VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HOLISTICKÉ ROZPOZNÁNÍ REGISTRAČNÍ ZNAČKY POMOCÍ KONVOLUČNÍCH NEURONOVÝCH SÍTÍ

HOLISTIC LICENSE PLATE RECOGNITION BASED ON CONVOLUTION NEURAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ANH LE HOANG**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAKUB ŠPAŇHEL**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Le Hoang Anh**
Program: Informační technologie
Název: **Holistické rozpoznání registrační značky pomocí konvolučních neuronových sítí**
Holistic License Plate Recognition Based on Convolution Neural Networks
Kategorie: Zpracování obrazu

Zadání:

1. Zorientujte se v současných metodách rozpoznávání registračních značek (RZ) vozidel.
2. Prostudujte dostupné materiály na téma rozpoznávání pomocí konvolučních neuronových sítí.
3. Vyberte vhodnou metodu a navrhněte konvoluční neuronovou síť pro rozpoznání RZ vozidla holistickým způsobem.
4. Experimentujte s vaší implementací a návrh případně iterativně vylepšujte.
5. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
6. Vytvořte stručný plakát a video prezentující vaši bakalářskou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Špaňhel Jakub, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 6. listopadu 2018

Abstrakt

Cílem práce bylo vytvořit holistický rozpoznávač registračních značek, kde byl kladen důraz na dosáhnutí co nejvyšší přesností na snímcích horší kvality. Byla navrhnuta a implementována kombinace konvoluční a rekurentní neuronové sítě, implementované pomocí LSTM a CTC, kde vstupem jsou výřezy získané z celé značky. Dále byly také implementovány konkurenční sítě pro porovnání výsledků. Sítě byly porovnány na celkem 4 datových sadách, a výsledkem bylo, že vlastní návrh dosáhl nejlepších výsledků s celkovou přesností rozpoznávání 97.6%.

Abstract

Main goal of this work was to create a holistic license plate reader, with an emphasis on achieving the highest possible accuracy on low quality images. Combination of convolutional and recurrent neural networks was designed and implemented, with usage of LSTM and CTC, where the inputs are cut-outs from the entire license plate. Competitive networks were also implemented to compare results. Networks were compared on a total of 4 datasets and the results were, that my design has achieved the best results with a recognition accuracy of 97.6%.

Klíčová slova

registrační značka, strojové učení, CNN, RNN, OCR, LSTM, CTC

Keywords

license plate, machine learning, CNN, RNN, OCR, LSTM, CTC

Citace

LE HOANG, Anh. *Holistické rozpoznání registrační značky pomocí konvolučních neuronových sítí*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jakub Špaňhel

Holistické rozpoznání registrační značky pomocí konvolučních neuronových sítí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jakuba Špaňhela. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Anh Le Hoang
16. května 2019

Poděkování

Chtěl bych poděkovat svému vedoucímu práce panu Ing. Jakubovi Špaňhelovi za jeho věnovaný čas a rady, které mi poskytoval po celou dobu tvorby práce. Dále bych chtěl také poděkovat své rodině za jejich podporu.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá problematikou počítačového vidění, kterou je rozpoznávání registračních značek. V současné době má mnoho využití v dopravě např. pro automatické účtování elektronického mýtného, ovládání přístupu do parkoviště, monitorování dopravy a nebo pro určení vozidla, které se dopustilo dopravního přestupku.
Existuje mnoho metod pro přepis snímku registrační značky do textové podoby, ale získání korektního výstupu mohou ztížit různé faktory, např. úhel pohledu, šum, rozostření, osvětlení nebo rozlišení snímku, se kterými se metody, založené na segmentaci znaků, nejsou schopny vždy vypořádat nejlépe. Ale v současné době lze díky neuronovým sítím zcela přeskočit nutnost segmentace a pro snímky s horší kvalitou získat přesný přepis.
Cílem této práce bylo navrhnout a implementovat rozpoznávač registračních značek, pomocí neuronových sítí, se zaměřením na přesnost. Návrh využívá kombinaci konvoluční a rekurentní sítě, která se velice často objevovala při řešení této problematiky, a která vychází z práce [18]. Vlastní návrh spočívá v úpravě architektury sítě ve zmíněné práci, aby umožnila spojení oddělených podsítí do jedné. Dále byly pro porovnání implementovány ostatní konkurenční sítě. Jedna z nich také využívá podobnou kombinaci a druhá využívá konvoluční síť s rozděleným výstupem.
Kapitola 2 obsahuje popis registrační značky a stručný úvod do různých typů neuronových sítí. Následující kapitola 3 obsahuje možné přístupy pro rozpoznávání, v kapitole 4 se pak nachází informace o implementaci vybraných holistických rozpoznávačů sítí a pak návrh a implementace vlastní sítě. V kapitole 5 se nachází informace o použitých datových sadách, vyhodnocení jednotlivých sítí a celkové zhodnocení dosažených výsledků.

Kapitola 2

Analýza problému

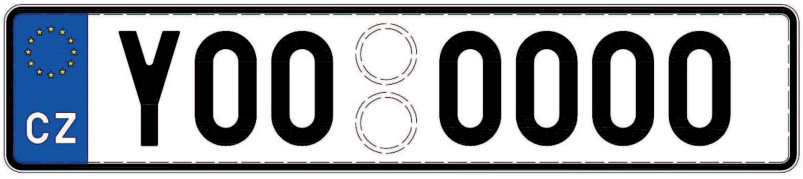
Rozpoznávání textu (OCR[1]) spočívá v převodu obrázků obsahující text do elektronické podoby, se kterým lze pak dále pracovat. Příkladem mohou být přepisy dokumentů, získání textu z fotografií nebo rozpoznání registračních značek. Správný přepis může ovlivnit mnoho faktorů, jako je úhel pohledu, rozostření, osvětlení, stíny nebo poškození textu. Výhodou přepisu registračních značek od přepisu textu z fotografií je, že nepotřebují zpracovávat písmena různých typů a barev. Další výhodou je, že pozadí registračních značek je ve většině stejné barvy.

Následující sekce obsahuje informace o registračních značkách a lehký úvod do neuronových sítí.

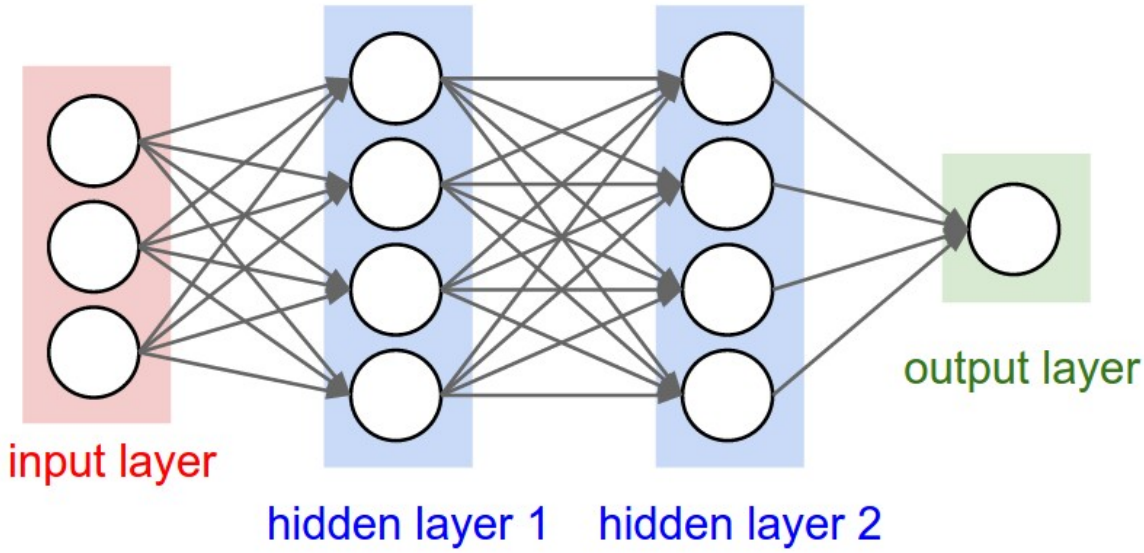
2.1 Registrační značka

Vyhláška č. 343/2014 Sb.[2] rozlišuje několik druhů registračních značek, které se dělí na standardní, na přání, pro elektrická vozidla, zvláštní a na umístění na nosné zařízení. Standardní registrační značka je tvořena kombinací nejméně pěti a nejvíce osmi znaků arabských číslic a velkých písmen kromě G, O, Q a W, kvůli potenciální záměně s podobnými znaky, např. číslice 0 s písmenem O nebo písmenem Q. Na značce musí být vždy nejméně 1 číslice a 1 písmeno. Nejlevější písmeno značí kód kraje. Písmena mohou být uspořádány na 1 nebo 2 řádcích. V případě, že jsou na 2 řádcích, jsou na prvním řádku maximálně 3 znaky a na druhém tedy 5. Na standardní značce se nachází mezi třetím a čtvrtým zleva místo, sloužící pro umístění nálepky o pravidelné technické prohlídce. Tabulka s registrační značkou odpovídá obdélníku o rozměrech 520 x 110 mm pro osobní automobily. Tvoří ji černé znaky na bílém pozadí a na levé straně se nachází modrý pruh s rozlišovací značkou České republiky. Využívají se ale i jiné kombinace barev. Registrační značky musí být vyrobeny ze slitin lehkých kovů a podklad musí být reflexního provedení. Značka by měla být na podélné ose vozidla, dále by měla být kolmo ke směru jízdy a vodorovně k vozovce. Značka musí být na zadní i přední straně vozidla a měla by být za nesnížené viditelnosti čitelná na 40 metrů. Příklad registrační značky lze vidět na obrázku 2.1.

[1] https://en.wikipedia.org/wiki/Optical_character_recognition
[2] https://www.zakonyprolidi.cz/cs/2014-343#cast5



Obrázek 2.1: Ukázka registrační značky. Převzato z [5]



Obrázek 2.2: Ukázka neuronové sítě s 2 skrytými vrstvami. Převzato z [4]

2.2 Umělá neuronová síť

Neuronová síť [14, 21] je výpočetní model, který je inspirován biologickými neuronovými sítěmi. Hlavní složkou, kterou ji pak tvoří jsou neurony, které obsahují libovolný počet vstupů, ale jen jeden výstup. Výstupu neuronu odpovídá následující vzorec

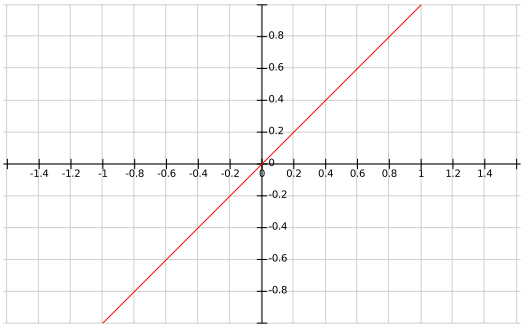
$$y = f(\sum_{i=1}^{n} w_i x_i + b), \tag{2.1}$$

kde f značí aktivační funkci, x_i značí vstupní hodnotu, w_i značí váhu, která představuje důležitost daného vstupu, a b značí práh. Model neuronu je znázorněn na obrázku 2.9.

Neurony jsou poté uspořádány do vrstev, které se dělí na vstupní, skryté a výstupní. V případě že výstupy neuronů jedné vrstvy jsou vstupem pro následující vrstvu bez žádných cyklických propojů mezi vrstvami nebo mezi neurony stejné vrstvy, tak se nazývají dopředné. Sítě které pak obsahují více skrytých vrstev se nazývají hluboké. Ukázka neuronové sítě je na obrázku 2.2

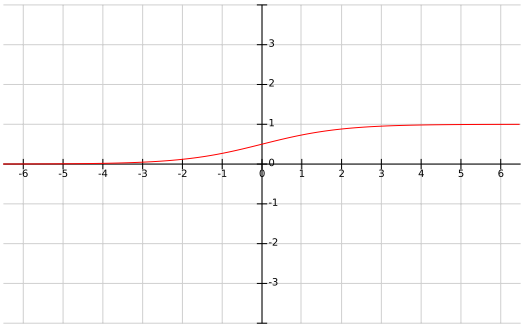
2.2.1 Hyperparametry

Hyperparametry jsou parametry sítí, které se během trénování nemění a slouží pro nastavení vlastností sítí. Může se jednat o nastavení počtu trénovacích epoch, volbu optimalizační, chybové funkce, nebo určení vlastností vrstev neuronové sítě.

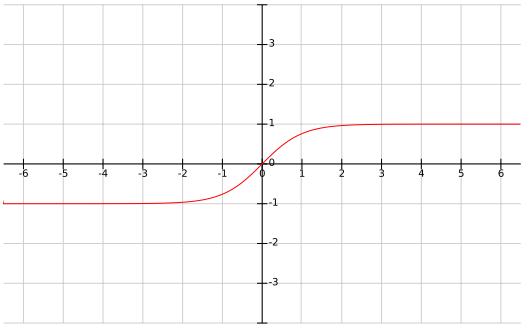


Obrázek 2.3: Ukázka lineární funkce

Obrázek 2.4: Ukázka funkce ReLU



Obrázek 2.5: Ukázka sigmoidy



Obrázek 2.6: Ukázka hyperbolického tangensu

2.2.2 Datové sady

Jedním z použití neuronových sítí je klasifikace, která se pro vstup snaží určit k jaké třídě patří. Příkladem klasifikace může být určení, co za písmeno se nachází na obrázku. Způsob jak natrénovat síť pro klasifikaci spočívá v porovnání výstupu sítě a očekávaným výstupem. Tomuto způsobu se pak říká učení s učitelem, který využívá pro trénování datovou sadu tvořenou dvojicemi (vstup, očekávaný výstup). Datové sady, lze pak rozšířit pomocí augmentace[3], která spočívá v úpravě vstupního obrázku, např. rotací, posuvem nebo přidáním šumu a tím zvýšit schopnost generalizace sítě. Augmentace lze provést před trénováním a to uložením jako součást datové sady nebo lze augmentaci provést během trénování a vytvořené obrázky neukládat. Další způsob jak získat data je vytvoření syntetických snímků. Výhodou syntetických dat je, že máme v podstatě neomezený počet trénovacích dat, ale vytvořené snímky nemusí vždy odpovídat skutečným datům, což by se mohlo projevit na přesnosti klasifikace.

2.2.3 Aktivační funkce

Aktivační funkce [25] určují, jakých hodnot může nabývat výstup neuronu. Dělí se na lineární a nelineární, ale důvod proč se nevyužívá lineární bude popsána v následující sekci.

[3] https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Lineární funkce

Lineární funkci odpovídá vzorec

$$f(x) = ax \text{ pro } a \in \mathbb{R}, \tag{2.2}$$

avšak jako aktivační funkce se nevyužívá, protože by se jakýkoliv počet skrytých vrstev s lineární aktivační funkcí, rovnal jedné vrstvě. Toto lze dokázat na jednoduchém příkladě. Mějme tedy příklad se 2 vrstvami, kde jedna z nich je napojená na druhou a obsahují každá 1 neuron. x bude značit vstup do sítě a výstupy jednotlivých vrstev budou označeny jako y_1 a y_2, které odpovídají následujícím rovnostem vycházející ze vzorce 2.1

$$\begin{aligned} y_1 &= 5x + 3, \\ y_2 &= 3y_1 + 2, \\ \implies y_2 &= 15x + 11. \end{aligned}$$

Výstup y_2 je opět lineární a lze tedy získat pomocí jedné skryté vrstvy. Dalším důvodem je, že parciální derivace výstupu neuronu podle x by byla konstantní a nebyla by závislá na x, což je nevhodné pro trénování sítě. Funkce je znázorněna na obrázku 2.3.

Funkce rectified linear unit (ReLU)

ReLU [22] je v současné době jedna z nejpoužívanějších aktivačních funkcí. Výhoda ReLU je rychlost trénování sítě oproti jiným aktivačním funkcím např. *sigmoid*, protože nevyužívá náročné operace, jako je dělení a sítě dosahují lepších výsledků trénování. Funkce je znázorněna na obrázku 2.4.

$$f(x) = max(0, x). \tag{2.3}$$

Sigmoida

Vrací hodnoty z rozsahu $(0, 1)$, nevýhodou je problém s *vanishing gradient* pro hodnoty x, jejichž hodnota $f(x)$ se blíží 0 nebo 1. Funkce je znázorněna na obrázku 2.5 a její matematické vyjádření je

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{2.4}$$

Hyperbolický tangens

Vrací hodnoty z rozsahu $(-1, 1)$, funkce je znázorněna na obrázku 2.6 a její matematické vyjádření je

$$f(x) = \frac{2}{1 + e^{-2x}} - 1. \tag{2.5}$$

Funkce softmax

Funkce převádí vstup na výstupní vektor, kde jednotlivé hodnoty leží v intervalu $\langle 0, 1 \rangle$ a celkový součet je roven 1. Každá z hodnot udává pravděpodobnost příslušnosti k určité třídě. Výpočet jednotlivých hodnot výstupu odpovídá následující funkci

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}, \tag{2.6}$$

kde J značí dimenzi vstupního vektoru a x_i značí i-tou hodnotu ze vstupního vektoru.

2.2.4 Trénování sítí

Cílem trénování je najít takové váhy, aby se minimalizovala hodnota objektivní funkce, která udává, jak moc se výstup pro trénovací data ze sítě liší od očekávaného výstupu. Proces trénování lze rozdělit do 2 kroků na dopředný a zpětný průchod. Při dopředném průchodu se pro vstupní data určí výstup sítě a spočítá se celková hodnota objektivní funkce. Jako jednu z objektivních funkcí pro řešení klasifikace lze zmínit *cross-entropy*

$$H(p, q) = -\sum_{x} p(x) \log q(x), \tag{2.7}$$

kde p je vektor obsahující hodnotu 1 pro korektní třídu a 0 pro ostatní třídy. q obsahuje distribuci pravděpodobností jednotlivých tříd, která je výstupem *softmax* vrstvy. Hodnota objektivní funkce je pak využita při zpětném průchodu, kde úpravy vah jsou provedeny pomocí optimalizačního algoritmu v závislosti na gradientech, které určuje zpětná propagace. Zpětná propagace využívá řetězového pravidla, která je definována tak, že pro $y = g(x)$ a $z = f(g(x)) = f(y)$ platí

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}. \tag{2.8}$$

Gradient se pro určitou váhu počítá, jako parciální derivace objektivní funkce podle dané váhy. Váha, pro kterou byl gradient vypočten, bude ovlivněna opačnou hodnotou než udává gradient, který vždy ukazuje směrem pro zvýšení hodnoty objektivní funkce. Jako první optimalizační algoritmus pro hledání minima lze zmínit *stochastic gradient descent* (SGD). Algoritmus spočívá v tom, že se váhy upravují pro každý trénovací prvek $x^{(i)}$ a jeho očekávaný výstup $y^{(i)}$.

$$\theta = \theta - \epsilon\delta\theta J(\theta; x^{(i)}; y^{(i)}), \tag{2.9}$$

kde ϵ značí učící koeficient, který je jeden z hyperparametrů sítě, ϵ značí váhy, δ značí gradient a J značí objektivní funkci. Existují i podoby, které provádí úpravu po dávkách vstupních dat a nebo se provádí úprava až po průchodu přes všechny trénovací data. Jako další optimalizační algoritmy lze pak zmínit *Adadelta* a *Adam*, které jsou popsány v práci [28].

2.2.5 Přeučení

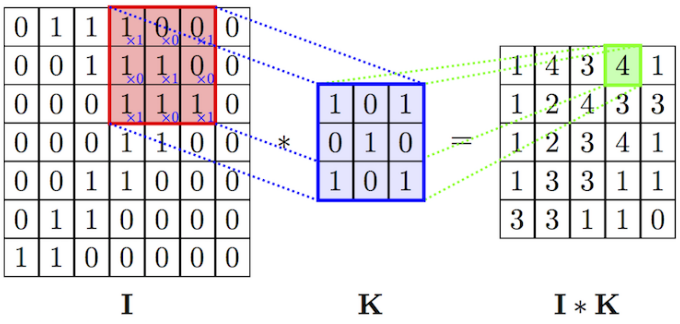
Přeučení, nebo anglicky *overfitting*, je situace, kdy během trénování sítě hodnota chybové funkce nad trénovacími daty klesá, ale schopnost sítě klasifikovat správně testovací data také klesá.

2.3 Konvoluční neuronová síť (CNN)

Konvoluční sítě [14] jsou typem hlubokých sítí, které navíc využívají konvoluční vrstvy, umožňující zpracovat vstupní data např. v podobě obrázků.

2.3.1 Popis vrstev neuronové sítě

V následujících sekcích budou informace o jednotlivých vrstvách, které se mohou nacházet v konvolučních sítích. Vrstvy kromě trénovatelných parametrů obsahují i hyperparametry.



Obrázek 2.7: Ukázka konvoluce. Převzato z [1]

Konvoluční vrstva

Slouží pro extrakci příznaků získanou konvolucí vstupu a konvolučního jádra, jinak také nazývaný filtr. Diskrétní konvoluce je definována jako

$$(x * w) = \sum_{i=-\infty}^{\infty} x(i)w(t-i), \tag{2.10}$$

kde x značí vstup a w filtr. Ale jelikož chceme provádět konvoluci nad obrázky, je potřeba ještě zadefinovat dvourozměrnou konvoluci, kde x značí dvourozměrný obrázek a w značí dvourozměrný filtr

$$(x * w) = \sum_{i} \sum_{k} x(i,k)w(t-i, u-k). \tag{2.11}$$

Hyperparametry pro tuto vrstvu je velikost posuvu, rozměry filtru, které jsou pro všechny filtry ve vrstvě stejné, počet filtrů a zda-li má být vstup vyplněn nulami na okraji, aby se konvolucí nesnížily rozměry výstupu. Počet výstupních kanálů je roven počtu filtrů, nikoliv počtu vstupních kanálů, ale tento počet odpovídá počtu kanálů ve filtru. Výstupem konvoluční vrstvy je příznaková mapa. Konvoluční vrstvy na nižší úrovni detekují jednoduché útvary například hrany a okraje, ale následující vrstvy už detekují složitější útvary. Počet trénovatelných parametrů odpovídá vzorci

$$p = (xym + 1)n, \tag{2.12}$$

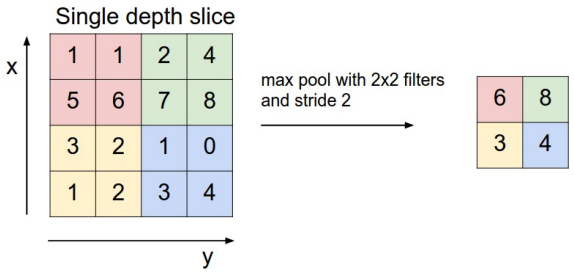
kde x a y jsou rozměry filtru, m je počet kanálů vstupu a n je počet filtrů. Ukázka konvoluce je znázorněna na obrázku 2.7

Maxpooling vrstva

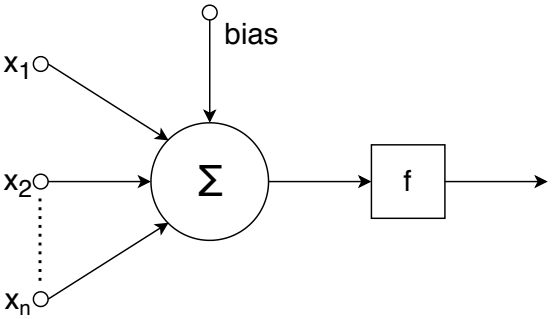
Podvzorkovací vrstva slouží pro zmenšení rozměrů vstupu a tím zároveň redukuje počty parametrů pro další vrstvy. Výstupem je tenzor obsahující maximální hodnoty z podoblastí o specifikované velikostí, která se posouvá podle určení kroku nad vstupem. Nemá žádné parametry, které by se při trénování měnily. Ukázka maxpoolingu je znázorněna na obrázku 2.8.

Dropout vrstva

Dropout [31] vrstva zahazuje některé vstupní neurony v závislosti na zadané pravděpodobnosti. Cílem vrstvy je zabránění přeučení. Nemá žádné parametry, které by se při trénování měnily.



Obrázek 2.8: Ukázka max poolingu s velikostí podoblasti 2x2 a posuvu rovno 2. Převzato z [3]



Obrázek 2.9: Model neuronu

Plně propojená vrstva

Jedná se o základní vrstvu neuronových sítí, kde všechny neurony dané vrstvy jsou napojené na všechny vstupy a výstupem je vektor, kde prvky odpovídají výstupům jednotlivých neuronů. Počet trénovatelných parametrů odpovídá následujícímu vzorci

$$p = n(d + 1), \tag{2.13}$$

kde n značí počet neuronů plně propojené vrstvy a d značí dimenzi vstupního vektoru

Batch Normalizace

Vrstva normalizuje hodnoty vstupů po dávkách, aby se výsledná průměrná hodnota rovnala 0 a rozptyl se rovnal 1. Normalizace se provádí pro každou dimenzi vstupu. Díky této vrstvě lze použít vyšší učící koeficient a snižuje kladený důraz správnou inicializaci počátečních vah. Podobně jako *dropout* vrstva snižuje přeučení, pomocí přidání šumu do vstupu, ale neztrácí se tolik informací, jak při použití *dropout* vrstvy. Čerpáno z [19].

2.4 Rekurentní neuronová síť (RNN)

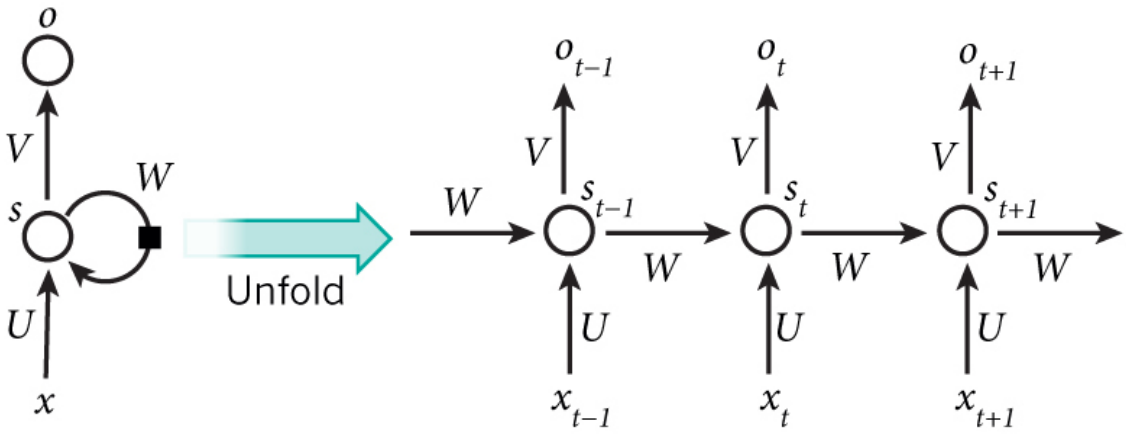
Rekurentní sítě [11, 15] jsou typy neuronových sítí, které mají cyklické propoje. Jejich hlavním účelem je, že umožňují pracovat se sekvencemi dat různých délek a vracet výstupy různých délek. Na rozdíl od dopředných sítí, kdy vstupy jsou na sobě nezávislé, využívá síť pro výpočet výstupu kromě aktuálního vstupu i interní stav, obsahující hodnotu získanou z předchozích vstupů v sekvenci. Mějme sekvenci $(x_1, x_2, \cdots, x_t)$, pak pro interní stav h_t platí

$$h_t = \begin{cases} 0 & \text{pro } t = 0, \\ f(h_{t-1}, x_t) & \text{pro } t \neq 0, \end{cases} \tag{2.14}$$

kde f značí nelineární funkci sigmoida nebo hyperbolický tangens. Úprava interního stavu se implementuje jako

$$h_t = f((Wx_t + Uh_{t1})), \tag{2.15}$$

kde f značí nelineární funkci, W a U značí váhy, x_t aktuální vstup a h_{t1} je předchozí stav. Výhodou rekurence je, že frekvenci úpravy vnitřního stavu lze ovlivnit pomocí vah, což umožňuje robustnost vůči zkresleným místům ve vstupních datech [17]. Schéma modelu



Obrázek 2.10: Ukázka rekurentní sítě po rozvinutí. Převzato z [6]

rekurentní sítě je znázorněno v 2.10. Při trénování se používá upravený algoritmus zpětného šíření nazývaný jako *backpropagation through time* [9], která také využívá řetězového pravidla. Nevýhodou samotné RNN jednotky je, že není schopna udržet vzdálené vazby, kvůli opakovanému maticovému násobení s maticí obsahující malé hodnoty. Tomuto jevu se říká *vanishing gradient*, kdy gradient po určitém počtu kroků zmizí. Tento problém byl však vyřešen pomocí LSTM bloku popsané v další části.

2.4.1 Long short-term memory (LSTM)

Existuje několik variant LSTM bloků, ale zde popsaná obsahuje paměť a 3 multiplikativní brány. Jedná se o vstupní, *forget* a výstupní bránu. Na rozdíl od běžných rekurentních jednotek, které přepisují svůj stav po každém časovém kroku, je LSTM blok schopen udržet si paměť pomocí bran a zároveň je schopen určit důležité příznaky ze vstupní sekvence velice brzy, což umožní uchovat potenciální vzdálené závislosti. Schéma LSTM vrstvy je na obrázku 2.11. Každý $j-$tý LSTM blok si uchovává hodnotu paměti c_t^j v čase t. Výstup bloku je pak roven

$$h_t^j = o_t^j tanh(c_t^j), \tag{2.16}$$

kde c_t^j značí paměť a o_t^j značí výstupní bránu, která určuje, co z paměti má projít na výstup. Vypočtu výstupní brány odpovídá

$$o_t^j = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t)^j, \tag{2.17}$$

kde σ je aktivační funkce sigmoida. Nový obsah paměťové buňky je vytvořen z části aktuálního obsahu, který nebyl zapomenut, a přidáním nového obsahu

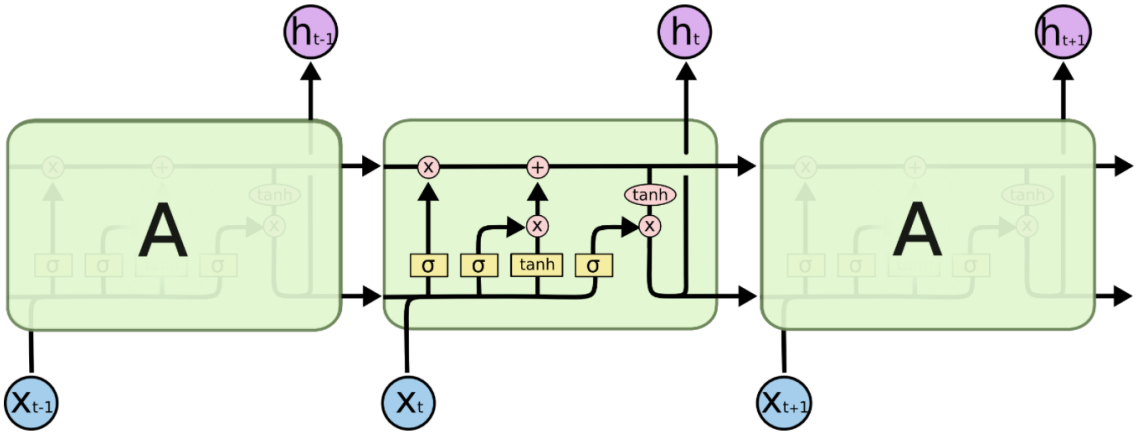
$$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j, \tag{2.18}$$

a pro nový obsah platí

$$\tilde{c}_t^j = tanh(W_c x_t + U_c h_{t-1})^j, \tag{2.19}$$

kde *tahn* značí aktivační funkci hyperbolický tangens. Množství zachované paměti je ovlivněno *forget* bránou, definovanou jako

$$f_t^j = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1})^j. \tag{2.20}$$



Obrázek 2.11: Ukázka LSTM. Převzato z [2]

a množství přidané paměti určuje vstupní brána, definovaná jako

$$i_t^j = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1})^j, \tag{2.21}$$

2.4.2 Obousměrné RNN (BRNN)

Jedná se o rekurentní vrstvu, která je rozdělena na dvě části tak, že jedna část vstup $(x_1, x_2, \cdots, x_t)$ zpracuje běžným způsobem od 1 do t a druhá bude mít opačné pořadí zpracování sekvence od t do 1. Výstupy jsou poté propojeny. Výstupní vrstva bude mít tedy přístup k budoucím a minulým stavům, což pro problematiku rozpoznávání textu je užitečné.

2.4.3 Connectionist temporal classification (CTC)

CTC [16, 15] je metoda pro zpracování výstupů rekurentní sítě. Metoda je vhodná pro sekvenční označování, kde není známo zarovnaní mezi vstupem a očekávaným výstupem a snaží se tedy zabránit nutnosti mapovat pro jednotlivé segmenty očekávaný cíl na trénovacích datech. Počet výstupů *softmax* vrstvy v neuronové síti odpovídá počtu označení včetně prázdného znaku a hodnoty vrstvy udávají pravděpodobnost výskytu daného označení nebo prázdné pozice v určitém čase v sekvenci. Mějme vstupní sekvenci x obsahující T časových kroků, rozšířenou množinu značení $L' = L \cup \{\epsilon\}$, kde ϵ značí prázdný znak, a pravděpodobnost označenou jako y_k^t, získanou z výstupu sítě pro označení k v čase t, pak pravděpodobnost řetězce π obsahující prvky z L' lze vyjádřit

$$p(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^t, \quad \forall \pi \in L'^T. \tag{2.22}$$

Nyní je ještě zapotřebí definovat mapování $L'^T \mapsto L'^{\leq T}$ označené jako $\mathcal{F}$, které odstraňuje duplicitní znaky, které stojí bezprostředně vedle, a prázdné znaky z řetězce. Např. platí, že

$$\mathcal{F}(aa - a - bb - -) = \mathcal{F}(aaa - -a - bbb) = aab. \tag{2.23}$$

Pro řetězec $l \in L^{\leq T}$ platí, že pravděpodobnost jeho výskytu se rovná součtu pravděpodobností všech možných π, které se na něj mapují. Lze vyjádřit jako

$$p(l|x) = \sum_{\pi \in \mathcal{F}^{-1}(l)} p(\pi|x). \tag{2.24}$$

Slučování různých cest na stejný řetězec je pro trénování CTC velice podstatné, protože umožňují sítím odhadnout znaky bez znalosti, kde se přesně nachází.

Zajímavostí je, že dříve metoda nepoužívala prázdný znak, což neumožňovalo mít několik stejných znaků vedle sebe, protože duplicitní znaky byly odstraněny. Dalším problémem bylo, že metoda musela neustále odhadovat znaky i v případech, kdy na vstupu byl šum, což způsobovalo zbytečnou zátěž na systém.

Výstupem klasifikátoru by měl nejpravděpodobnější řetězec odpovídající vstupní sekvenci

$$h(x) = \arg \max_{l \in L^{\leq T}} p(l|x). \tag{2.25}$$

Jako jedno z řešení pro získání výstupního řetězce bude zmíněn *best path decoding*. Pro každý časový krok vybírá nejpravděpodobnější znak a konkatenací vzniká výsledný řetězec. Jedná se o rychlou metodu, avšak výsledný řetězec nemusí být ten nejpravděpodobnější. Metoda lze vyjádřit jako

$$h(x) \approx \mathcal{F}(\pi^*),$$
$$\pi^* = \arg \max_{\pi \in N^t} p(\pi|x).$$

Chybová funkce je definována jako

$$\mathcal{L}(S) = -\sum_{(x,z) \in S} \ln p(z|x), \tag{2.26}$$

kde $p(z|x)$ značí pravděpodobnost správného označení všech trénovacích dat v datové sadě S.

Kapitola 3

Současná řešení

V současné době lze způsoby rozpoznávání rozdělit na 2 hlavní způsoby.

- Detekce všech znaků v registrační značce a pak následná identifikace každého znaku zvlášť.
- Z celé značky bez nutnosti segmentace se určí posloupnost znaků, kterou obsahuje.

3.1 Rozpoznávání za pomoci segmentace

Čerpáno z [30, 18]. Nevýhodou těchto způsobů je nutnost zajistit správnou segmentaci, která je navíc ovlivněná také binarizací obrázků. I v případech, kdy rozpoznávač jednotlivých znaků má vysokou přesnost, tak bez správné segmentace budou znaky špatně identifikovány. Pro následnou identifikaci znaků se používá porovnání se vzory [13] nebo strojové učení. Porovnání se vzory spočívá v tom určit, jak moc se vstup podobá šabloně a podle toho, která se nejvíc shodovala, bude pak znak podle ní identifikován. Nevýhodou ale je, že se nedokáže vypořádat s rotací nebo s neuceleným písmem. Metody založené na strojovém jsou schopny se s těmito problémy vypořádat. Jako jedním se zástupců lze zmínit např. neuronové sítě [26].
Způsoby segmentace lze dále rozdělit na 2 způsoby popsané v následujících sekcích.

3.1.1 Connected Component Analysis (CCA)

Prvním krokem je převod vstupního obrázku do binární podoby např. pomocí prahování, za účelem co nejvíce rozlišit znaky od pozadí. Metoda CCA [24, 12] v binarizovaném obrázku vyhledává propojené plochy na základě *4-connectivity*[1] nebo *8-connectivity* a v případech, že jsou tvořeny určeným minimálním počtem pixelů, budou považovány za plochy obsahující znak, jinak budou plochy odstraněny. Problémové mohou být ale znaky, které nejsou v obrázku spojité nebo v případech, kdy je vícero znaků propojeno.

3.1.2 Projection-based

Metoda Projection-based [27] vytváří pro sloupce a řádky registrační značky v binární podobě graf počtu černých pixelů nacházející se v daném řádku nebo sloupci. Z vertikální projekce lze pak pomocí počtu černých pixelů určit vertikální hranice jednotlivých znaků,

[1] http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/connect.html

protože z grafů pak lze nalézt volné pozice, protože oproti sloupcům obsahující znak, bude počet černých pixelů jiný. Podobně i pro horizontální projekci lze určit horní a dolní hranice znaků. Z takto získaných hranic lze určit pozice jednotlivých znaků. Problémové pro tuto metodu mohou být však rotace a zkreslení vstupu.

3.2 Holistické rozpoznávání

Holistická řešení přeskakují krok pro segmentaci a hlavní metody, které jsou zde zmíněny, jsou založené na neuronových sítích, díky kterým lze extrahovat významné příznaky, potřebné pro správnou klasifikaci obrázků registračních značek nízké kvality.

3.2.1 Sweeping OCR

V práci [7] je popsán postup, který využívá syntetická data, kde OCR klasifikátor prochází celou značku a vrací jednotlivé znaky a jejich pozice pomocí metody *probabilistic inference* založené na skrytých Markovových modelech a společně s jazykovým modelem určí nejpravděpodobnější přepis.

3.2.2 Sémantická segmentace

V práci [36] popisují postup, kde vytváří sémantickou mapu, která velikostně odpovídá vstupnímu obrázku a jednotlivé hodnoty v ní určují k jaké třídě znaku patří daný pixel. Sémantickou segmentaci implementovali pomocí konvoluční sítě a výstupem z ní se poté určí počáteční sekvence znaků. Poté následovala další síť, která ze vstupního obrázku, sémantické mapy a z počáteční sekvence určí počet jednotlivých znaků a tedy i výslednou sekvenci znaků. Obě zmíněné sítě se trénovaly zvlášť a jako modely použili pro znakovou klasifikaci *Inception-v3* a pro určení počtu znaků *AlexNet*. Síť pro klasifikaci znaků byla natrénovaná na syntetických značkách doplněné o reálné, které byly navíc augmentované. Trénování probíhalo porovnáním s referenční sémantickou mapou registrační značky, kde místo označovaní jednotlivých pixelů, využívají jednoduché značení, ve které určují obdélníkové hranice pro jednotlivé znaky, ve kterých všechny pixely uvnitř nabývají stejných hodnot, určují příslušnost k určitému znaku.

3.2.3 Konvoluční sítě s rozděleným výstupem

Jeden ze způsobů popsaných v pracích [30, 20], využívá konvoluční síť, kde pro celý výřez značky síť vrací příznaky, které jsou vstupem pro větve obsahující plně propojené vrstvy. Každá z těchto větví se pak naučí rozpoznávat znaky na určité pozici v registrační značce a výstupem je tedy vektor obsahující pravděpodobnosti jednotlivých znaků na dané pozici. K rozpoznávaným třídám znaků je potřeba přidat substituční znak za prázdnou pozici, pro případy kdy počet znaků v registrační značce je menší než počet větví. Maximální počet rozpoznávaných pozic v registrační značce tedy odpovídá počtu větví. Pro trénování se všechny výstupy porovnávají se znakem, který je převeden do kódu 1 z n[2], na odpovídající pozici z referenčního přepisu.
Podobný způsob je zmíněn i v práci [8], kde jako konvoluční síť využili model *VGGNet*[3] a jednotlivé větve jsou ještě propojené pomocí RNN. V práci byly natrénovany 2 varianty.

[2] https://en.wikipedia.org/wiki/One-hot
[3] https://neurohive.io/en/popular-networks/vgg16/

Jedna, která obsahovala jen CNN a druhá, obsahující zmíněnou kombinaci s RNN. Přesnosti rozpoznání s druhou sítí byly mnohem lepší a lišily se celkem o 40%. Z výsledků vyvodili, že pro vstupy, kterými jsou celé registrační značky, je přístup k celé sekvenci znaků velice podstatný. Výsledná přesnost se také zvýšila o celkem 7% po augmentaci dat.

3.2.4 Rekurentní sítě

V práci [18] je zmíněna architektura, skládající se z oddělené konvoluční a rekurentní podsítě. Vstupem pro konvoluční síť je sekvence výřezů, vzniklé posuvem okna nad obrázkem registrační značky, která má stejnou výšku jako dané okno. Příznaky z konvoluční a plně propojené mezivrstvy, které po konkatenaci, umožňují využít jak globální, tak i lokální příznaky jednotlivých výřezů, což by mělo vylepšit přesnost rozpoznávání. Po snížení dimenze a normalizaci konkatenovaných příznaků, jsou poté vstupem pro obousměrnou rekurentní síť, implementovanou pomocí LSTM a CTC, která bude schopna určit jednotlivé znaky v sekvenci. Konvoluční síť byla natrénovaná zvlášť na výřezech znaků a pak rekurentní síť na reálných a syntetických registračních značkách. Zajímavostí je i samotný vstup do sítě, kde místo obrázku v RGB, využívají 2 kanálový vstup, kde jedním z nich je šedotónový obrázek a druhý je získán metodou LBP [34], která nabízí robustnost vůči osvětlení. Tento přístup dosáhl *state-of-art* výsledků na datové sadě AOLP [10] .
Další přístup popsané v pracích [33, 29, 23] využívá architekturu, kde konvoluční a rekurentní sítě jsou propojené do jedné. Vstupem je celá značka a z ní extrahované příznaky, pomocí konvoluční sítě, je potřeba upravit na tvar (počet časových kroků, počet příznaků pro jednotlivý krok), aby byl vstup kompatibilní pro obousměrnou rekurentní síť, která využívá LSTM a CTC.

Kapitola 4

Návrh řešení a implementace

V této sekci bude popsán postup řešení a popis implementovaných sítí. Veškeré sítě byly implementovány tak, že rozeznávají 35 různých znaků, které leží v následující množině {A, B, C, D, E, F, G, H, I, J, K, L, M, N, P, Q,R , S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Vynecháno bylo písmeno O kvůli podobnosti s číslicí 0. Další vlastností testovacích a trénovacích dat je, že registrační značky jsou jednořádkové a mohou obsahovat 5 až 8 znaků. Sítě byly vytvořeny v jazyce *python* pomocí knihoven *Keras*[1] verze 2.2.4 a *TensorFlow*[2] verze 1.10.0 s možností trénování na grafické kartě. Trénování původně probíhalo přes *Colaboratory*[3], kde výhodou bylo, že nebylo potřeba instalovat žádné další knihovny a připojení disku s datovými sadami bylo jednoduché, ale hlavní nevýhodou byla příliš dlouhá doba načítání obrázků z datových sad. Poté mi byl poskytnut školní počítač s grafickou kartou GTX 980 s pamětí 4GB, na kterých pak probíhalo trénování sítí. Vyhodnocení sítí bylo provedeno na jiném počítači s procesorem Intel i5 3.5 GHz a grafickou kartou GTX 970 s pamětí 4 GB.

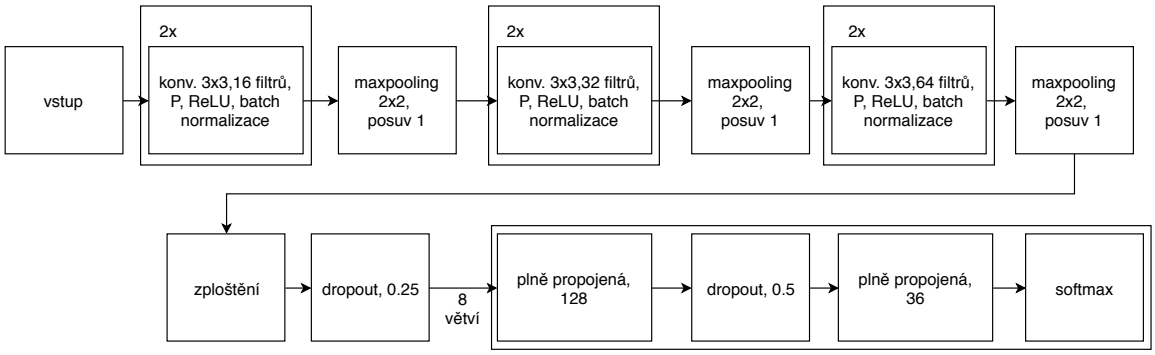
4.1 Srovnání současných řešení

Za účelem získat více zkušeností a srovnat vlastnosti jednotlivých implementací, jsem se rozhodl reimplementovat sítě z vybraných prací. Budou zde zmíněny 2 reimplementace, kde jedna z nich se skládá jen z konvoluční sítě a druhá využívá kombinaci konvoluční a rekurentní sítě.

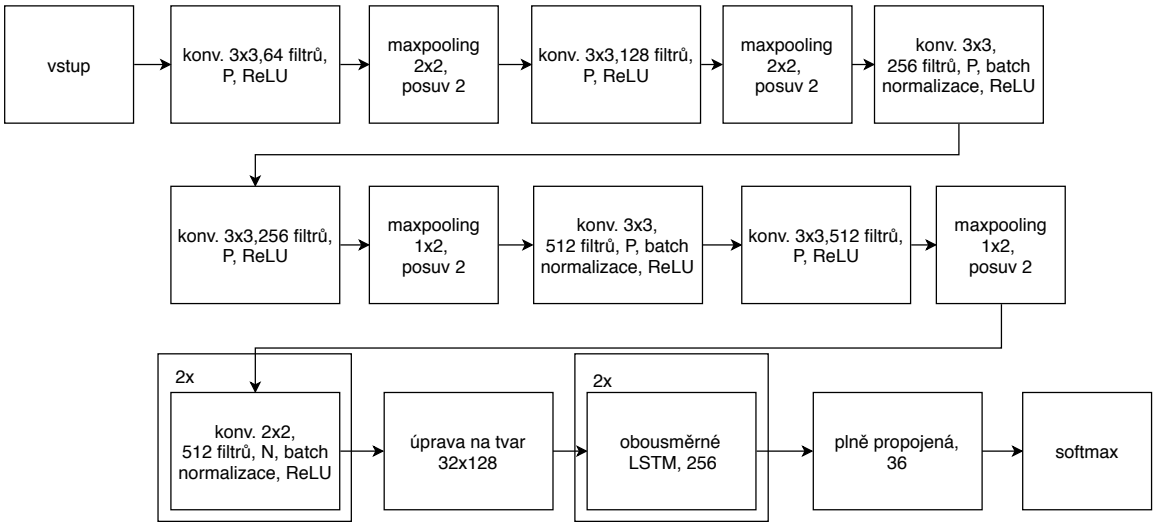
4.1.1 CNN s rozděleným výstupem

Jako první model pro implementaci jsem zvolil menší síť popsánou v práci *Holistic Recognition of Low Quality License Plates by CNN using Track Annotated Data* [30]. Rozdíl v přesnosti mezi sítěmi je minimální, ale rychlost zpracování obrazu je třikrát rychlejší. Síť jsem implementoval tak, že umožňuje identifikovat značky o maximální délce rovno 8. Součástí rozeznávaných znaků je ještě substituce za prázdný znak, v této implementaci se jedná o #. Pokud je značka kratší, doplňuje se # na čtvrtou pozici zprava, dokud výsledná délka nedosahuje 8. Do sítě byly ještě přidány vrstvy *dropout* do každé větve před poslední plně propojenou vrstvou s pravděpodobností zahazovaní 0.5 a před rozvětvením s pravděpodobností zahazování 0.25, které přesnost zvýšily, ale výsledná přesnost i přes to

[1] https://keras.io/
[2] https://www.tensorflow.org/
[3] https://colab.research.google.com/



Obrázek 4.1: Schéma CNN s rozděleným výstupem, kde P u konvoluční vrstvy značí zachování velikosti.



Obrázek 4.2: Schéma spojené CNN a BRNN, kde P u konvoluční vrstvy značí zachování velikosti a N možnou úpravu velikosti.

nedosahuje výsledků ve zmíněné práci, za což by mohla augmentace dat či jiné modifikace sítě. Vstupem do sítě je barevný obrázek s velikostí 200 x 40 a byla natrénovaná na 80 epochách s velikostí dávek 64. Schéma sítě je znázorněno na obrázku 4.1.

4.1.2 Spojená CNN a BRNN

Jako zástupcem těchto sítí jsem zvolil architekturu popsanou v práci [33]. Vstupem je obrázek o velikosti 160 x 48 a výstupem konvoluční sítě je celkem 4096 příznaků. V práci ale chyběly informace o počtu časových kroků, pro převod na tvar vhodný jako vstup do rekurentní sítě. Rozhodl jsem se proto natrénovat 2 verze, které upravily vstup na tvar 16 x 256 a 32 x 128, kde první číslo značí počet kroků. Porovnáním přesností bylo zjištěno, že druhá varianta dosáhla značně lepších výsledků. Síť byla natrénovaná na 60 epochách s velikostí dávek 32. Schéma sítě je znázorněno na obrázku 4.2.

4.2 Vlastní návrh

Cílem práce je vytvořit rozpoznávač registračních značek nízké kvality bez nutnosti segmentace. Z výsledků popsaných v práci [8] jsem chtěl, aby můj návrh využíval kombinaci konvoluční a rekurentní sítě, která se velice často objevovala. Původně jsem chtěl reimplementovat síť, která nevyužívá segmentaci, podle práce *Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs* [18] a popsané v sekci 3.2.4, ale výsledná reimplementace nekonvergovala. Proto jsem se rozhodl vytvořit návrh, který vychází z daného článku a spočívá ve spojení obou podsítí do jedné. Výhodami jsou, že půjde trénovat obě sítě zároveň pomocí celé registrační značky a není pak potřeba použít další datovou sadu s jednotlivými výřezy znaků. Možným negativem by pak mohla být časová a prostorová náročnost trénování.

Aby výsledná síť odpovídala síti v článku, je potřeba extrahovat příznaky pro jednotlivé výřezy registrační značky, které pak společně tvoří vstup pro rekurentní síť.

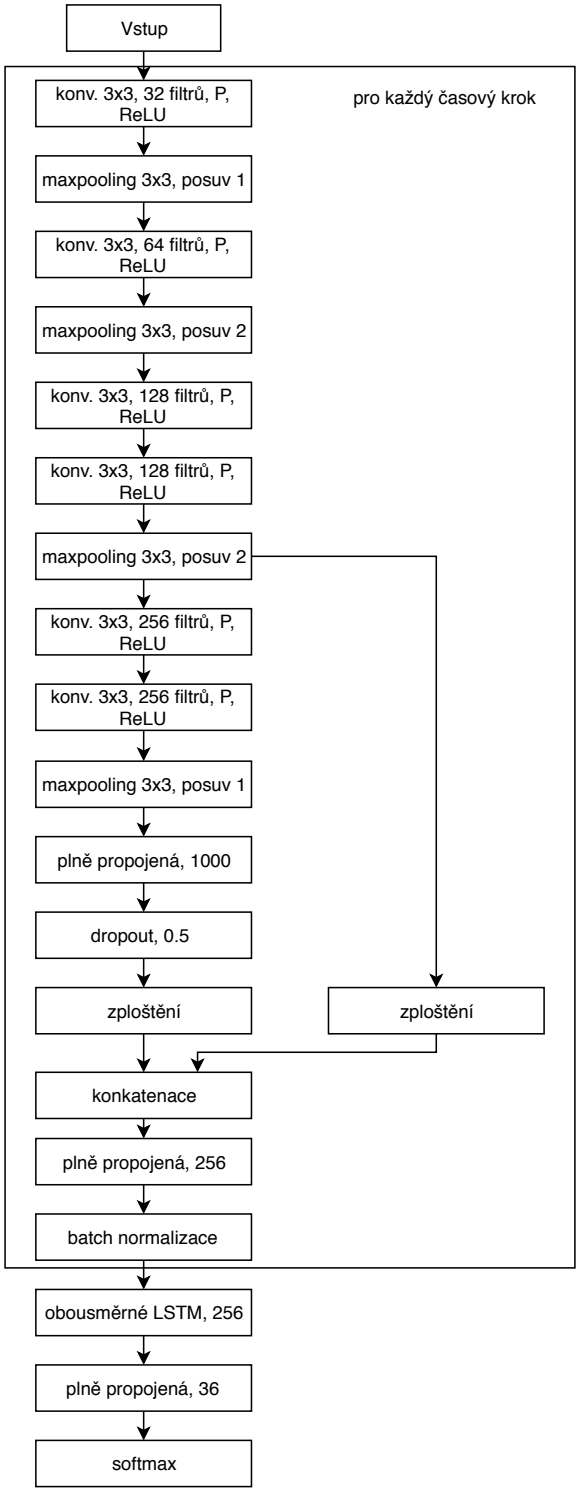
Následná úprava spočívala v tom, že místo běžného 3 rozměrného vstupu do sítě, bude vstup odpovídat tvaru (71, 24, 24, počet kanálů), kde první rozměr určuje počet výřezů, druhý a třetí určují velikosti jednotlivých výřezů. Výřezy jsou získávány průchodem posuvného okna o velikosti 24 x 24 nad obrázkem registrační po změně na velikost 94 x 24. Zajímalo mě dále, jestli bude mít vstup vliv na výslednou přesnost, a proto byly natrénovány 2 varianty s různým vstupem *grayscale* + LBP (tvořící 2 kanály), který použili v původním článku, a RGB (tvořící 3 kanály), který používá většina ostatních článků. Aby se extrakce prováděla pro každý výřez v sekvenci zvlášť, byla použita funkce, kterou nabízí *Keras*, označenou jako **TimeDistributed**[4]. Veškeré vrstvy, které jsou v ní zabaleny budou vracet odděleně výstupy pro každý prvek ve vstupní sekvenci.

Konkatenovaným příznakům z plně propojené a *maxpooling* vrstvy je nyní potřeba snížit dimenzi a provést následnou normalizaci. Toho je docíleno pomocí plně propojené vrstvy s 256 neurony a batch normalizace.

Vektor o velikosti 71 x 256, získaný redukcí dimenze, je pak vstupem pro obousměrné LSTM, které vrací výstup pro každý časový krok. Jako objektivní funkce bylo použito CTC.

Další úpravou bylo zredukování počtu filtrů na polovinu, protože s původním počtem filtrů byla síť příliš náročná na natrénování. Za optimalizátor byl ze začátku zvolen *stochastic gradient descent*, avšak síť nedosahovala dobrých výsledků, kvůli nutnosti zvolit vhodný učící koeficient. Z toho důvodu byla poté zvolena *Adadelta* [35], která byla použita i práci [33]. Výhodou je, že rychle konverguje a nepotřebuje nastavení některých hyperparametrů. Pro následné dekódování byla použita metoda *best path decoding*. Schéma sítě je znázorněno na obrázku 4.3. Hlavní rozdíly oproti síti popsané v sekci 4.1.2, je v počtu příznaků, které jsou vstupem do rekurentní sítě, kde rozdíl je více jak čtyřnásobný. Dalším rozdílem je, že síť extrahuje jak globální tak lokální příznaky pro každý výřez v sekvenci. Jako další variantu dané sítě, jsem zkoušel augmentaci dat a doplnění batch normalizací před každou *maxpooling* vrstvou. Augmentace otáčí snímky o 10 stupňů libovolným směrem a posouvá je o 0.1 délky stran do všech směrů. Sítě byly trénovány 50 epoch s velikostí dávek 16.

[4] https://keras.io/layers/wrappers/#timedistributed



Obrázek 4.3: Schéma návrhu, kde P u konvoluční vrstvy značí zachování velikosti.

Kapitola 5

Vyhodnocení návrhu

Kapitola obsahuje porovnání výsledků jednotlivých sítí a informace o použitých datových sadách. Následující sekce obsahuje informace o použitých datových sadách a o provedených úpravách.

5.1 Použité datové sady

Celkem byly použity 4 datové sady obsahující anotované výřezy s registrační značkou. Ukázky ze všech datových sad lze vidět na obrázku 5.2 a rozložení značek podle počtu znaků lze vidět v tabulce 5.1.

5.1.1 ReId

Pro trénování byla použita trénovací část datové sady **ReId** [30] obsahující 105924 obrázků. Datová sada byla získána z kamerových záběrů nad mosty simulující dohledovou kameru mýtných bran. Z 9.5 hodinového záznamu z 8 různých lokalit, se získaly výřezy, které byly poté ručně anotovány, pomocí webové aplikace. Následné vyhodnocení bylo provedeno na testovací části **ReId**, která obsahuje 76412 obrázků.

5.1.2 HDR

Pro vyhodnocení byla dále použita datová sada **HDR** [30], obsahující 652 obrázků, získané z digitální zrcadlovky, které pak byly ručně oříznuty a anotovány. Rozdílem od trénovacích dat je, že obsahuje i snímky s otočenými registračními značkami.

5.1.3 blur a deblur

Poslední dvě datové sady, které byly použity pro vyhodnocení, budou v této práci označeny jako **blur** a **deblur**. Jedná se o snímky převzaté z práce [32]. Datové sady obsahují šedotónové obrázky. Jedna z nich obsahuje rozmazané registrační značky a druhá obsahuje jejich nerozmazané podoby získané pomocí neuronové sítě. Z datových sad byly odstraněny značky, jejichž anotace obsahovala otazníky a přepisy obsahující písmeno O, byly upraveny, aby místo nich obsahovaly číslici 0. Dalším problémem byly samotné obrázky, které obsahovaly větší části nesouvisející s registrační značkou, a proto byly celkové výsledky nad danými daty špatné. Přesnost rozpoznávání však výrazně stoupla při upravení původních obrázků o konstantní velikosti 264 x 128 pomocí oříznutí, provedené při generování dávek

Obrázek 5.1: Ukázka úpravy obrázků z datové sady **blur**

pro testování, na 234 x 74. Ukázka způsobu oříznutí a výsledného snímku je znázorněna na obrázku 5.1.

5.2 Srovnání implementací

Jako metrika pro přesnost byl použit vzorec

$$\text{přesnost} = \frac{\text{počet správně rozpoznaných značek}}{\text{celkový počet v datové sadě}}, \tag{5.1}$$

a pro změření rychlostí byla naměřena průměrná rychlost na 40 dávkách o 16 snímcích v milisekundách. Součástí měřeného času bylo získání výstupu ze sítě a následné dekódování, pro získání přepisu. Měření probíhalo zároveň na grafické kartě a na procesoru, popsané v kapitole 4. Výsledky měření jsou v tabulce 5.4.

V tabulce 5.3 byly doplněny přesnosti nad datovými sadami **ReId**, **HDR**, **blur** a **deblur** pro *OpenALPR*[1] a *UnicamLPR*[2], převzaté z [30]. Je ale potřeba zmínit, že hodnoty převzatých přesností nemusí odpovídat kvůli možným rozdílům testovacích dat, protože jejich úpravy obrázků, provedené nad datovými sadami **blur** a **deblur**, nemusí být totožné s použitými v této práci, ale pro ostatní by měly být identické, protože nebyly provedeny jiné změny než úpravy velikostí obrázků. V tabulce 5.2 lze vidět počet chybně identifikovaných značek, rozdělené podle počtu znaků ve značce.

Sítě na datové sadě **ReId** dosahovaly dobrých výsledků a navzájem se od sebe příliš neliší, díky podobnosti testovacích dat s trénovacími. Na datové sadě **HDR** dosáhly lepších výsledků obě varianty vlastního návrhu, oproti oběma reimplementacím, které se výsledkem příliš nelišily. Hlavní rozdíl se ale projevil na datové sadě **blur**, kde rekurentní sítě dosáhly lepších výsledků. Rozdíly mezi variantami vlastního návrhu byly minimální, ale zajímavé jsou výsledky nad neořezanými značkami z datových sad **blur** a **deblur**, kde obě varianty byly schopny některé značky rozpoznat a kombinace *grayscale*+LBP dosáhla dokonce 54% na druhé datové sadě. Všechny implementace dosáhly lepších výsledků nad všemi datovými než *OpenALPR* a *UnicamLPR*, i přes odlišnosti od trénovacích dat.

5.2.1 Celkové zhodnocení

Ke každé datové sadě byla vytvořena tabulka 5.5, 5.6, 5.7 a 5.8 s náhodně vybranými obrázky a s odpovídajícími přepisy ze všech implementovaných sítí. Nejlepšího výsledků dosáhla síť vycházející z práce [18], ale při porovnání s výsledky větší sítě, popsané v práci [30], tak dosáhla lepších výsledků jen na datové sadě **blur**. Síť jsem poté zkoušel vylepšit pomocí augmentace dat a přidáním batch normalizací, ale výsledkem bylo zlepšení o pouhá 2 procenta nad **blur**, ale došlo ke značnému snížení přesností nad všemi ostatními datovými

[1] https://www.openalpr.com/
[2] https://www.camea.cz/cz/

Obrázek 5.2: Náhodně vybrané ukázky z datových sad **ReId**, **HDR**, **blur** a **deblur**

sadami. Za zhoršením výsledku by mohla být špatná volba augmentačních parametrů nebo dokonce i chybné použití augmentace.

Tabulka 5.1: Tabulka s počtem registračních značek v každé datové sadě, rozdělené podle počtu znaků

datove sady	**5**	**6**	**7**	**8**
ReId trén.	53	208	105410	253
ReId test.	0	126	76032	254
HDR	4	2	646	0
blur	1	6	701	3
deblur	1	6	701	3
celkem test.	6	140	78080	260

Tabulka 5.2: Tabulka s počtem chybných rozeznání, rozdělené podle počtu znaků ve značce

datové sady		**Hol. CNN**	**CNN + BRNN**	**vl. ná. (LBP)**	**vl. ná. (RGB)**
ReId	5	-	-	-	-
	6	63	56	52	59
	7	1486	1541	1240	1214
	8	183	177	166	154
Hdr	5	4	4	1	1
	6	2	2	0	2
	7	132	138	97	92
	8	-	-	-	-
blur	5	1	1	0	1
	6	6	4	4	4
	7	364	302	245	252
	8	3	2	3	3
deblur	5	1	1	1	1
	6	6	4	5	4
	7	95	108	64	56
	8	3	2	3	3

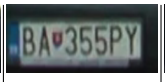
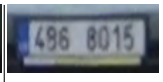
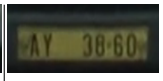
Tabulka 5.3: Tabulka přesností jednotlivých sítí na datových sadách v procentech. Datové sady s hvězdičkou značí přesnosti rozpoznávání bez ořezávání. Hodnoty pro *OpenALPR* a *UnicamLPR* byly převzaty z [30]

systémy	**ReId**	**HDR**	**blur**	**deblur**	**blur***	**deblur***
OpenALPR	34.0	40	1.0	69.6	–	–
UnicamLPR	70.0	31.9	38.3	83.3	–	–
Holistic CNN [30]	97.7	78.8	47.3	85.2	1.5	3.3
CNN + BRNN [33]	97.6	77.9	56.5	83.8	7.4	10.5
vlastní návrh (RGB)	98.1	84.4	63.4	89.7	24.5	42.9
vlastní návrh (LBP)	98.1	84.3	64.5	89.7	28.9	54.2

Tabulka 5.4: Naměřené rychlosti rozpoznávání v ms

Holistic CNN	**CNN + BRNN**	**vlastní návrh (RGB)**	**vlastní návrh (LBP)**
5.06	9.60	13, 55	13, 69

Tabulka 5.5: Tabulka s ukázkami z **ReID**

obrázek					
skutečný přepis	B0L4750	BA355PY	RUGAB180	NLM222	4B68015
Holistic CNN	B0L4750	BA355EY	RU6AB810	WL4222	4B68015
CNN + BRNN	8B0L4750	BA355PY	8UGAA10	NX222	4B68015
vlastní návrh (RGB)	B0L4750	BA355PY	RUGAB180	NM222	4B68015
vlastní návrh (LBP)	B0L4750	BA355PY	RUGAB10	NLM222	4B68015
obrázek					
skutečný přepis	NDK745	BZC0683	0FDN2400	2B64065	AY3860
Holistic CNN	TDK745	BZC0683	0ED2200	2B64065	4Y3860
CNN + BRNN	1BZN1085	BZC0883	0FJ2400	2B64065	AVY3860
vlastní návrh (RGB)	NDK745	BZC0683	03FDN2400	2B64065	VY3860
vlastní návrh (LBP)	NDK745	BZC0683	0DN2400	2B64055	Y3860

Tabulka 5.6: Tabulka s ukázkami z **HDR**

obrázek					
skutečný přepis	BZN1065	BMZ3933	6B84949	B13AEB	9B33297
Holistic CNN	BZN1065	BB73930	6B89444	BB3AE5S	9B33297
CNN + BRNN	1BZN1085	BMZ3933	6B84949	BAED	9B33297
vlastní návrh (RGB)	BZN105	BM3933	6B84949	BBEB	9B33297
vlastní návrh (LBP)	BZN1019	BM3933	6B8949	B13AEB	9B33297
obrázek					
skutečný přepis	4B10036	B8121	BMZ3933	5C65578	BZD1223
Holistic CNN	4B10026	8B8121	BMZ3933	5C65578	4B9000
CNN + BRNN	4B10006	8B8121	5S2880	5C65578	5B90046
vlastní návrh (RGB)	4B10006	B8121	5B2883	5C65578	B700U
vlastní návrh (LBP)	4B10036	B8121	5B28893	5C625578	B0

Tabulka 5.7: Tabulka s ukázkami z **blur**

obrázek					
skutečný přepis	AKE5751	KJA652	AKR9508	A6670	3U42123
Holistic CNN	4KE5751	KJ4652	4KR9508	4AI6670	3U42123
CNN + BRNN	AAKE5751	KJA652	AAK9508	4AI6670	3U42123
vlastní návrh (RGB)	KE5751	KA652	AKR9508	6670	3U42123
vlastní návrh (LBP)	KE5751	KJA652	AR9508	A6670	3U42123
obrázek					
skutečný přepis	NGW591	6A23733	BH4301AA	ZH238BT	CA2548CT
Holistic CNN	CAC4591	6A23733	WH49144	Z02386T	GA846T
CNN + BRNN	WM591	6AZ3733	BH4301AA	7H238BT	CB234C0
vlastní návrh (RGB)	NGW591	6A23733	B4301AA	7H23BBT	K2564CY
vlastní návrh (LBP)	NGW591	6A23733	B4301AAA	7B238BT	SA2566C19

Tabulka 5.8: Tabulka s ukázkami z **deblur**

obrázek					
skutečný přepis	AKE5751	KJA652	AKR9508	A6670	3U42123
Holistic CNN	4KE5751	KR4652	BKR9508	4AI6670	3U42123
CNN + BRNN	AKE5751	KJA652	AK9508	4AI6670	3U42123
vlastní návrh (RGB)	AKE5751	KJA652	AKR9508	6670	3U42123
vlastní návrh (LBP)	K5751	KA652	A9508	A66670	3U42123
obrázek					
skutečný přepis	NGW591	6A23733	BH4301AA	ZH238BT	CA2548CT
Holistic CNN	DNC4591	6A23733	ZH2385T	Z02386T	CA2846T
CNN + BRNN	NGN591	6A23733	BH4301AA	ZH238BT	CA584GT
vlastní návrh (RGB)	NG591	6A23733	BM4301AA	ZH238BT	CA2584CT
vlastní návrh (LBP)	NGW591	6A23733	B4301AA	ZH238BT	CA2584CT

Kapitola 6

Závěr

Cílem práce bylo vytvořit neuronovou síť pro rozpoznávání značek ze snímků nízké kvality, se snahou dosáhnout co nejvyšší přesnosti. Zaměřil jsem se na vytvoření návrhu kombinující konvoluční a rekurentní sítě, vycházející z práce [18], kterou jsem poté natrénoval na reálných snímcích registračních značek. V práci byly popsány různé metody rozpoznání a některé byly i reimplementovány a poté vyhodnoceny. Natrénovány byly celkem 2 varianty navrhované sítě s různými vstupy, které dosáhly podobných výsledků na všech testových datových sadách. Kombinace *grayscale* a LBP dosáhla přesnosti rozpoznávání na datové sadě **ReId** 98.1%, na **HDR** dosáhla 84.3%, na **blur** dosáhla 63.4% a na **deblur** dosáhla 89.7% a celková přesnost tedy byla 97.6%.
Přesnost navrhované sítě by dále šla zkusit vylepšit pomocí natrénování s větším počtem filtrů a augmentací dat. Samotná práce by pak šla dále rozšířit implementací další sítě kombinující konvoluční sít s rozděleným výstupem a rekurentní sítě, podobně popsanou v práci [8].

Literatura

[1] Obrázek - konvoluce. https://blog.francium.tech/machine-learning-convolution-for-image-processing-42623c8dbec0.

[2] Obrázek - LSTM. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[3] Obrázek - maxpooling. http://cs231n.github.io/convolutional-networks/.

[4] Obrázek - neuronová síť. http://cs231n.github.io/convolutional-networks/.

[5] Obrázek - registrační značka. https://www.zakonyprolidi.cz/cs/2014-343.

[6] Obrázek - rekurentní síť. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[7] Bulan, O.; Kozitsky, V.; Ramesh, P.; aj.: Segmentation- and Annotation-Free License Plate Recognition With Deep Localization and Failure Identification. *IEEE Transactions on Intelligent Transportation Systems*, ročník 18, č. 9, Sep. 2017: s. 2351–2363, ISSN 1524-9050, doi:10.1109/TITS.2016.2639020.

[8] Cheang, T. K.; Chong, Y. S.; Tay, Y. H.: Segmentation-free Vehicle License Plate Recognition using ConvNet-RNN. *CoRR*, ročník abs/1701.06439, 2017, 1701.06439. URL http://arxiv.org/abs/1701.06439

[9] Chen, G.: A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation. *CoRR*, ročník abs/1610.02583, 2016, 1610.02583. URL http://arxiv.org/abs/1610.02583

[10] Chowdhury, S.; Das, A.; Punitha, P.: Projection Profile Based Number Plate Localization and Recognition. 05 2016, s. 185–200, doi:10.5121/csit.2016.60615.

[11] Chung, J.; Gulcehre, C.; Cho, K.; aj.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. 12 2014.

[12] Giannoukos, I.; Anagnostopoulos, C.-N.; Loumos, V.; aj.: Operator context scanning to support high segmentation rates for real time license plate recognition. *Pattern Recognition*, ročník 43, 11 2010: s. 3866–3878, doi:10.1016/j.patcog.2010.06.008.

[13] Goel, S.; Dabas, S.: Vehicle registration plate recognition system using template matching. In *2013 INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING AND COMMUNICATION (ICSC)*, Dec 2013, s. 315–318, doi:10.1109/ICSPCom.2013.6719804.

[14] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[15] Graves, A.: *Supervised Sequence Labelling with Recurrent Neural Networks*. 2011.

[16] Graves, A.; Fernández, S.; Gomez, F.; aj.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, s. 369–376.

[17] Graves, A.; Liwicki, M.; Fernández, S.; aj.: A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 31, č. 5, May 2009: s. 855–868, ISSN 0162-8828, doi:10.1109/TPAMI.2008.137.

[18] Hui Li, Chunhua Shen: Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs. Jan 2016.

[19] Ioffe, S.; Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, ročník abs/1502.03167, 2015, 1502.03167.
URL http://arxiv.org/abs/1502.03167

[20] Jain, V.; Sasindran, Z.; Rajagopal, A.; aj.: Deep automatic license plate recognition system. 12 2016, s. 1–8, doi:10.1145/3009977.3010052.

[21] Karpathy, A.: "Neural Networks Part 1: Setting Up the Architecture." Notes for CS231n Convolutional Neural Networks for Visual Recognition. Stanford University, http://cs231n.github.io/, zobrazeno 18.4.2019.

[22] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012, s. 1097–1105.

[23] Li, H.; Wang, P.; Shen, C.: Toward End-to-End Car License Plate Detection and Recognition With Deep Neural Networks. *IEEE Transactions on Intelligent Transportation Systems*, ročník 20, č. 3, March 2019: s. 1126–1136, ISSN 1524-9050, doi:10.1109/TITS.2018.2847291.

[24] Llorens, D.; Marzal, A.; Palazón, V.; aj.: Car license plates extraction and recognition based on connected components analysis and HMM decoding. In *Iberian Conference on Pattern Recognition and Image Analysis*, Springer, 2005, s. 571–578.

[25] Nwankpa, C.; Ijomah, W.; Gachagan, A.; aj.: Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *CoRR*, ročník abs/1811.03378, 2018, 1811.03378.
URL http://arxiv.org/abs/1811.03378

[26] Oz, C.: Vehicle Licence Plate Recognition Using Artificial neural Networks. 01 2001.

[27] Rahman, C. A.; Badawy, W.; Radmanesh, A.: A real time vehicle's license plate recognition system. In *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.*, July 2003, s. 163–166, doi:10.1109/AVSS.2003.1217917.

[28] Ruder, S.: An overview of gradient descent optimization algorithms. 2016, 1609.04747.

[29] Shivakumara, P.; Tang, D.; Asadzadehkaljahi, M.; aj.: CNN-RNN based method for license plate recognition. *CAAI Transactions on Intelligence Technology*, ročník 3, 07 2018, doi:10.1049/trit.2018.1015.

[30] Špaňhel, J.; Sochor, J.; Juránek, R.; aj.: Holistic recognition of low quality license plates by CNN using track annotated data. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, Aug 2017, ISBN 978-1-5386-2939-0, s. 1–6, doi:10.1109/AVSS.2017.8078501.

[31] Srivastava, N.; Hinton, G.; Krizhevsky, A.; aj.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, ročník 15, č. 1, Leden 2014: s. 1929–1958, ISSN 1532-4435.
URL http://dl.acm.org/citation.cfm?id=2627435.2670313

[32] Svoboda, P.; Hradis, M.; Marsik, L.; aj.: CNN for License Plate Motion Deblurring. 2016, 1602.07873.

[33] Wang, X.; You, M.; Shen, C.: Adversarial Generation of Training Examples for Vehicle License Plate Recognition. *CoRR*, ročník abs/1707.03124, 2017, 1707.03124.
URL http://arxiv.org/abs/1707.03124

[34] Xiaoxuan Chen; Qi, C.: A super-resolution method for recognition of license plate character using LBP and RBF. In *2011 IEEE International Workshop on Machine Learning for Signal Processing*, Sep. 2011, ISSN 2378-928X, s. 1–5, doi:10.1109/MLSP.2011.6064550.

[35] Zeiler, M. D.: ADADELTA: An Adaptive Learning Rate Method. *CoRR*, ročník abs/1212.5701, 2012, 1212.5701.
URL http://arxiv.org/abs/1212.5701

[36] Zhuang, J.; Hou, S.; Wang, Z.; aj.: Towards Human-Level License Plate Recognition. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Příloha A

Paměťové médium

Médium obsahuje složky:

src Obsahuje zdrojové kódy.

add Obsahuje video a plakát.

doc Obsahuje PDF soubor práce.