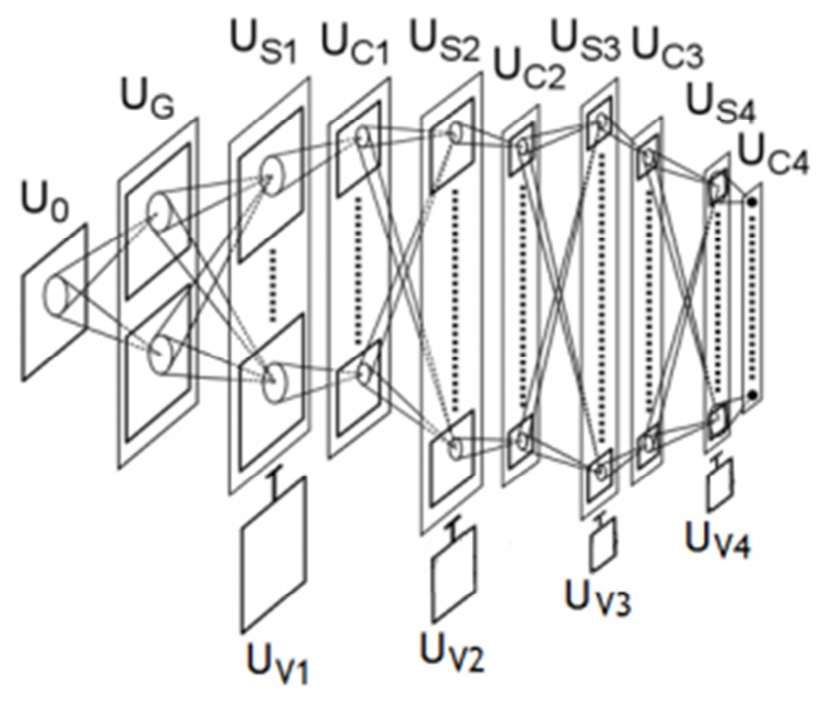
Neprobíhá vyhodnocování výstupu, výstup není znám. Síť třídí vzory do skupin dle typických zástupců.

5.1.2 S učitelem

Založeno na základě biologických sítí, kdy je zde využita zpětná vazba. Proces je založený na předložení vzoru, který po zpracování sítí s momentálním nastavením vah a dalších parametrů, vrátí výsledek, který je porovnán s požadovaným výsledkem a následně je určena chyba. Na základě vypočítané korekce jsou upraveny váhy a prahy neuronové sítě a síti je opět předložen vzor. Takto se učení opakuje až po dosažení minimální chyby.

5.2 Konstrukce Neocognitronu

Neocognitron je vícevrstvá hierarchická neuronová síť. Její výhoda spočívá ve schopnosti správně identifikovat nejen vzory již naučené, ale také vzory částečně posunuté, rotované nebo jinak deformované. Tato hierarchičnost spočívá ve schopnosti rozpoznávat v nižších úrovních jednoduché znaky. Na vyšších úrovních jsou tyto symptomy více a více komplexní. Kromě nulté úrovně, která obsahuje pouze vstupní vstvu pro uchování vstupní informace, obsahuje každá další úroveň tři vrstvy: S-vrstva, C-vrstva, V-vrstva. Jednotlivé vrstvy obsahují několik oblastí, které jsou pod vrstvami S, C a V. Tyto oblasti obsahují dvourozměrné pole buněk. Síť samotná je složena ze 4 základních typů buněk: S-buňky, C-buňky, V-buňky a receptorové buňky. Jednotlivé buňky pracují s reálnými kladnými hodnotami. Zjednoduššená struktura sítě Neocognitron je zobrazena na obr. 11.



Obr. 11 Struktura sítě Neocognitron (Bishop, 2006)

Příklad oblastí a vrstev s buňkami pro každou úroveň je ukázána v Tab. 2. Jedná se o příklad ke klasické aplikaci neuronové sítě pro ručně psané znaky bez diakritiky obsahující číslice 0 - 9 a kapitálky A-Z.

Vrstva	Velikost buňky	Počet oblastí	Počet buněk
U0	19 x 19	-	361
US1	19 x 19	12	4 332
UV1	19 x 19	1	361
UC1	21 x 21	8	3 528
US2	21 x 21	80	35 280
UV2	21 x 21	1	441
UC2	13 x 13	33	5 577
US3	13 x 13	97	16 393
UV3	13 x 13	1	169
UC3	7 x 7	64	3 136
US4	3 x 3	46	414
UV4	3 x 3	1	9
UC4	1 x 1	35	35
Celkem	-	380	70 045

Tab 2 Oblasti a buňky v síti Neocognitron

5.2.1 Propojení v síti

Neocognitron je typický velkým počtem vazeb mezi těmito buňkami, kde každá buňka je propojena na skupinu buněk v předchozí vrstvě, pomocí připojovací oblasti. Vstupní vstva buněk nebo předchozí C-vrstvy jsou napojeny na buňky v S-

vrstvě a V-vrstvě. Každá buňka je napojena na S-buňku. Každé propojení má určitou váhu.

5.2.1.1 Typy vah

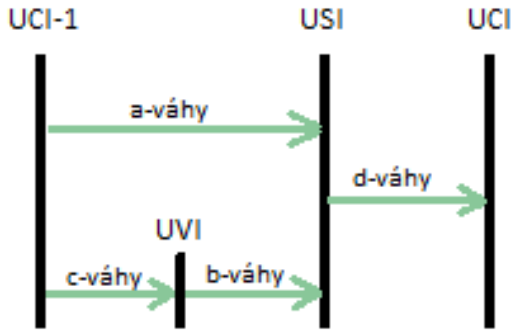
V neuronové síti Neocognitron existují čtyři typy vah, viz obr. 12:

- A-váhy slouží k připojení připojovací oblasti C-ploch L-1 úrovně s S-buňkami úrovně následující. S-buňky v jedné S-ploše mají stejnou hodnotu a-váhy.

- B-váhy spojují V-buňky s příslušnými S-buňkami a jsou také modifikovány učením. Stejně jako předchozí a-váhy jsou i b-váhy sdílené, tzn. všem S-buňkám v S-ploše náleží stejné b-váhy. Vzhledem k tomu, že počet buněk v ploše ve V a S-vrstvě je totožný, připadá ke každé buňce pouze jedna b-váha.

- C-váhy mají nastaveny hodnotu pevně při konstrukci sítě a jsou jimi spojeny připojovací oblasti C-buněk L-1 s V-buňkami následující úrovně. Váhy jsou nastaveny za účelem omezení přenosu informací z okrajů připojovacích oblastí (Fukushima, 1980).

- D-váhy - jsou spojením mezi připojovací oblastí S-buněk a příslušnou C-buňkou. Váhy jsou pevně nastaveny při konstrukci a stejně jako c-váhy tlumí přenos informace směrem k okrajům připojovací oblasti.



Obr. 12 Váhy spojení v Neocognitronu (vlastní zdroj)

5.2.2 Typy buněk

Jak bylo zmíněno v kapitole 5.2, neuronová síť Neocognitron obsahuje čtyři typy buněk. První a nejjednodušší buňky jsou receptorové buňky, umístěné ve vstupní vrstvě. Jejich jedinou funkcí je uložit vstupní informaci o každém jednom pixelu vstupního obrazu. Počtem receptorových buněk je dáno maximální rozlišení vstupní vrstvy a tedy i vstupního vzoru.

5.2.2.1 S-Buňky

Dalším typem buněk jsou buňky typu S. Cílem S-buněk je detekovat na jejich předdefinovaných pozicích ve vrstvě vzruchovou informaci, získanou pomocí každé jednotlivé buňky z předchozí C-úrovně, a také obsahuje informace z inhibičních V-buněk. Tato informace indikuje průměrnou aktivitu v oblasti. Výstupní hodnota S-buněk je získána rovnicí (7).

$$u_{Sl}(n,k) = r_l(k) \cdot \varphi \left[\frac{1+\sum_{\kappa=1}^{\kappa_{Cl-1}} \sum_{\nu \in Al} a_l(\nu,\kappa,k) \cdot u_{Cl-1}(n+\nu,k)}{1+\frac{r_l(k)}{1+r_l(k)} b_l(k) \cdot u_{Vl}(n)} \right] \quad (7)$$

Vrstva	l
Koordináty konkrétní buňky	n
Počet oblastí mezi vrstvami	a
Koordináty buňky v připojovací oblasti	ν
Počet C-oblastí v C-vrstvě	κCl
Oblast připojená k S-buňkám	A
Selektivita	r
Prahová přenosová funkce	φ
a-váhy	a
b-váhy	b
Výstupní hodnota V-buněk	uVl

Při vyšších hodnotách se zvyšuje přesnost rozpoznávání, ale je snížena schopnost reagovat na více odlišné vzory. Naopak snížením selektivity se zvyšuje schopnost třízení deformovaných vzorů, ale je snížena přesnost rozřazení. Každá S-vrstva může mít specifickou hodnotu selektivity. Kvalitní nastavení je nutné pro optimální funkci neuronových sítí (Fukushima, 1992).

5.2.2.2 V-Buňky

Hlavním cílem je přenést informaci o průměrné aktivitě spojení C-prostoru s S-buňkami. Výstup V-buněk je popsán vztahem

$$u_{Vl}(n) = \sqrt{\sum_{\kappa=1}^{\kappa_{Cl-1}} \sum_{\nu \in Al} c_l(\nu) \cdot u_{Cl-1}^2(n+\nu, \kappa)} \quad (8)$$

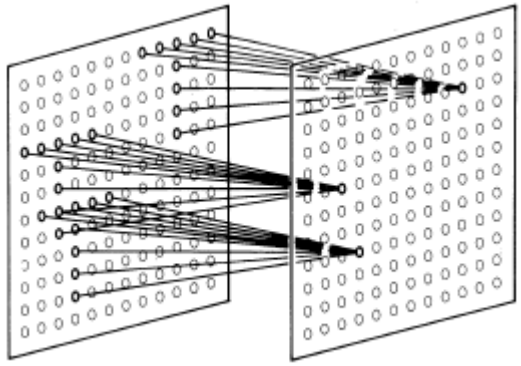
C-váhy	c
Vrstva	l
Koordináty konkrétní buňky	n
Koordináty buňky v připojovací oblasti	ν
Počet C-oblastí v C-vrstvě	κCl
Oblast připojená k S-buňkám	A
Výstupní hodnota V-buněk	uVl

5.2.2.3 C-Buňky

Cílem C-buněk je poskytnout odolnost proti posunutí či otočení vzoru. Aktivita těchto buněk je přímo odpovídající d-váze a hodnotě S-buněk v připojovací oblasti. Připojovací oblast jednotlivých buněk C-buněk v S-vrstvě se překrývá, viz obr. 13. S-buňka tedy ovlivňuje více C-buněk. Jelikož C-oblast je rozostřením S-vzorové oblasti je tímto je zajištěna určitá odolnost oproti deformacím. Výstupní hodnota C-buněk je popsána následujícím vztahem (Fukushima 1992).

$$u_{Cl}(n,k) = \psi\left[\sum_{\kappa \in K_{Sl}} \sum_{\nu \in Dl} d_l(\nu) \cdot u_{Sl}(n+\nu, \kappa)\right] \quad (9)$$

d-váha	d
počet S-oblastí v S-vrstvě	KSl
oblast spojující C-buňky	Dl
přenosová funkce	Ψ



Obr. 13 Překrytí mezi C a S buňkami (Fukushima, 1980)

5.3 Učící fáze

Učení se skládá z předkládání učících vzorů a nastavení upravených vah tak, aby byla síť schopná úspěšně detekovat vzory. Na počátku učení jsou veškeré váhy nastaveny na hodnotu 0 a učení začíná od nejnižších vrstev. Nejdříve se učí jednotlivé S-vrstvy v první úrovni. Vybraná S-oblast vybere jednu buňku zvanou „seed“ a na vstup je předložen požadovaný vzor. Pro tento seed jsou nastaveny seed váhy, které jsou pro stejnou oblast sdílené a jsou automaticky nastaveny s ohledem na váhy ostatních buněk. Takto se pokračuje s ostatními S-povrchy. Po nastavení všech S-oblastí pokračuje učící proces v S-vrstvě vyšší úrovně. Každé S-ploše se předkládá jeden trénovací vzor, může jich být předloženo i více. Pro každou úroveň se předkládá vzor jen s určitým příznakem (rozpoznává se pouze charakteristická část daného vzoru). Cílem nastavení příznaků je, aby se pro danou úroveň detekoval rozdíl, který odlišuje jednotlivé znaky.

Po kompletním naučení sítě, má síť podobnost s hierarchickým neuronovým systémem navrženým Hubelem a Wieselem v roce 1962. Pro učení neocognitron sítě se osvědčily algoritmy Levenberg-Marquard, nebo Gauss-Newton (Fletcher, 2000) (Ranganathan, 2004).

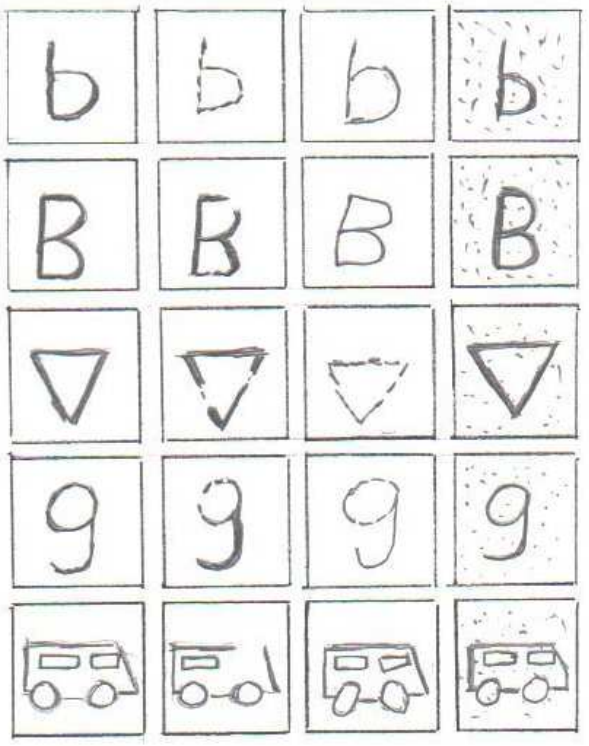
5.4 Rozpoznávací proces

Proces rozpoznávání, zvaný vybavovací fáze, spočívá v rozpoznávání předkládaných vzorů. Cílem je rozlišit do které kategorie určité vzory patří. Proces se skládá z následujících kroků:

1. Vzor je předložen na receptorové buňky vstupní vrstvy.
2. Je určena hodnota V-buněk v první úrovni ve V-vrstvě.
3. Výpočet hodnot S-buněk (detekce jednoduchých příznaků).
4. Zpracování C-vrstvou, která obraz rozostří.
5. Proces je analogicky opakován na vyšších vrstvách. Výstupem C-buněk je míra podobnosti předloženého vzoru s kategorií, kterou buňka reprezentuje.

5.5 Biologická analogie

Mezi vlastnosti Neocognitronu patří schopnost rozpoznávat s nízkou chybovostí při vysokém šumu, při předložení tvarů s různými pozicemi, deformacemi, či velikostmi. Popsané defekty jsou zobrazeny na obrázku 14. Tyto schopnosti jdou dány využíváním Gestal psychologie popsané v kapitole 5.5.1.

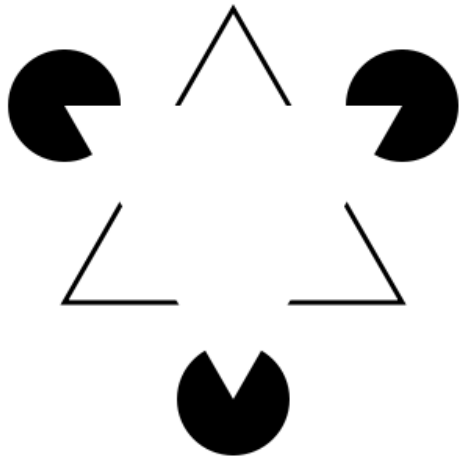


Obr. 14 Defekty rozpoznávaných vzorů (vlastní zdroj)

5.5.1 Gestalt psychologie

Gestalt psychologie (Gestaltismus) vznikla z německého Gestalt na základě myšlenky, že na vše je potřeba pohlížet v celku a ne jen jako na sumu částí. Teorie je tedy založená na teorii celistvosti, ze které vyplývá, že rozpoznávání tvarů je výsledkem umělé abstrakce a konstrukce. Tímto přístupem se vysvětluje způsob, jak je člověk schopen rozpoznávat bez velké náročnosti a přemýšlení, tvary, které jsou např. nekompletní, zašuměné apod. Typickým příkladem potvrzujícím teorie Gestaltismu je tzv. Kaniszův trojúhelník, viz obr. 15, který slouží jako názorná ukázka, jak si lidské oko a mysl dokáží vytvořit a doplnit chybějící tvary. Postup rozpoznávání zobrazeného obrazce by na základě metod, které by posuzovaly

pouze jednotlivé části obrazce, nebyl možný a výsledek by neodpovídal požadované hodnotě (Green, 2000).

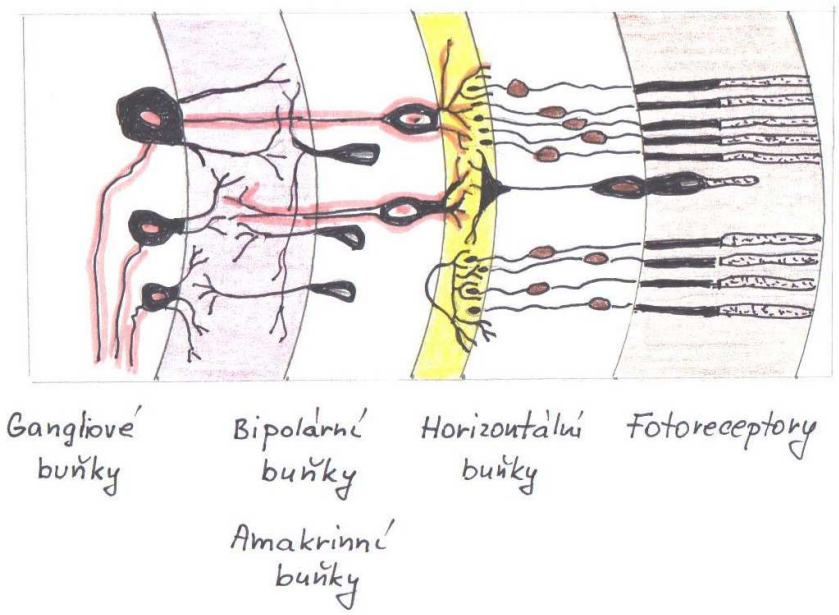


Obr. 15 Kaniszův trojúhelník (Ueda, 1998)

5.5.1.1 Gestalt zákony

- Zákon blízkosti – tendence vnímat podobné objekty jako skupiny, nebo série.
- Zákon podobnosti - skupiny podobných a odlišných objektů vidíme po skupinách.
- Zákon pokračování nebo směru - v obrazcích hledáme čáry s nepřerušeným pokračováním.
- Zákon výstižnosti - tendence vidět nejjednodušší tvar.
- Zákon dobrého tvaru - tendence doplňovat obrazce.
- Vnímání figury a pozadí - schopnost mysli zaměřit pozornost na smysluplný tvar a ignorovat zbytek.
- Konstantnost velikosti - schopnost vnímání perspektivy.

5.5.2 Biologické procesy



Obr. 16 Sítnice oka a struktura buněk (vlastní zdroj)

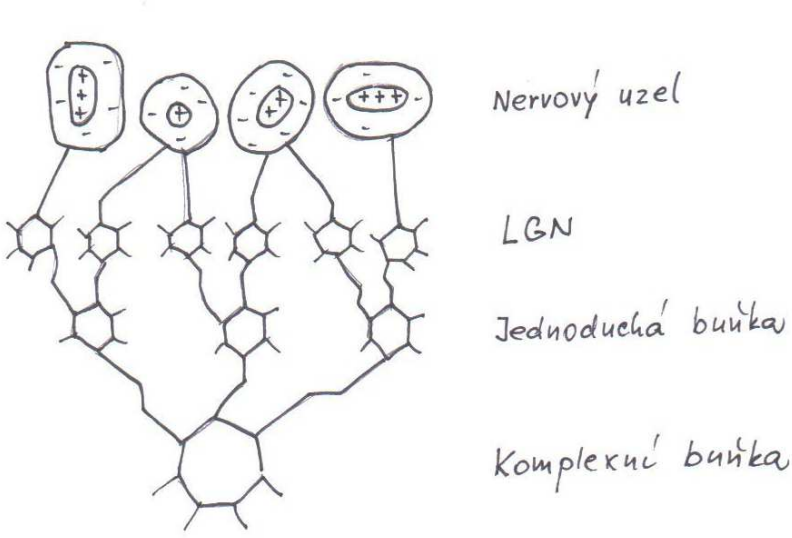
Proces rozpoznávání se dá analogicky pozorovat i v biologii. Z fotoreceptorů se signál šíří k horizontálním a bipolárním buňkám (přenášejí obrazové informace a částečně je předzpracovávají) a poté k buňkám amacrine a ganglion. Celý tento proces je názorně zobrazen na obr. 16. V Gangliových buňkách jsou detekovány hrany a tyto buňky jsou dvou typů:

- On center – detekce světlých míst obklopených tmavým
- Of center – detekce tmavých míst obklopených světlým

Z výše uvedeného vyplývá, že na oční sítnici dochází k preprocessingu signálu a získání hran obrazu. Touto detekcí hran jsou získány pouze signifikantní data (rozpoznávají se pouze důležité informace, znamená to i úsporu přenosu těchto informací).

Ve zrakovém centru mozku se nachází jednoduché, komplexní buňky a komplexní buňky s inhibicí, které jsou určeny pro detekci jednoduchých, a složitějších tvarů, viz obr 17.

- Jednoduché buňky - detekují rovné hrany v konkrétním místě a s konkrétní orientací.
- Komplexní buňky - spojují informace z jednoduchých buněk, v případě, že je alespoň jedna ze zpracovávaných jednoduchých buněk aktivní, aktivuje se i komplexní.



Obr. 17 Proces rozpoznávání v centru mozku (vlastní zdroj)

Z výše stručně popsaného biologického procesu je na první pohled patrná podobnost s neuronovou sítí Neocognitron a i při zkoumání jednotlivých částí procesu je možné na obou procesech (jak v sítí Neocognitron, tak v biologickém procesu) pozorovat analogii.

6 DETEKCE HRAN

Detekce hran v obraze je disciplnína hledající skupiny pixelů, jejichž jas se značně mění. Tato skupina pixelů je vnímána jako okraj objektu. Typickým příkladem jsou okraje tabulky. Tato detekce na poli OCR je zjednodušená díky rozpoznávání formulářů v odstínech šedi. Převod do černé a bílé je jedním z částí přípravné fáze OCR procesu.

Přechod mezi těmito dvěmi barvami, neboli hrana, může být popsána jako přechodová funkce získaná pohybem obrazu. Tyto přechody jsou vyhledávány algoritmy pro detekci hran (Javidi, 2002).

6.1 Detektory hran

Existuje více druhů detektorů hran. Tyto detektory se liší v mnoha parametrech a možných nastaveních. Základní parametry jsou:

- způsob hledání okraje,
- typ okraje,
- přesnost rozpoznání,
- odolnost proti šumu,
- odolnost proti dalším chybám v obraze.

Detektory hran jsou rozděleny na základě způsobu jejich rozhodování o hranách do dvou kategorií:

- detektory využívající první derivace jasové funkce,
- detektory používající druhou derivaci jasové funkce.

V prvním případě je gradient hrany porovnáván s prahem a výsledek určuje existenci hran. Algoritmus závisící na druhé derivaci jasové funkce porovnává změnu polarity a její důležitost.

Nejvýznamnější změna v jasové funkci indikuje přítomnost hrany a je nejsilnější v kolmém směru. Této vlastností je využíváno v praxi. Kalkulace je provedena pouze ve dvou nebo čtyřech směrech a výsledek gradientu je implementován jako konvoluční filtr obrazu. Všechny hranové filtry se liší v jádru filtrování. V maticích jsou zapsány hodnoty, které určují kolik bodů je použito ke kalkulaci velikosti a vah v této kalkulaci. Velikost matic ovlivňuje odolnost vůči šumu a dalším chybám v obraze.

Metody založené na druhé derivaci jasové funkce hledají přechod nulou touto derivací. Využívá se fakt, že je jednodušší najít průchod nulou spíše než funkce extrému. Druhá derivace je však mnohem citlivější na šum než derivace první, takže je vhodné kombinovat výpočet s takovým vyhlazení, který odstraňuje maximální množství šumu a zároveň nepoškozuje hrany. Tyto metody často pracují s Laplaceovým operátorem - Laplaciánem, který aproximuje druhou derivaci všesměrové funkce jasu. Výpočet používá pouze jednu matici a špatně zpracovává tenké okraje. V případě tenkých okrajů reaguje dvakrát. Nejznámější hranové detektory jsou uvedeny na obr. 18.

	G_x	G_y
Robertsův	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Prewittové	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobelův	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Robinsonův	$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
Kirsehův	$\begin{bmatrix} 3 & 3 & -5 \\ 3 & 0 & -5 \\ 3 & 3 & -5 \end{bmatrix}$	$\begin{bmatrix} -5 & -5 & -5 \\ 3 & 0 & 3 \\ 3 & 3 & 2 \end{bmatrix}$
Frei-Chenův	$\begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 0 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$
Prewittové(5x5)	$\begin{bmatrix} 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \end{bmatrix}$	$\begin{bmatrix} -2 & -2 & -2 & -2 & -2 \\ -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$

Obr. 18 Nejznámější hranové detektory

6.1.1 Cannyho hranový detektor

Cannyho hranový detektor je jedním z hranových detektorů a je obecně považován za optimální. Je navržen tak aby splňoval tři základní požadavky na hranový detektor:

- Minimální počet chyb – je potřeba nalézt všechny hrany v obraze a v místech bez hran musí být návrat roven nule.
- Přesnost – přesné určení pozice detekovaných hran.
- Unikátnost – každá hrana musí být detekována pouze jednou, nejsou povoleny duplicitní hrany.

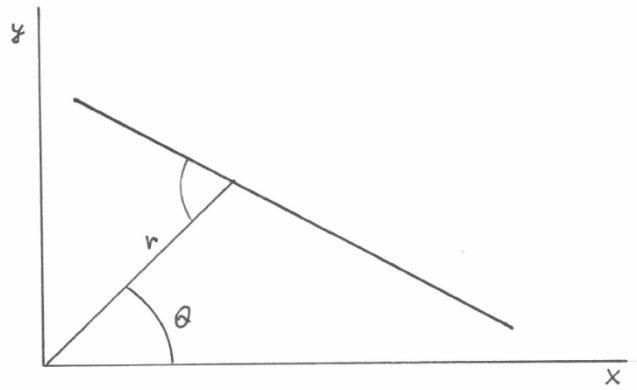
Proces detekce hran se skládá z několika kroků. Prvním krokem je odstranění šumu - na obrázku je aplikován Gaussův šum, a je aplikován filtr, používající konvoluční masku. Dalším krokem je aplikace Sobel operátoru. Na obraz je aplikována konvoluční maska detektoru Sobel a je nalenen směr gradient hrany. Ve ztenčovacím procesu jsou detekované hrany zeslabeny a umístěny na správnou pozici. Princip této fáze spočívá v označení hran pouze body a sousedící body v kolmém směru gradient (směr gradientu je známý, je vrácen Sobelovým detektorem) mají nižší hodnotu gradientu. V posledním kroku je provedeno zpracování obrazu, při kterém jsou ohodnoceny nalezené hrany významností. Canny detektory využívají dvou prahů; když hodnota překročí horní mez prahu, je detekován hrana gradientu. Pokud je hodnota nižší než nižší úroveň gradientu, není hrana detekována. Pokud je hodnota mezi prahy, je hrana rozeznána pouze, pokud sousedí s jinou hranou. Canny hranový detektor je v současnosti považován za optimální detektor (Ali, 2001).

6.1.2 Houghova transformace

Houghova transformace je metoda, která pomáhá najít parametrický popis objektů v obraze. Nevýhodou je předpoklad znalosti analytického popisu tvaru objektu, který hledáme. Z tohoto důvodu je metoda používána pouze pro nalezení známých objektů, jako jsou přímky, kružnice, elipsy a složené tvary čtverce, trojúhelníky. Největší výhodou této metody je robustnost a odolnost vůči šumu. Metoda používá převod z kartézského do polárního systému. Detekce přímek v dokumentu je použita k detekci rotace dokumentu a může být použita ke korekci. Příklad parametrického popisu je zapsán následující rovnicí a je dále vysvětlen na obr. 19.

$$r = x \cdot cos\theta + y \cdot \theta \tag{11}$$

Kde r je vzdálenost od normály k přímce a θ je úhel mezi normálou a osou x.



Obr. 19 Parametrický popis přímky (vlastní zdroj)

Transformace je provedena tak, že parametry x and y každého z počátečních bodů obrazu reprezentující bod, přímku nebo segment, nahrazeny do rovnice. Hodnota θ roste náhradou každé možné hodnoty v intervalu <0, 360>

(jedná se nekonečnou sadu reálných čísel představujících úhel θ, ale v případě implementace jsou hodnoty intervalu závislé na nastavení Houghovy transformace). V parametrickém prostoru jsou křivky generovány pro každý počáteční bod. Pokud jsou počáteční body kolineární, pak křivky protínají parametrický prostor v bodě představujícím nejpravděpojší umístění přímky (Guo, 2008).

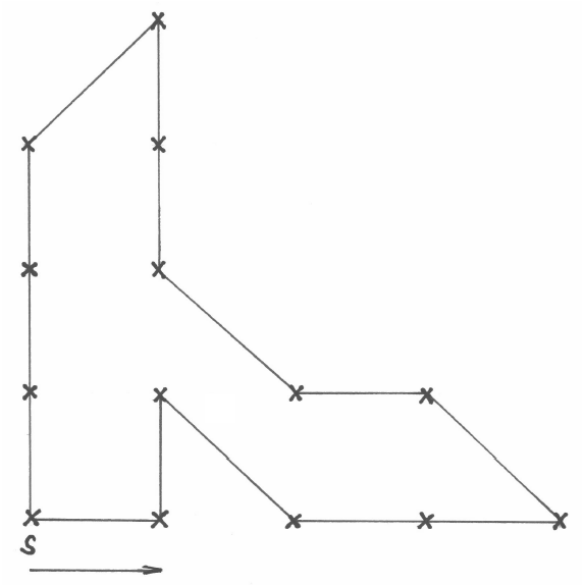
6.1.3 Reprezentace hran

Reprezentace hran nám poskytuje možnosti, pomocí kterých můžeme ukládat a prezentovat data získané hranovými detektory. Prezentace těchto hran je ve většině případů nedostatečná a je potřeba provést úpravy před jejich dalším použitím v dalším procesu. Po zisku obrysů v obraze je potřeba zpracovat tyto obrysy zpracovat. Jedním z kritérií pro zpracování těchto obrysů je jejich původ. V případě ukládání hran, kde je předpoklad dlouhých přímek nebo malý počet ostrých zakřivení je výhodné použít metodu ukládání matematickými funkcemi. Pokud je nutné zpracovávat rastrová data, kde se očekává velké množství ostrých zakřivení, zatížení šumem a dalšími obrazovými vadami nebo je zde velmi mnoho nepravidelných tvarů, je lepší použít řetězce kódů.

6.1.4 Řetězcové kódy

Řetězcové kódy pracují na principu, který je založen na faktu, že cestu z jednoho bodu do sousedního bodu, popisované kontury, lze označit – například číslicí. Pokud uvažujeme, že se pohybujeme na úrovni jednotlivých pixelů, musíme podle zvolené provázanosti takto označit 4 resp. 8 směrů. Freemanův řetězcový kód pracuje s 8mi směry a s jejich označením číslicemi od 0 do 7. Tyto směry si můžeme představit jako možnosti tahu s dámou na šachovnici. Potom řetězcový

kód kontury s počátkem v bodě S bude vypadat takto {2,0,3,2,2,7,6,7,0,0,5,4,4,4}. Tento řetězcový kód je také uveden na obr. 20.



Obr. 20 Kontura řetězcového kódu [vlastní zdroj]

PRAKTICKÁ ČÁST

7 NALEZENÍ VHODNÉ NEURONOVÉ SÍTĚ PRO ICR SYSTÉM

Pro splnění jednoho z hlavních cílů této disertační práce bylo potřeba zvolit vhodnou neuronovou síť, na které by byla následně provedena měření na různých testovacích vzorcích, poté byla optimalizována na těchto testovacích datech a následně použita jako hlavní rozpoznávací prvek v navrženém ICR systému. Za tímto účelem byly ve skriptovacím prostředí Matlab implementovány algoritmy pracující neuronovými sítěmi. V rámci implementovaných algoritmů bylo možné nastavit veškeré potřebné parametry pro otestování možností, které nabízejí a optimalizovat je pro účely disertační práce. Implementované a optimalizované algoritmy byly následně otestovány na testovacích datech získaných z databáze MNIST, viz kapitola 8.1. Získané výsledky z měření byly porovnány a pro následující práci byla vybrána síť Neocognitron. Implementace a testování neuronových sítí je detailně popsáno v následujících kapitolách.

7.1 Testování perceptron sítě

První sítí, vybranou k testování, byla vícevrstvá neuronová sít Perceptron. Ve skriptovacím prostředí Matlab byl vytvořen algoritmus, který pracuje s vícevrstvou neuronovou sítí Perceptron. Pomocí algoritmu je možné nastavit počet neuronů ve skrytých vrstvách, typ přenosových funkcí, a upravit metody učení a mnoho dalších parametrů.

První navržená síť odpovídala síti s určitým počtem výstupních neuronů odpovídajícím počtu klasifikačních tříd. Druhá byla navržena jako „pletivo“ neuronových sítí s jedním výstupem. Počet kombinací byl dán $\frac{n!}{2(n-2)!}$, kde n je počet znaků. Navržená druhá varianta byla ve výsledku mnohem více náročná na dobu potřebnou k naučení sítě.

Klíčovou vlastností při rozpoznávání znaků při použití neuronových sítí je vstup neuronové sítě a s ním vztažená extrakce markantů (významných znaků). Hlavním cílem extrakce je získání sady charakteristických znaků, pro zajištění nejlepší klasifikace s nejmenším počtem elementů. V průběhu řešení byly testovány různé nastavení, pokrývající jak statistické (počet zón, sekcí) tak i strukturální (histogram, Fouriérova transformace) charakteristiky. Bylo provedeno množství testovacích měření za účelem nalezení nejvhodnější kombinace, která by byla nejvhodnější ve smyslu správného rozpoznávání. Důležitým znakem navržené sítě byl nepoměr mezi schopností rozpoznávání a správnou klasifikací známých a neznámých znaků. V případě známých znaků se úspěšnost pohybovala přes 90%. Po předložení testovací sady neznámých znaků se však procentuální úspěšnost snížila pod 50%, Dále bylo na základě testů zjištěno, že síť je velmi náchylná k chybám při předložení znaků s posunutím, odlišnou rotací, a odlišnou velikostí, což korespondovalo s teoretickými poznatky, získanými rešeršemi v teoretické části. Na základě těchto poznatků byla tato síť vyhodnocena jako nevyhovující pro použití v dalším výzkumu.

7.2 Testování Neocognitron sítě

V prostředí Matlab byl vytvořen algoritmus, pracující s konvoluční sítí. Podobně, jako v případě testování perceptron sítě v kapitole 7.1, bylo možné zvolit počet skrytých vrstev, počet buněk ve skrytých vrstvách, typy přenosových funkcí a mnoho dalších parametrů. V tomto případě byly pro testování použity dva zdroje dat. Prvním zdrojem byla podobně jako v předchozím případě databáze MNIST. Druhým zdrojem dat byla vlastní databáze znaků získaná sběrem podrobněji popsaným v kapitole 8.

7.2.1 Testování vlivu počtu znaků

V následujících kapitolách jsou provedeny testy učení neuronových sítí a jsou prezentovány výsledky těchto testů formou zobrazení učících křivek a důležitých parametrů potřebných pro učení a také získaných měřením při učení. Mezi tyto důležité parametry patří:

- RMSE (Root Mean Square Error) parametr reprezentuje střední kvadratickou chybu.
- CR (Classification Rate) parametr představuje úspěšnost procesu klasifikace, která se v průběhu učení mění (zvyšuje). V průběhu času učení můžeme pozorovat, že síť se již více neučí. Tento poznatek lze využít pro úsporu času, neboť není nutné síť učit na více vzorech znaků než je potřeba. Z důvodu úspory času, byly nadále používány testovací sady znaků obsahující v řádu několika set vzorů každého znaku.
- Iterace – odpovídá počtu variant všech znaků.
- Epochy – počet opakování všech iterací.

Výše uvedené testování učení sítě bylo provedeno na různých počtech vzorků, aby bylo možné určit závislost tohoto počtu na kvalitě naučení výsledné sítě.

7.3 Testování Neocognitron sítě na znacích s diakritikou

Jedním z hlavních cílů této práce bylo optimalizovat stávající systémy pro rozpoznávání ručně psaného písma v českém jazyce s diakritikou. Základní tezí, vedoucí k optimalizaci těchto systémů, bylo optimalizovat stávající přístup řešení problému rozpoznávání ručně psaného českého textu o prvek inteligentního rozpoznávání, který má za účel zjednoduššení komplexnosti vstupu do neuronové sítě a z toho plynoucí lepší výsledky poskytované neuronovou sítí. Pro ověření této teze byl implementován kód pracující s neuronovou sítí Neocognitron ve

skriptovacím prostředí Matlab a bylo provedeno měření pro jednotlivé typy vstupů viz kapitola 7.3.1. Testování této neuronové sítě, jejíž výsledky, prezentované v následujících kapitolách, potvrzují teoretický předpoklad vysoké úspěšnosti v rozpoznávání znaků, ale také potvrzují předpokládané schopnosti sítě odolávat negativním vlivům, jako je posunutí, rotace a defekty předložených znaků. Na základě těchto vlastností, konzultací se školitelem i diskuse k rámci disertační práce, byla síť vybrána jako stěžejní pro další pokračování ve výzkumu.

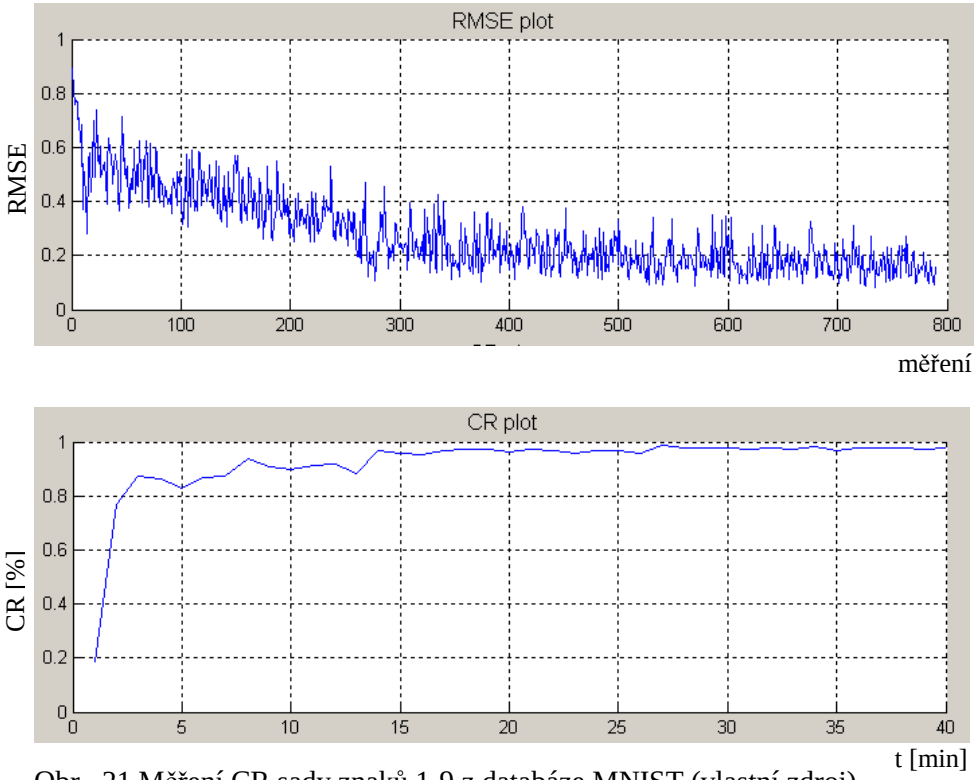
7.3.1 Vliv komplexnosti znaků - diakritiky

Pro ověření vlivu komplexnosti vstupních znaků na schopnosti sítě rozpoznávat jednotlivé znaky bylo potřeba provést měření CR pro jednotlivé varianty. Pro urychlení těchto měření a názornější zobrazení bylo toto měření provedeno v rámci učícího procesu, kdy po každém kroku učení je provedeno okamžité vyhodnocení CR předložením testovací sady znaků. CR odpovídá procentuálnímu vyjádření správně rozpoznaného počtu znaků z kompletního počtu znaků, použitých pro testovací rozpoznávání.

Prvním provedeným měřením CR neuronové sítě Neocognitron bylo měření na sadách číslic získaných vlastním sběrem a dále na vzorcích získaných z databáze MNIST. Tato měření neměla vliv na optimalizaci systému pro čtení českého písma s diakritikou a byla provedena za účelem ověření základní funkčnosti implementovaných algoritmů neuronové sítě Neocognitron a také jejího správného nastavení. Z důvodu zajištění srovnatelných výsledků byl test proveden na sadě stejného počtu znaků, jako v případě vlastní sady znaků, která v době měření čítala 264 variant každého znaku. Z uvedeného grafu na obr. 21 je patrné, že vytvořený algoritmus byl schopen rozpoznávat znaky číslic získané z databáze MNIST s úspěšností 97% a tento výsledek byl potvrzením, že algoritmy neuronové sítě (včetně algoritmů pro preprocessing znaků a rozpoznávacího algoritmu) jsou

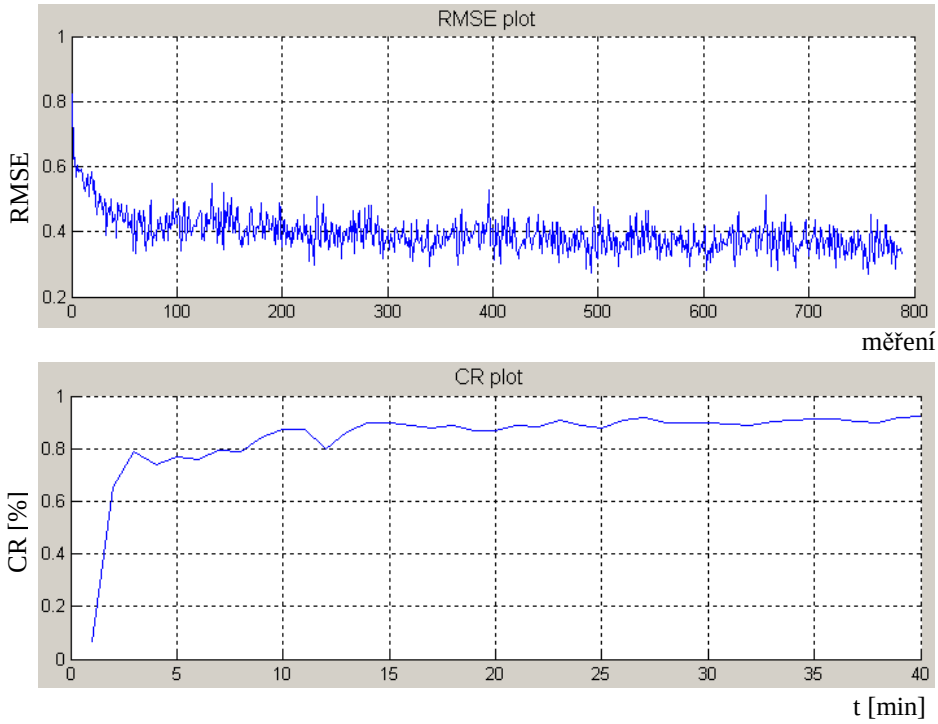
ve skriptovacím prostředí Matlab implementovány správně a je možné pokračovat s těmito algoritmy v další práci.

V uvedeném grafu na obr. 21 je zobrazen průběh učení sítě s průběžným vyhodnocováním úspěšnosti rozpoznávání a střední kvadratické odchylky. Z průběhu měření je patrné, že CR roste pouze na počátku učení a v dalším průběhu už schopnost učení slábne. Na základě tohoto zjištění bylo z důvodu úspory času následné učení prováděno pouze ve třech epochách. Finální učení sítě pro produkční řešení by bylo samozřejmě provedeno ve více epochách z důvodu kvalitnějšího naučení sítě.



Obr. 21 Měření CR sady znaků 1-9 z databáze MNIST (vlastní zdroj)

Nastavení:	0-9_MNIST_28x28_3e_2640i_2340train_84n_04t
Epochy:	3
Iterace:	2 640
RMSE:	0,16
CR:	97%



Obr. 22 Měření CR sady znaků 1-9 z vlastního sběru (vlastní zdroj)

Nastavení:	0-9_vlastni_28x28_3e_2640i_2340train_84n_04t
Epochy:	3
Iterace:	2 640
RMSE:	0,33
CR:	92%

Druhou variantou, která byla otestována, byla síť naučená na vlastní sadě číslic. Při tomto měření byla zjištěna hodnota CR na úrovní 92%, viz obr. 22, což je ve srovnání s přechozí variantou, učenou na sadě znaků z databáze MNIST, velmi dobrý výsledek. Při srovnání je potřeba přihlížet k faktu, že byla použita pouze omezená sada znaků z MNIST databáze, aby byly výsledky srovnatelné. Pro případ použití kompletní databáze znaků byly získány výsledky 97% úspěšnosti rozpoznávání, viz kapitola 7.2.1.

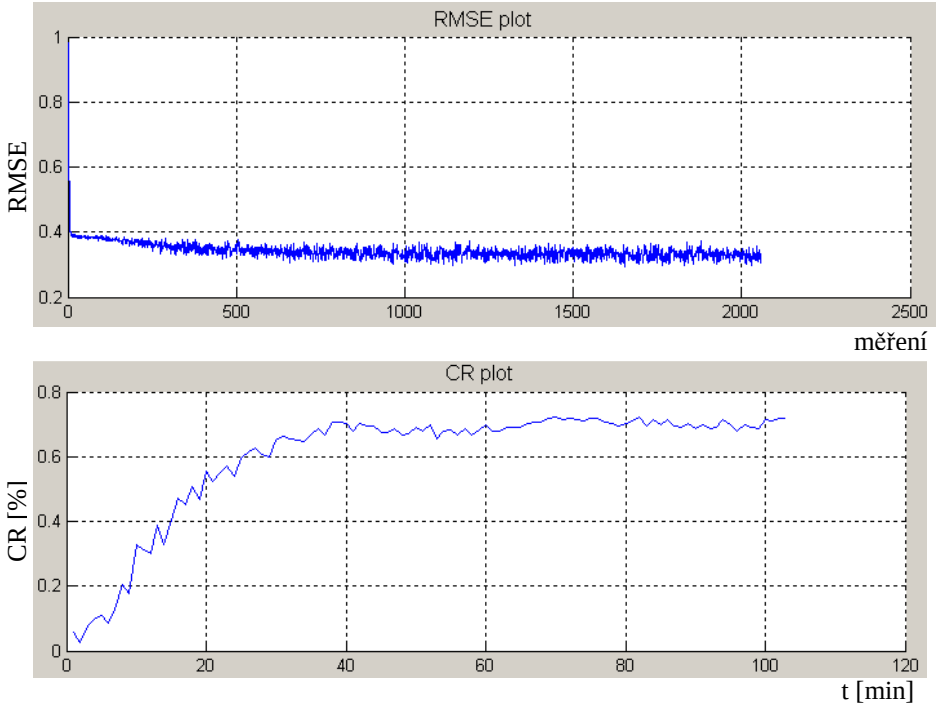
Dalšími provedenými experimenty, již více relevantními pro další pokračování v práci, byla měření úspěšnosti rozpoznávání implementovaných algoritmů, učených na sadách znaků abecedy bez diakritiky, znacích s diakritikou a kompletní sadě znaků včetně diakritiky i bez diakritiky. Výsledky z provedených měření jsou uvedeny v tab. 3.

Sada znaků	**CR**	**Popis**
0-9	93%	sada číslic
0-9	97%	sada číslic z databáze MNIST
A-Z	72%	sada písmen bez diakritiky
A-Ž	50%	sada písmen kompletní (s i bez diakritiky)
Á-Ž	71%	sada písmen pouze s diakritikou

Tab 3 Naměřené CR v závislosti typu vstupních vzorků

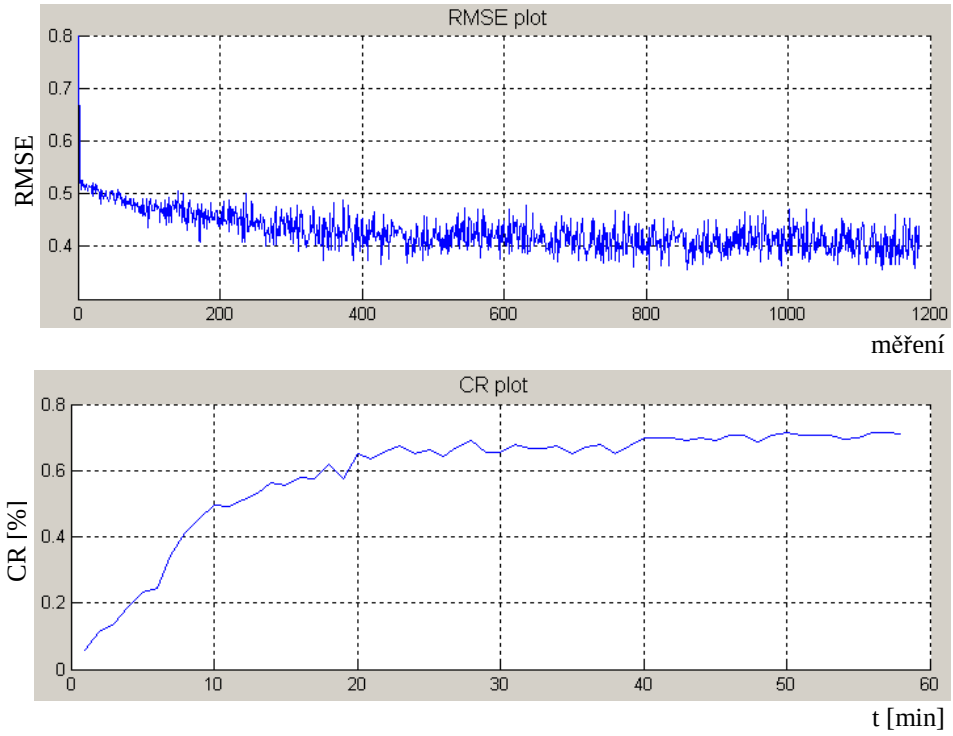
Z výše uvedených výsledků vyplývá závislost CR neuronové sítě na sadě použitých znaků pro učení sítě, a je přímo úměrná počtu znaků, na které je potřeba síť naučit a komplexnosti těchto znaků. Tyto výsledky potvrzují navržený plán optimalizovat výsledný systém pomocí prvku, který by byl schopný rozlišit, zda znak obsahuje diakritiku či nikoliv.

V případě zařazení uvedeného prvku inteligentního preprocesingu lze dosáhnout snížení počtu znaků na vstupu neuronové sítě a tímto krokem zvýšit úspěšnost rozpoznávání. V případě kdyby prvek inteligentního preprocesingu fungoval se 100% úspěšností, bylo by možné rozpoznávat znaky z kompletní sady znaků s úspěšností rozpoznávání na úrovni neuronových sítí naučených pouze pro znaky bez diakritiky a neuronové sítě pro znaky pouze s diakritikou. Průběhy učení neuronových sítí na jednotlivé varianty vstupů, pomocí implementovaných algoritmů, jsou zobrazeny na následujících grafech.



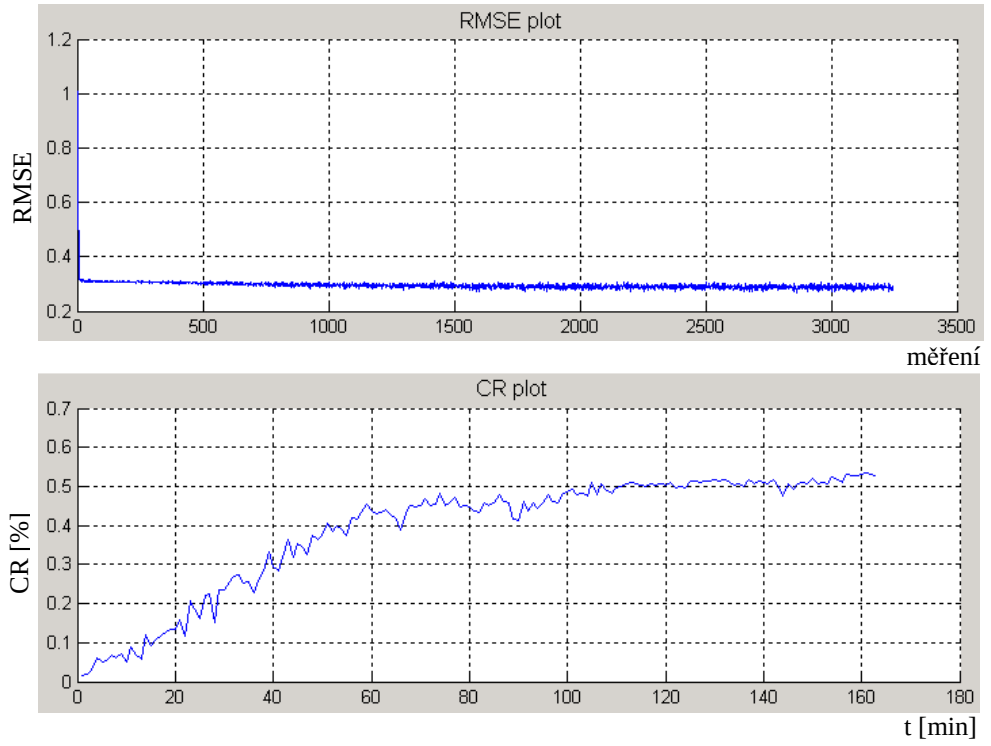
Obr. 23 Měření CR sady znaků A-Z (vlastní zdroj)

Nastavení:	A-Z_vlastni_28x28_3e_6864i_6564train_84n_04t
Epochy:	3
Iterace:	6 864
RMSE:	0,31
CR:	72%



Obr. 24 Měření CR sady znaků Á-Ž (vlastní zdroj)

Nastavení:	Á-Ž_vlastni_28x28_3e_3960_3660train_84n_04t
Epochy:	3
Iterace:	3 960
RMSE:	0,39
CR:	71%



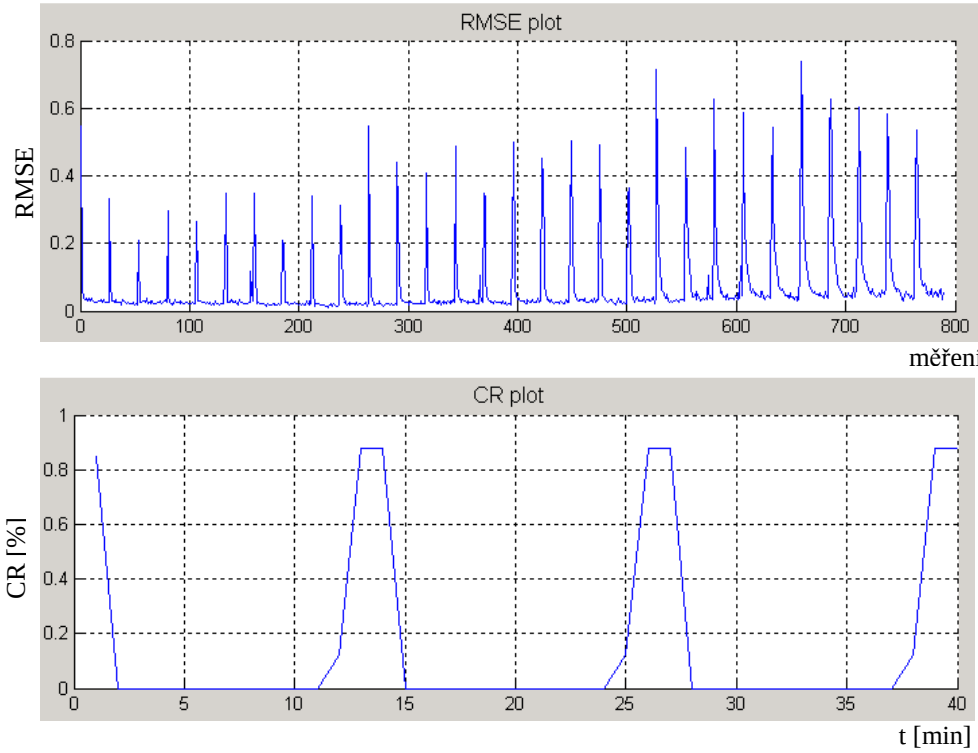
Obr. 25 Měření CR sady znaků A-Ž (vlastní zdroj)

Nastavení:	A-Ž_vlastni_28x28_3e_10824_10524train_84n_04t
Epochy:	3
Iterace:	3 960
RMSE:	0,29
CR:	52%

7.3.2 Vliv pořadí znaků na kvalitu učení neuronové sítě

V průběhu testování vytvořených algoritmů na různých sadách znaků, viz kapitola 7.3.1, bylo zjištěno, že na kvalitu naučení neuronové sítě, která v našem případě odpovídá CR, má vliv, v jakém pořadí předkládáme znaky k učení této neuronové síti. Tato vlastnost je znázorněna na obr. 26, který znázorňuje proces učení neuronové sítě na sadě číslic 0-9 z vlastní databáze znaků, které jsou síti

předkládány v pořadí dle abecedy. Z uvedeného grafu lze vypozorovat, že proces učení sítě neprobíhá správně a i výsledná úspěšnost po skončení učení je menší než v případě, kdy jsou znaky zpřeházeny. Konkrétně se jedná o pokles úspěšnosti o 5% na výsledné CR = 88%. Z tohoto důvodu bylo vždy pro další učení sítě pořadí znaků zpřeházeno do náhodného pořadí.



Obr. 26 Měření CR sady znaků bez random funkce 0-9 (vlastní zdroj)

Nastavení:	0000-9999_vlastni_28x28_3e_2640_2340train_84n_04t
Epochy:	3
Iterace:	2 640
RMSE:	0,06
CR:	88%

8 VYTVOŘENÍ DATABÁZE VZORŮ V ČESKÉM JAZYCE

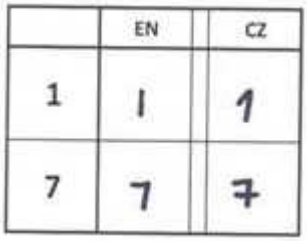
V rámci řešení rámce disertační práce je také řešen jeden ze současných problémů optimalizací OCR systémů pro český jazyk. Spočívá v nedosažitelnosti veřejně dostupné databáze, ručně psaných znaků, v českém jazyce s diakritikou. V počátku výzkumu byla z důvodu nedostatečného množství těchto vzorů využita veřejně dostupná databáze MNIST, vytvořená a poskytovaná Národním Institutem Standardů a Technologií ve Spojených Státech.

8.1 MNIST

Databází MNIST ručně psaných znaků je více druhů. V našem případě byla použita databáze obsahující trénovací sadu 60 000 vzorů a sadu 10 000 vzorů pro testování vybavovací fáze. Jedná se o část velikostně normalizovaných a vycentrovaných obrázků stejné velikosti z kompletní NIST databáze. Tato databáze je také poskytována Národním Institutem pro standardy a technologie. Databáze je vhodná pro testování učících algoritmů, pro rozpoznávání vzorů a klasifikaci. Veškeré znaky jsou normalizovány a vycentrovány na střed. Jednoznačnou výhodou použití této databáze pro předběžný výběr neuronových sítí a následné testování je úspora času, protože databáze je již předpřipravena a není potřeba žádného dalšího preprocessingu nebo jiných zásahů. Jednotlivé vzory v databázi jsou uloženy ve stupních šedi a jsou normalizovány na velikost 20x20 pixelů při zachování poměrů stran.

Přes veškerá pozitiva, která přináší použití databáze znaků MNIST nebo rozšířené databáze NIST (pro kompletní sadu písmen) bylo potřeba vytvořit vlastní databázi, a to z důvodu nepoužitelnosti těchto databází k testování, učení, optimalizaci a další práce se sadou znaků v českém jazyce s diakritikou. Chybějící diakritika není jediným problémem, který s sebou nese použití NIST databáze. Dalším aspektem ovlivňujícím úspěšnost rozpoznávání je také styl písma, kdy

ačkoliv téměř celý svět používá latinku, v regionálním použití jsou jemné nuance, kterými se liší. Typickým příkladem odlišných znaků v americké angličtině, která je obsahem databáze MNIST, jsou číslice jedna a sedm. Číslice jedna je v angličtině ukončena v horní části zalomením, zatímco v českém jazyce je psána jednička se zalomením. Tyto a podobné rozdíly jsou zobrazeny na obr. 27.



Obr. 27 Rozdíly ve stylu ručně psaného písma (vlastní zdroj)

Vzhledem k tomu, že hlavním cílem disertační práce, jak již z názvu práce vyplývá, bylo optimalizovat rozpoznávácí systémy ručně psaného písma v českém jazyce, bylo potřeba přistoupit k vytvoření vlastní databáze vzorů. Postup vytvoření této databáze je popsán detailněji v následujících kapitolách (Wang, 2011).

8.1.1 National Institute of Standards and Technology

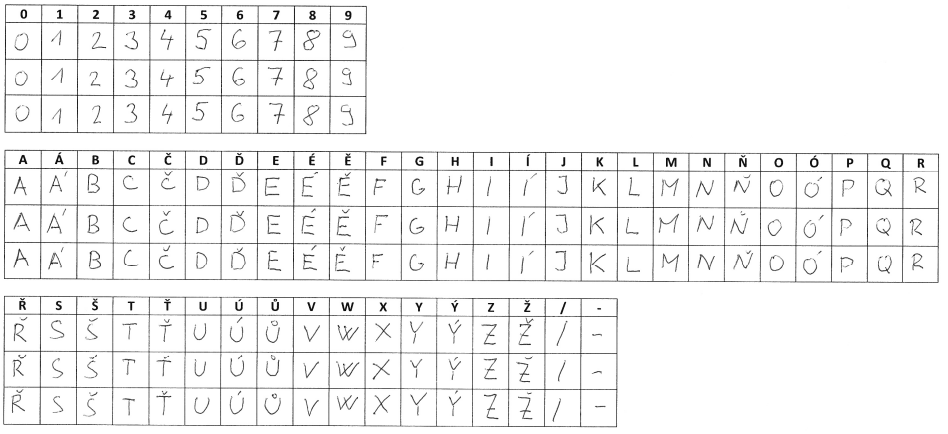
Národní institut pro standardy a technologie, který je federální agenturou pod ministerstvem obchodu USA, byl založen v roce 1901, a jedná se o jednu z nejstarších vědeckých fyzikálních laboratoří. Výzkum je zaměřen od nejmenších nano technologií až po největší a nejkomplexnější lidské výtvory, např. zemětřesení odolné mrakodrapy, trysková letadla. Jednou ze služeb institutu je poskytování databází pro vědecké účely. Pro účel této práce byla použita výše uvedená databáze znaků číslic MNIST. Nicméně NIST nabízí i širší databázi znaků, obsahující kompletní databázi znaků (číslic, malá písmena, kapitálky) v anglickém jazyce v celkovém počtu 800 tis. obrázků od 3 600 autorů. Veškeré znaky jsou ručně zkontrolovány.

8.2 Vlastní databáze vzorů

V rámci disertační práce bylo zapotřebí pro účely učení neuronových sítí, jejich testování a následné praktické použití, vytvořit dostačně velkou databázi ručně psaných znaků. Základní požadavky na vytvořenou databázi jsou uvedeny v následujícím přehledu:

- Databáze obsahující hůlkovým písmem číslice a znaky české abecedy.
- Obsahující diakritiku.
- Standardizována a normalizována.

Pro vytvoření databáze s výše popsanými parametry byl vytvořen standardizovaný formulář pro sběr dat. Tento formulář byl v rámci vypsané diplomové práce studenta naší univerzity optimalizován pro snadnější rotaci do vodorovné polohy a pro eliminaci chyb způsobených ořezaním nesprávným ořezáním polí. Z důvodu zachování stejného formátu formuláře po celou dobu testů, analýz a vzhledem k již naprogramovanému funkčnímu vytěžování těchto formulářů v rámci skriptovacího prostředí Matlab, bylo od použití optimalizovaného formuláře upuštěno a byl nadále používán původní navržený formulář, viz obrázek obr 28.



Obr. 28 Ukázka vytvořeného formuláře pro sběr dat (vlastní zdroj)

Sběr dat byl prováděn vlastními silami, za pomoci studentů a vyučujících, na půdě univerzity mezi studenty. V první fázi sběrů v rámci tvorby rámce disertační práce bylo získáno přes 300 unikátních vzorů, tzn. přes 30 tis. znaků české abecedy a číslic.

Vzhledem k tomu, že sběr probíhal v homogenní skupině přispěvatelů, kde většina respondentů byla věkové kategorie 20 - 25 let, s minimálně středoškolským vzděláním, byl ve druhé fázi proveden sběr i mimo univerzitní prostředí mezi respondenty vyššího věku, a jiné úrovně vzdělání. Protože je rozpoznávání ručně psaného písma na těchto parametrech velmi závislé, bylo tímto zajištěno větší rozmanitosti což by mělo vést k lepšímu naučení neuronové sítě. Neboť byl sběr prováděn především na kvantitu, byly formuláře označeny velmi zjednodušenou formou a to znakem, zda sběr probíhal na univerzitě či nikoliv. Dalším zaznamenávaným znakem byl věk respondenta pouhým zaznačením, zda je starší či mladší 30 let.

Výsledky z výše popsaného sběru testovacích formulářů jsou zobrazeny v tab. 4. Pro porovnání z hlediska kvantity znaků ve zdroji získaném vlastním sběrem slouží tab. 5, ve které jsou porovnány počty znaků dle jejich typu mezi vlastním zdrojem znaků a zdrojem získaného z databází institutu NIST viz kap. 8.1.1.. Pro ukázku jsou některé formuláře použité pro sběr dat uvedeny v příloze A.

8.2.1 Struktura uložení databáze znaků v digitální formě

Veškerá data, získaná sběrem dat, byla převedena skenováním, viz kapitola 11, do digitální formy a pomocí vytvořených algoritmů byla převedena do datových struktur. Tyto struktury se skládají ze dvou polí:

- All_pism – obsahuje pole charů, pro popis obsažených znaků.

- Letter – obsahuje pole, které obsahuje vnořená pole (stejného počtu a pořadí korespondující s polem All_pism. Každé toto pole dále obsahuje všechny varianty naskenovaných vzorů znaků.

Struktura je uložena v datovém formátu Matlab (*.mat), a těchto struktur je několiv verzí, odlišných podle fáze sběru a podmnožiny znaků které obsahuje. V pokračujících fázích byla již ukládána struktura s kompletní množinou sbíraných znaků, a jejich výběr byl prováděn až v algoritmech používaných pro učení sítí.

	I. Fáze	**II. fáze**	**Celkem**
Formulářů (Věk < 30 let)	265ks	212ks	477ks
Formulářů (Věk > 30 let)	5ks	110ks	115ks
Celkem formulářů	270ks	322ks	592ks
Celkem číslic (znaků)	8 100	9 660	17 760
Celkem A-Z znaků bez diakritiky (znaků)	21 060	25 116	46 176
Celkem A-Z znaků s diakritikou (znaků)	12 150	14 490	26 640
Celkem (znaků)	**41 310**	**49 266**	**90 576**

Tab 4 Výsledky sběru testovacích dat

	NIST	**Vlastní zdroj**
Číslice (znaků)	50 000	17 760
Celkem A-Z a číslic bez diakritiky (znaků)	500 000	63 936
Celkem A-Z znaků včetně diakritiky (znaků)	není dostupné	72 816

Tab 5 Srovnání vlastní databáze znaků s NIST

9 OPTIMALIZOVANÝ SYSTÉM ICR

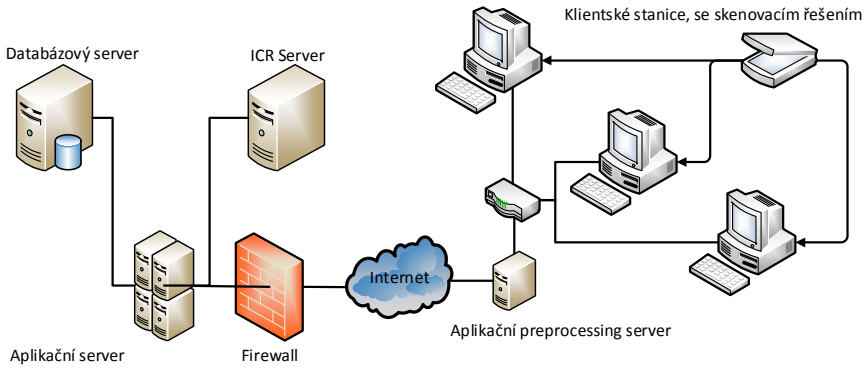
Poznatky získané výzkumem a testováním v rámci této disertační práce byly využity pro návrh optimalizovaného ICR systému pro rozpoznávání ručně psaného textu s českou diakritikou. Funkční struktura systému vychází z dnešních možností výpočetní techniky a požadavků na ni kladených. Mezi požadavky na tuto strukturu patří:

- **Bezpečnost** – z hlediska zabezpečení know-how systému, kterým je bezesporu optimalizovaná a naučená neuronová síť.
- **Škálovatelnost** – umožnit rozšířitelnost systému pro splnění větších požadavků.
- **Uživatelská dostupnost** – možnost snadné implementace do existujících systémů.

9.1 Struktura ICR systému

Splnění výše uvedených požadavků prezentuje diagram na obr. 29. Systém je rozdělen na klientskou a serverovou část, která je rozdělena pomocí brány firewall na zabezpečenou a nezabezpečenou část v souladu s tímto klient-server rozdělením. Tímto rozdělením je zajištěno zabezpečení neuronové sítě proti jejímu zcizení, či prozrazení. Další výhodou je přenesení výpočetních operací s klientských stanic na vyhrazený server, kterým vzhledem k dnešním možnostem (například cloudingu) může mít výkonově teoreticky neomezenou škálovatelnost. Tímto je zajištěna bezkonkurenční rychlost ve srovnání se systémy, instalovanými přímo na klientské stanice. Jedním z negativních aspektů navrženého systému je závislost na přenosu dat přes síť internet, a tedy na kvalitním a stabilním připojením. Z tohoto důvodu a jeho vlivu na rychlost výsledného systému, byla struktura optimalizována předsazením preprocessing serveru do klientské nezabezpečené části. Tento vyhrazený preprocessing server, jak název napovídá, předzpracovává obrazová data pro ICR server, díky této činnosti jsou přes síť

přenášeny již pouze potřebné vzory k rozpoznání v rozlišení a barevné hloubce potřebné pro ICR server. Na základě této úpravy dochází k velké úspoře dat a tedy i zvýšení rychlosti.



Obr. 29 Struktura systému z pohledu funkčních celků (vlastní návrh)

9.1.1 Zabezpečení komunikace přes síť internet pomocí VPN

V případě nasazení systému do oddělených lokací je potřeba zajistit bezpečnou komunikaci mezi serverovou rozpoznávací částí a částí pro získávání dat a jejich předzpracování. Vzhledem k tomu, že veškeré součásti potřebné pro rozpoznávání jsou dostupné interně, a není potřeba přístupu k externím zdrojům, je možné zabezpečit komunikaci přes nedůvěryhodnou počítačovou síť pomocí virtuální privátní sítě (VPN). Takto navázaná komunikace je ověřena pomocí digitálních certifikátů a po její autentizaci je veškerá komunikace šifrována a může být považována za bezpečnou.

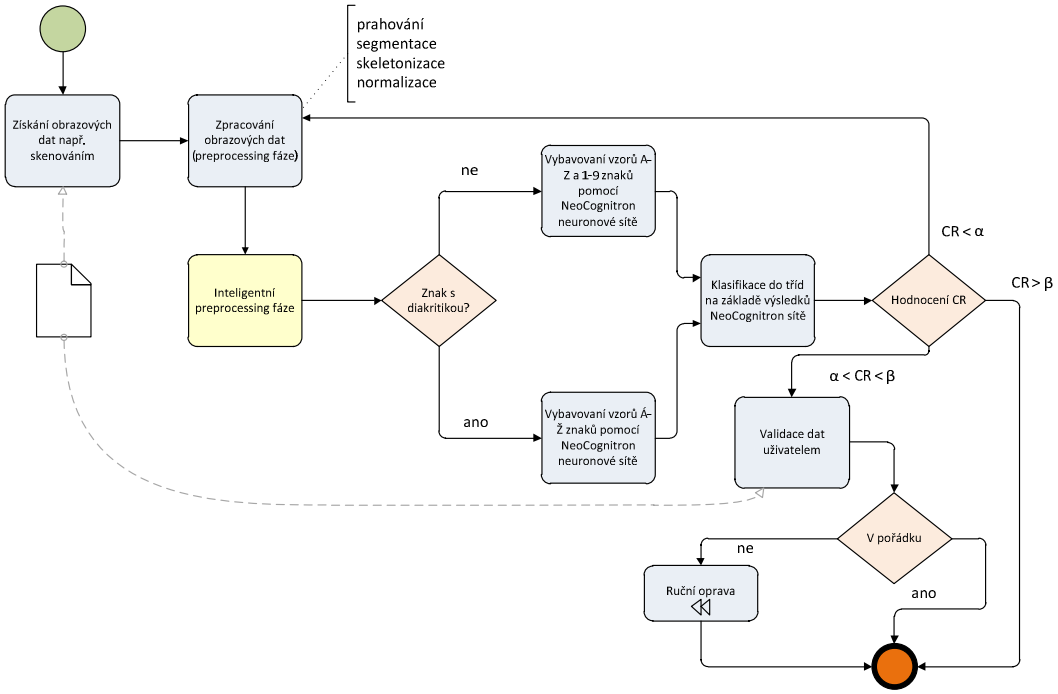
9.2 Inteligentní preprocessing fáze

Z experimentů provedených v rámci testování Neocognitron sítě vyplynulo, že na hodnotu CR má negativní vliv jak počet rozpoznávaných znaků tak i podobnost těchto znaků a jejich komplexnost. V těchto případech není neuronová síť schopná s dostatečnou úspěšností rozpoznat o který vzor (znak) se jedná. Tato

vlastnost je zejména patrná u znaků s diakritikou, které ze své podstaty jsou velmi podobné svým párům znaků bez diakritiky a také přispívají k vyššímu počtu znaků. Na základě této předložené teze byla navržena fáze inteligentního preprocessingu.

9.2.1 Princip

Preprocesing fáze je založená na rozpoznání problémových znaků, v našem případě českých znaků s diakritikou a jejich následné zpracování speciálně nastavenou neuronovou sítí s nastavenými parametry přímo za účelem rozpoznávání těchto znaků. Vzhledem k tomu, že tato speciální sít je naučená pouze na úzký okruh znaků s diakritikou vykazuje mnohem lepší CR, než kdyby byla použita síť s kompletním rozsahem znaků. Další pozitivem zařazení této preprocesing fáze je i menší počet znaků rozpoznáváných v neuronové síti, což vede také k lepšímu CR. V případě českých znaků s diakritikou je pro tuto specifickou dedikovanou neuronovou síť potřeba naučit pouhých 15 znaků. Zbývající znaky a číslice jsou poslány k rozpoznání do standardní Neocognitron neuronové sítě. Proces obohacený o tuto fázi předzpracování znaků je zobrazen na obr. 30.



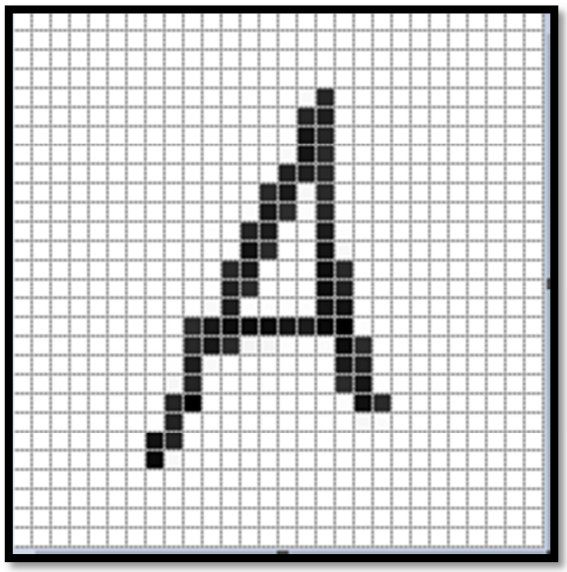
Obr. 30 Preprocessing fáze (vlastní návrh)

9.2.2 Funkce inteligentní preprocesing fáze

Fáze inteligentního preprocesingu spočívá v analýze rastrového obrázku, která zjišťuje, zda obrázek obsahuje či neobsahuje diakritiku a na základě této analýzy je analyzovaný obrázek znaku poslán do příslušné neuronové sítě. Funkce inteligentní preprocesing fáze je detailně popsán v následujících kapitolách.

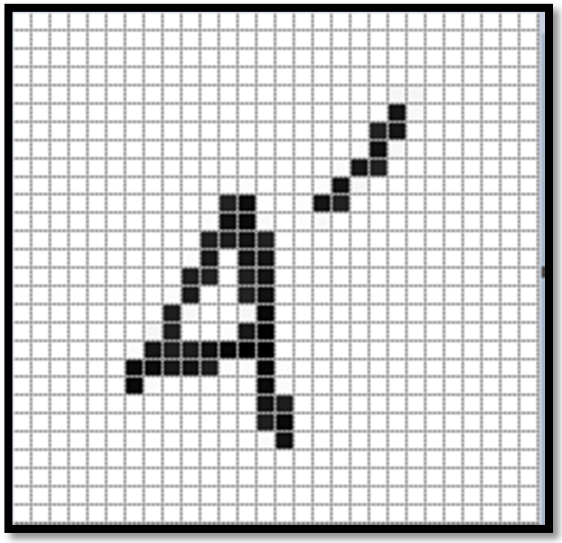
9.2.2.1 Analýza rastrového obrázku

Pro stanovení, zda znak zapsaný ve zkoumaném obrázku může obsahovat diakritiku, byla připravena metoda ParseObjects. Tato metoda v první řadě analyzuje předaný rastr za účelem nalezení samostatných skupin pixelů, které se navzájem dotýkají. Vychází se z předpokladu, že základní znaky vytváří jednu ucelenou skupinu. Viz Obr. 31, písmeno A.



Obr. 31 Písmeno bez diakritiky (vlastní zdroj)

Naopak znak s diakritikou bude tvořen více skupinami. Viz Obr. 32, písmeno Á, které obsahuje skupinu pixelů popisujících symbol diakritiky a skupinu samotného znaku A.



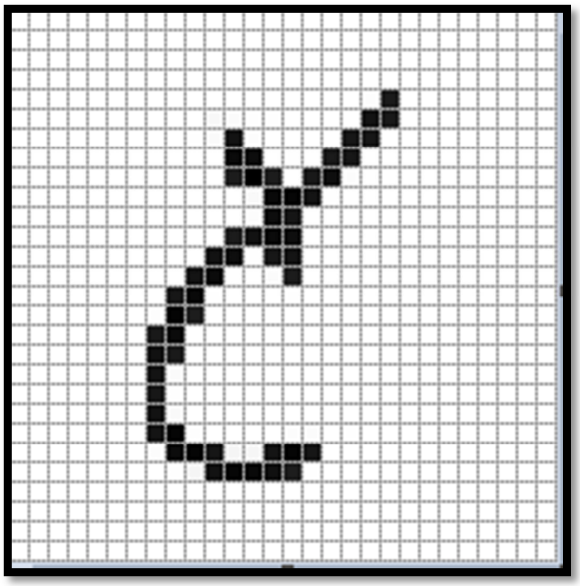
Obr. 32 Písmeno s diakritikou (vlastní zdroj)

Rozhodovací pravidlo, zda analyzovaný rastr obsahuje znak s diakritikou, nebo nikoli, vychází z níže demonstrovaných pravidel. Tato pravidla je vhodnější definovat negací.

Pravidlo 1:

Je-li v rastru pouze jedna skupina pixelů, nejedná se o znak s diakritikou

Toto pravidlo je v implementovaném řešení považováno za postačující pro vyloučení diakritiky. Znaky s diakritikou, které jsou zapsány jako jeden celek, nebudou správně označeny. Viz Obr. 33, písmeno Č, ve kterém symbol háčku splývá se znakem písmene C.



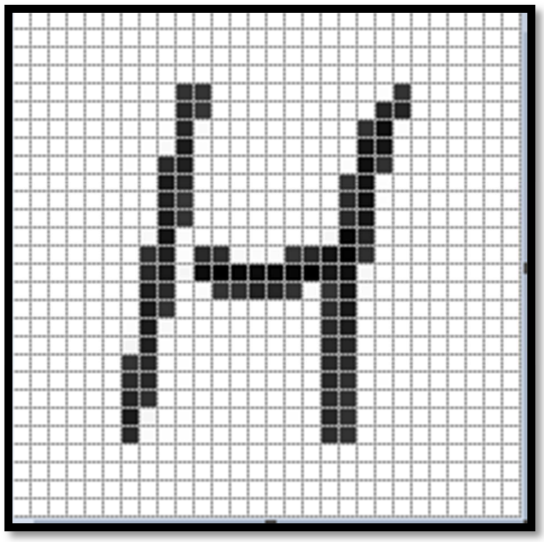
Obr. 33 Ukázka písmene se spojenou diakritikou (vlastní zdroj)

Pravidlo 2:

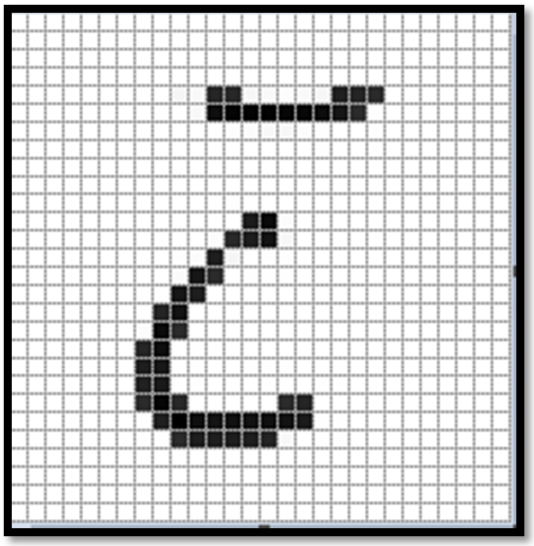
Je-li v rastru více skupin pixelů a skupina, která leží nejvýše, zasahuje do dolní poloviny rastru, nejedná se o znak s diakritikou.

Toto pravidlo omezuje rozhodovací logiku, dle které by více objektů automaticky vždy znamenalo existenci diakritiky, o případy, kdy je znak z důvodu nesprávného zápisu, nebo nekvalitního naskenování rozdělen. Viz obr. 34, písmeno H, které díky nespojenému zapsání vytvoří dvě skupiny.

Pokud pravidlo 1 nebo 2 není splněno, je znak považován za písmeno s diakritikou.



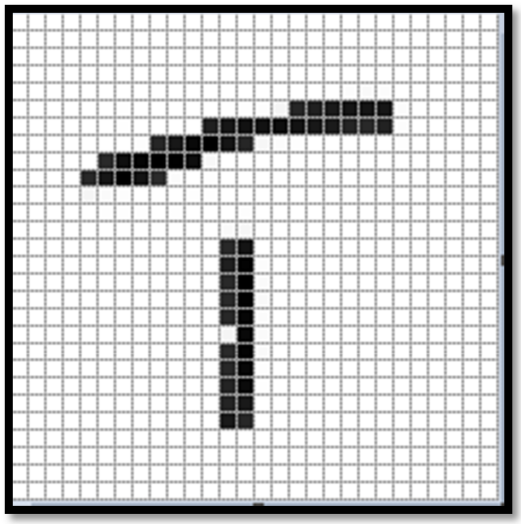
Obr. 34 Písmeno tvořené dvěma skupinami bez diakritiky (vlastní zdroj)



Obr. 35 Písmeno s diakritikou (vlastní zdroj)

Při analýze písmena Č z Obr. 35 jsou nalezeny dvě skupiny pixelů. Skupina ležící nejvýše je reprezentována pixely symbolu háček. Tato skupina rovněž nesplňuje podmínku 2 – nezasahuje do dolní poloviny rastru, znak je považován za písmeno s diakritikou.

Popisovaný algoritmus, mimo příkladu na obrázku 35, nepracuje správně také pro případy nevhodného zapsání některých znaků bez diakritiky, např.: T, F, E. viz Obr. 36, písmeno T, u kterého není spojena horní vodorovná část se zbytkem symbolu.

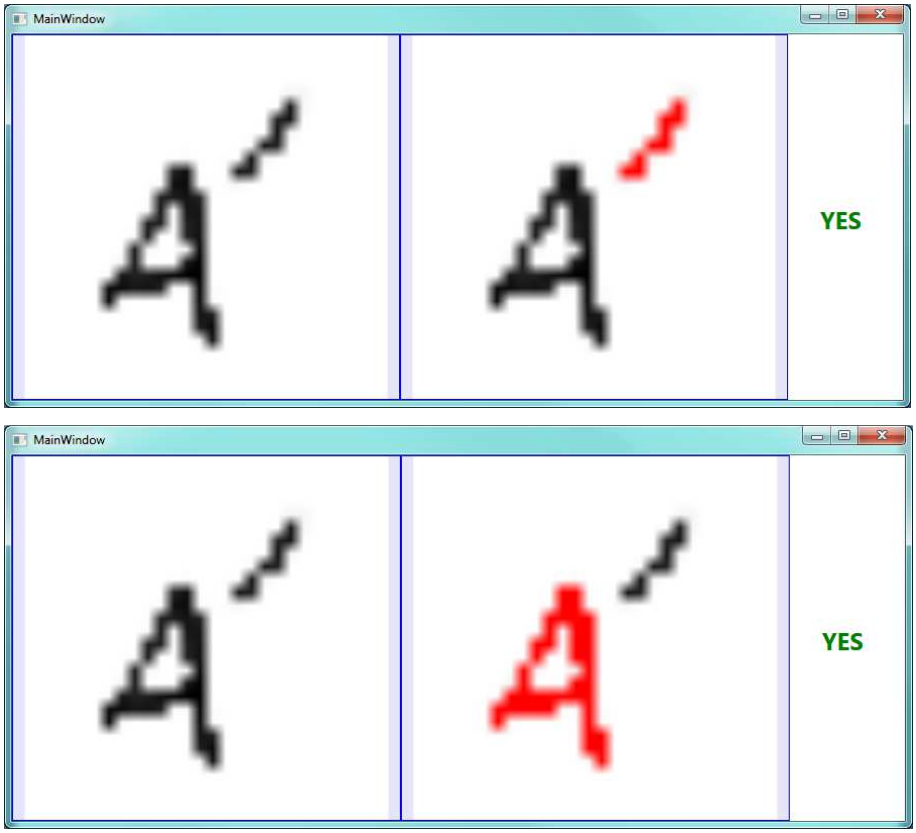


Obr. 36 Ukázka písmene T bez propojení (vlastní zdroj)

Při analýze písmene T z Obr. 36 jsou nalezeny dvě skupiny pixelů. Skupina ležící nejvýše však nezasahuje do dolní poloviny rastru, tedy podmínka 2 není splněna, a znak tedy nebude považován za písmeno s diakritikou.

9.2.2.2 Podrobnější popis algoritmu analýzy

Pro demonstraci algoritmu byla připravena jednoduchá aplikace, vizualizující výsledky analýzy. Ve vzorovém příkladu na obr. 37 jsou vyobrazeny všechny nalezené objekty pro písmeno „Á“, a zobrazen první objekt „čárka“ a druhý objekt písmeno „A“. V pravé části okna je vyznačen výsledek analýzy, v tomto případě zelené slovo „YES“ znamená, že vložené písmeno obsahuje diakritiku.



Obr. 37 Výstup z vytvořené testovací aplikace zobrazující nalezení diakritiky (vlastní zdroj)

Jako doplňková funkce byla v algoritmu analýzy implementována rovněž logika pro odstranění fragmentů, které nejsou součástí symbolu. Za tyto fragmenty jsou považovány malé skupiny pixelů, které vznikly například chybou skenování. Předpokladem však zůstává, že fragmenty budou odstraněny sofistikovanějším způsobem již v úvodních fázích zpracování obrázku. Do algoritmů analýzy diakritiky už budou poté přicházet obrázky bez fragmentů. Velikost skupiny, která je považována za fragment, byla experimentálně stanovena na 4 pixely. Jedná se o hodnotu stanovenou pro vstupní rastr o rozměru 28x28 pixelů. Hodnota 4 je minimální počet pixelů, které mohou tvořit symbol diakritiky nebo interpunkčního znaménka. Obecně je tuto hodnotu nutné stanovit v závislosti na rozlišení

vstupního rastru, jelikož ve velkém rozlišení i tečka za větou má např. 8px a naopak v malém rozlišení může být tečka popsána pouze dvěma pixely.

9.2.2.3 Vyhledávání objektů v předloženém znaku

V rámci testovací aplikace je vytvořena metoda ParseObjects, která ukazuje implementaci hlavní části algoritmu pro vyhledávání skupin pixelů. Vstupem funkce je objekt Bitmap, tedy jeden konkrétní rastrový obrázek obsahující písmeno a návratovou hodnotou je List<PixGroup>, který obsahuje všechny nalezené skupiny pixelů v rastru.

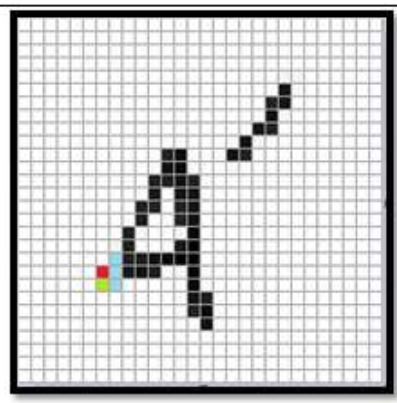
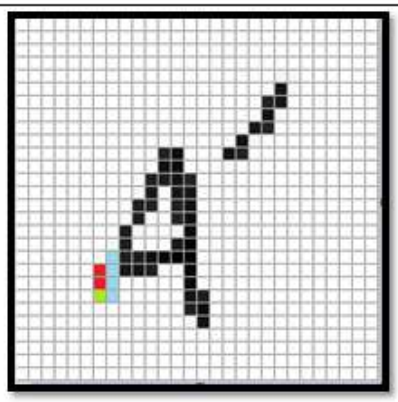
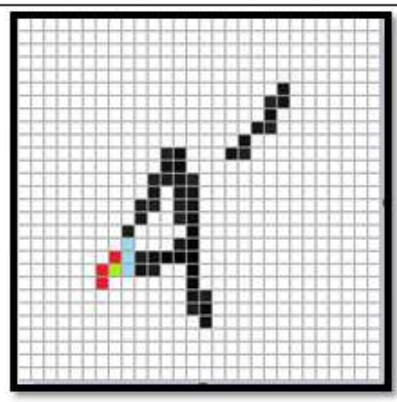
```
public static List<PixGroup> ParseObjects(Bitmap bmp)
{
	List<PixGroup> chars = new List<PixGroup>();

	List<int> added = new List<int>();
	byte whiteLevel = 150;
	System.Drawing.Color c;
	int i, n = bmp.Width;
	int j, m = bmp.Height;
	int k, s;
	for (i = 0; i < n; i++) //sloupce
	{
		for (k = 0; k < chars.Count; k++)
		  chars[k].NewColumn();

		for (j = 0; j < m; j++)
		{
			c = bmp.GetPixel(i, j);

			//kdyz hledam cernou tak  >
			if (whiteLevel > c.R)
			{
			   //pixel, ktery je povazovan za soucast pismena
```

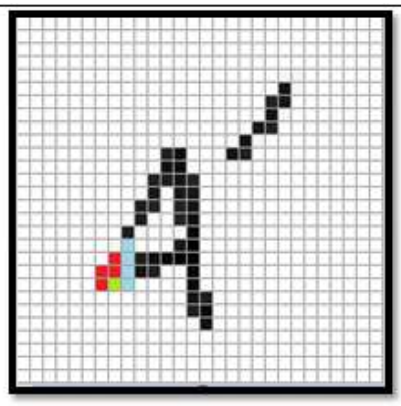
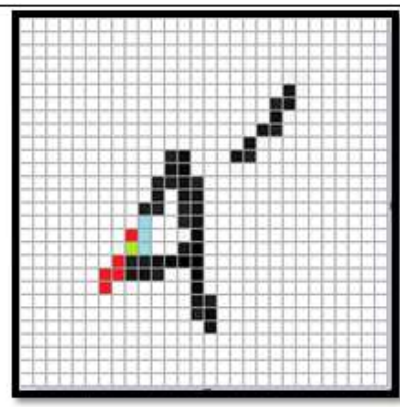
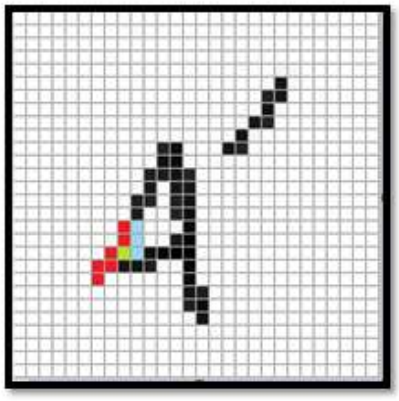
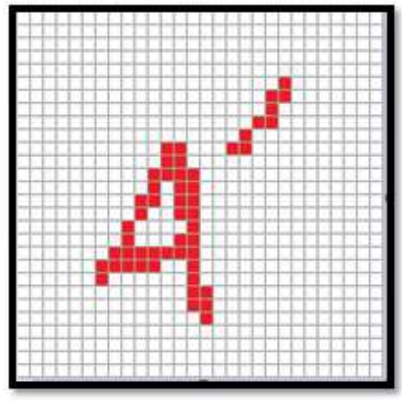
```
                    Pix px = new Pix(i, j, c.R);
                    s = chars.Count;
                    added.Clear();
                    for (k = 0; k < s; k++)
                    {
                       //projdu existujici pismena
                        if (chars[k].AddIfNeighbor(px))
                            added.Add(k);
                    }
                    if (added.Count == 0)
                    {
                       //pixel nepasoval do zadn.objektu, vytvorim novy
                        chars.Add(new PixGroup());
                        chars[s].Add(px);
                    }
                    if (added.Count > 1)
                    {
                        //pasoval do vice nez jednoho = sloucim objekty
                        for (k = added.Count - 1; k > 0; k--)
                        {
                            chars[added[0]].Fuse(chars[added[k]]);

                            //slouceny vyradim ze seznamu
                            chars.RemoveAt(added[k]);
                        }
                    }
                  }
                }
              }
              return chars;
}
```

9.2.2.4 Princip vyhledávání

V tab. 6 je znázorněno několik průchodů algoritmu při vyhledávání jednotlivých skupin pixelů. Algoritmus jehož ukázka je uvedena v kapitole prochází jednotlivé pixely obrázku po sloupcích. V případě, že nalezne černý pixel, je vytvořena skupina, do které je uložena pozice pixelu. Poté jsou k nalezenému černému pixelu vygenerováni sousedé. Do souseda Y je vygenerován soused pro aktuální sloupec, ve kterém se vyhledává (označen zeleně). Do dalšího souseda se vygenerují další možní sousedé (označeno modře), kde by se mohly nacházet pixely patřící do stejné skupiny. Vygenerované hodnoty jsou uloženy do aktuálních sousedů.

Metoda AddIfNeighbor pak slouží pro určení, jestli se pixel hodí do vytvořené skupiny pixelů. V případě, že je vyhledán černý pixel jehož hodnota Y je obsažena v aktuálních sousedech nebo se soused Y se rovná pixelu Y pak je nalezený pixel přidán do skupiny pixelů.

<table>
<tr>
<td> 
Vstupní obrázek</td>
<td> 
Krok 1: První černý pixel nalezen na pozici [6, 19] (červeně), do yNeighbor je uloženo y=20 (zeleně) a do nextNeighbor je uloženo y=18,19,20 pro analýzu dalšího sloupce (modře)</td>
</tr>
<tr>
<td> 
Krok 2: Druhý černý pixel nalezen na pozici [6, 20] (červeně), do yNeighbor je uloženo y=21 (zeleně) a do nextNeighbor je uloženo y=18,19,20,21 pro analýzu dalšího sloupce (modře)</td>
<td> 
Krok 3: Třetí černý pixel nalezen na pozici [7, 18] (červeně), do yNeighbor je uloženo y=19 (zeleně) a do nextNeighbor je uloženo y=17,18,19 pro analýzu dalšího sloupce (modře)</td>
</tr>
</table>

 Krok 4: Čtvrtý černý pixel nalezen na pozici [7, 19] (červeně), do yNeighbor je uloženo y=20 (zeleně) a do nextNeighbor je uloženo y=17,18,19,20 pro analýzu dalšího sloupce (modře)	 Krok 5: Pátý černý pixel nalezen na pozici [8, 16] (červeně), do yNeighbor je uloženo y=17 (zeleně) a do nextNeighbor je uloženo y=15,16,17 pro analýzu dalšího sloupce (modře)
 Krok 6: Šestý černý pixel nalezen na pozici [8, 17] (červeně), do yNeighbor je uloženo y=18 (zeleně) a do nextNeighbor je uloženo y=16,17,18 pro analýzu dalšího sloupce (modře)	 Konečný výsledek: po projití celého algoritmu byly nalezeny dva objekty

Tab 6 Znázornění několika průchodů vyhledávání skupin pixelů (vlastní zdroj)

9.2.2.5 Pravidla vytváření objektu

Souvisí s předchozím zdrojovým kódem pro vyhledávání.

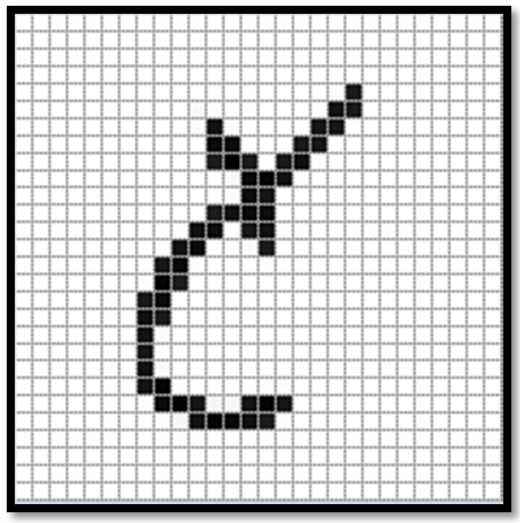
```
#region Vytvareni objektu
       internal bool AddIfNeighbor(Pix px)
        {
            if (yNeighbor == px.Y || curNeighbors.Contains(px.Y))
            {//pixel se hodi do daneho pismene
                Add(px);
                return true;
            }
            return false;
        }

        //sloucim dva PixGroup bez generovani sousedu
        internal void Assembly(PixGroup px)
        {
            this.AddRange(px);
            analyzed = false;
        }

        private void Fuse(PixGroup px)
        {
            this.RemoveAt(this.Count - 1);
//posledni prvek je stejny, aby se neopakoval tak je z jednoho listu vyrazen
            this.AddRange(px);

            //sloucim sousedy
            foreach (int i in px.nextNeighbors)
            {
                this.nextNeighbors.Add(i);
            }
            foreach (int i in px.curNeighbors)
            {
                this.curNeighbors.Add(i);
            }
```

```
        }

        //volano pred zacatkem skenovani noveho sloupce
        internal void NewColumn()
        {
            yNeighbor = -1;
            curNeighbors.Clear();
            foreach (int n in nextNeighbors)
            {
                curNeighbors.Add(n);
            }
            nextNeighbors.Clear();
        }

        internal new void Add(Pix px) //ok
        {
            base.Add(px);
            //vygeneruji mista co znamenaji sousedni body
            yNeighbor = px.Y + 1;
            nextNeighbors.Add(px.Y);
            nextNeighbors.Add(px.Y + 1);
            nextNeighbors.Add(px.Y - 1);
        }
        SortedSet<int> curNeighbors = new SortedSet<int>(); //zajisti
jedinecnost vyskytu v listu
        internal SortedSet<int> nextNeighbors = new SortedSet<int>();
        int yNeighbor = -1; //jako soused se predpoklada i predchozi vlozeny
#endregion Vytvareni objektu
```

9.2.2.6 Rozhodnutí zda písmeno obsahuje diakritiku:

Zdrojový kód ukazuje návrh rozhodovacích pravidel pro určení, zda-li písmeno obsahuje diakritiku či nikoliv. V předchozí části byly pomocí metody ParseObjects nalezeny skupiny pixelů v obrázku. V případě, že obrázek obsahuje pouze jeden objekt, tak systém vyhodnotí, že v daném obrázku není obsažena

diakritika. Existuje varianta, kdy je diakritika spojena s písmenem, jak je vidět na obrázku níže.



Obr. 38 Ukázka zobrazující písmeno Č se spojenou diakritikou (vlastní zdroj)

V takovém případě je dané písmeno pomocí takového pravidla označeno za písmeno bez diakritiky. V případě, že je nalezeno více skupin pixelů v obrázku, tak metoda ParseChars setřídí nalezené objekty obrázku od shora dolů, tedy první skupinou pixelů bude taková, jejíž hodnota na ose Y je nejmenší.

```
class DCharSeparator : CharSeparator
        {
            public static List<PixGroup> ParseChars(Bitmap img)
            {
                List<PixGroup> rv = CharSeparator.ParseChars(img, 1);
                rv.Sort((ar1, ar2) => ar1.Top - ar2.Top);
                return rv;
            }

            public static bool IsDiacritic(List<PixGroup> list, int height)
            {
                if (list.Count < 1)
```

```
                    return false;

if (list[0].Bottom < height / 2)
                    return true;

                return false;
            }
        }
```

9.2.3 Zhodnocení

Po implementaci prvku inteligentní preprocesingu se zlepšila hodnota schopnost rozpoznávat znaky celého systému o cca 15%. Hodnota neodpovídá přesně 21% nárustu, jak by se dalo předpokládat z měření, provedených v kapitole 7.3.1. Tento výsledek je způsoben především tím, že ne vždy je znak s diakritikou správně zařazen, a tím pádem poslán do správné neuronové sítě. Nicméně, i hodnota 15% nárustu úspěšnosti je velmi znatelným zlepšením funkce navrženého ICR systému.

Sada znaků	CR	Popis
A-Ž	50%	bez prvku inteligentního preprocessingu
A-Ž	65%	s prvkem inteligentního preprocessingu

Tab 7 Výsledky rozpoznávání optimalizovaného systému

9.3 Implementace navržených algoritmů do systému

Pro ověření teoretického návrhu optimalizovaného ICR systému byl jedním z cílů disertační práce vytvoření dílčích částí tohoto systému do testovacích aplikací, pomocí kterých bylo možné ověřit navržené algoritmy v praxi. Vzhledem k tomu, že implementace, algoritmů pracujících s neuronovými sítěmi, není v programovacích jazycích triviální a je velmi časově náročná, bylo nejdříve

přistoupeno k jejich implementaci ve skriptovacím prostředí Matlab a teprve po prověření funkčnosti byly přepsány do jazyka C#, který je vhodnější pro implementaci do kompletního ICR systému.

9.3.1 Matlab

Po provedení potřebné rešerše a analýze problému, popsané v teoretické části této disertační práce, bylo potřeba provést potřebná simulace a testování neuronových sítí, potřebných pro určení jejich vhodnosti pro následnou další práci. Pro tento účel bylo zvoleno skriptovací prostředí Matlab, protože nabízí snadnou implementaci algoritmů pro jejich otestování. Díky svým vlastnostem umožňuje jejich rychlou úpravu a ověření těchto úprav. Za účelem těchto testů byl tedy, na základě získaných zkušeností z provedené rešerše ve skriptovacím prostředí Matlab, vytvořen algoritmus pro učící a vybavovací fázi neuronové sítě Neocognitron a výsledky z testování těchto algoritmů byly použity v této práci. Vytvořený kód je rozdělen do tří funkčních celků:

- Předzpracování.
- Učící fáze sítě.
- Vybavovací fáze sítě.

9.3.1.1 Předzpracování

Prvním krokem předzpracování znaků pro učení sítí a jejich rozpoznávání je převedení naskenovaných obrázků do vhodných standardizovaných matic, které jsou vhodné pro další použití v neuronovových sítích. Základem je tedy převod obrázků jednotlivých znaků do požadovaných rozměrů, v našem případě 28x28 px jak je popsáno v kapitole 11.1.1. Tento proces je podrobně podrobně popsán v ukázce kódu v příloze B1. Jak z tohoto kódu vyplývá, prvním krokem je samotné načtení obrázků znaků z matic, vyřezaných znaků, získaných skenováním, a jejich uložením do výsledné matice po provedení úprav v metodě Uprava1. Metoda Uprava1 se skládá z následujících kroků:

- Vyčištění okrajů z každé strany.
- Odstranění samostatných pixelů.
- Prahování.
- Změna velikosti na požadovanou velikost.

Dalším krokem je vytvoření výsledných matic párů, vhodných pro použití v neuronových sítích. Tento krok je zobrazen v příloze B2 a jsou v něm provedeny následující kroky:

- Načtení znaků z předpřipravených matic.
- Vytvoření sady značek znaků.
- Převod znaku do negativu.
- Doplnění 2px okrajů znaku.
- Mapstd – normalizace vstupu, nastavením průměrných hodnot na 0 a odchylek na 1.
- Reshape.
- Uložení párů do výsledné matice.

9.3.1.2 Učící fáze sítě

V rámci učící fáze byly implementovány algoritmy na základě podkladů, získaných rešeršemi, provedených v kapitole 7.2. Prvním krokem bylo nastavení parametrů neuronové sítě pro učení a optimalizace těchto nastavení na základě učení neuronové sítě znaky s diakritikou. Optimalizace se odvíjela pouze v několika drobných změnách a ve většině případů se jednalo o úpravy v literatuře, popisované jako možné, nejednalo se tedy o velké zásahy do struktury neuronové sítě jako je počet vrstev a podobně. Ukázkový kód nastavení parametrů neuronové sítě je uveden v příloze B3. Po nastavení těchto parametrů je provedeno samotné učení neuronové sítě. Učící fáze je implementací algoritmů procesů nastíněných v kapitole 7.2.

9.3.1.3 Vybavovací fáze sítě

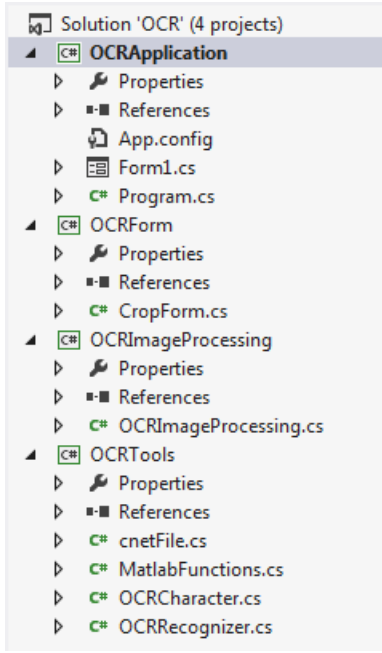
Zpracování vybavovací fáze ve skriptovacím prostředí matlab je podrobně zobrazeno v příloze B4, a je zpracováno také na základě poznatků získaných analýzou a rešeršemi v teoretické části. V první části procesu rozpoznávání je načtena správně naučená neuronová síť pro danou sadu znaků a této síti je předložen znak, který musí projít totožným předzpracováním, jako v případě učení sítě. Musí mít tedy stejné parametry jako znaky, kterými byla síť učena.

9.3.2 Systém v jazyce C#

Skriptovací prostředí Matlab je ideální pro testování neuronových sítí, návrh prototypů pracujících s neuronovými sítěmi, ověření jejich funkčnosti a prezentaci výsledků těchto testů. Nicméně pro účely navrhovaného systému a pro splnění požadavků popsaných v kapitole 9 není použití použití Matlabu vhodné. Z tohoto důvodu byl výsledný systém navržen v programovacím prostředí Visual Studio s využitím programovacího jazyka C#. Přepsáním, již navržených algoritmů v prostředí Matlab do jazyka C#, byla vytvořena knihovna OCRTools, kterou je možné referencovat v jakémkoliv projektu a využívat. Pro snadnější testování implementovaných algoritmů v jazyce C# a prezentaci jejich výsledků byly vytvořeny dvě testovací aplikace TestLetters, FormRecognizer, které jsou podrobněji popsány v kapitole 10.

9.3.2.1 Struktura systému

Struktura realizovaného projektu je rozdělena do 4 částí (projektů), za účelem zajištění možnosti rozdělení na jednotlivé funkční celky, dle návrhu provedeného v kapitole 9.1. Jednotlivé projekty ze struktury řešení, viz obr. 39, jsou podrobněji popsány v následujících kapitolách.



Obr. 39 Struktura systému ve Visual Studiu 2012 (vlastní zdroj)

9.3.2.2 OCRApplication

První částí systému je samotný prototyp funkční aplikace, který slouží jako vzor výsledné implementace řešení do cílové aplikace. V případě nasazení řešení v praxi, nemusí být aplikace vytvořená přímo pro tento účel, ale může se jednat o již existující aplikaci, například komplexní aplikaci pro správu dokumentů, apod. Jedinou podmínkou je správně nareferencovat potřebné knihovny (další části ICR systému), a poté volat příslušné potřebné funkce. Způsob volání knihoven je podrobně zobrazen v přílohách C1, C2 a C3.

9.3.2.3 Vytvořená OCRTools knihovna

Jádrem navrženého systému je vytvořená knihovna OCRTools, která obsahuje veškeré potřebné metody pro rozpoznávání znaků. Tato knihovna vznikla přepsáním algoritmů, vytvořených a otestovaných v rámci skriptovacího prostředí

Matlab, do jazyka C#. Porovnání kódu implementace v Matlabu a C# je uvedeno na ukázce části převedeného kódu v příloze C4. Kompletní kód samotné knihovny OCRTools čítá tisíce řádků kódu a proto jsou v přílohách disertační práce uvedeny pouze jednotlivé ukázky. Výsledný projekt OCRTools obsahuje následující 4 části:

- cnetFile – obsahuje metody pro práci s naučenou neuronovou sítí. Slouží k načtení naučené neuronové sítě z prostředí Matlab (učící fáze není, vzhledem k náročnosti pro převedení a jednorázovému použití této fáze, převedeno do jazyka C# a naučená neuronová síť je použita přes vytvořené algoritmy v cnetFile).
- MatlabFunctions – implementace integrovaných funkcí z Matlabu (všechny musely být ověřeny, zda vrací stejné výsledky jako funkce původní)
- OCRCharacter – finální preprocessing znaku pro zpracování v rozpoznávání.
- OCRRecognizer – implementace jádra rozpoznávání, se vstupem typu bitmapa 32x32px a návratem ve formě seznamu nejpravděpodobnějších výsledků.

9.3.2.4 OCRImageProcessing knihovna

Knihovna obsahující metody preprocessingu, převedené z Matlab. Obsahuje metody pro načtení obrázku ze souboru v PDF a extrakci obrázků z tohoto PDF. Grafické operace s obrázkem, jako jsou změna velikosti, převod do 256 stupňů šedi, ořezání obrázku.

9.3.2.5 OCRForm knihovna

Knihovna obsahující metody pro vytěžování formulářů. Vstupem je formulář ve formátu PDF (Portable Document Format), výstupem je pole bitových map jednotlivých polí. Získání obrazu z PDF, jeho ořezání a další zpracování je

podrobněji popsané v kapitole 10.2. Výsledná vytěžená pole lze následně použít pro rozpoznávání pomocí OCRTools knihovny.

10 OVĚŘENÍ SYSTÉMU NA TESTOVACÍCH VZORCÍCH

Z důvodu ověření navržených algoritmů byly vytvořeny dvě testovací aplikace implementující navržené algoritmy v knihovně OCRTools.

10.1 Aplikace TestLetters

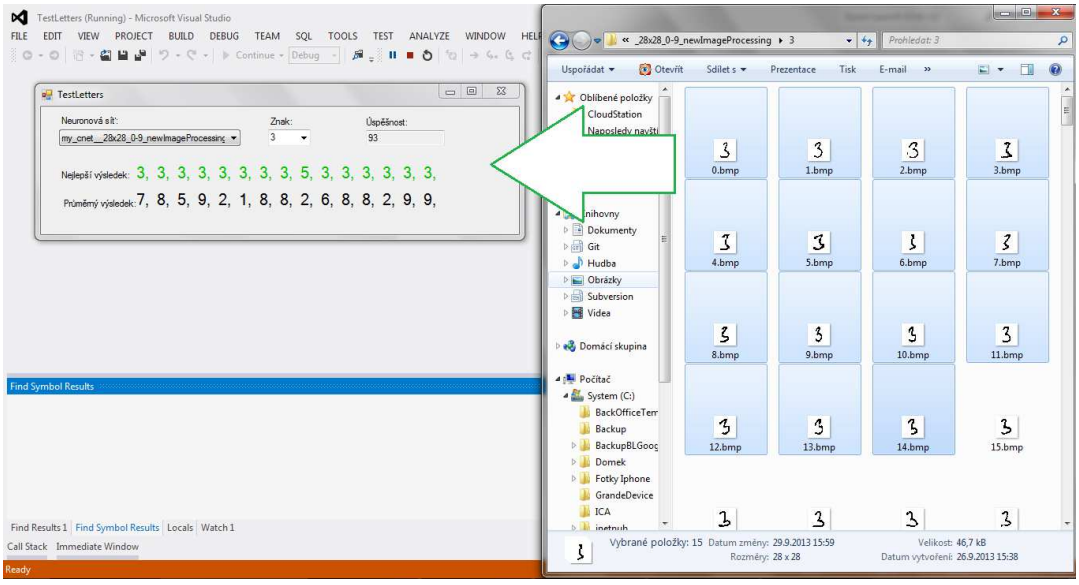
Aplikace TestLetters využívá algoritmu implementovaných v knihovně OCRTools a slouží ke snadnému testování úspěšnosti naučené sítě na předložených znacích. Díky této aplikaci bylo snadné testovat závislost úspěšnosti rozpoznávání sítě na předloženém znaku. Velmi rychle bylo možné otestovat odolnost sítě oproti uměle zaneseným chybám do obrazu znaku, případně dohledávat důvod, proč některý znak je špatně rozpoznán.

V adresářové struktuře aplikace je definována složka pro předem naučené neuronové sítě ve formátu *.MAT (Matlab File Format). Při běhu aplikace je poté možné vybrat jednu z těchto sítí pomocí výběrovníku „Neuronová síť“ a vybrat znak, který očekáváme na výsledku. Dalším krokem je zvolení znaků, které se mají pomocí aplikace rozpoznat, pomocí přetažení těchto znaků do aplikace. Tyto rozpoznávané obrazky znaků musejí mít rozměr čtverce o straně 28 px a musejí být předzpracované např. dle kapitoly 9.3.1.1.

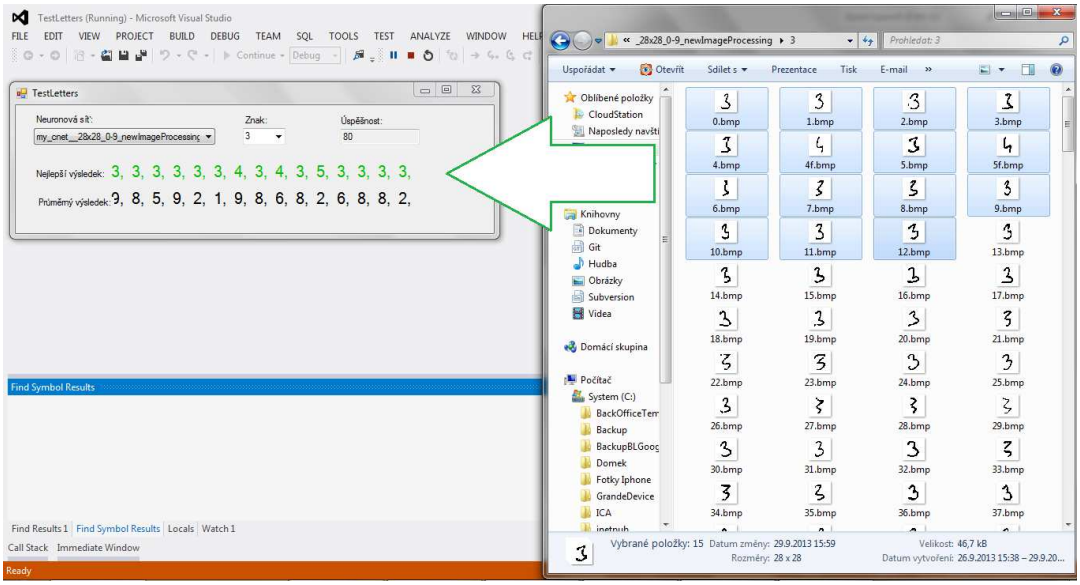
Výsledkem rozpoznávání aplikace je zobrazení rozpoznaného znaku pro každý předložený znak a druhého nejlépe ohodnoceného znaku. Dále je vypočítána průměrná úspěšnost rozpoznávání všech znaků. Pro správný výpočet je potřeba správně nastavit znak, který očekáváme. Výše popsané, vlastnosti jsou zobrazeny na ukázce aplikace na obr. 40. Z výsledku zobrazeného na ukázce lze vyčíst, že předložené znaky číslice 3 byly rozpoznány s 93% úspěšností.

Na obr. 41 je zobrazen výsledek aplikace reagující na předložené znaky, v tomto případě však má úmyslně podvrženy dvě číslice chybně. Z výsledku je patrný pokles úspěšnosti na průměr 80%. Vzory číslice „3“ jsou opět správně

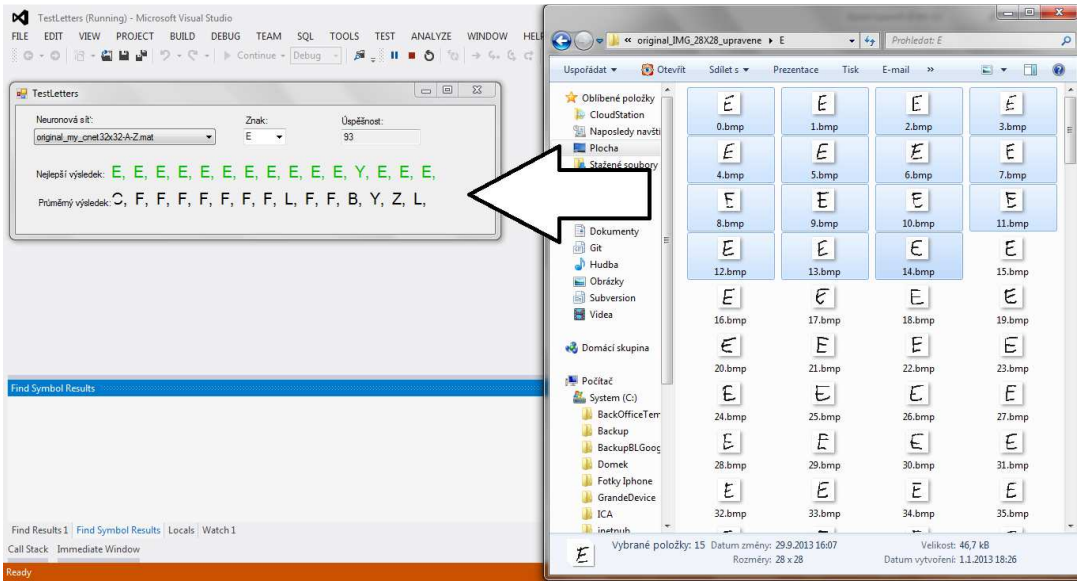
rozpoznány jako číslice 3, pouze v jednom případě totožném s rozpoznáváním na obr. 40 je zachycena chyba. Podvržené číslice „4“ jsou započítány do chybných, protože jsou rozpoznány správně jako číslice „4“.



Obr. 40 Ukázka testování znaku “3“ pomocí vytvořené TestLetters aplikace (vlastní zdroj)



Obr. 41 Ukázka testování znaku "3" pomocí vytvořené TestLetters aplikace s chybami (vlastní zdroj)

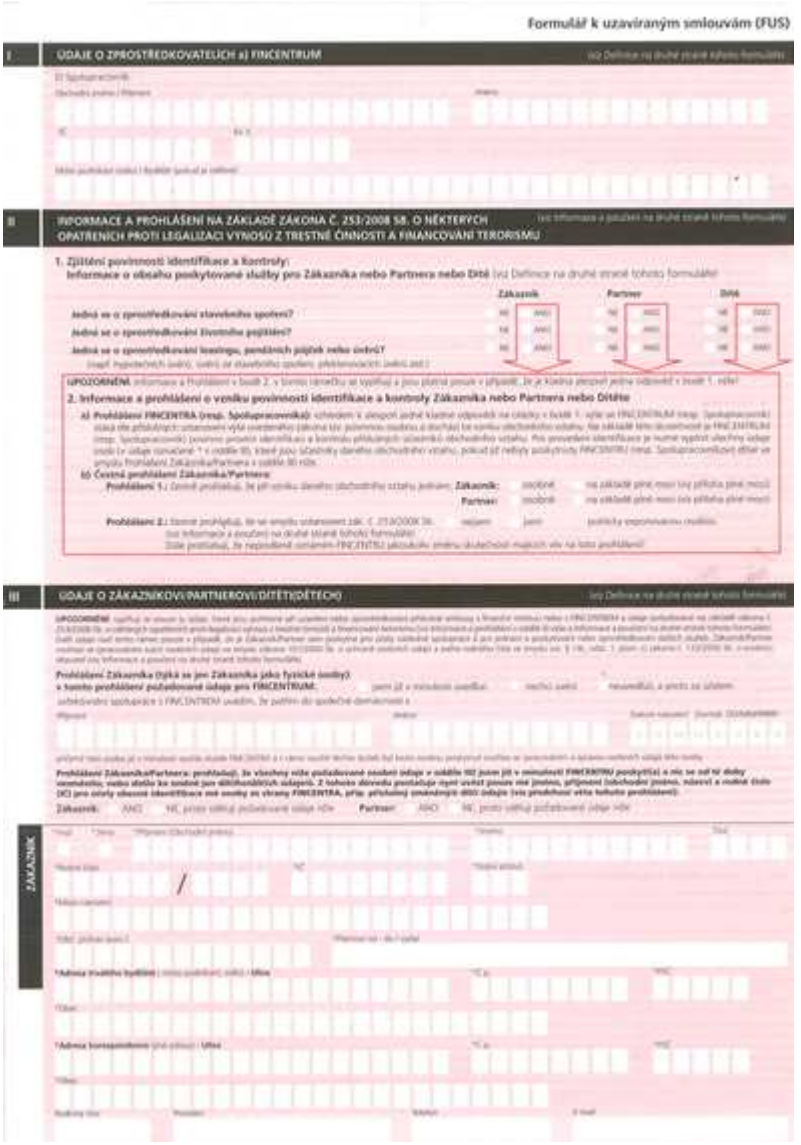


Obr. 42 Ukázka testování znaku "E" pomocí vytvořené TestLetters aplikace (vlastní zdroj)

10.2 Aplikace FormRecognizer

Dalším krokem, pro ověření navržených algoritmů v reálném provozu, bylo vytvoření testovací aplikace, sloužící k rozpoznávání formulářů, používaných v praxi. Vytvořená aplikace je schopna pomocí nastavených parametrů přes XML, nalézt veškerá zadávací pole a tato vytěžit do datových struktur použitelných pro následné rozpoznání pomocí jádra ICR.

Navržené algoritmy byly testovány na standardizovaném formuláři k uzavíraným smlouvám (FUS) používaném finančními institucemi jako doprovodný dokument ke smlouvám uzavíraným s klientem. Ukázka originálního formuláře FUS je zobrazena na obr. 43.

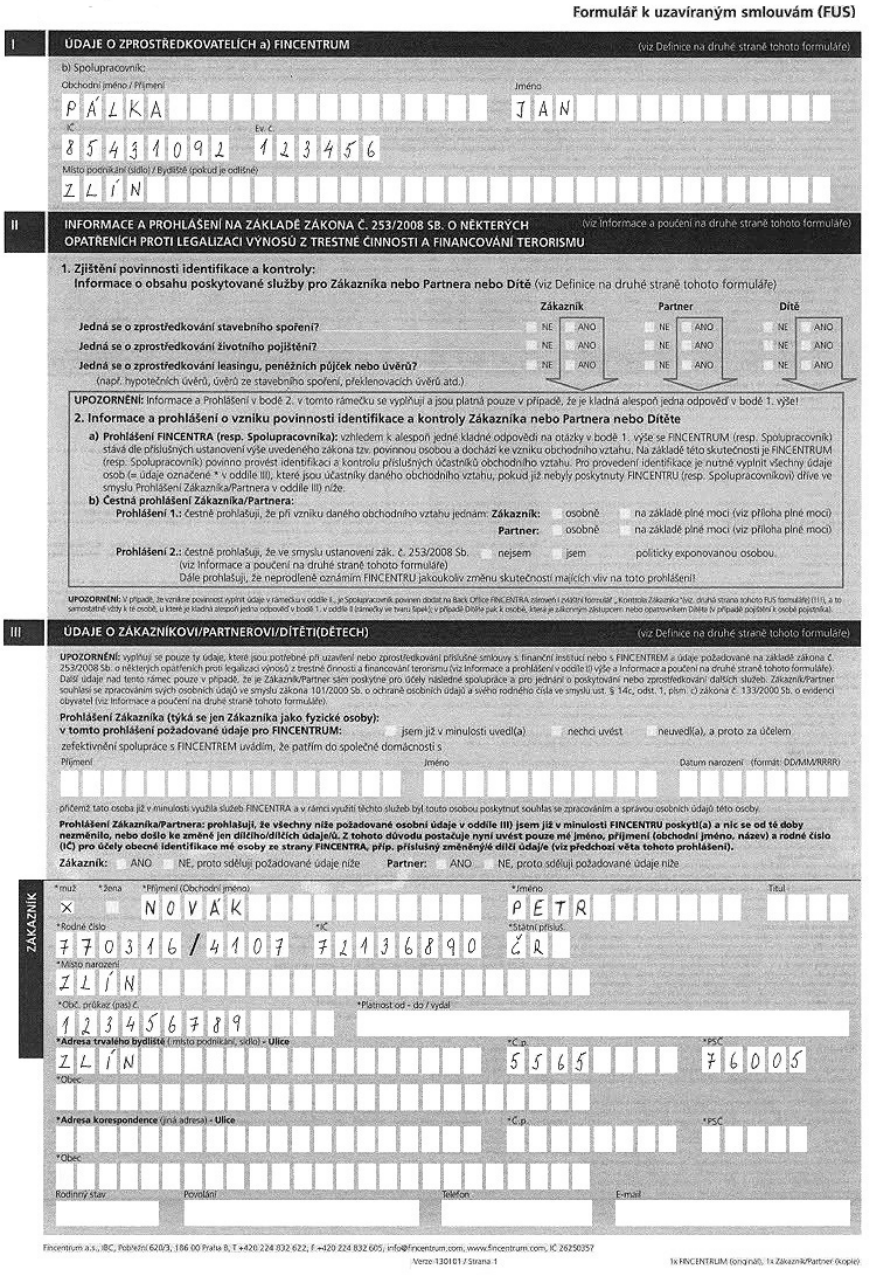


Obr. 43 Ukázka formuláře k uzavíraným smlouvám FUS.(vlastní zdroj)

10.2.1 Funkcionalita aplikace pro vyhledání zadávacích polí

Vstupem do aplikace je naskenovaný formulář, viz obr. 44. Naskenovaný formulář je vstupem do preprocessingové části, která slouží pro úpravu a zlepšení

kvality formuláře pro snadnější vyhledávání jednotlivých rámečků. V tomto formuláři.



Formulář k uzavíraným smlouvám (FUS)

I ÚDAJE O ZPROSTŘEDKOVATELÍCH a) FINCENTRUM (viz Definice na druhé straně tohoto formuláře)

b) Spolupracovník:

Obchodní jméno / Příjmení: PÁLKA — Jméno: JAN

IČ: 85431092 — Ev. č.: 123456

Místo podnikání (sídlo) / Bydliště (pokud je odlišné): ZLÍN

II INFORMACE A PROHLÁŠENÍ NA ZÁKLADĚ ZÁKONA Č. 253/2008 SB. O NĚKTERÝCH OPATŘENÍCH PROTI LEGALIZACI VÝNOSŮ Z TRESTNÉ ČINNOSTI A FINANCOVÁNÍ TERORISMU (viz Informace a poučení na druhé straně tohoto formuláře)

1. Zjištění povinnosti identifikace a kontroly:
Informace o obsahu poskytované služby pro Zákazníka nebo Partnera nebo Dítě (viz Definice na druhé straně tohoto formuláře)

	Zákazník		Partner		Dítě	
Jedná se o zprostředkování stavebního spoření?	NE	ANO	NE	ANO	NE	ANO
Jedná se o zprostředkování životního pojištění?	NE	ANO	NE	ANO	NE	ANO
Jedná se o zprostředkování leasingu, peněžních půjček nebo úvěrů? (např. hypotečních úvěrů, úvěrů ze stavebního spoření, překlenovacích úvěrů atd.)	NE	ANO	NE	ANO	NE	ANO

UPOZORNĚNÍ: Informace a Prohlášení v bodě 2. v tomto rámečku se vyplňují a jsou platná pouze v případě, že je kladná alespoň jedna odpověď v bodě 1. výše!

2. Informace a prohlášení o vzniku povinnosti identifikace a kontroly Zákazníka nebo Partnera nebo Dítěte

a) Prohlášení FINCENTRA (resp. Spolupracovníka): vzhledem k alespoň jedné kladné odpovědi na otázky v bodě 1. výše se FINCENTRUM (resp. Spolupracovník) stává dle příslušných ustanovení výše uvedeného zákona tzv. povinnou osobou a dochází ke vzniku obchodního vztahu. Na základě této skutečnosti je FINCENTRUM (resp. Spolupracovník) povinno provést identifikaci a kontrolu příslušných účastníků obchodního vztahu. Pro provedení identifikace je nutné vyplnit všechny údaje osob (= údaje označené * v oddíle III), které jsou účastníky daného obchodního vztahu, pokud již nebyly poskytnuty FINCENTRU (resp. Spolupracovníkovi) dříve ve smyslu Prohlášení Zákazníka/Partnera v oddíle III) níže.

b) Čestná prohlášení Zákazníka/Partnera:

Prohlášení 1.: čestně prohlašuji, že při vzniku daného obchodního vztahu jednám: Zákazník: osobně / na základě plné moci (viz příloha plné moci)
Partner: osobně / na základě plné moci (viz příloha plné moci)

Prohlášení 2.: čestně prohlašuji, že ve smyslu ustanovení zák. č. 253/2008 Sb. (viz Informace a poučení na druhé straně tohoto formuláře) nejsem / jsem politicky exponovanou osobou.
Dále prohlašuji, že neprodleně oznámím FINCENTRU jakoukoliv změnu skutečností majících vliv na toto prohlášení!

UPOZORNĚNÍ: V případě, že vzniklou povinnost vyplnit údaje v rámečku v oddíle II. je Spolupracovník povinen dodat na Back Office FINCENTRA zřízením zvláštní formuláře „Kontrola Zákazníka" (viz. druhá strana tohoto FUS formuláře) (III.), a to samostatně vždy k té osobě, u které je kladná alespoň jedna odpověď v bodě 1. v oddíle II tamtéž ve tvaru Spolu, v případě Dítěte pak k osobě, která je zákonným zástupcem nebo opatrovníkem Dítěte (v případě pojištění k osobě pojistníka).

III ÚDAJE O ZÁKAZNÍKOVI/PARTNEROVI/DÍTĚTI(DĚTECH) (viz Definice na druhé straně tohoto formuláře)

UPOZORNĚNÍ: vyplňují se pouze ty údaje, které jsou potřebné při uzavření nebo zprostředkování příslušné smlouvy s finanční institucí nebo s FINCENTREM a údaje požadované na základě zákona č. 253/2008 Sb. o některých opatřeních proti legalizaci výnosů z trestné činnosti a financování terorismu (viz Informace a prohlášení v oddíle II) výše a Informace a poučení na druhé straně tohoto formuláře). Další údaje nad tento rámec pouze v případě, že je Zákazník/Partner sám poskytne pro účely následné spolupráce a pro jednání o poskytování nebo zprostředkování dalších služeb. Zákazník/Partner souhlasí se zpracováním svých osobních údajů ve smyslu zákona 101/2000 Sb. o ochraně osobních údajů a svého rodného čísla ve smyslu ust. § 14c, odst. 1, písm. c) zákona č. 133/2000 Sb. o evidenci obyvatel (viz Informace a poučení na druhé straně tohoto formuláře).

Prohlášení Zákazníka (týká se jen Zákazníka jako fyzické osoby):
v tomto prohlášení požadované údaje pro FINCENTRUM: jsem již v minulosti uvedl(a) / nechci uvést / neuvedl(a), a proto za účelem zefektivnění spolupráce s FINCENTREM uvádím, že patřím do společné domácnosti s

Příjmení — Jméno — Datum narození (formát: DD/MM/RRRR)

přičemž tato osoba již v minulosti využila služeb FINCENTRA a v rámci využití těchto služeb byl touto osobou poskytnut souhlas se zpracováním a správou osobních údajů této osoby.

Prohlášení Zákazníka/Partnera: prohlašuji, že všechny níže požadované osobní údaje v oddíle III) jsem již v minulosti FINCENTRU poskytl(a) a nic se od té doby nezměnilo, nebo došlo ke změně jen dílčího/dílčích údaje/ů. Z tohoto důvodu postačuje nyní uvést pouze mé jméno, příjmení (obchodní jméno, název) a rodné číslo (IČ) pro účely obecné identifikace mé osoby ze strany FINCENTRA, příp. příslušný změněnýé dílčí údaje (viz předchozí věta tohoto prohlášení).

Zákazník: ANO / NE, proto sdělují požadované údaje níže — Partner: ANO / NE, proto sdělují požadované údaje níže

ZÁKAZNÍK

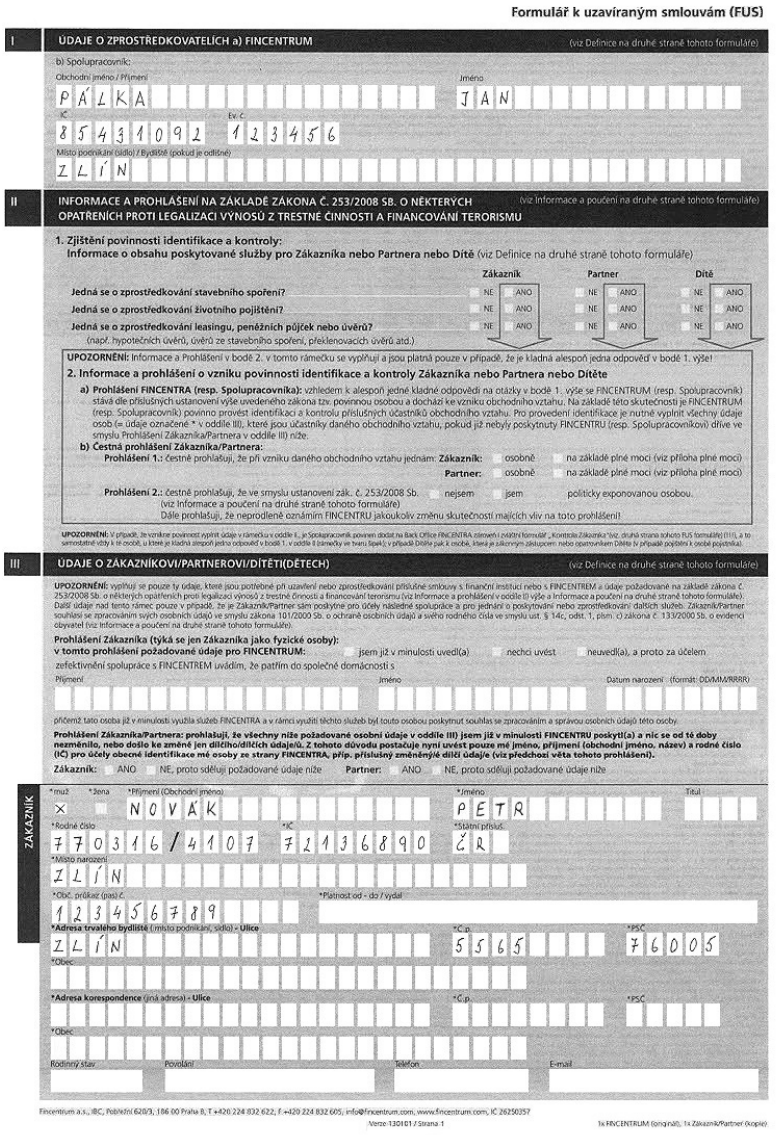
*muž: X	*žena:	*Příjmení (Obchodní jméno): NOVÁK	*Jméno: PETR	Titul:
*Rodné číslo: 770316/4107		*IČ: 72136890	*Státní příslušnost: ČR	
*Místo narození: ZLÍN				
*Obč. průkaz (pas) č.: 123456789		*Platnost od - do / vydal:		
*Adresa trvalého bydliště (místo podnikání, sídlo) - Ulice: ZLÍN		*Č.p.: 5565	*PSČ: 76005	
*Obec:				
*Adresa korespondence (jiná adresa) - Ulice:		*Č.p.	*PSČ	
*Obec:				
Rodinný stav	Povolání	Telefon	E-mail	

Fincentrum a.s., IBC, Pobřežní 620/3, 186 00 Praha 8, T +420 224 832 622, f. +420 224 832 605, info@fincentrum.com, www.fincentrum.com, IČ 26250357

Verze 130101 / Strana 1 — 1x FINCENTRUM (originál), 1x Zákazník/Partner (kopie)

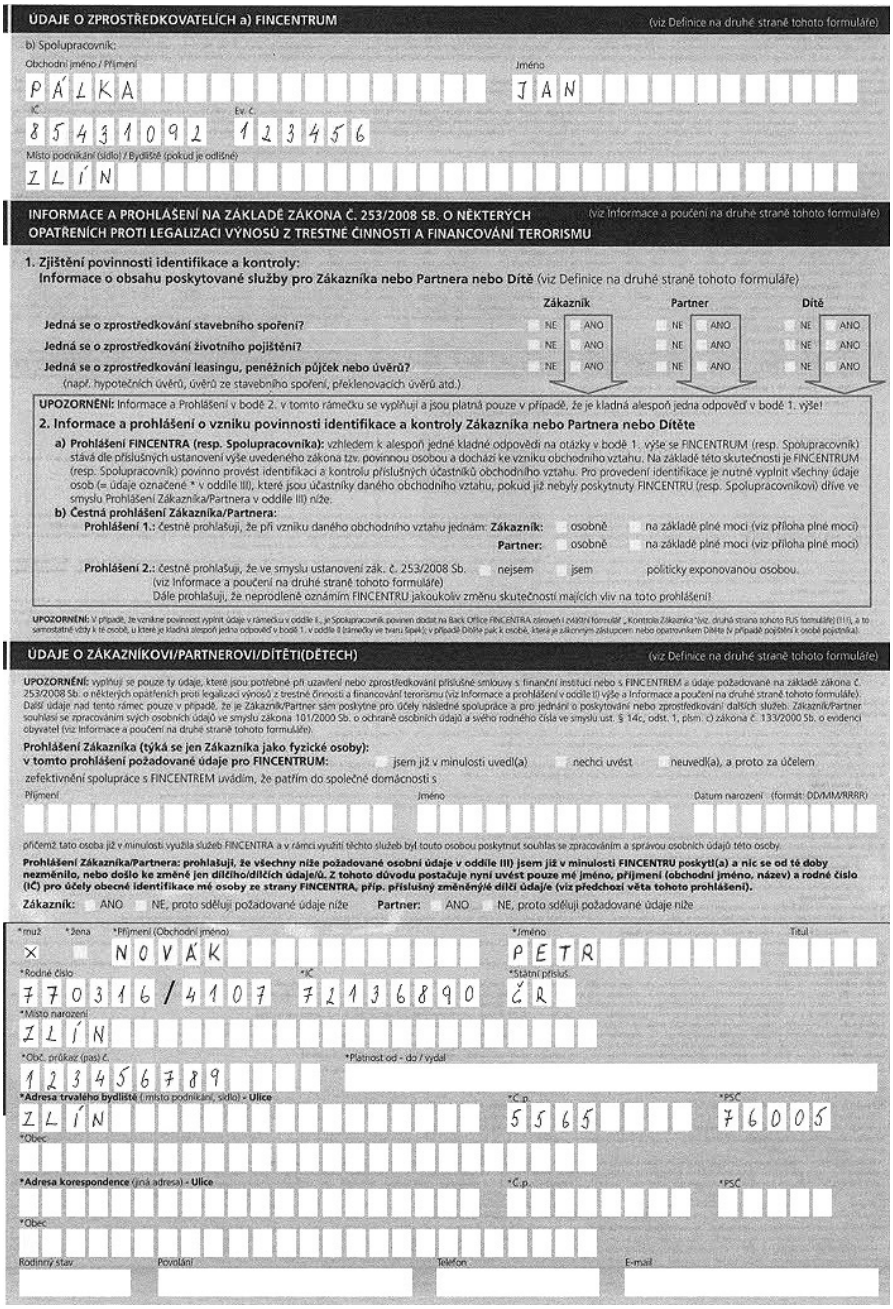
Obr. 44 Naskenovaný formulář na vstupu aplikace (vlastní zdroj)

Naskenovaný formulář je nejprve upraven do takové podoby, aby vyhledávání bílých rámečků, obsahující písmena, bylo co možná nejpřesnější. Úprava formuláře je založena na převedení tohoto formuláře do odstínů šedi a navíc byly z formuláře pomocí vytvořené metody CropBlackEdge(), ořezány tmavé okraje způsobené skenováním, kdy se v okolí formuláře nachází část oskenované plochy skeneru, viz obr. 45.



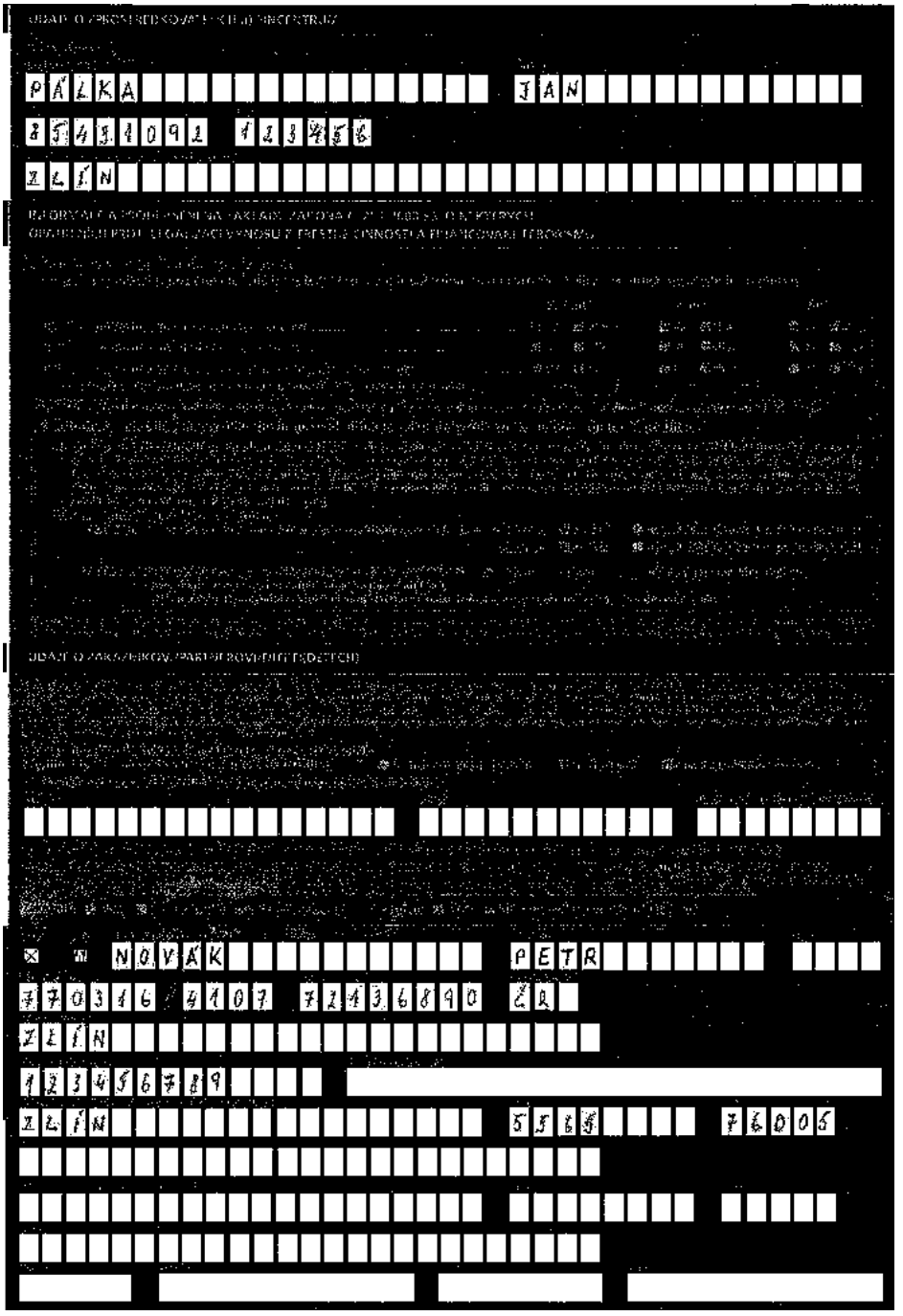
Obr. 45 Naskenovaný formulář po derotaci a ořezání okrajů (vlastní zdroj)

Po úpravě metodou CropBlackEdge() program ještě kontroluje, zda není oskenovaný formulář otočen o nějaký úhel, což by mohlo mít negativní vliv na vyhledávání jednotlivých bílých oblastí, tedy řádků a sloupců. V případě nutnosti je před rotací potřeba formulář oříznout, aby bylo možné vypočítat úhel pomocí specifického prvku formuláře např. černé dělící proužky formuláře. Pro každý formulář je tak vypočten úhel a v případě, že je formulář otočen, tak pomocí metody RotateImage() je daný formulář o vypočtený úhel rotován zpět. Po výše uvedených úpravách je spuštěna vytvořená metoda CenterCrop(), která slouží pro ořezání okrajů formuláře od středu, kde jsou hledány bílé pixely, výsledný formulář připravený pro samotné vytěžení znaků je zobrazen na obr. 46.



Obr. 46 Naskenovaný formulář použití funkce CenterCrop (vlastní zdroj)

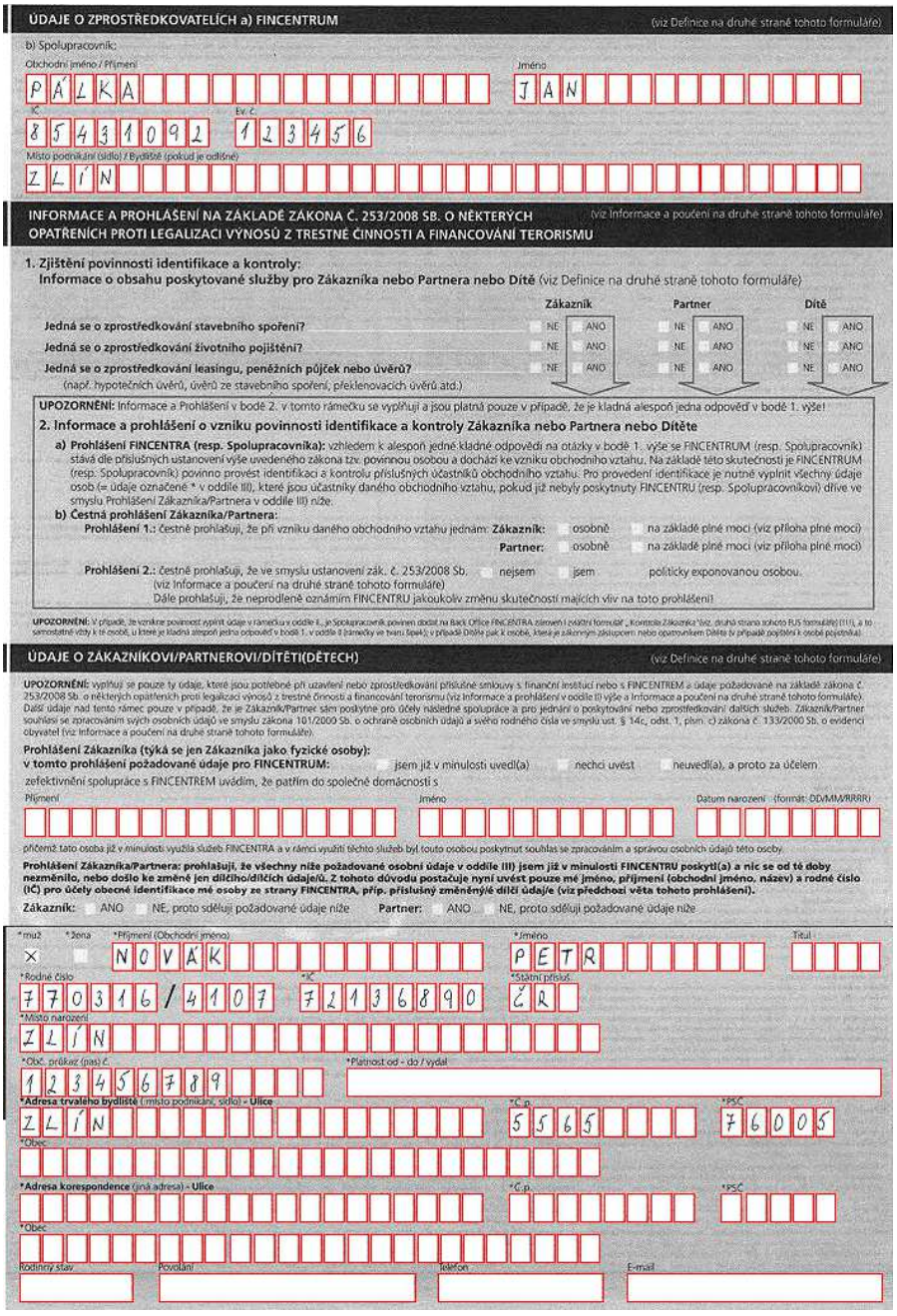
Formulář je poté pomocí metody BWThreshold ztmaven tak, aby tmavší odstíny šedé byly černé (0) a světlejší byly bílé (1). Tím je dosaženo, že jednotlivé zadávací pole jsou bílé a zvýrazněné od svého okolí viz Obr.47.



Obr. 47 Aplikace funkce BWThreshold (vlastní zdroj)

V oblastech formulářů, kde jsou očekávány zadávací pole, jsou hledány bílé pixely nejprve na ose Y, které definují počet nalezených řádků ve formuláři a poté na ose X, které definují počet sloupečků, můžeme také říci počet jednotlivých rámečků. Takto nalezené oblasti bílých pixelů jsou třízeny, neboť algoritmus vyhledávání najde všechny oblasti s bílými pixely, tedy i takové, co nepatří mezi kolonky pro vyplňování. V takovém případě je nutné zahazovat tyto nadbytečné oblasti bílých pixelů, jejichž počet je natolik malý, že nemůže popisovat jednotlivé kolonky. O nalezení a vyřezání jednotlivých rámečků se stará metoda SearchFrame(), jejíž výstup je znázorněn na obr. 48. Pro zvýraznění nalezených polí je použita metoda RedSurroundFrame(), pomocí které jsou nalezená pole orámována.

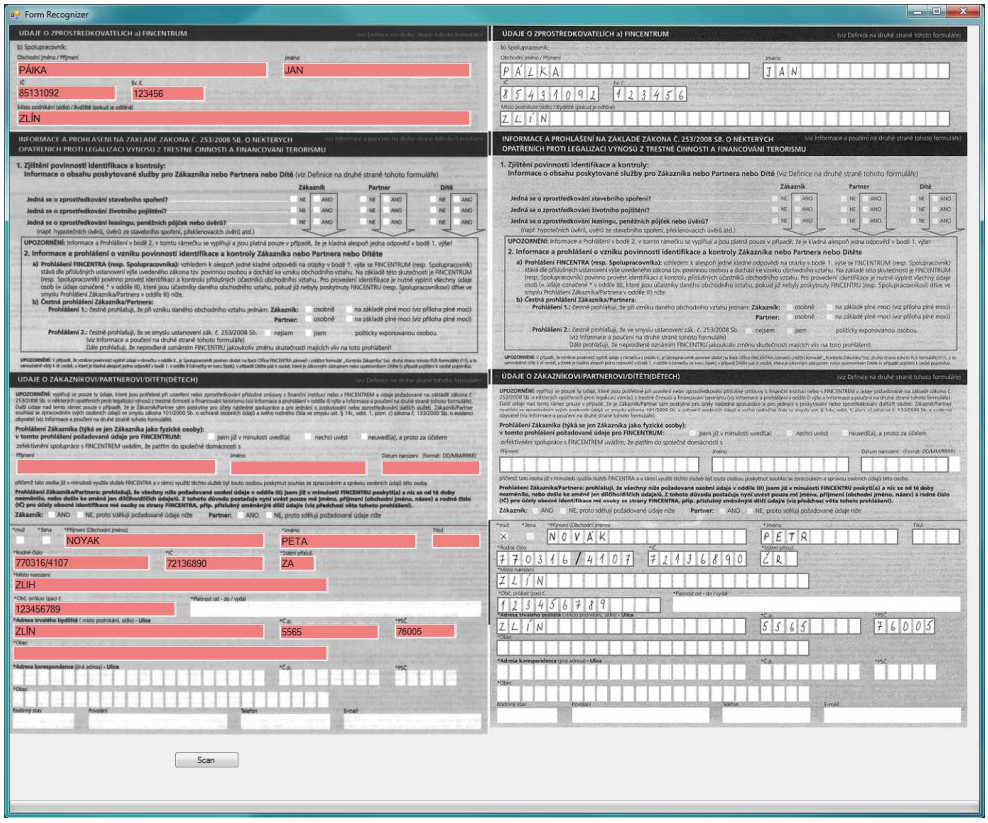
Před vstupem jednotlivých rámečků do aplikace OCRRecognizer jsou jednotlivé rámečky upravovány tak, aby neobsahovaly zbytky okolí formuláře nebo šum kolem obrázku. I přesto je potřeba podotknout, že na výsledky hledání polí měl vliv typ použitého skeneru. V případě použití vysoce kvalitního a také velmi finančně nákladného řešení Canon DR-X10C, nebylo potřeba šum odstraňovat a pole byla správně pomocí výše popsaných metod rozpoznána. Pro případ použití levnějších skenerů bylo potřeba zvýšit práh jasu pro rozpoznání polí na formulářích.



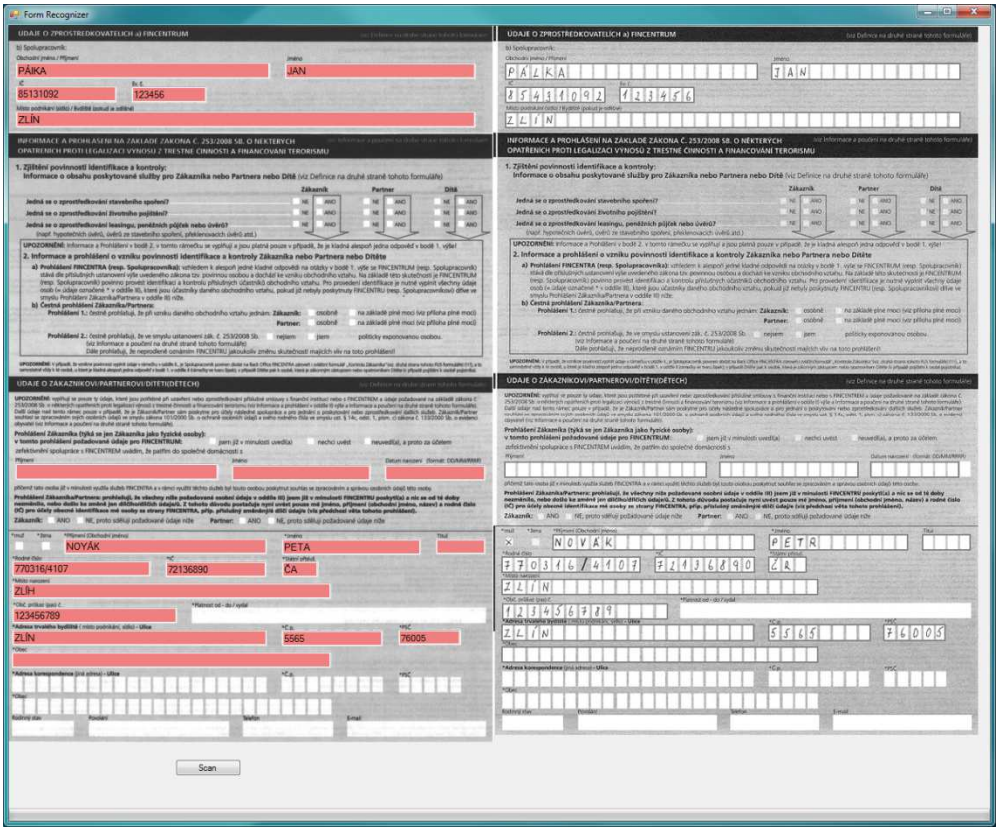
Obr. 48 Formulář po vyhledání jednotlivých polí (vlastní zdroj)

10.2.2 Testování rozpoznávání formulářů

Pomocí vytvořené aplikace, obsahující vyhledávání polí ve formuláři, viz kapitola 10.2.1, bylo provedeno testování rozpoznávání formulářů a vzhledem k bezchybnému fungování části aplikace, určené k lokalizaci jednotlivých zadávacích polí a separaci obrazů znaků v nich obsažených, výsledky odpovídaly již získaným měřením v rámci kapitoly 7.3.1 a korespondovaly s testováním jednotlivých znaků v rámci kapitoly 10.1. Ukázka funkce aplikace bez zařazeného prvku inteligentního preprocessingu rozlišujícího, zda jde o znak s diakritikou či bez diakritiky, je zobrazena na obr. 49. Z výsledného rozpoznaného formuláře lze vypozorovat velmi dobrá úspěšnost systému rozpoznávat jednotlivá pole ve formuláři, konkrétně se jedná především o chyby ve slovech s diakritikou. Toto je patrné např. u hodnoty státní příslušnosti, kde má neuronová síť problémy se znakem s diakritikou Č. Po zapojení prvku inteligentního preprocessingu viz obr. 50, lze pozorovat zlepšení schopnosti rozpoznávat znaky s diakritikou, ale i bez diakritiky, z důvodu užší sady znaků, na které musí být síť naučena. Dle tohoto konkrétního testu, ale mnoha dalších provedených v rámci analýz viz kapitola 7.3.1 odpovídá zlepšení cca 20%.



Obr. 49 Ukázka rozpoznání FUS pomocí FormRecognizer aplikace bez prvku IP (vlastní zdroj)



Obr. 50 Ukázka rozpoznání FUS pomocí FormRecognizer aplikace s prvkem IP (vlastní zdroj)

Ačkoli byla experimentálně ověřena velmi dobrá schopnost aplikace rozpoznávat ručně psaná data zadaná do formuláře, bylo zjištěno, že teoretická úspěšnost okolo 90%, dosažená implementovanými algoritmy, není v praxi dostatečná. Hlavním důvodem potřeby vysoké úspěšnosti rozpoznávání je potřeba zajištění 100% správnosti převedených dat, není tedy potřeba převést data se 100% správností, ale dokázat určit, která data jsou s určitostí správná. Na základě tohoto poznatku byla, po konzultaci se školitelem, zpracována kapitola, navrhující další postup zabývající analýzou možnosti implementací prvku pro validaci a korekci převedených dat.

10.2.3 Návrh dalšího postupu pro validaci a korekci převedených dat

V rámci disertační práce byl rozpracován stručný návrh metod pro validaci a korekci převáděných dat. Vzhledem k tomu, že tato problematika je velmi obsáhlá a nepatří do oblasti řešení této disertační práce, mohla by být vhodným námětem pro navazující základní i aplikovaný výzkum v této problematice.

10.2.3.1 Validace dat

Pro řešení problematiky validace a korekce převáděných dat se nabízí použití slovníkové metody pro kontrolu dat. Proces kontroly dat by byl založen na kontrole rozpoznaných slov oproti slovníku možných kombinací. V případě, že je rozpoznané slovo nalezeno ve slovníku může být považováno za správné. Slovníkovou metodu však nelze použít pro všechna pole na formulářích a záleží na jejich typu. Typickými případy polí vhodnými pro použití slovníkové metody jsou:

- Příjmení
- Jméno
- IČ, DIČ
- Titul
- PSČ

Kromě výše uvedených polí existují na formulářích také pole, která jsou pro použití slovníkové metody méně vhodná a pro jejich kontrolu je potřeba dodatečné logiky a algoritmů; mezi tato pole můžeme uvést např. Adresu, Název společnosti a další. Poslední kategorií, kterou nelze pomocí této metody vůbec validovat, jsou pole pro hodnoty, které nejsou obsaženy v žádném slovníku. Typickým příkladem může být pole pro zadání názvu společnosti ve formuláři pro založení nové společnosti.

10.2.3.2 Korekce dat

Korekce dat může být také řešena pomocí slovníkové metody, je zde však potřeba vytvoření pravidel a algoritmů, kterými by tato korekce byla řízena. Ukázkou algoritmů a výsledných pravidel mohou být následující pravidla:

1. V případě, že náhradou každého jednoho znaku je nalezeno právě jedno odpovídající slovo ve slovníku, je toto slovo považováno za správné.
2. V případě, že náhradou každého jednotlivého znaku ve slově není nalezeno právě jedno odpovídající slovo ve slovníku pokračuje se pravidlem 3
3. Pokud nahrazením libovolné varianty dvou znaků ve slově je nalezeno právě jedno odpovídající slovo ve slovníku, je toto slovo považováno za správné.

Další korekce jsou závislé na konkrétním typu zpracovávaného pole. Příkladem může být pole pro zadání rodného čísla, jehož struktura je řízena určitými pravidly. Tato pravidla je možné využít pro jejich validaci i korekci.

11 PŘEVOD FORMULÁŘŮ DO DIGITÁLNÍ FORMY

Formuláře získané skenováním bylo potřeba pro splnění cílů disertační práce převést do digitální formy vhodné pro použití v neuronových sítích. Pro tento převod bylo potřeba stanovit parametry potřebné u výsledných digitálních obrázků. Mezi tyto parametry řadíme barevná hloubka a rozlišení obrázku.

11.1 Skenování

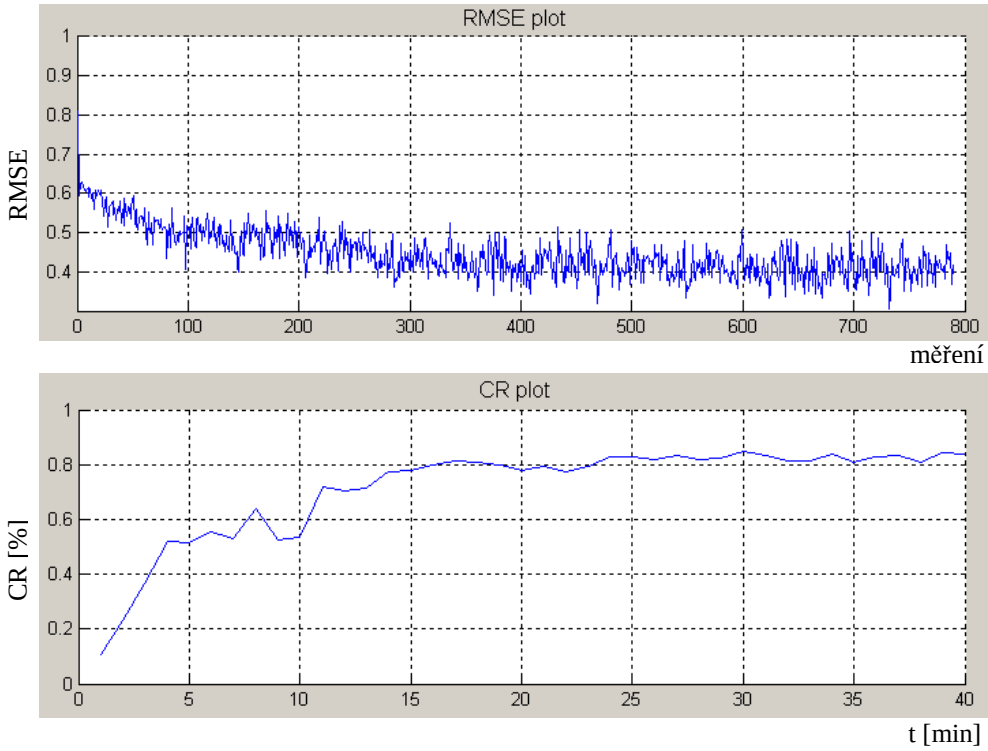
11.1.1 Rozlišení

Prvním krokem bylo skenování formulářů pomocí skenovacích zařízení. Konkrétně bylo skenování provedeno pomocí skenerů popsaných podrobněji v kapitole 12.1.

Skenování formulářů bylo provedeno v rozlišení 3519 px x 2550 px, což odpovídá cca 300 DPI. Takto vysoké rozlišení bylo použito především z neznalosti potřebného rozlišení pro následné použití v disertační práci a bylo vybráno hlavně za účelem získání dostatečné rezervy pro případ, že by bylo potřeba vyšší rozlišení. Z testů provedených na neuronových sítích, zobrazených na ukázkách obr. 51, 52, 53 a kompletního srovnání uvedeného v grafu na obr. 54 nicméně vyplývá, že potřebné rozlišení na jeden znak odpovídá zhruba 28x28 px, což přepočítáno na jednu stranu A4 odpovídá cca 79 DPI viz vztah 10.

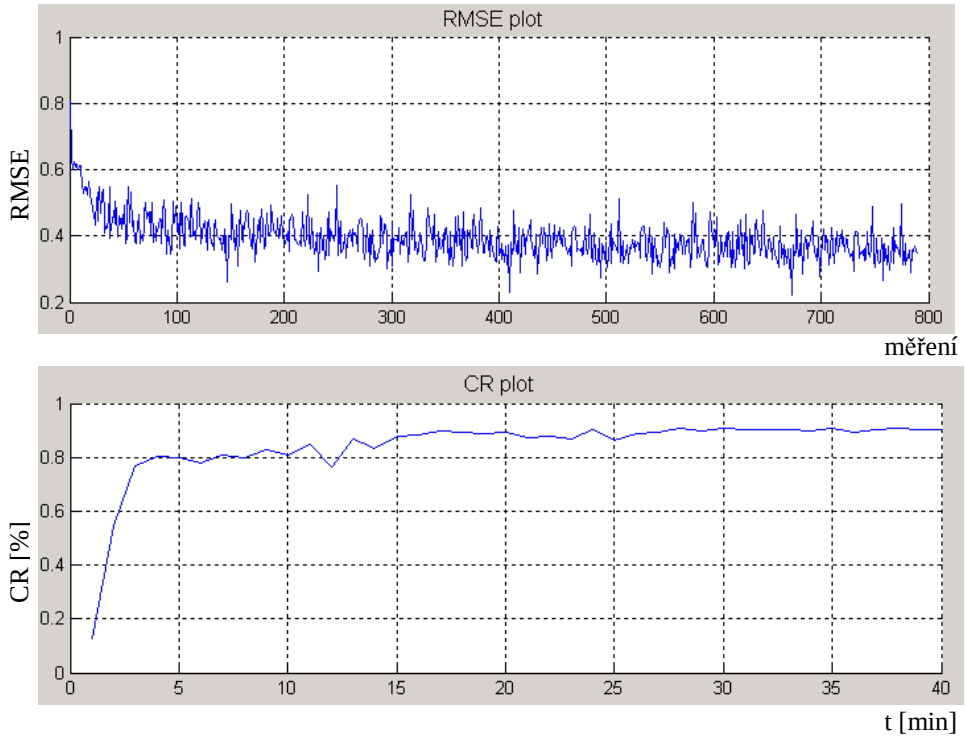
$$resA4 = \left(\frac{pw}{\mathrm{piw}*\mathrm{cw}}\right) \times \mathrm{cwn} = \left(\frac{3519}{8{,}267*130}\right) \times 28 = 78{,}903\mathrm{DPI} \qquad (10)$$

ResA4	potřebné rozlišení v palcích
piw	šířka stránky v palcích
pw	šířka stránky v px
cw	šířka políčka v px
cwn	požadovaná šířka políčka v px



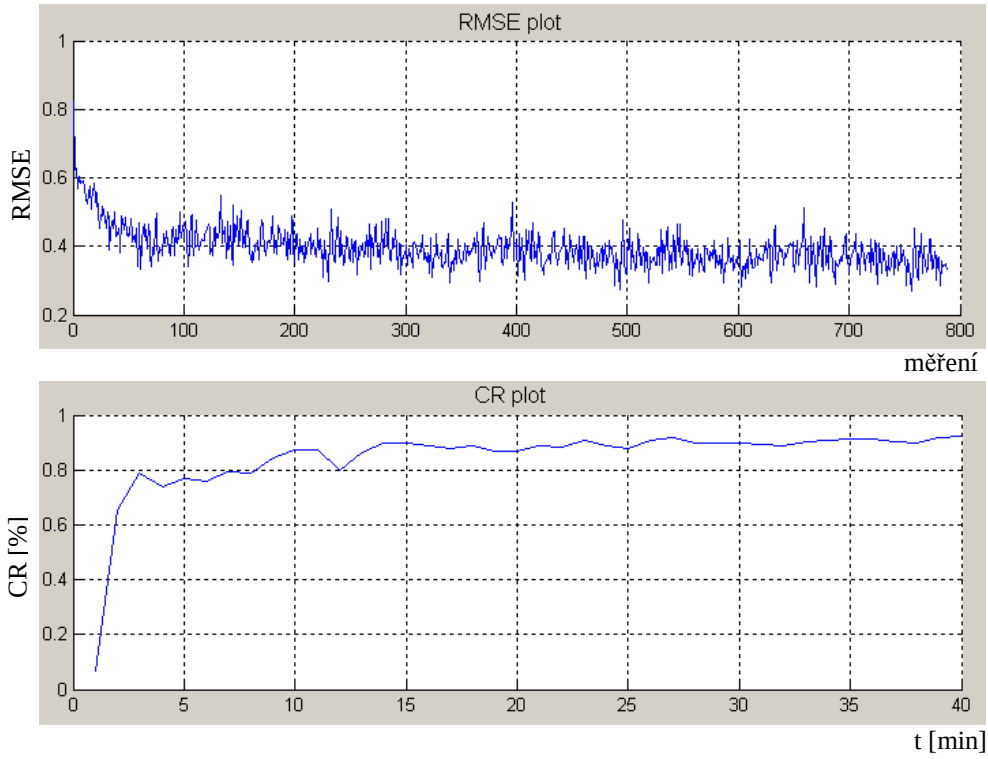
Obr. 51 Měření CR vstupních vzorů 18x18px (vlastní zdroj)

Nastavení:	1-9_originalAll18x18_3e_2640i_2340train_84n_04t
Epochy:	3
Iterace:	2 640
RMSE:	0,42
CR:	83%



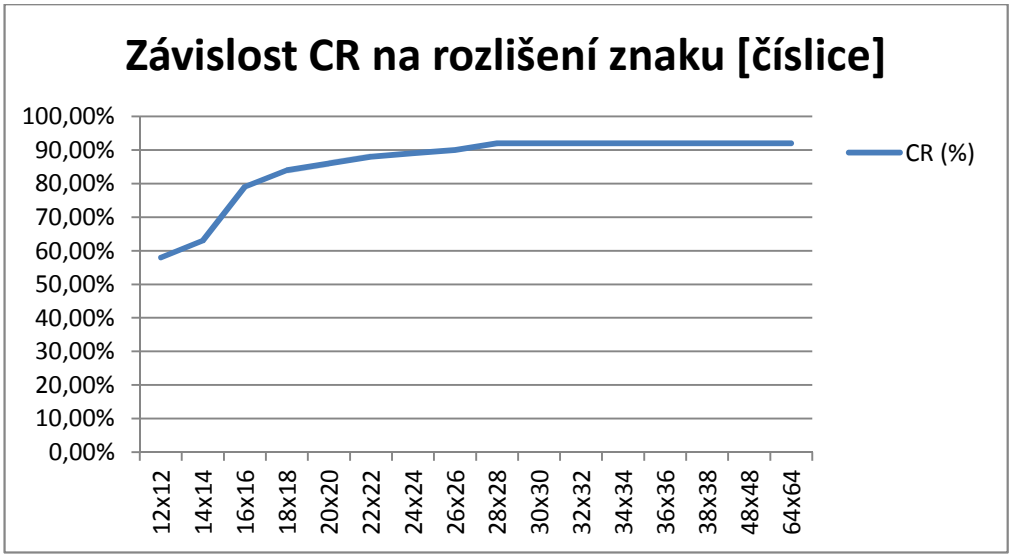
Obr. 52 Měření CR vstupních vzorů 24x24px (vlastní zdroj)

Nastavení:	1-9_originalAll24x24_3e_2640i_2340train_84n_04t
Epochy:	3
Iterace:	2 640
RMSE:	0,35
CR:	90%



Obr. 53 Měření CR vstupních vzorů 28x28px (vlastní zdroj)

Nastavení:	1-9_originalAll28x28_3e_2640i_2340train_84n_04t
Epochy:	3
Iterace:	2 640
RMSE:	0,33
CR:	92%



Obr. 54 Závislost CR a rozlišení vzorů na vstupu neuronové sítě (vlastní zdroj)

11.1.2 Barevná hloubka

Zvolení vhodné barevné hloubky skenovaných formulářů bylo dalším z problémů, které bylo potřeba vyřešit v rámci skenování formulářů. Stejně jako v případě specifikace rozlišení uvedeném výše, se v případě skenování pro výzkum zvolila dostatečná hodnota s rezervou, bez ohledu na potřebné parametry. Touto hodnotou byla maximální barevná hloubka podporovaná skenerem, tedy 24 bitů barevné hloubky. Z další práce s obrázky v neuronových sítích vyplynula potřebná barevná hloubka pro skenovací proces. Vzhledem k tomu, že neuronové sítě pracují se vstupem znaků ve 256 stupních šedi, je určení potřebné barevné hloubky pro skenování triviální, a je rovné těmto 256 stupním šedi. Příklad vzoru číslice „0“, o velikosti 10x10px (pro lepší čitelnost s invertovanými barvami), je zobrazen na obr. 55.

```
    act_letter=letter(act_digit);
    total = length(act_letter);

%     disp(['TRAIN> Processing letter: "' pism(d) '"']);


        for c=1:total

            celk = celk + 1;
            img=act_letter(c);
            img=imresize(img,[10 10],'bicubic');

            img=abs(255-img);
```

```
img: 10x10 uint8 =

    0    0    0    0    0    0    0    0    0    0
    0    0    0    0   70   83   11    0    0    0
    0    0    0   78  103   58  120   34    0    0
    0    0   21  152    7    0   28   91    0    0
    0    0   30   57    0    0   40   76    0    0
    0    0   25    3    0    0   99   44    0    0
    0    4  146   10    0   13  112    6    0    0
    0    0   84  101   49  119   46    0    0    0
    0    0    0   79  108   22    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
```

```
        end;

end;
```

Obr. 55 Ukázka struktury vzoru číslice „0“ (vlastní zdroj)

Z pohledu disertační práce a možného navazujícího výzkumu však byly obrázky skenovány ve 24 bitových barvách, a to hlavně z důvodu, aby v případě zvolení jiných metod, neuronových sítí, či postupů nebyl výzkum touto barevnou hloubkou v budoucnu omezen.

12 PROSTŘEDKY VYUŽITÉ PRO VÝVOJ A EXPERIMENTY

Pro provádění výpočtů, experimentů a při sestavení aplikace, implementující neuronové sítě Perceptron a Neocognitron, byly použity nástroje od společností Microsoft a MathWorks. Tato volba byla především z důvodu mnohaletých zkušeností autora s vývojem v tomto jazyce. Vývojovými prostředími použitými pro vývoj, byla nejnovější verze vývojového prostředí Microsoft Visual Studio 2012 a skriptovací prostředí Matlab ve verzi R2011b. Při výzkumu bylo využito jejich vhodných vlastností pro určité oblasti výzkumu.

12.1 Hardware

Veškeré výpočty, adaptace sítí a následné rozpoznávání předložených vzorů probíhalo na dostatečně výkonném hardware, jehož parametry jsou podrobněji zobrazeny v tab. 8.

Název	**Parametry**
Operační systém	Microsoft Windows Server 2008 R2, Hyper-V virtualizace
Typ systému	64bit
Procesor	Intel I7-930 2.8Ghz (4 jádra, 8 logických jader)
Paměť RAM	24GB DDR3 (použito 20GB)

Tab 8 Parametry použitého HW

V rámci disertační práce bylo vedle potřeby digitalizovat mnoho formulářů pro učení neuronových sítí, následné testování také digitalizovat reálné formuláře používané institucemi pro získávání dat od klientů případně o klientech. Pro skenování těchto formulářů byly použity skenovací zařízení uvedené v tab. 9. Je nutno podotknout, že použitý typ skeneru měl vliv na způsob předzpracování obrazu, neboť při použití jiných skenerů než Canon DR-X10C byl obsažen v bílé

barvě šum, který bránil rozpoznávání typovacích polí viz kapitola 10.2.1. Z důvodu nutnosti podporovat i levnější skenery než Canon DR-X10C (cena skeneru v době provádění analýz byla cca 420 tis. Kč bez DPH.) bylo potřeba zvýšit práh jasu pro rozpoznání polí na formulářích. Tento parametr by tedy v budoucnu mohl být dynamický dle kvality použitého skeneru.

Název	Parametry
Canon i-SENSYS MF4730	Optické rozlišení: 600 dpi, 24 bit, multifunkce
HP G3010	Optické rozlišení: 480 dpi, 24 bit
Canon DR-X10C	Optické rozlišení: 600 dpi, 24 bit, rychlost sken. 100 str./min.
XEROX WorkCentre 6015N	Optické rozlišení: 1 200 dpi, 24bit, multifunkce

Tab 9 Hardware použitý ke skenování

12.2 Software

12.2.1 C#, .Net Framework

C# je objektově orientovaný programovací jazyk od společností Microsoft, vydaný v první verzi v roce 2002 společně s .NET Frameworkem 1.0. Vychází z jazyka C++ v kombinaci s jazykem Java. Jedná se o jednoduchý, moderní, typově bezpečný a objektově orientovaný jazyk. Kód jazyka C# se kompiluje jako strukturovaný kód, z čehož plyne, že čerpá výhod ze služeb CLR (Troelsen, 2010). V současné době je hojně využívaný pro vývoj robustních enterprise aplikací fungujích pod .NET frameworky. .NET Framework je prostředí potřebné pro běh aplikací obsahující potřebné knihovny. Ačkoliv platforma .NET nevyžaduje konkrétní programovací jazyk, můžeme konstatovat, že nejpoužívanějšími jazyky jsou C#, Visual Basic .NET, a Delphi. Jak bylo zmíněno výše, pro vývoj potřebných aplikací k této práci byl vybrán jazyk C#, s .NET Frameworkem 4.0., z nichž byly využity převážně technologie LINQ a WCF (Stellman, 2010).

12.2.2 Matlab

Matlab je interaktivní programové prostředí a skriptovací programovací jazyk, který je vyvíjen společností MathWorks. MATLAB umožňuje počítání s maticemi, vykreslování 2D i 3D grafů funkcí, implementaci algoritmů, počítačovou simulaci, analýzu a prezentaci dat i vytváření aplikací včetně uživatelského rozhraní, (Moore, 2011). V rámci analýz bylo použito ve verzi R2011b a bylo využito především k experimentálnímu ověření implementace algoritmů neuronových sítí a porovnání hodnot úspěšnosti rozpoznávání těchto sítí na různých druzích vstupních znakových sad (Gilat, 2011).

12.2.3 Microsoft Visual Studio 2012

Programování veškerého kódu v jazyce C# implementovaného v navrženém systému bylo provedeno ve vývojovém prostředí Microsoft Visual Studio 2012. Jedná se o poslední verzi tohoto vývojového prostředí a jedná se o univerzální nástroj pro vývoj konzolových, grafických a webových aplikací. Dále je možné také toto prostředí pro vývoj mobilních aplikací.

12.2.4 Microsoft SQL Server 2008 R2

V rámci analýz možnosti implementace slovníkového řešení pro validaci a korekci převáděných dat, byl využit Microsoft SQL Server ve verzi 2008 R2. Tento server byl využit pro uložení dat potřebných pro slovníkové metody. Pro komunikaci s aplikací bylo využito technologie uložených procedur. (BEN-GAN, 2012) (Rankins, 2010)

12.2.5 Microsoft Visio 2010

Pro lepší čitelnost návrhů optimalizovaného systému a procesů v něm probíhajících byl pro vizualizaci použit nástroj Visio 2010 od společnosti

Microsoft. Modelování systému a funkčních procesů bylo prováděno v souladu s BPMN notací popsanou detailněji v kapitole 12.2.6.

12.2.6 BPMN Notace

Business Process Modelling Notation (BPMN) byla vyvinuta iniciativou Business Process Management Initiative (BPMI) a její první specifikace byla uvolněna v květnu 2004. Hlavním cílem snahy o vytvoření BPMN bylo poskytnutí notace, která by byla snadno pochopitelná pro každého čtenáře od analytiků, kteří se zabývají tvorbou návrhů procesů přes technické vývojáře odpovědné za implementace technologií, které vykonávají navržené procesy, až po uživatele, kteří budou mít na starost tyto procesy monitorovat. Mimo jiné BPMN lze vhodně použít jako můstek vyplňující mezeru mezi návrhem procesů a jejich následnou implementací. Výhodou použití této notace je mimo standardizace popisu a tím i usnadněné komunikace její názornost a čitelnost i pro uživatele bez detailních znalostí o projektu (White, 2004).

13 PŘÍNOS PRÁCE PRO VĚDU A PRAXI

Navržený způsob optimalizace systémů pro rozpoznávání ručně psaného písma pomocí prvku inteligentního předzpracování, přináší znatelné zlepšení vlastností těchto systémů a nabízí široké možnosti uplatnění těchto systémů v oblasti zpracování ručně psaných dat, jako je oblast zabývající se sběrem dat formou formulářů a dotazníků, či oblast týkající se získávání informací o klientech různých institucí a společností.

V rámci plnění cílů disertační práce byla vytvořena databáze znaků, kterou je možné využít jako přínosný zdroj sady znaků české abecedy s diakritikou pro další výzkum OCR, ICR systémů a může také sloužit jako validní zdroj dat pro další zájemce o výzkum neuronových sítí a jejich implementaci v ICR systémech. Dle znalostí autora, získaných v průběhu výzkumu a analýz, provedených v rámci práce na této disertaci, se jedná o nejobsáhlejší zdroj ručně psaných znaků české abecedy včetně diakritiky.

Možnou oblast využití navrženého řešení spatřuji, mimo jiné, v prostředí informačních systémů určených pro zpracovávání dat používaných v bankovnictví, finančních institucích, pojišťovnictví a poradenských společnostech. Hlavním přínosem zavedení automatizace převodu ručně psaného textu do digitální formy je bezesporu úspora nákladů na straně pracovních sil. Vedle úspory lidského času na přepisování, je možné rozšířit oblast zpracovávaných dat a díky tomu je možnost využití těchto dat jejich vytěžování a efektivnějšímu využití. V neposlední řadě lze navržené řešení využít pro kontrolu a validaci již dnes typovaných dat (zpracovávaná data by byla párována vůči rozpoznaným datům pomocí ICR).

Díky vlastnostem navrženého systému, jako je škálovatelnost či odělitelnost jádra rozpoznávání od klientské části, jej lze využít jak v mobilních aplikacích, tak v aplikacích desktopových systémů s velkým množstvím zpracovávaných dat. V nejbližší době je, z důvodu prověření funkčnosti systému v praxi, plánované

nasazení prototypu systému do informačního systému pro zpracování tištěných smluv v přední české poradenské společnosti s objemem zpracování těchto smluv v řádech jednotek tisíc smluv denně.

Zjištěné poznatky, získané v rámci testování převodu reálných formulářů a komunikací s pracovišti zabývajících se touto činností, vedly k potřebě verifikace a korekce převáděných dat. Řešením těchto problémů se autor práce již v současné době zabývá v rámci své profese a mohla by být námětem pro navazující základní i aplikovaný výzkum v této problematice.

14 ZÁVĚR

Systémy pro rozpoznávání ručně psaného textu nacházejí vlivem rozvoje informačních technologií své uplatnění v mnoha oborech lidských činností jako můstek, zprostředkovávající převod textu z tištěné formy do formy digitální. Jedná se především o obory, ve kterých se v současnosti stále pracuje s velkým objemem dat v tištěné podobě, jako jsou archívy dokumentů ve státní či komerční sféře nebo oblasti, ve kterých je stále prováděn sběr dat formou tištěných formulářů. Vzhledem k tomu, že rozpoznávání ručně psaného textu má v každém jazyce svá specifika, je nutné správně navrhnout řešení těchto rozpoznávacích systémů pro každý jazyk zvlášť.

Tato disertační práce se zabývá problematikou optimalizace systémů pro rozpoznávání ručně psaného textu se zaměřením na rozpoznávání textu s českou diakritikou. Jsou zde popsány základní typy algoritmů využívaných v současné době v oblasti rozpoznávání textu, a je zde provedena detailní rešerše konvoluční neuronové sítě Neocognitron, která byla v rámci práce, díky svým vlastnostem, zvolena jako stěžejní neuronová síť pro implementaci ve výsledném ICR systému. Pro funkčnost rozpoznávacích systémů je důležité zjednodušení předložených obrazů těmto systémům a z tohoto důvodu je v závěru teoretické části proveden rozbor, týkající se detekce hran, a metod, které se k této detekci používají. Experimentální část práce popisuje výběr vhodné neuronové sítě pro účel daný tématem disertační práce. Součástí je taktéž experimentální ověření úspěšnosti rozpoznávání neuronové sítě Neocognitron na různých typech vstupních sad znaků provedené simulacemi na implementovaných algoritmech této sítě ve skriptovacím prostředí Matlab R2011b.

V rámci práce bylo realizováno vytvoření vlastního zdroje ručně psaných znaků, včetně znaků s diakritikou. V první fázi sběr probíhal především mezi studenty, na půdě fakulty aplikované informatiky, nicméně ve druhé fázi bylo

přistoupeno ke sběru těchto znaků i mimo univerzitu od starších respondentů z důvodu možného ovlivnění stylu písma věkem pisatele. Praktické testy provedené pomocí vytvořených algoritmů na číslicích prokázaly, že tento vlastní zdroj je srovnatelný s referenčním zdrojem dat, získaným z databáze MNIST, a je možné jej použít pro další výzkum v této práci i mimo ni.

Implementované algoritmy v Matlab byly ve spojení s vlastním zdrojem znaků využity k analýze potřebného rozlišení a barevné hloubky pro kvalitní naučení neuronové sítě. V dalším kroku byly také použity k testování různých vstupních sad znaků do neuronových sítí a tímto způsobem byla odhalena slabá místa rozpoznávání ručně psaného písma s českou diakritikou.

Na základě poznatků, získaných provedenou rešerší, analýzou a testováním, byl vytvořen návrh ICR systému pro rozpoznávání ručně psaného písma, který splňuje veškeré dnešní požadavky na něj kladené, mezi něž patří především škálovatelnost a bezpečnost z pohledu zabezpečení know-how, obsaženém v nastavení použité neuronové sítě. Navržený systém je dále obohacen o prvek inteligentního preprocesingu, který si klade za cíl zjednodušení práce jádra ICR systému, pomocí rozlišení, zda rozpoznávaný znak obsahuje diakritiku či nikoliv. Takto optimalizovaný systém vykazoval dle testů provedených zlepšení hodnoty CR o více než 20 %.

Dílčí výstupy práce, týkající se zejména implementace algoritmů neuronové sítě Neocognitron jsou, díky profesnímu zaměření autora, již v současné době prakticky testovány v rámci prototypů pro nasazení v komeční sféře.

Návrhy pro navazující výzkum v této problematice:

- Navrhnout a implementovat komponentu zajišťujicí validaci dat pomocí slovníkové metody.

- Ověřit možnosti využití jiných moderních neuronových sítí ART, LeNet, Deep Belief Network.
- Otestovat systém na kompletní sadě znaků NIST.
- Zpracovat analýzu slovníkové metody založené na neuronových sítích.

15 LITERATURA

ALI, M. *Using the Canny edge detector for feature extraction and enhancement of remote sensing images*, Geoscience and Remote Sensing Symposium, 2001. ISBN: 0-7803-7031-7

BISHOP, Ch. *Pattern Recognition and Machine Learning.* Springer, 2006. 738 s. ISBN: 978-0387310732

BEN-GAN, I. *Microsoft SQL Server 2012 T-SQL Fundamentals.* Microsoft Press, 2012. ISBN: 978-0-7356-5814-1

BUCKLAND, M., K. *Electronic Document Retrieval, And Vannevar Bush's Memex*, Journal of the American Society for Information Science. 1992. Dostupné z: http://people.ischool.berkeley.edu/~buckland/goldbush.html

CASTELLANO, G. *Handwritten digits recognition using Hough transform and neural networks.* Circuits and Systems, ISCAS '96 IEEE International Symposium. 1996.

DASARATHY, B. V. *Nearest Neighbor: Pattern Classification Techniques.* Ieee Computer Society, 1990. 550 s. ISBN: 978-0818689307.

FINK, G. A. *Markov Models for Pattern Recognition: From Theory to Applications*. Springer, 2007. 260 s. ISBN: 978-3540717669.

FLETCHER. R. *Practical Methods of Optimization.* 2. vyd. Wiley, 2000. 450 s. ISBN: 978-0471494638

FUKUSHIMA, K. *Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position*. Biological Cybernetics, 1980. 202 s.

FUKUSHIMA, K. *Neural Networks and Information Processing.* Addison Wesley Longman, 340 s. ISBN: 978-0201525748

GILAT, A. *MATLAB, 4th Edition*, Wiley, 2011. ISBN: 978-1-11813-662-1.

GORMAN, L., SAMMON, M. J. *Practical Algorithms for Image Analysis*. Cambridge University Press, 2000. 302 s. ISBN: 978-0521884112

GREEN, CH. *Perception: An introduction to the Gestalt-Theorie by Kurt Koffka,* York University, 2000. Dostupné z WWW: http://psychclassics.yorku.ca/Koffka/Perception/intro.htm

GUO, S. *Probabilistic Hough transform for line detection utilizing surround suppression,* Machine Learning and Cybernetics International Conference, 2008. ISBN: 978-1-4244-2096-4.

HAILONG, L. *Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes.* Document Analysis and Recognition International Conference, 2005. ISBN: 0-7695-2420-6

HASNAT, A., HABIB, M., KHAN, M. *Segmentation Free Bangla OCR using HMM: Training and Recognition*. [online]. [cit. 2012-06-12]. Dostupné z WWW: http://www.panl10n.net/english/outputs/Working%20Papers/ Bangladesh/Microsoft%20Word%20-%2029_N_176.pdf

HAYKIN, S. *Neural Networks and Learning Machines*. Third Edition - New York: Prentice Hall 2009. 936 s. ISBN: 978-0131471399

HOUDEK, M., SVOBODA, T., PROCHÁZKA, T. *Klasifikace podle nejbližších sousedů*. FEL ČVUT Praha, Přednášky [online]. [cit. 2012-05-26]. Dostupné z WWW: < http://cmp.felk.cvut.cz/cmp/courses/recognition/ zapis_prednasky/zapis_01/4/rpz4.pdf>

HOANG, L., THUAN, L., T. *NET Framework Essentials, 2nd Edition*. O'Reilly Media 2002. ISBN: 978-0-596-00302-9.

CHERIET, M., KHARMA, N., LIU, Ch. *Character Recognition Systems: A Guide for Students and Practitioner*. Wiley-Interscience, 2007. 360 s. ISBN: 978-0471415701

IOANNIDIS, I., NOVIKOV B., RACHEV, B. *Advances in Databases and Information Systems,* 11th East European Conference, 2007. ISBN: 9783540751847.

JAIN, A. K. *Fundamentals of Digital image processing.* Prentice Hall, 1988.

JAVIDI, B. *Image Recognition and Classification.* Marcel Dekker, Inc. 2002. 506 s. ISBN: 0-8247-0783-4

KAŇA, M. *Kohonenova síť*. Bakalářská práce, Vysoké učení technické v Brně. Brno 2011.

KYRYATI, N., ELDAR, Y., BRUCKSTEIN, A., M. *A probabilistic Hough transform,* Department of Electrical Engineering Institute of Technology, Haifa, 1991.

LENT, S. C. *Learning to Program with MATLAB,* Wiley, 2013. ISBN: 978-0-470-93644-3.

LIN, H. *A weighted minimum distance classifier for pattern recognition.* Electrical and Computer Engineering, Canadian Conference, 1993. ISBN: 0-7803-2416-1.

LIU, Y., STARZYK, J., ZHU, Z. *Optimized Approximation Algorithm in Neural Networks Without Overfitting*. IEEE Transactions on neural networks, 2008. 995 s. ISSN: 1045-9227.

LU, Z., BAZZI, I., KORNAI, A., MAKHOUL, J., NATARJAN, P., SCHWARTZ, R. *A Robust, Language-Independent OCR System. BBN Technologies*, GTE Internetworking, Cambridge, MA 02138. [cit. 2012-07-22]. Dostupné z WWW: < http://citeseerx.ist.psu.edu >

MICROSOFT, Windows Image Acquisition (WIA). Microsoft, dostupné z: http://msdn.microsoft.com/en-us/library/ms630368(VS.85).aspx

MOORE, H. *MATLAB for Engineers.* 3 vyd. Prentice Hall, 2011. 672 s. ISBN: 978-0132103251

NAVA-ORTIZ, M. Digit recognition system for camera mobile phones, Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on. 2011.

PETERKA, J. *Báječný svět elektronického podpisu, CZ.NIC, 2011. ISBN: 978-80-904248-3-8*

PLOTZ, T., FINK, G. A. *Markov Models for Handwriting Recognition*. Springer, 2011. 84 s. ISBN: 978-1447121879

RABINER, L. R. *A tutorial on hidden Markov models and selected applications in speech recognition.* Proceedings of the IEEE, vol. 77, 1989.

RANGANATHAN, A. *The Levenberg-Marquard Algorithm.* 2004. [cit. 2012-07-22]. Dostupné z WWW: <http://citeseerx.ist.psu.edu>

RANKINS, R., BERTUCCI, P. T., GALLELLI, Ch., SILVERSTEIN, A. T. *Microsoft SQL Server 2008 R2 Unleashed.* Sams, 2010. 1704 s. ISBN: 978-0672330568

RITCHIE, S., D. *Pro .NET Best Practices,* Apress, 2011. ISBN: 978-1-4302-4023-5.

ROSENBLATT, F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, 1962. 616 s. ASIN: B0006AXUII

SHEPHERD, A. *Second-Order Methods for Neural Networks: Fast and Reliable Training Methods for Multi-Layer Perceptrons.* Springer, 1997. 145 s. ISBN: 978-3540761006

SHI, S. *Taking time seriously: hidden Markov experts applied to financial engineering,* Computational Intelligence for Financial Engineering (CIFEr), 1997. ISBN: 0-7803-4133-3.

SINGH, D., MEHTA, N., PUROHIT, P. *Text Based Image Recognition using Multilayer Perceptron*, IP Multimedia Communication. [cit. 2012-05-18]. Dostupné z WWW: http://www.ijcaonline.org/

SINGH, S.,SINGH, M. *Pattern Recognition and Data Mining.* Springer, 2005. 715 s. ISBN-13: 978-3540287575

STAIR, R., REYNOLDS, G. *Fundamentals of Information Systems*. 2.vyd. Course Technology, 2003. 432 s. ISBN-10: 0619064919

STELLMAN, A., GREENE, J. *Head First C#, 2nd Edition.* O'Reilly Media. 2010. ISBN: 978-1-4493-8034-2.

UEDA, M. : *Toward Dialogue Documents as Creative Conversational Tools*. Center for Coordination Science, Massachusetts Institute of Technology, 1998. Dostupný z: http://ccs.mit.edu/papers/CCSWP206/

TROELSEN, A. *Pro C# 2010 and the .NET 4 Platform.* 5vyd. Apress, 2010. 1752 s. ISBN: 978-1430225492.

TWAIN Working Group. *TWAIN Specification 2.2 [online]*, 16.2.2012. Dostupné z: www.twain.org

VOJÁČEK, A. *Samoučící se neuronová síť* [online]. [cit. 2012-03-01]. Dostupné z WWW: <http://automatizace.hw.cz/mereni-a-regulace/ART244-samoucici-se{neuronova-sit{som-kohonenovy-mapy.html>.

WANG, H., BENGIO, S. *The MNIST Database Of handwritten upper-case letters,* Dalle Molle Institute for Perceptual Artificial Intelligence. [cit. 2011-12-02]. Dostupné z WWW: http://www.idiap.ch

WHITE, A. S. *Introduction to BPMN,* IBM Corporation, 2004. Dostupné z WWW: www.bpmn.org

WHITE, A. S. *Workflow Patterns with BPMN and UML,* IBM Corporation, 2004. Dostupné z WWW: www.bpmn.org

nové

16 PUBLIKAČNÍ AKTIVITY

1. PÁLKA, J.: Application of the OLAP cubes technology in the Data mining process. *Informační a datová bezpečnost ve vazbě na strategické rozhodování ve znalostní společnosti.* Zlín 24. - 25. 3. 2009. ISBN 80-238-6785-7.
2. PÁLKA, J., PÁLKA J.: Principy a vlivy zavedení bezpečnostní politiky na chod společnosti. *Bezpečnostní technologie Systémy a Management*. Zlín 9. - 11. 9. 2009. ISBN 978-80-7318-864-1.
3. PÁLKA J., PÁLKA J.: Application of the Olap Cubes Technology in the Data Mining. *Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium „Intelligent Manufacturing & Automation: Focus on Theory“*, 2009, ISBN 978-3-901509-70-4.
4. PÁLKA J., MOTÝL, J., JAŠEK, R.: Use of active directory in securing the client applications. *Internet, bezpečnost a konkurenceschopnost organizací*. Zlin 17-18. 3. 2010. ISBN 978-83-61645-16-0.
5. PÁLKA, J., MOTÝL, I.:Securing the client-server communication in WCF. In Int. 2010. *Internet, bezpečnost a konkurenceschopnost organizací*. Zlin 17-18. 3. 2010. ISBN 978-83-61645-16-0.
6. PÁLKA J., MOTÝL I., JAŠEK R.: Cybercrime. *Internet, bezpečnost a konkurenceschopnost organizací*. Zlin 17-18. 3. 2010. ISBN 978-83-61645-16-0.
7. PÁLKA, J., MOTÝL, I., JAŠEK, R.: Application of hash function to increase security level of the information system. *Internet, bezpečnost a konkurenceschopnost organizací*. Zlin 17-18. 3. 2010. ISBN 978-83-61645-16-0.
8. PÁLKA, J., MOTÝL, I.: Ways to protect information system against techniques of SQL injection. *Internet, bezpečnost a konkurenceschopnost organizací*. Zlin 17-18. 3. 2010. ISBN 978-83-61645-16-0.
9. MOTÝL, I., PÁLKA, J., PÁLKA, J.: Advanced Methods for Securing the Information Systems. *Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium*, 2010, ISBN 978-3-901509-73-5.

10. PÁLKA, J., PÁLKA, J, MOTÝL, I.: Securing the Client-Server Communication in WCF. *Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium*, 2010, ISBN 978-3-901509-73-5.

11. PÁLKA, J., PÁLKA, J, MOTÝL, I.: Use of Active Directory in securing the client applications. *Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium*, 2010, ISBN 978-3-901509-73-5.

12. PÁLKA, J., MOTÝL, I., PÁLKA, J.: Cybercrime – website injection techniques. In Int. 2011. *Internet, competitiveness Organizational Security*. Zlin 16-17. 3. 2011. ISBN 978-83-61645-16-0.

13. PÁLKA, J., PÁLKA, J., NAVRÁTIL, M.: OCR Systems in language specific environments. In Int. 2011. *The 13th WSEAS International Conference. Lanzarote*, Canary Islands, 27 - 29. 5. 2011. ISBN 978-1-61804-003-9.

14. PÁLKA, J., PÁLKA, J., NAVRÁTIL, M.: OCR Systems based on convolutional neocognitron network. *NAUN International Journal of Mathematical Models and Methods in Applied Sciences*. ISSN 1998-0140.

15. PÁLKA, J., PÁLKA, J.: OCR Systems based on Neural Network. *Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium,* Volume 22. ISBN 978-3-901509-83-4

17 ŽIVOTOPIS

OSOBNÍ INFORMACE

Jméno:	Jan Pálka	
Datum narození:	6. 4. 1984	
Adresa:	Kúty 1941 Zlín, Česká republika	
Rodinný stav:	svobodný	
Kontakt:	telefon: email:	+420 774 498 509 jpalka@fai.utb.cz

VZDĚLÁNÍ

1999 – 2003	Gymnázium Zlín, Lesní čtvrť
2003 – 2006	Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, Informační technologie (Bc.)
2006 – 2008	Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, Informační technologie (Ing.)
2008 – dosud	Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, doktorský studijní program.

STÁŽE A MOBILITY

2010	Univerzidade de Vigo – Španělsko, studijní pobyt v rámci ERASMUS

PRAXE

2005 – 2007	Programování ASP.NET aplikací a podpůrné práce při správě informačních systémů. (externí spolupráce pro Aquila SW)
2008	Vývoj ASP.NET aplikací s komunikací s MS SQL (OSVČ, externí spolupráce pro Edhouse s.r.o.)
2008 - 2012	Programování komplexních IS systémů v ASP.NET, Winforms za použití MS SQL, (OSVČ): Tvorba a správa ekonomického magazínu PROFIT.cz včetně podpůrných aplikací Tvorba a správa týdeníku Czech Business Weekly včetně podpůrných aplikací Spolupráce na vývoji jádra ERP systému pro poradenskou společnost Fincentrum a.s., včetně podpůrných modulů
2012	Vedení týmu programátorů v technologiích MS.NET na pozici projektový manažer a vývoj mobilních aplikací na platformě IOS. (zaměstnanecký poměr ve společnosti Business Logic s.r.o.)
2013	Programování v technologiích MS.NET na pozici Senior Developer. (zaměstnanecký poměr ve společnosti Business Logic s.r.o.)

JAZYKOVÉ ZNALOSTI

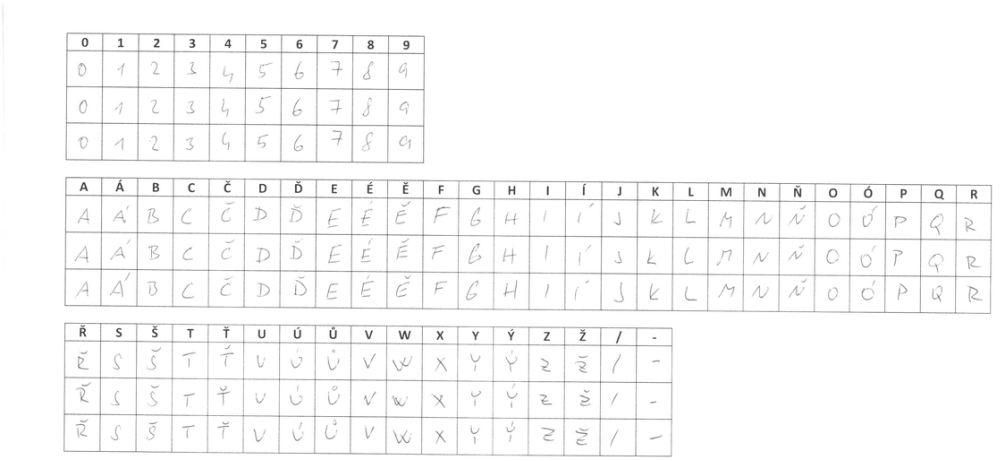
Český jazyk	nativně
Anglický jazyk	pokročilý
Německý jazyk	pokročilý
Španělský jazyk	základy

Výuka	Databázové systémy, Univerzita Tomáše Bati ve Zlíně (2009, 2010)
	Instrumentace a měření, Univerzita Tomáše Bati ve Zlíně (2008 - 2011)
	Technologie WWW, Univerzita Tomáše Bati ve Zlíně (2011)
Konference	Spolupráce na přípravě konference: Internet, bezpečnost a konkurenceschopnost organizací (2010)

PŘÍLOHY

Příloha A: Ukázka formulářů použitých pro sběr dat

Ukázka formuláře pro sběr testovacích dat č.1



Ukázka formuláře pro sběr testovacích dat č.2

Ukázka formuláře pro sběr testovacích dat č.3

Příloha B1 - Převod znaků do požadované velikosti

```
% program nacte ze souboru s nazvem "all.mat"
% vsechny pismenka z formularu
% a zkonvertuje je do souboru h_w_all.dat
%kde h a w je vyska a sirka obrazku

clear all;
clc;
h=28; w=28;

% nacteni dat ze souboru
load('all.mat');

new_letter=[];
n_pism=length(letter);

% celkovy pocet znaku
for p=1:n_pism
  %  celkovy pocet vzoru
  pism = letter{p};
  pocet = length(pism);

  temp=[];
  for c = 1:pocet
        img = pism{c};
        [nimg, newIMG,n,final_img]=uprava1(img,20,[h
w],0);
        temp=[temp; {final_img}];
  end
        new_letter=[new_letter; {temp}];
end

letter=new_letter;
filename = ['all_' num2str(h) 'x' num2str(w) '.mat'];
save (filename,'letter','all_pism');
disp('');
disp(['Soubor ' filename ' ulozen!']);
```

Příloha B2 - Načtení znaků z uložených dat z formulářů

```
function [I,labels,pism] = load_myDB(random)

% nacteni dat ze souboru
load C:\DOKUMENTY\13_05_14_POKUS_OCR\OCR\all_28x28.mat

% vybrani prislusne sady znaku
% pism='0123456789';
  pism='ABCDEFGHIJKLMNOPQRSTUVWXYZ';
% pism='AÁBCČDĎEÉĚFGHIÍJKLMNŇOÓPQRŘSŠTŤUÚŮVWXYÝZŽ';
% pism='ÁČĎÉĚÍŇÓŘŠŤUÚŮÝŽ';

% vstupni a vystupni vektor
I=[];
labels=[];
celk=0;

% pocet znaku
n_pism=length(pism);

for d=1:n_pism

    % vyhledat v obrazech znaku aktualni znak
    act_digit = find(all_pism==pism(d));
    act_letter= letter{act_digit};

    % pocet variant daneho znaku
    total = length(act_letter);

    for c=1:total
        celk = celk + 1;

        % vzit aktualni variantu daneho znaku
        img=act_letter{c};

        % negativ daneho znaku
        img=abs(255-img);
```

```
        randd = zeros(32,32);
        randd(3:30,3:30)=img;  %resImg;   %puvodne img

        img = reshape(mapstd(reshape(randd,1,[])),32,32);

        I{celk}=(img);
        labels(celk)=d;

        end;

end;

% nahodne prohozeni
if (random==1)
    [I, labels] = random_order(I, labels);
end
```

Příloha B3 - Nastavení parametrů pro trénování neuronové sítě

```
clear;
clc;

% pocet znaku, musí souhlasit s vybranou sadou
ntotal= 6864;
ntrain= 1716;

[I, labels,pism] = load_myDB(1);
classes = length(pism);

%%
%pocet vrstev
numLayers = 8;
%pocet podvzorkovacích vrstech
numSLayers = 3;
%pocet konvolucnich vrstev
numCLayers = 3;
%pocet plne propojenych vrstev
numFLayers = 2;
```

```
numInputs = 1;
%Image width
InputWidth = 32;
%Image height
InputHeight = 32;
%Number of outputs
numOutputs = classes;
% vytvoreni neuronove site
sinet =
cnn(numLayers,numFLayers,numInputs,InputWidth,InputHeight,numOutpu
ts);

% parametry neuronove site
sinet.SLayer{1}.SRate = 1;
sinet.SLayer{1}.WS{1} = ones(size(sinet.SLayer{1}.WS{1}));
sinet.SLayer{1}.BS{1} = zeros(size(sinet.SLayer{1}.BS{1}));
sinet.SLayer{1}.TransfFunc = 'purelin';

% 2. vrstva
sinet.CLayer{2}.numKernels = 6;
sinet.CLayer{2}.KernWidth = 5;
sinet.CLayer{2}.KernHeight = 5;

% 3. vrstva
%Subsampling rate
sinet.SLayer{3}.SRate = 2;

% 4. vrstva
sinet.CLayer{4}.numKernels = 16;
sinet.CLayer{4}.KernWidth = 5;
sinet.CLayer{4}.KernHeight = 5;

% 5. vrstva
sinet.SLayer{5}.SRate = 2;
sinet.CLayer{6}.numKernels = 120;
sinet.CLayer{6}.KernWidth = 5;
sinet.CLayer{6}.KernHeight = 5;
sinet.FLayer{7}.numNeurons = 20;
sinet.FLayer{8}.numNeurons = numOutputs;
sinet = init(sinet);
```

```
%Nastaveni pripojovaci mapy dle Yann Lecun
sinet.CLayer{4}.ConMap = ...
[1 0 0 0 1 1 1 0 0 1 1 1 1 0 1 1;
 1 1 0 0 0 1 1 1 0 0 1 1 1 1 0 1;
 1 1 1 0 0 0 1 1 1 0 0 1 0 1 1 1;
 0 1 1 1 0 0 1 1 1 1 0 0 1 0 1 1;
 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1;
 0 0 0 1 1 1 0 0 1 1 1 1 0 1 1 1;
]';
sinet.SLayer{1}.WS{1} = ones(size(sinet.SLayer{1}.WS{1}));
sinet.SLayer{1}.BS{1} = zeros(size(sinet.SLayer{1}.BS{1}));

%%
% pocet epoch
sinet.epochs = 8;
%Mu koef pro Levenberg-Markvardt
sinet.mu = 0.001;
%trenovaci koeficient teta
sinet.teta =  0.0005;

sinet.HcalcMode = 1;
sinet.Hrecalc = 300;
sinet.HrecalcSamplesNum = 50;

%Teta decrease coefficient
sinet.teta_dec = 0.4;

% trenovani
sinet = train(sinet,I,labels,ntotal,ntrain,pism);
disp('Konec');
save('my_cnet.mat','sinet','pism');
```

Příloha B4 - Rozpoznávání znaků v neuronové síti

```
function [out, cnet] = sim(cnet,inp)

% podvzorkování
cnet.SLayer{1}.SS{1} = subsample(inp,cnet.SLayer{1}.SRate);
cnet.SLayer{1}.YS{1} = cnet.SLayer{1}.SS{1}.*cnet.SLayer{1}.
    WS{1}+cnet.SLayer{1}.BS{1} ;

% přenosová funkce
cnet.SLayer{1}.XS{1} = feval(cnet.SLayer{1}.TransfFunc,cnet.SLayer{1}.YS{1});

% smyčka hlavní vrstvy
for k=2:(cnet.numLayers-cnet.numFLayers)
    if(rem(k,2))
        for l=1:cnet.CLayer{k-1}.numFMaps % pro vsechny mapy z n-1 vrstvy
            % podvzorkování, výstup matice na 1-D vektor
            XC = reshape(cnet.CLayer{k-1}.XC,1,[]);
            cnet.SLayer{k}.SS{l} = subsample(XC{l},cnet.SLayer{k}.SRate);
            cnet.SLayer{k}.YS{l} = cnet.SLayer{k}.SS{l} *
                cnet.SLayer{k}.WS{l}+cnet.SLayer{k}.BS{l};
            % aplikovat přenosovou fci
            cnet.SLayer{k}.XS{l} =
                feval(cnet.SLayer{k}.TransfFunc,cnet.SLayer{k}.YS{l});
        end
    else

    % C-vrstva
    YC = num2cell(zeros(cnet.CLayer{k}.numKernels,1));
            for l=1:cnet.CLayer{k}.numKernels
                % pro vsechny konvolucni jadra
                for m=find(cnet.CLayer{k}.ConMap(l,:))
                    % pro vsechny mapy predchozi vrstvy dle pripoj. mapy
                     YC{l} = YC{l}+fastFilter2(cnet.CLayer{k}.WC{l},
                     cnet.SLayer{k-1}.XS{m},'valid')+cnet.CLayer{k}.BC{l};
                end
            end

     cnet.CLayer{k}.YC = YC;
     cnet.CLayer{k}.XC=cnet.CLayer{k}.YC;
```

```
    end
end

% po posledni c-vrstve jsou vsechny mapy velikosti 1x1
% bunky v poli do vektoru
XC = cell2mat(cnet.CLayer{k}.XC)';
for k=(cnet.numLayers-cnet.numFLayers+1):cnet.numLayers
    if (k == cnet.numCLayers+cnet.numSLayers+1)
     cnet.FLayer{k}.Y = XC*cnet.FLayer{k}.W+cnet.FLayer{k}.B;
     cnet.FLayer{k}.X = feval(cnet.FLayer{k}.TransfFunc,cnet.FLayer{k}.Y);
    else
     cnet.FLayer{k}.Y = cnet.FLayer{k-1}.X*cnet.FLayer{k}.W+cnet.FLayer{k}.B;
     cnet.FLayer{k}.X = feval(cnet.FLayer{k}.TransfFunc,cnet.FLayer{k}.Y);
    end
end

out = cnet.FLayer{k}.X;
```

Příloha C: Ukázky zdrojového kódu C#

Vzhledem k rozsáhlosti vytvořeného kódu (v řádu tisíců řádků kódu jsou uvedeny pouze ukázky jednotlivých části – viz kapitoly disertační práce)

Příloha C1 - ICR systém, OCRApplication

```
     private void btDo_Click(object sender, EventArgs e)
        {
            // načíst pdf s formulářem
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.InitialDirectory = @"C:\temp\pdf";
            ofd.Filter = "PDF files|*.pdf|" + "JPG files|*.jpg|" + "All
files|*.*";
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                tbPath.Text = ofd.FileName;
            }
            string fileName = tbPath.Text;

            // počáteční čas rozpoznávání
            DateTime startTime = DateTime.Now;
            Cursor = Cursors.WaitCursor;
            lNotify.Text = "";
            lNotify.ForeColor = Color.Red;
            // vyčistit richbox
            rtbCompareString.Clear();

            // vytvořit data pro gridview
            DataTable dt = null;
            dt = new System.Data.DataTable();
            dt.Columns.Add("Item");
            dt.Columns.Add("Result");
            dt.Columns.Add("Original");
            resultGrid.DataSource = dt.DefaultView;

            // volání metody OCRFormRecognizer z knihovny OCR form pro
rozpoznávání formuláře a vyhledání polí s hodnotami
            List<CropForm.OCRString> allList =
CropForm.OCRFormRecognizer(dt,fileName);

            // zobrazeni vysledku (vcetne zavolání rozpoznávací metody)
            ShowFormResults(allList, dt, fileName);
```

```
            // konečný čas rozpoznávání
            DateTime endTime = DateTime.Now;
            Cursor = Cursors.Arrow;

            // doba trvání rozpoznávání
            TimeSpan tsInterval = endTime - startTime;
            tbDuration.Text = tsInterval.ToString();
        }
```

Příloha C2 - ICR systém, OCRApplication – metoda ShowFormResult

```
void ShowFormResults(List<CropForm.OCRString> inList, DataTable dt ,
string fileName)
        {
            string name = Path.GetFileNameWithoutExtension(fileName);

            #region fieldList

            TextWriter testTXT = new
StreamWriter(@"C:\temp\porovnani.txt", true);

            // pocet chybných znamének
            int numErrors = 0;
            // pocet správných znamének
            int numCorrect = 0;

            for (int i = 0; i < inList.Count; i++)
            {
                if (inList[i].Name == "Rodné číslo")
                {

                    LoggingExceptions(0,"Rozpoznavani..");


// volani metody UnknowenImageRecognize - predana bitmapa a typ neur.site
                    OCRStringRecog(inList[i]);

                    for (int o = 0; o < lOrg.Count; o++)
                    {
                        if (name == o.ToString())
                        {

        dt.Rows.Add(inList[i].Name, inList[i].GetBestResult(), lOrg[o]);
```

```
        testTXT.WriteLine(name + "\t" + inList[i].Name + "\t" +
inList[i].GetBestResult() + "\t" + lOrg[o]);

                                string s = lOrg[o];
                                string s2 = inList[i].GetBestResult();


                                for (int p = 0; p < s.Length; p++)

                                {
                                    if (s[p] != s2[p])
                                    {
                                        rtbCompareString.SelectionColor =
Color.Red;

rtbCompareString.AppendText(s2[p].ToString());
                                        numErrors++;
                                    }
                                    else
                                    {
                                        rtbCompareString.SelectionColor =
Color.Black;

rtbCompareString.AppendText(s2[p].ToString());
                                        numCorrect++;
                                    }
                                }

                                tbErrors.Text = numErrors.ToString();
                                tbCorrect.Text = numCorrect.ToString();
                                tbPercentage.Text =
CalculPerc((double)numCorrect, (double)(numCorrect +
numErrors)).ToString();
                                break;
                            }
                            else
                                continue;
                        }
         testTXT.WriteLine("Percentage: " + tbPercentage.Text + "\r\n");
                    }
                }
                testTXT.Close();

            }


            public void OCRStringRecog(CropForm.OCRString s)
            {
                OCRResult result;
                s.Clear();
                for (int j = 0; j < s.BmChars.Count; j++)
                {
```

```
                if (s.BmChars[j] != null)
                {
                    // predani bitmapy z listu a typu neuronove site
kterou bude rozpoznavan
                    result = UnknowImageRecognize(s.BmChars[j], s.RcTyp);
                    s.AddOCRChar(result);
                }
                else
                {
                    if (s.Length > 0)
                    {
                        s.AddOCRChar(null);
                    }
                    else
                        s.LeftTrim++;
                }
            }
            s.RightTrim();

        }
```

Příloha C3 - ICR systém, OCRApplication - metoda UnknownImageRecognize

```
        /// rozpoznání jednotlivych znaku pomoci naucene neuronove site.
        private OCRResult UnknownImageRecognize(object input,
CropForm.OCRString.RecognizerType rcType)
        {
            Bitmap imageOriginal;
            if (input.GetType() == typeof(string))
                imageOriginal =
ImageProcessing.LoadFromFile((string)input);
            else if (input.GetType() == typeof(Bitmap))
                imageOriginal = (Bitmap)input;
            else
                throw new ArgumentException("Invalid argument!");

            if (rcType == CropForm.OCRString.RecognizerType.Alpha)
                /// volání OCRTools knihovny
                return recognizer.RecognizeChar(imageOriginal);

            else

            {
                OCRResult result =
recognizerNum.RecognizeChar(imageOriginal);
                return result;
            }
```

```
        }
```

Příloha C4 - srovnání implementace části rozpoznávací funkce Matlab vs C#

MATLAB

```
for k=2:(cnet.numLayers-cnet.numFLayers) %(First layer is
dummy, skip it)
    if(rem(k,2))
    %S-layer
        for l=1:cnet.CLayer{k-1}.numFMaps
            XC = reshape(cnet.CLayer{k-1}.XC,1,[]);
            cnet.SLayer{k}.SS{l} =
subsample(XC{l},cnet.SLayer{k}.SRate);
            cnet.SLayer{k}.YS{l} = cnet.SLayer{k}.SS{l}*
cnet.SLayer{k}.WS{l}+cnet.SLayer{k}.BS{l};
            %Apply transfer function
            cnet.SLayer{k}.XS{l} =
feval(cnet.SLayer{k}.TransfFunc,
cnet.SLayer{k}.YS{l});
        end
    else
```

C#

```
// Main layer loop - comment from Matlab
for (k = 1; k < (cnet.numLayers - cnet.numFLayers); k++)
{
// S-layer
if (k % 2 == 0)
{
numFMaps = (int)((MLDouble)cnet.GetLayerData("CLayer", k - 1,
"numFMaps")).Get(0);
for (l = 0; l < numFMaps; l++)
  {
   MLCell cLayXC = (MLCell)cnet.GetLayerData("CLayer", k - 1, "XC");
   XC = cnetFile.reshape(cLayXC);
   ratio = (int)((MLDouble)cnet.GetLayerData("SLayer", k,
"SRate")).Get(0);
   ss = cnetFile.subsample(XC[l], ratio);
   cnet.SetLayerData("SLayer", k, "SS", l, ss);
```

```
   ws = ((MLDouble)((MLCell)cnet.GetLayerData("SLayer", k,
"WS")).Cells[l]).GetArray();
   bs = ((MLDouble)((MLCell)cnet.GetLayerData("SLayer", k,
"BS")).Cells[l]).GetArray();
   ys = OCRMat.Plus(OCRMat.Multiple(ss, ws[0][0]), bs[0][0]);
   cnet.SetLayerData("SLayer", k, "YS", l, ys);
   tfce = cnetFile.TransferFunction((MLChar)cnet.GetLayerData("SLayer",
k, "TransfFunc"));
   xs = OCRMat.feval(tfce, ys);
   //Apply transfer function
   cnet.SetLayerData("SLayer", k, "XS", l, xs);
  }
}
```

Jan Pálka

Disertační práce

Optimalizace systémů pro rozpoznávání ručně psaného textu pomocí metod umělé inteligence

Univerzita Tomáše Bati ve Zlíně – Fakulta aplikované informatiky

Zlín 2013, počet stran 154.