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

ZLEPŠOVÁNÍ KVALITY DIGITALIZOVANÝCH TEXTOVÝCH DOKUMENTŮ

DOCUMENT QUALITY ENHANCEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JAN TRČKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ROMAN JURÁNEK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Trčka Jan, Bc.**
Program: Informační technologie Obor: Inteligentní systémy
Název: **Zlepšování kvality digitalizovaných textových dokumentů**
Document Quality Enhancement
Kategorie: Zpracování obrazu
Zadání:

1. Prostudujte základy konvolučních sítí a generativních sítí.
2. Vytvořte si přehled o současných metodách zlepšování kvality obrazu a převodu obrazu mezi různými doménami pomocí neuronových sítí.
3. Vyberte nebo navrhněte metodu aplikovatelnou na zlepšování kvality textových dokumentů.
4. Obstarejte si databázi vhodnou pro experimenty.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte prezentační materiály demonstrující výsledky vaší práce.

Literatura:

- Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros: Image-to-Image Translation with Conditional Adversarial Networks, arXiv:1611.07004.
- Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, in IEEE International Conference on Computer Vision (ICCV), 2017.
- HRADIŠ Michal, KOTERA Jan, ZEMČÍK Pavel a ŠROUBEK Filip. Convolutional Neural Networks for Direct Text Deblurring. In: Proceedings of BMVC 2015, Swansea, The British Machine Vision Association and Society for Pattern Recognition, 2015.
- Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: A Neural Algorithm of Artistic Style, arXiv:1508.06576, 2015.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Juránek Roman, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Cílem této práce je zvýšení úspěšnosti při rozpoznávání textových dokumentů. Práce je zaměřena především na texty nacházející se na degradovaném materiálu jako jsou noviny nebo staré knihy. K řešení tohoto problému jsou analyzovány současné metody a problémy spojené s rozpoznáváním textu. Na základě získaných poznatků je zvolena implementovaná metoda založena na GAN sítích. Na těchto sítích jsou provedeny experimenty pro nalezení jejich vhodné velikosti a parametrů učení. Následně je provedeno testování pro porovnání různých metod učení a srovnání jejich výsledků. Trénování a testování je provedeno na umělém datovém setu, u kterého se zvýší přesnost přepisu z 65.61 % pro nezpracované řádky textu na 93.23 % u řádků zpracovaných sítí GAN.

Abstract

The aim of this work is to increase the accuracy of the transcription of text documents. This work is mainly focused on texts printed on degraded materials such as newspapers or old books. To solve this problem, the current method and problems associated with text recognition are analyzed. Based on the acquired knowledge, the implemented method based on GAN network architecture is chosen. Experiments are a performer on these networks in order to find their appropriate size and their learning parameters. Subsequently, testing is performed to compare different learning methods and compare their results. Both training and testing is a performer on an artificial data set. Using implemented trained networks increases the transcription accuracy from 65.61 % for the raw damaged text lines to 93.23 % for lines processed by this network.

Klíčová slova

Neuronové sítě, hluboké neuronové sítě, konvoluční neuronové sítě, GAN sítě, TensorFlow, zlepšování kvality obrazu

Keywords

Neural networks, deep neural networks, convolution neural networks, GAN networks, TensorFlow, image quality enhancement

Citace

TRČKA, Jan. *Zlepšování kvality digitalizovaných textových dokumentů.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Roman Juránek, Ph.D.

Zlepšování kvality digitalizovaných textových dokumentů

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Ing. Romana Juránka, Ph.D. Další informace mi poskytl pan Ing. Michal Hradiš, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Trčka
3. června 2020

Obsah

Kapitola 1

Úvod

Ke sběru informací docházelo již odpradávna, avšak až příchod počítačů v minulém století toto shromažďování odstartoval v masovém měřítku. Vše bylo ještě umocněno internetovým připojením, které umožnilo tyto informace jednoduše sdílet mezi lidmi a institucemi. Stejně tak byly počítače stále více využívány k automatizaci lidských úkolů, které mohou provádět mnohonásobně rychleji. Tento přechod ovšem není vždy zcela jednoduchý, neboť problémy, pro člověka jednoduché až triviální, nemusí být jednoduše implementovatelné.

S postupným nárůstem výpočetního výkonu začalo docházet k většímu výzkumu v oblasti strojového učení, neuronových sítí a umělé inteligence jako takové. Vše spočívá ve vytvoření programu, jehož inspirací je fungování lidského mozku, avšak s konkrétním cílem k řešení. Na rozdíl od běžných programů v tomto případě není poskytnut návod, jak k danému výsledku dojít. K požadovaným výsledkům se tyto programy musí dopracovat postupným získáváním zkušeností. Naučí-li se programy provádět jednoduché činnosti s uspokojivými výsledky, je možné je následně kombinovat do složitějších celků k řešení problematičtějších úloh. Tento přístup kombinace jednodušších konceptů do složitějších a jejich vzájemné propojení se v umělé inteligenci označuje jako hluboké učení.

Neuronové sítě a text v obraze jsou v dnešní době převážně známé ve spojitosti s lokalizací těchto textů a jejich následná transkripce. Tento proces je znám jako proces pro optické rozpoznávání znaků či OCR proces (z anglického Optical Character Recognition). S postupnou dostupností výkonných počítačových zařízení v běžném životě vzrůstá i uplatnění této úlohy. Může se jednat o přepis textu snímaného mobilním zařízením a jeho následném překladu do různých jazyků nebo v automatickém kategorizování digitalizovaných dokumentů na základě jejich obsahu.

Tato práce se zabývá principy fungování neuronových sítí, konvolučních sítí a GAN sítí ve spojitosti se zlepšováním kvality textových obrazů a jejich převodu mezi různými doménami. Konkrétně se práce zaměřuje na problematiku zlepšení čitelnosti textových dokumentů, kde výstup bude opět v obrazové podobě a může být následně využit pro další zpracování. V následující kapitole jsou uvedeny základní informace o neuronových sítích, v krátkosti jsou zmíněny problémy, jaké se jimi řeší, možné způsoby učení sítí a jejich variace. Podrobněji jsou popsány konvoluční neuronové sítě a GAN sítě, které jsou v rámci této práce využity. V kapitole 3 je představen problém rozpoznávání textu a jednotlivé části, které tento problém řeší s detailnějším zaměřením na předzpracování obrazu. V podkapitole 3.2 jsou představeny existující metody zabývající se problémem zlepšení kvality obrazu. Na základě poznatků z těchto existujících metod je v kapitole 4 navrhnuta architektura sítě, která bude implementována a následně testována, a data, která k tomuto účelu budou použita. Kapitola 5 obsahuje stručné informace o použitých nástrojích a implemen-

tačních prostředích, na kterých byl projekt testován. Součástí této kapitoly je i přehled implementovaných souborů a jejich účel. V kapitole 6 jsou popsány dvě sady provedených experimentů a jejich vyhodnocení.

Kapitola 2

Strojové učení

V této kapitole je představena problematika strojového učení, společně s vybranými kategoriemi problémů, které se tímto způsobem dají řešit, a jejich stručný popis. Následně je podrobněji popsána základní struktura umělých neuronových sítí, aktivační funkce, které se v nich používají, a způsoby učení sítí. V podkapitole 2.2 je představena třída neuronových sítí, která se nejčastěji využívá ke zpracování obrazu – tzv. konvoluční neuronové sítě. V závěrečné podkapitole 2.3 je popsána architektura GAN (z anglického pojmenování generative adversarial network) sítí, její výhody a problémy.

Existuje mnoho úkolů, které jsou snadno řešitelné lidmi, avšak z mnoha důvodů, ať již kvůli časové či prostorové náročnosti, je vhodnější zvolit řešení pomocí počítačů. Ovšem ne vždy jsou tyto úlohy snadno převoditelné na konkrétní programy nebo je jejich převod značně komplikovaný. Velká variace rozhodnutí na základě vstupních dat či i pouhá neznalost všech informací daného tématu může vývoj značně zkomplikovat. Chybné výsledky špatně navrhnutých a implementovaných aplikací můžou způsobit vážné komplikace. Proto jsou tyto úlohy vhodným uchazečem pro řešení pomocí metod strojového učení.

V těchto případech je cílem vytvořit program se specifickým formátem vstupních dat. Ten je obvykle definován jako jednorozměrný vektor či n-rozměrná matice. Tato vstupní data jsou často již předzpracována a normalizována. Následně se specifikuje formát dat výstupních a struktura učícího se stroje. Tento stroj může mít mnoho různých podob, a to jako uzavřený nebo otevřený systém s agenty či neuronovými sítěmi. Tento systém se sám snaží naučit, jak daný problém řešit na základě definovaných pravidel a zvoleného způsobu učení. Důležitou vlastností takto učeného stroje je, aby i po jeho zastavení a následném spuštění nedocházelo ke ztrátě naučeného procesu a aby jeho činnost byla opakovatelná se stejnými nebo lepšími výsledky.

Nejčastější třídy úloh řešené pomocí umělých neuronových sítí a jejich problematika jsou:

- **Klasifikace**
  Úlohy tohoto typu obvykle mají za cíl zařadit prvky vstupní množiny dat do jedné z k kategorií. Dalším možným výstupem může být i procentuální příslušnost do jednotlivých tříd s případným následným zpracováním těchto výstupních hodnot. Zpravidla se jedná o úlohy založené na tzv. učení s učitelem (supervised learning). Trénování neuronové sítě probíhá na datech, která mají správně přiřazenou kategorii, a tedy se ví, jakých výsledků se má v rámci trénovací a testovací množiny docílit.

- **Shluková analýza**
  Shluková analýza se snaží řešit podobný problém jako klasifikace, avšak s tím rozdí-

lem, že rozdělení do jednotlivých tříd není předem známo. Vše probíhá na principu tzv. učení bez učitele (unsupervised learning), při kterém se neuronová síť sama snaží rozpoznat, která data patří do stejných kategorií. Toto rozdělení probíhá na základě podobnosti jednotlivých dat, ať už se jedná o vzdálenosti numerických hodnot, nebo stejných či podobných vlastností mezi jednotlivými daty.

- **Strojový překlad**
  Obvykle se jedná o překlad textu z jednoho jazyka do jiného. Tradiční překlad slovních spojení je založen na rozdělení textu na mnoho menších komponent, které jsou překládány odděleně. Naopak, překlady neuronovými sítěmi se snaží o nalezení větných vazeb a docílit uspokojivějších výsledků, než kdyby se překládala jednotlivá slova odděleně [4].

- **Transkripce**
  Programy s tímto úkolem obvykle převádějí informace z jedné datové domény do jiné. Toto má aplikaci v mnoha různých směrech, ať už při digitalizaci tištěných nebo ručně psaných textů (OCR – Optical Character Recognition), hlasových zvukových vln do jejich textové podoby, ale i kupříkladu převod hudebních not do jejich zvukové podoby. Tyto převody mohou fungovat všemi směry, tedy i převod textu do jeho zvukové podoby.

- **Detekce objektů**
  Cílem této úlohy je rozpoznání a lokalizace objektů ve vstupních obrazových datech. Jakožto jeden ze základních problémů počítačového vidění detekce objektů může poskytnout důležité informace pro hlubší porozumění obrázků a videí. Tato kategorie je často spojována s problémy zabývající se obrazovou klasifikací, analýzou lidského chování, rozpoznávání obličejů nebo autonomními vozidly [34].

2.1 Umělé neuronové sítě

„Umělá neuronová síť je tvořena matematickými neurony, primitivními jednotkami, kde každá zpracovává váhované vstupní signály a generuje výstup [15].“ Hlavní myšlenka za takto vytvořenými programy je pokus o simulaci činnosti lidského mozku s určitou úlohou k řešení.

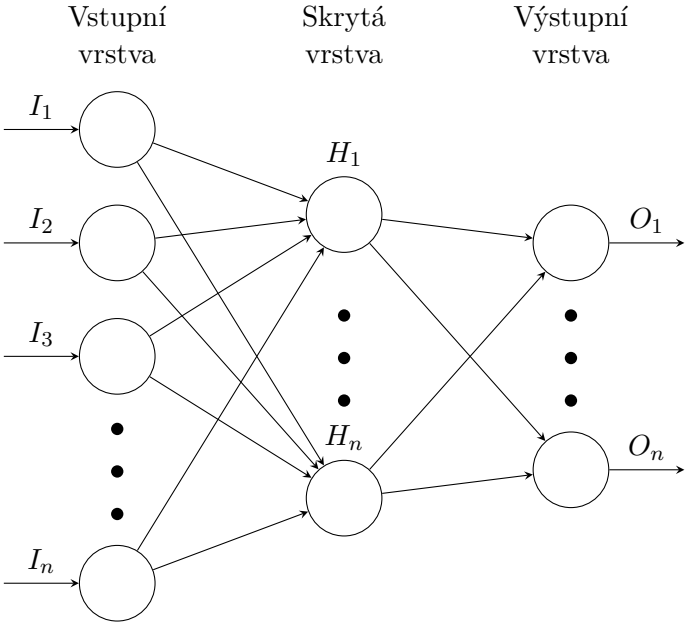
Jednotlivé neurony sítě pracují samostatně. Na základě obdržených hodnot vstupů vygenerují vlastní výstupní hodnotu, a to bez znalosti chování jiných neuronů a hodnot v síti. Chování každého neuronu je založeno na shromáždění hodnot ze svých n vstupů. Tyto hodnoty tvoří vstupní vektor neuronu $\vec{n} = (x_1, x_2, \ldots, x_n)$. Tento vektor hodnot je dále ovlivněn váhovým ohodnocením příslušných hran, ovlivňujících míru dopadu dané hodnoty na výpočet výstupní hodnoty neuronu. Toto váhové ohodnocení je přiřazeno vstupním hranám neuronu a tvoří váhový vektor $\vec{w} = (w_1, w_2, \ldots, w_n)$, hodnoty tohoto váhového vektoru násobí hodnoty vstupního vektoru. Jako doplňující vstupní hodnota se obvykle používá tzv. bias, který má konstantní hodnotu a v průběhu učení se nemění. Formálně lze rovnici výpočtu neuronu zapsat

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

kde y značí výstupní hodnotu neuronu, x_i a w_i i-té prvky vstupního a váhové vektoru ovlivněné jejich velikostí n. Suma vzájemně vynásobených hodnot těchto vektorů je doplněna

o hodnotu proměnné bias b a následně je aplikována přenosová, známá také jako aktivační, funkce neuronu, která je v rovnici značená f. Nejčastěji používané aktivační funkce jsou popsány v 2.1.1.

Takto definované základní jednotky lze skládat a propojovat do komplexnějších architektur tvořící výslednou umělou neuronovou síť. Jedno z těchto propojení je vyobrazeno na obrázku 2.1. Je zde předvedena základní struktura sítí, ve kterých jsou jednotlivé neurony uspořádány do vrstev, které jsou navzájem propojeny. Tyto orientované spoje určují předávání informací mezi neurony a jejich váha určuje intenzitu předávané informace.



Obrázek 2.1: Dopředná neuronová síť a její vrstvy.

Zde zobrazená síť je tzv. dopředná síť, ve které jsou jednotlivé vrstvy zpravidla plně propojeny a neurony ve vrstvách samotných vzájemně propojeny nejsou. Všechny spoje jsou orientovány od vstupní vrstvy, jejíž neurony na své vstupy obdrží vstupní data, k vrstvě výstupní, která generuje data výstupní. Existuje mnoho rozšířených verzí tohoto základního modelu, které upravují jeho chování. Mezi tyto úpravy patří např. vynechání některých propojení mezi vrstvami neuronů, nebo přidání zpětných spojů z některých neuronů k neuronům ve vrstvách předcházejících. Tyto úpravy mohou hrát důležitou roli při řešení vybraných úloh.

2.1.1 Aktivační funkce

Aktivačních funkcí existuje několik a slouží k transformaci vnitřního potenciálu neuronu na výstupní hodnotu. Tato transformace je dána použitou aktivační funkcí. V závislosti na jejím umístění v síti a dané úloze může být vhodné využít několik aktivačních funkcí v rámci jednoho modelu. Nejčastěji se používají aktivační funkce, které umožňují použití učení sítě pomocí algoritmu zpětné propagace (backpropagation), který vyžaduje, aby daná funkce byla diferencovatelná.

Rectified Linear Unit

V současnosti velice používaná aktivační funkce známá především pod zkratkou ReLU je dána předpisem:

$$f(x) = \begin{cases} x & \text{pro} x > 0 \\ 0 & \text{jinak.} \end{cases} \tag{2.2}$$

Tato funkce má velké uplatnění v konvolučních sítích, kde vyniká svou jednoduchostí, a tedy i rychlostí výpočtu. Její nevýhoda se projeví, nastane-li, že během učení se nastaví příliš vysoká váha propojení. Toto bude mít za následek, že daný neuron již nebude aktivován. Tomuto dopadu lze předejít vhodnou volbou míry učení, jejíž vysoká hodnota vede k tomuto nežádoucímu efektu.

Sigmoida

Jedná se o první používanou nelineární funkci, která stále nachází využití. Její výhodou je především její nelineárnost, existence derivace, což umožňuje použití algoritmu učení zpětné propagace, a její omezený obor hodnot. Nejčastěji používaný tvar této aktivační funkce je dán rovnicí:

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{2.3}$$

jejíž výstupní hodnoty spadají do intervalu (0;1).

SoftMax

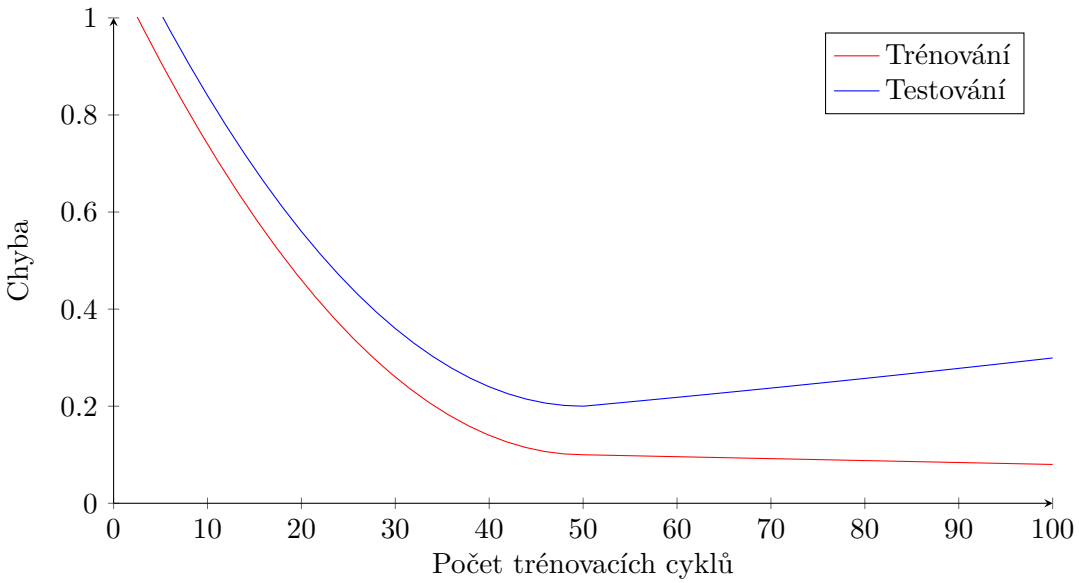
Tato aktivační funkce je nejčastěji používána u výstupních vrstev neuronových sítí řešící klasifikační úlohy. Tato funkce normalizuje vstupní vektor hodnot na pravděpodobnost, s jakou vstupní data sítě patří do daných výstupních kategorií. Součet pravděpodobností všech tříd je roven 1. Tato funkce je dána rovnicí 2.4, kde K udává počet neuronů ve výstupní vrstvě, který je roven počtu rozpoznávaných kategorií. Z této rovnice je vidět, že pro výpočet je nutné pracovat s hodnotami všech neuronů. Je-li k danému vstupu hledána pouze jedna kategorie, pak je tato kategorie dána nejvyšší pravděpodobností na výstupním neuronu.

$$f(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \tag{2.4}$$

2.1.2 Učení neuronových sítí

Učení a ohodnocování neuronových sítí se zpravidla provádí na dvou odlišných datových sadách. Datová sada s dostatečným množstvím datových vzorků se používá k učení neuronové sítě a odlišná skupina datových vzorků k ohodnocení její úspěšnosti. Přestože počet testovacích vzorků není stanoven, měl by být vybírán s ohledem na složitost daného problému a velikost neuronové sítě. Velké množství testovacích dat může vést ke zdlouhavému procesu testování s výsledky poskytující stejné informace, kterých by bylo možno dosáhnout menší testovací množinou vzorků. Tento výběr však musí být vhodně vybranou reprezentací testovacích případů. Naopak malé množství vzorků může mít za následek zkreslení výsledků úspěšnosti neuronové sítě.

Podobná pravidla se vztahují k trénovací datové sadě, avšak zde je vyšší množství trénovacích dat obvykle přínosem. Nicméně, použití datasetu s nedostatečným množstvím dat může vést k tzv. přeučení sítě. Tento efekt se projeví adaptací sítě na testovací vzorky,



Obrázek 2.2: Přeučení neuronové sítě

což vede ke stále lepším výsledkům během trénování, na úkor výsledků během praktického použití. Pro vyvarování se tomuto problému se doporučuje neuronovou síť průběžně testovat na odlišných datech, než probíhá trénování a testování. Tímto vznikne třetí skupina dat, označována jako validační, která je určena právě k tomuto ohodnocení během trénování. Výsledky ohodnocení napomáhají k rozpoznání momentu, kdy začíná docházet k přeučení. Tento jev je znázorněn na grafu 2.2. Testování pomocí validační sady se obvykle provádí po každém trénovacím cyklu, tedy po průchodu všech trénovacích dat.

Volně dostupné, případně vytvořené, datové sady obvykle nejsou rozděleny do uvedených skupin. V těchto případech se daná sada rozdělí v určitém poměru na trénovací, validační a testovací. Často je uveden poměr 8:1:1, ovšem správné rozdělení by mělo být ovlivněno velikosti dané sady, typu řešené úlohy a její složitostí. Rozdělení je tedy velice individuální, avšak stále je potřeba mít na paměti problémy spojené s nesprávným výběrem daných datových sad, které byly zmíněny v předcházejícím textu této podkapitoly.

Úspěšnost učení lze hodnotit ze dvou pohledů, a to jeho přesnosti (accuracy) vyjadřující míru správnosti výstupů daného modelu a naopak, jako jeho chybovosti (error rate), která vyjadřuje míru nesprávnosti výstupů.

Samotný proces učení je možné rozdělit do několika kategorií dle přístupu a informací v použité datové sadě: učení s učitelem (supervised learning), učení bez učitele (unsupervised learning) a zpětnovazební učení (reinforcement learning). Jako samostatná kategorie je občas uváděno i tzv. semi-supervised learning, při kterém se využívá kombinace prvních dvou výše zmíněných kategorií učení. Tento typ je především zmiňován ve spojení s GAN sítěmi, ve kterých se využívá jak učení s učitelem, tak učení bez učitele. Tyto sítě budou podrobněji popsány v 2.3.

Učení s učitelem

Učení s učitelem je založeno na datových sadách obsahující dvojice hodnot, které odpovídají vstupním datům sítě a požadovaným výstupům, též známým z anglické literatury jako ground truth. Vstupní data této trénovací sady jsou postupně přiváděna na vstup sítě.

Vnitřní hodnoty vah této sítě jsou zpočátku nastaveny na hodnoty inicializační, mnohdy na hodnoty náhodné. V každém kroku trénování sítě proběhne zpracování vstupu a vygenerování výsledné hodnoty nebo hodnot výstupní vrstvou sítě. Ty jsou pomocí vhodné chybové funkce porovnány s očekávanou výstupní hodnotou a dle výsledků tohoto porovnání dojde ke zpětné úpravě vah sítě, pomocí některého z algoritmů učení, nejčastěji pomocí zpětné propagace chyby.

Dle typu úlohy a výstupních dat je nutné zvolit vhodnou chybovou funkci. Mezi často používané funkce a funkce použité v této práci patří:

- **Mean squared error**
  Funkce pro výpočet rozdílu čtverců, též známá pod zkratkou **MSE**, se používá, je-li výstupem n-dimenzionální vektor. Výsledná hodnota udává, jak se v průměru liší odhad modelu od skutečných hodnot, měříme-li jejich vzdálenost jako druhou mocninu rozdílu těchto hodnot. Tento výpočet je dán rovnicí, která je uvedena v 2.5, kde n je počet vzorků, y_i výstupní vektor sítě a $\hat{y}_i$ požadovaný výstup. Trénováním sítě se snažíme o snižování hodnoty této chybové funkce.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.5}$$

- **Mean absolute error**
  Tato chybová funkce je založena na podobném principu jako MSE, avšak funkce **MAE** (Mean Absolute Error) nepracuje s druhými mocninami rozdílů hodnot, ale jejich absolutní hodnotou. Tento vztah je znázorněn rovnicí 2.6, kde n je počet vzorků, y_i výstupní vektor sítě a $\hat{y}_i$ požadovaný výstup. Trénováním sítě modelu se opět snažíme o snížení této hodnoty.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.6}$$

- **Cross entropy**
  Použití funkce CE (Cross Entropy) je vhodné, je-li úlohou neuronové sítě zařadit vstup do jedné z C kategorií. V tomto případě je na výstupní vektor sítě obvykle aplikována sigmoida či softmax aktivační funkce, která převede vektor výsledných hodnot do požadované podoby. Rovnice pro výpočet této chyby je uvedená v 2.7, kde t_i znázorňuje požadovaný výstup, s_i výstupní ohodnocení neuronové sítě a C celkový počet tříd.
  Existují další varianty této základní funkce. Například **binary cross-entropy** má pevně zadané dvě kategorie a **categorical cross-entropy** zase pracuje s pravděpodobnostmi, že daný vstup patří do některé z C kategorií. To je využíváno při rozřazování vstupů do více tříd.

$$CE = -\sum_{i}^{C} t_i \log(s_i) \tag{2.7}$$

Učení bez učitele

Při použití této metody učení nedochází k žádnému ohodnocení výstupních dat během trénovací fáze, a tedy předání zpětné vazby učené síti. Samotné učení probíhá na základě

rozpoznávání podobností mezi vstupními daty a jejich rozdíly. To má za následek úpravu vnitřních vah neuronů tak, aby byl výstup sítě konzistentní a na vstupní data s podobnými vlastnostmi se vytvořil podobný výstup. Kupříkladu při problémech klastrování dochází k hledání podobností a roztřiďování vstupů do hledaných skupin, jejichž počet může být předem znám.

Zpětnovazební učení

Zpětnovazební učení probíhá na základě snahy maximalizovat obdrženou odměnu získávanou pomocí některého z hodnotících algoritmů. Ty svou odměnu vypočítávají dle výstupních dat nebo změn, které tyto algoritmy zpozorovaly.

Optimalizační algoritmy

Optimalizační algoritmy pomáhají při učení sítě, a tedy k minimalizování výsledné chyby sítě. Gradientní metoda (gradient descent) je jedna z nejpopulárnějších metod pro provádění optimalizací a zdaleka nejběžnější k optimalizování neuronových sítí. Použitím této metody se postupuje ve směru největšího spádu chybové funkce ve snaze nalézt její minimum. V současnosti každá moderní knihovna pro hluboké učení obsahuje implementaci některého z mnoha algoritmů k optimalizaci gradientní metody. Nicméně tyto algoritmy jsou často používány jako černé skříňky a praktické vysvětlení jejich výhod a nevýhod je obtížné získat [30].

Cílem těchto algoritmů je nalezení lokálního či globálního minima a tím snížení chyby sítě. V dnešní době existuje mnoho obměn algoritmu gradient descent např. batch gradient descent, stochastic gradient descent a mini-batch gradient descent. To stejné lze říci i o optimalizačních algoritmech, které jsou na gradientní metodě založené – Adadelta, RMSprop, AdaMax, AMSGrad, Nadam nebo Adam. Volba konkrétní metody může být proto obtížná a její výběr může být také ovlivněn její dostupností v použité knihovně pro hluboké učení. Výběr vhodné metody také může záležet na množství trénovacích dat, kde při malém datasetu lze docílit lepších výsledků použitím adaptivní metody učení, a to i s použitím výchozího nastavení bez nutnosti hledání nejlepší hodnoty pro míru učení [30].

Optimalizační algoritmus **Adam** [19] (Adaptive Moment Estimation), obdobně jako jiné algoritmy, pracuje s uloženými hodnotami gradientů z předchozích kroků v_t, ale také si ukládá exponenciálně se snižující gradient m_t podobný hybnosti (momentu). Tato hybnost může být viděna jako míč kutálející se ze svahu, algoritmus Adam simuluje chování těžkého míče s vysokým třením, který preferuje hledaná minima [12]. Gradienty m_t a v_t se vypočítají dle rovnic uvedených v 2.8.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.8}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.9}$$

Autoři tohoto algoritmu zpozorovali, že tyto gradienty se nechovají zcela korektně v počátečních iteracích, kdy mají tendenci blížit se k nule. To má za následek, že je nutné vypočítat opravující odhady pomocí rovnic uvedené v 2.10, aby se zajistilo korektní chování převážně při inicializačním kroku a v momentě, kdy je míra snižování velmi malá.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.10}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.11}$$

Takto vypočítané hodnoty se následně použijí k aktualizaci parametrů pomocí rovnice 2.12, kde θ je aktualizovaný parametr, α je velikost kroku a β_1, β_2 a ϵ jsou proměnné nastavení programu. Autoři doporučují následující hodnoty: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ a $\epsilon = 10^{-10}$ [19].

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.12}$$

2.2 Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou speciální typ neuronových sítí pro zpracovávání dat, která mají maticovou topologii. Jako konvoluční sítě se označují ty, které obsahují alespoň jednu konvoluční vrstvu. Ke zpracování obrazů jsou preferovány konvoluční sítě, které svou architekturou a implementací zredukují množství parametrů v síti. Touto redukcí dojde ke snížení počtu propojení neuronů a jejich vah, kterých by bylo zapotřebí v běžné neuronové síti. Další výhodou těchto sítí je jejich schopnost posoudit sousedící hodnoty a tímto napomoci k odhalení vzorů zpracovávaných dat. Vzdálenost posuzovaných hodnot je dán velikostí filtru.

Během výpočtů v této vrstvě nedochází k obvyklému násobení matic jako u běžných vrstev umělých neuronových sítí, ale využívá se matematické operace zvané konvoluce. Vstupem této operace obvykle bývá vícerozměrné pole dat a filtrů. Tyto filtry jsou též označovány pod pojmem kernel a jejich hodnoty jsou průběžně upravovány na základě algoritmu učení. Operace konvoluce je definována rovnicí 2.13. V této rovnici je operace konvoluce značena symbolem $*$, první argument x označován jako vstup a druhý argument w jako kernel. Výstupem této operace je aktivační mapa (feature map).

$$s(t) = (x * w)(t) \tag{2.13}$$

Konvoluce se často počítají přes více než jednu osu najednou. Například, při použití dvourozměrného obrazu I jakožto vstupu, se bude také pravděpodobně používat dvourozměrný kernel K:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I\ (i-m, j-n) K\ (m,n), \tag{2.14}$$

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I\ (m,n) K\ (i-m, j-n), \tag{2.15}$$

kde funkce I vrací hodnotu vstupních dat a funkce K hodnotu kernelu (filtru) na příslušných souřadnicích. Z těchto rovnic je vidět, že operace konvoluce provádí vážený součet hodnot na daných souřadnicích a v jejím okolí ve vstupních datech s hodnotami kernelu.

Tato konvoluce má komutativní vlastnost, neboť kernel je v tomto případě obrácen relativně ke vstupu. Tedy navyšuje-li se hodnota m, pak se navyšuje i index do vstupu, avšak index do kernelu se snižuje. Zatímco komutativní vlastnost je užitečná pro psaní důkazů,

není příliš důležitá při implementacích neuronových síti. Místo toho mnohé knihovny implementují konvoluci jako funkci vzájemné korelace (cross-correlation), která je stejná jako konvoluční, ale jádro konvoluce se neotáčí [10], jak je znázorněno na rovnici 2.16.

$$S(i,j) = (K \ * I)(i,j) = \sum_m \sum_n I\ (i+m, j+n) K\ (m,n) \tag{2.16}$$

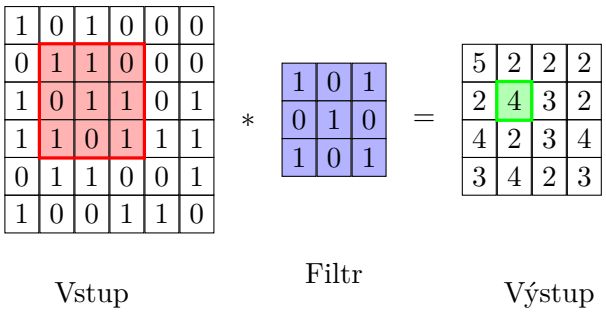
2.2.1 Vrstvy konvolučních sítí

Jak již bylo řečeno, za konvoluční sítě jsou považovány ty, které obsahují jednu nebo více konvolučních vrstev. V těchto sítích se nejčastěji vyskytují níže popsané typy vrstev, z nichž některé (konvoluční a plně propojená vrstva) pracují nejen se vstupními hodnotami, ale také s parametry vah a biasy neuronů. Aktivační a poolingová vrstva provádí své funkce bez těchto parametrů.

Konvoluční vrstva

Každá konvoluční vrstva obsahuje jeden nebo více filtrů, které se také označují jako kernel nebo jádro konvoluce, obsahující váhy dané vrstvy. Ve vrstvě zpravidla nebývá pouze jeden filtr, ale skupina filtrů, každý s vlastními váhami. Při průchodu vstupních dat se realizují konvoluce pro každý z přítomných filtrů. Každý kernel má stejný počet dimenzí jako vstupní data, avšak liší se v rozměrech, které jsou zpravidla v řádech jednotek.

V každém kroku zpracovávání vstupu se tyto filtry vynásobí jako skalární součin s malou plochou vstupních dat. Tato plocha rozměrově odpovídá velikosti filtrů a výsledkem je jedna hodnota, která se uloží na odpovídající místo ve výstupní aktivační mapě, která je tímto filtrem generována. Následně se vybraná plocha posune o předem daný krok a proces se opakuje, dokud nezpracují všechny vstupní data. Tento proces je znázorněn na obrázku 2.3.



Obrázek 2.3: Ukázka operace konvoluce. Na vybraná vstupní data daného kroku (červeně) je aplikována operace konvoluce s uvedeným filtrem (modře) a součet hodnot této operace je zapsán na určené místo ve výstupní aktivační mapě (zeleně).

Posuvný krok (stride) bývá stejný ve všech dimenzích a jeho velikost ovlivňuje velikost výstupní aktivační mapy, kde vyšší krok vyústí ve výstup o menších rozměrech. Problém také nastává s okrajovými hodnotami. Pokud tento problém není řešen a filtry jsou přikládány pouze tak, aby nepřekročily hranice dat, tyto hodnoty budou mít menší vliv na výstup a také dojde ke zmenšení výstupních aktivačních map. Obsahuje-li neuronová síť mnoho konvolučních vrstev, dojde k výraznému zmenšení dat, což není vždy žádoucí. Tento efekt lze kompenzovat, použije-li se zarovnání nulami tzv. padding. Takto se umožní přikládat filtr

i mimo hranice dat s tím, že při počítání výstupu se hodnoty za hranicemi dat doplní nulami. Takto nedochází ke zmenšení výstupu, a i okrajové hodnoty přispívají stejnou váhou. Dalším možným řešením je zkopírovat okrajové hodnoty a takto rozšířit vstup. Velikost kroku a rozměry filtrů by měly být voleny vhodně a to tak, aby při výpočtu dle vzorce 2.17 vycházelo celé číslo a zachovala se tak symetrie výstupu. V uvedené rovnici n značí jeden z rozměrů vstupu a kalkulovaného výstupu, p okrajové zarovnání, f rozměr filtru, u kterých se obvykle volí čtvercová podoba a s značí zvolený krok [18].

$$n = \frac{n + 2p - f}{s} + 1 \tag{2.17}$$

Aktivační vrstva

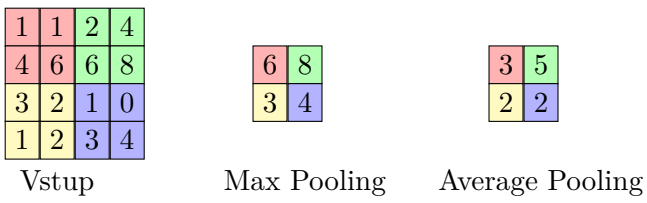
Jak bylo popsáno výše, výpočet konvolučních vrstev je založen na lineárním výpočtu. Neuronové sítě skládající se pouze z konvolučních vrstev, jsou ekvivalentní sítím skládajících se pouze z jedné konvoluční vrstvy a jinými hodnotami filtrů. Proto se za konvoluční vrstvy typicky vkládají vrstvy aktivační, které v síti naruší zmíněnou linearitu.

Aktivační vrstvy obvykle vyžadují pouze parametry v závislosti na zvolené aktivační funkci, kde většina z nich má již přednastavené hodnoty.

Jednou z často používaných aktivačních vrstev je založena na aktivační funkci ReLU, která byla popsána v 2.1.1.

Poolingová vrstva

Hlavním úkolem poolingové vrstvy v konvolučních sítích je zmenšení rozměrů vstupních dat na základě nastaveného kroku. Tento krok se obvykle volí v závislosti na velikosti filtru tak, aby se redukované oblasti nepřekrývaly. Zmenšení rozměrů vstupních dat se týká pouze rozměrů jednotlivých aktivačních map, ale jejich počet zůstává na výstupu zachován. Hodnoty výstupů se vypočítávají dle zvolené funkce. V praxi se nejčastěji používá tzv. max pooling, který z oblasti filtru vybere vždy nejvyšší hodnotu a tu umístí na příslušnou pozici ve výstupní aktivační mapě. Lze se také setkat s funkcí average pooling, který vytváří průměrné hodnoty z dat v oblasti filtru. Obě tyto funkce jsou znázorněny na obrázku 2.4.



Obrázek 2.4: Ukázka operace max pooling a average pooling s velikostí filtru 2×2 a krokem 2. Jednotlivé redukované oblasti a jejich výstupy jsou barevně odlišeny.

Použitím pooling vrstvy docílíme snížení předávaných informací mezi následujícími vrstvami a tím snížení výpočetních nároků při jejím učení, a tedy zrychlení tohoto procesu. Některé z úloh řešené pomocí konvolučních sítí vyžadují jako výstup jednu hodnotu. U těchto a podobných problémů má význam použít v síti více pooling vrstev. Nejčastěji se u této vrstvy používá krok s hodnotou 2, která zredukuje velikost aktivačních map o 75 %.

Umístěním pooling vrstvy na začátek neuronové sítě slouží k úpravě rozměrů vstupních dat. V tomto případě nezáleží na rozměrech předávaných dat s tím, že počáteční pooling

vrstva upraví vstupní data na požadované rozměry, se kterými následně pracuje zbylá část sítě. Tohoto lze docílit pomocí dynamicky volených parametrů této vrstvy, a to v závislosti na velikosti vstupu.

Tuto vrstvu lze také nahradit vrstvou konvoluční, která s vhodně nastaveným krokem dosáhne podobného chování, avšak toto řešení bude produkovat rozdílné výstupní aktivační mapy, než jaké by produkovala poolingová.

Plně propojená vrstva

Jak již název napovídá, neurony ve vrstvě tohoto typu jsou plně propojeny se všemi neurony ve vrstvě předchozí a ve vrstvě následující. Neurony ve vrstvě samotné však vzájemně propojeny nejsou. Plně propojené vrstvy odpovídají vrstvám používaným ve standartních neuronových sítích. V konvolučních sítích se tyto vrstvy obvykle nachází u vrstev výstupních např.: jako konečná vrstva klasifikace.

Každý neuron plně propojené výstupní vrstvy s aktivační funkcí SoftMax 2.1.1 slouží jako indikátor míry příslušnosti vstupu do jednotlivých kategorií. Tyto vrstvy lze ovšem použít i jako vrstvy skryté, jelikož plně propojené vrstvy lze převést na konvoluční a naopak.

2.3 Generative adversarial networks

Generative adversarial networks (dále jen GAN sítě) [11, 28] je architektura, která se skládá ze dvou vzájemně soupeřících neuronových sítí 2.5. Úlohou první sítě, generátoru G, je na základě vstupních dat vygenerovat výstup, který co nejpřesněji odpovídá reálné množině dat z určitého datového setu. Vzorky z tohoto datového setu jsou společně s vygenerovanými výstupy předloženy druhé neuronové síti – diskriminátoru D. Úlohou této sítě je naučit se rozlišovat reálná data datového setu a vygenerovaná data sítí G. Síť D své rozhodnutí vyjádří v podobě skalární hodnoty $D(x)$. Během trénování se snažíme maximalizovat pravděpodobnost, že D přiřadí svým vstupům správné označení a zároveň trénujeme G k minimalizování hodnoty $\log(1-D(G(z)))$. Toto soupeření lze vyjádřit jako hru minimax dvou hráčů, kterou lze vyjádřit rovnicí 2.18 [11].

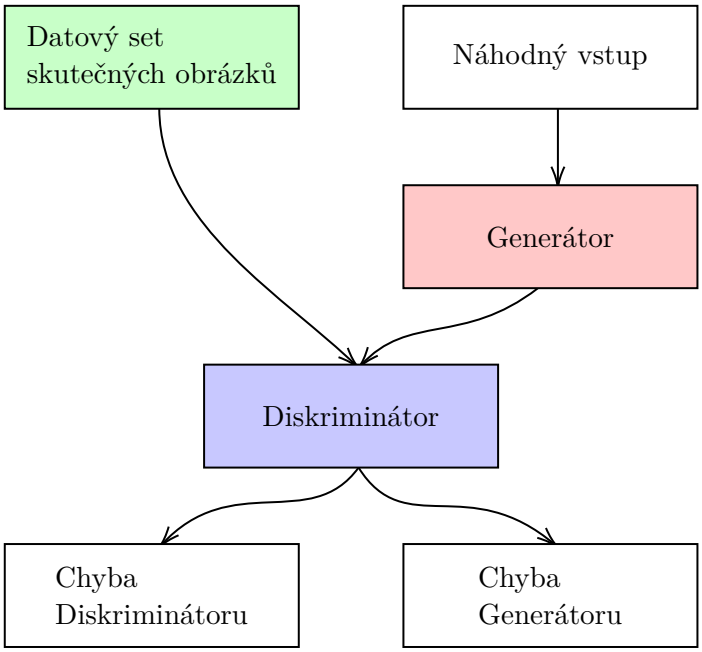
$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \tag{2.18}$$

Během trénování sítí dochází k jejich oddělenému upravování vnitřních vah. Při trénování diskriminátoru se mu předloží vzorky z datasetu a podvrhy vzorků vygenerované za pomoci generátoru. Výstupy diskriminátoru se porovnají s požadovanými a dojde k aktualizaci jeho vah za pomocí vzestupu jeho stochastického gradientu:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))]. \tag{2.19}$$

Trénování generátoru funguje obdobně. Generátorem poskytnuté vzorky jsou ohodnoceny sítí D. Informace o úspěšnosti obelstění diskriminátoru je předána zpět a na jejím základě jsou aktualizovány hodnoty vnitřních vah generátoru pomocí sestupu jeho stochastického gradientu:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)}))). \tag{2.20}$$



Obrázek 2.5: Model generátoru obdrží na vstup data, která má zpracovat. Jeho výstup a vzorek skutečných dat jsou porovnány diskriminátorem, který se snaží rozeznat, který je vytvořen generátorem. Výstup diskriminátoru, v kombinaci s požadovaným výsledkem, se použije k učení generátoru i diskriminátoru.

2.3.1 Model a chyba diskriminátoru

Při řešení úloh spojené s generováním obrazu mívají tradiční chybové funkce MSE 2.5 a MAE 2.6 problém s ostrostí generovaného výstupu. Přestože jejich výstupy přesně zachytí nízké frekvence obrazu, selžou v podporování generátoru o generování vysoko frekvenční ostrosti a výsledné obrazy jsou tak rozmazané [14].

Ke korekci tohoto problému lze sestavit síť diskriminátoru tak, aby se soustředila na přesnost obrazu ve vysokých frekvencích a spoléhala se, že nízké frekvence výstupů se převezmou ze zmíněných chybových funkcí. Takto sestrojený diskriminátor se nazývá PatchGAN. Jeho úlohou je, aby u každé $N \times N$ sekce vstupu rozhodl, zda je pravdivá nebo podvržená. Takto diskriminátor zpracuje celý vstupní obraz a vytvoří chybový výstup [14].

Na obrázku 2.6 jsou demonstrovány výstupy sítí, u nichž jsou použity různé metody výpočtu chyby. Tyto výstupy jsou generovány na základě vstupního obrazu ve formě sémantické segmentace odpovídajícího obrazu. Z těchto výstupů je vidět, že hodnota N může být výrazně menší než rozměry obrazu, a stále vytvořit výstup s vysokým rozlišením. To přináší výhody, protože menší PatchGAN pracuje s méně parametry, běží rychleji a může být aplikován na vstupy neomezených rozměrů [14].

2.3.2 Problémy GAN modelů

Sítě GAN dokáží generovat velmi přesvědčivá data. Nicméně stále čelí mnoha těžce řešitelným problémům. Obecně platí, že je časově obtížné tyto sítě natrénovat i za pomoci optimalizačních metod. Během trénování se mohou objevit i následující problémy, kterým není možné se s naprostou jistotou vyvarovat:

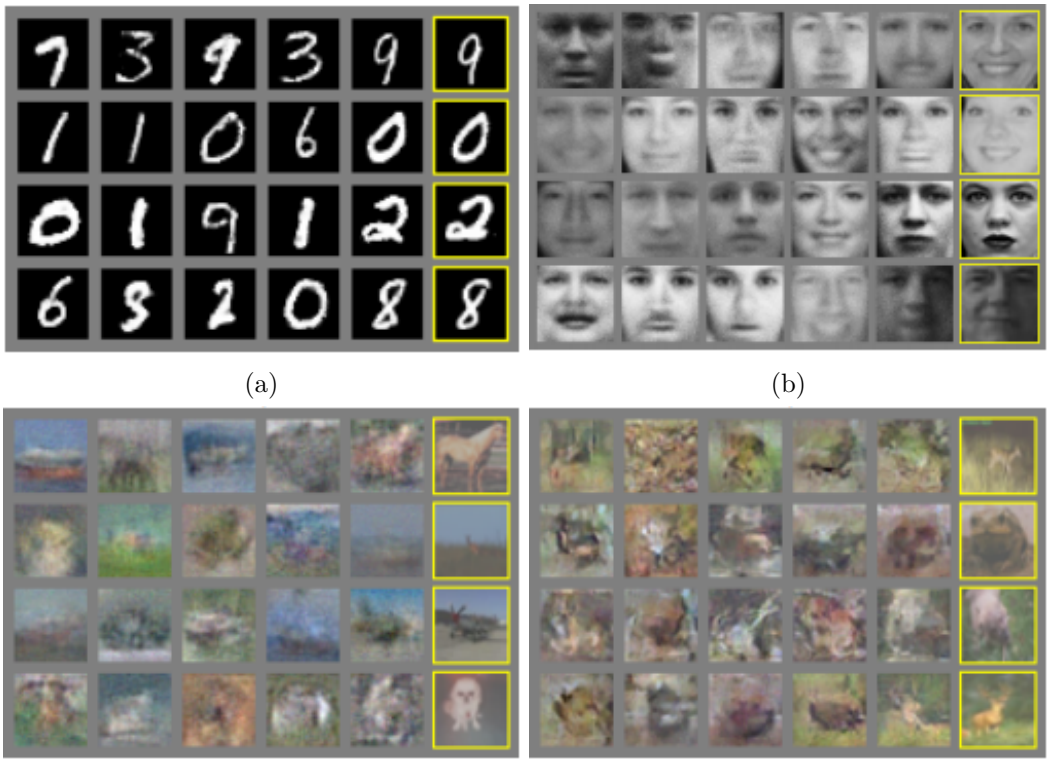
- **Mizející gradienty**
  Tato chyba souvisí převážně s hlubokými neuronovými sítěmi. Čím je síť komplexnější, tím je více ovlivněna zpětná propagace gradientů. V sítích GAN je tento problém umocněn tím, že chyba jedné sítě ovlivní i síť druhou. Při velké chybě diskriminátoru nedochází k dostatečnému předání informace generátoru a obdobný problém nastává i v opačném případě. Je proto nutné udržovat tyto sítě v rovnováze.

- **Mode collapse**
  Tuto chybu lze snadno odhalit, neboť se postupně začne projevovat na všech generovaných výstupech sítě generátoru. Jedná se o problém, při kterém tato síť dokáže opakovaně obelstít diskriminátor omezeným počtem vzorků. Extrémní situace nastává, generuje-li vzorek pouze jeden. Byly navrženy způsoby, jak tomuto problému předejít, kupříkladu použitím tzv. batch discrimination a repelling regularizer, avšak s různými stupni úspěchu [6].

Podaří-li se danou síť úspěšně natrénovat, mohou být výsledky velmi podobné skutečným obrázkům, jak je vidět na vygenerovaných výsledcích na obrázku 2.7.

(a) MAE (b) 1×1 (c) 16×16 (d) 70×70 (e) 286×

Obrázek 2.6: Variace posuzované velikosti sekcí. Nejednoznačnost ve výstupech se pro jednotlivé chybové funkce projevuje jinak. Tyto oblasti při použití funkce MAE jsou rozmazané a s menší intenzitou barev. PixelGAN 1×1 podpoří větší rozmanitost barev, avšak nemá žádný efekt na prostorovou statistiku. PatchGAN 16×16 vytvoří lokálně ostré výsledky, které vedou k tvorbě artefaktů za oblastí, se kterou pracuje. PatchGAN 70×70 si vynutí ostré výsledky, i když jsou nesprávné, jak v prostorovém, tak ve spektrálním rozměru. Plný 286×286 ImageGAN produkuje výsledky, které jsou vizuálně podobné 70×70 PatchGANu. Obrázky a popis převzat z [14].



(a) (b) (c) (d)

Obrázek 2.7: Ukázka vzorků vygenerovaných GAN modely. V pravých sloupcích obrázků jsou zobrazeny příklady skutečných vzorků, aby se demonstrovalo, že se model nenaučil kopírovat trénovací dataset. a) MNIST [25] b) TFD c) CIFAR-10 [23] (plně propojený model) d) CIFAR-10 (konvoluční diskriminátor a dekonvoluční generátor). Převzato z [11].

Kapitola 3

Neuronové sítě pro rozpoznávání textu

Neuronové sítě ve spojitosti s textem jsou známé především pro jejich využití při jeho vyhledávání v obraze a přepisu do jeho textové podoby. Tyto programy jsou obecně známy jako programy pro optické rozpoznávání znaků (OCR – Optical Character Recognition) a zahrnují několik oblastí pro práci s textem. Mezi tyto oblasti patří detekce textu v obraze, jeho extrakce, rozdělení po řádcích, slovech či znacích. Různé druhy přístupu v jednotlivých částech ovlivňují výstupní data, a tedy i celkové ohodnocení kvality rozpoznání a přepisu. Je dobré mít alespoň obecný přehled o těchto částech, neboť jsou vzájemně provázány. Tyto části jsou v krátkosti představeny v podkapitole 3.1.

Většina prací zaměřených na OCR sítě se soustřeďuje především na vyhledání, extrakci a přepis nalezeného textu. Vychází se z předpokladu, že textové dokumenty jsou digitalizovány vhodnými metodami nebo předem vhodně upraveny a samotný proces předzpracování tedy vypouštějí. Tato práce je zaměřena na zlepšení kvality těchto dokumentů před jejich zpracováním pomocí OCR sítí či jiných programů. Na proces předzpracování lze nahlížet jako na obecné zlepšování kvality obrazu. Techniky, zabývající se touto problematikou, soustředěné především na neuronové sítě využívající model GAN, jsou podrobněji rozebrány v podkapitole 3.2.

3.1 Rozpoznávání textu

V současnosti je vývoj v oblasti OCR soustředěn především na rozpoznávání ručně psaného textu a textu, jež je zasazen v určitém prostředí. Texty, nacházející se v libovolném prostředí, mohou mít různé styly a velikosti písma, být deformovány perspektivou a formou jejich vysázení, zčásti zakryty nebo být ovlivněny různou intenzitou světla. Všechny tyto problémy se mohou objevit i v rámci jednoho obrazu. Dalším zvýšením náročnosti rozpoznání je okolí textu, které může ovlivnit jeho detekci. Systém by si s těmito problémy měl umět poradit a extrahovat pouze požadovaný text [21].

Ručně psané texty jsou obvykle naskenovány nebo vyfotografovány, aby došlo k co nejmenší deformaci daného textu. Proto se u nich problémy, typické pro texty, ve scéně nevyskytují. U ručně psaného textu se naopak vyskytují problémy s jeho formou. Texty pocházející od různých autorů se mohou výrazně lišit ve formě tvaru znaků, jejich velikostí a úhlu. Drobné odlišnosti mezi jednotlivými znaky se nacházejí i v textech pocházejících

od jednoho autora. I když se problémy mezi ručně psaným textem a textem ve scéně mohou zdát odlišné, obvykle se k jejich řešení používají stejné techniky [21].

V této kapitole jsou stručně popsány jednotlivé kroky, vedoucí od pořízení obrazu textu, přes zpracování až po výslednou klasifikaci. Výstupem klasifikace je detekovaný text v podobě strojového přepisu, se kterým lze následně v počítači pracovat. Toto rozdělení je převzato z [22].

Digitalizace textových dokumentů

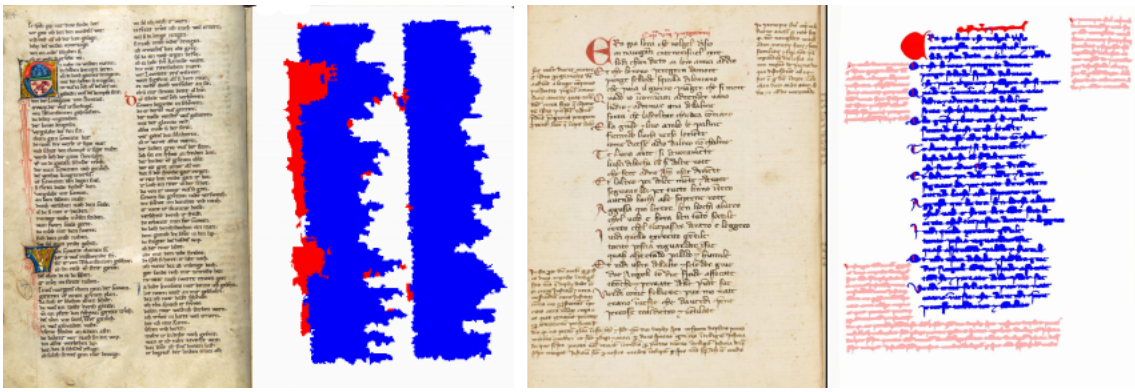
Digitalizací je převod daného textu do formy, se kterou je možné pracovat v počítačích. Obvykle se jedná o fotografie či skeny daného textu. Je důležité, aby všechny vstupy pro dané OCR byly snímány stejným nebo obdobným způsobem. Nelze předpokládat, že OCR natrénované na obrazech stránek knih pořízené profesionálním skenovacím zařízením s optimálním osvětlením, bude produkovat stejně kvalitní výsledky na rozmazaných snímcích pořízených mobilním fotoaparátem. Některé ze zmíněných vad lze odstranit nebo minimalizovat procesem předzpracování.

Předzpracování

Pojmem předzpracování je myšlen takový proces úpravy obrazu, aby se dosáhlo co nejpřesnějších výsledků v následujících částech procesu rozpoznávání textu. Jedná se tedy o odstranění šumu, rozmazání a deformace, ale také například o úpravu intenzity barev. Některé z metod provádějící tyto úpravy jsou podrobněji rozebrány v podkapitole 3.2.

Segmentace

Segmentace slouží k nalezení a rozdělení oblastí obsahující určitý typ požadované informace, v tomto kontextu se jedná o nalezení odstavců, řádků, slov a znaků. Tímto procesem se vyloučí nežádoucí oblasti a napomůže k detekci samotných řádků textu se správným rozdělením nesouvisejících textů. Segmentace stránky historického textu je ukázána na obrázku 3.1.



Obrázek 3.1: Ukázka segmentace textu (modře), dekorace (červeně), komentářů (růžově) a stránek (bíle) pomocí konvolučních neuronových sítí. Převzato z [7].

Detekce řádků

Následujícím krokem zpracování je detekce jednotlivých řádků textů, které se v daném obraze vyskytují. U každého z nalezených řádků je určena jeho výška a šířka. Je důležité, aby se detekovaly pouze jednotlivé řádky, které jsou následně přivedeny na vstup klasifikátoru.

Při lokalizování řádků mohou být na vstupní obraz použity různé transformace, které této detekci napomůžou. Při zpracování se také využívají charakteristiky obrazu. Tento popsaný postup je použit při implementování sítě v článku *STN-OCR: A single Neural Network for Text Detection and Text Recognition* [5], jehož ukázkový výstup detekce textu je na obrázku 3.2.

Obrázek 3.2: Vzorky z ICDAR [17], SVT [16] a IIIT5K [27] datových setů, které ukazují, jak dobře dokáže model vyhledat textové oblasti a je schopen sledovat sklon slov. Obrázek a popis převzat z [5].

Klasifikace

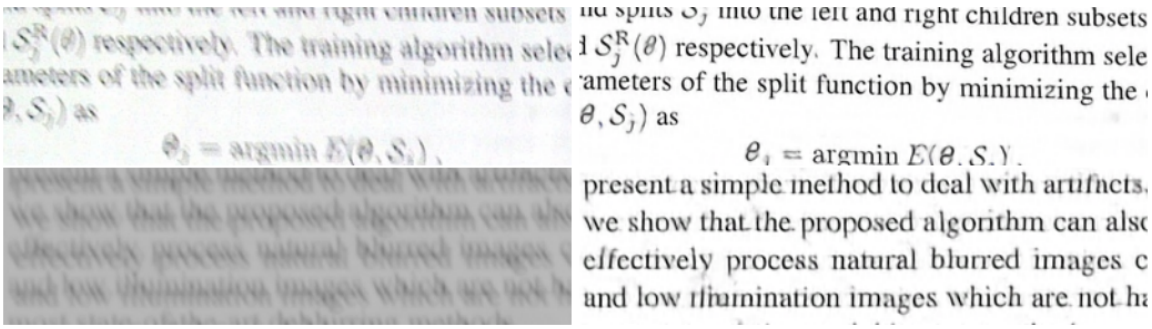
Posledním krokem rozpoznávání textu je samotná klasifikace vstupu do jeho textové podoby. Jedním z možných postupů je použití znakové a poziční větve, jak je uvedeno v článku *ImageNet Classification with Deep Convolutional Neural Networks* [24]. Na vstup větví se postupně přivádí výřezy vstupních dat. Znaková větev se snaží rozpoznat znak nejblíže středu daného výřezu a pravděpodobnosti jednotlivých znaků předá formou pravděpodobnostního vektoru na svůj výstup. Poziční větev má za úkol odhadnout vzdálenost k dalšímu znaku. Výstupem této větve je počet pixelů, o který se má výřez posunout a proces opakovat. Tento postup se opakuje, dokud nejsou zpracovány všechna data vstupního řádku.

3.2 Zlepšování kvality obrazu

Cílem předzpracování obrazu je upravit obraz do podoby, která umožní bezchybný přepis textu, jež se v obraze vyskytuje. Jak již bylo zmíněno, kvalita přepisu je ovlivněna všemi částmi procesu přepisu. Část předzpracování má za úkol zjednodušit zpracování obrazu dalšími částmi procesu. Její úloha obvykle spočívá v odstranění šumu a rozmazání, natočení textu do vhodné podoby, změně intenzity barev apod.

K řešení tohoto problému je možno postavit se několika způsoby. Častým přístupem je obstarání si datového setu obrázků, které jsou následně uměle poškozeny. Takto vytvořená data jsou použita k trénování, validaci a testování sítě. Nepoškozená data a parametry aplikovaného poškození jsou využita k ohodnocení výsledku a učení sítě.

Tento přístup je použit například v článku *Convolutional Neural Networks for Direct Text Deblurring* [13]. Na výřezy akademických článků jsou aplikovány efekty simulující pohybovou neostrost (motion blur) podobnou třesu fotoaparátu a rozostření (defocus blur). Na výsledcích a modelu konvoluční sítě z tohoto článku je založen článek *CNN for License Plate Motion Deblurring* [32]. I zde je využíván efekt pohybové neostrosti, kde je použit na výřezy poznávacích značek, a tím dochází k simulaci pohybu automobilu. Příslušný korespondující ostrý obraz z kamerového systému je opět použit k ohodnocení výsledků. Pro validaci a testování sítě byly použity obrazy zachycené pomocí kamerového systému, u nějž byl zvýšen čas expozice. Rozmazané obrazy z těchto článků a odpovídající výstupy, jež byly doostřeny konvolučními sítěmi, jsou znázorněny na obrázcích 3.3 a 3.4.

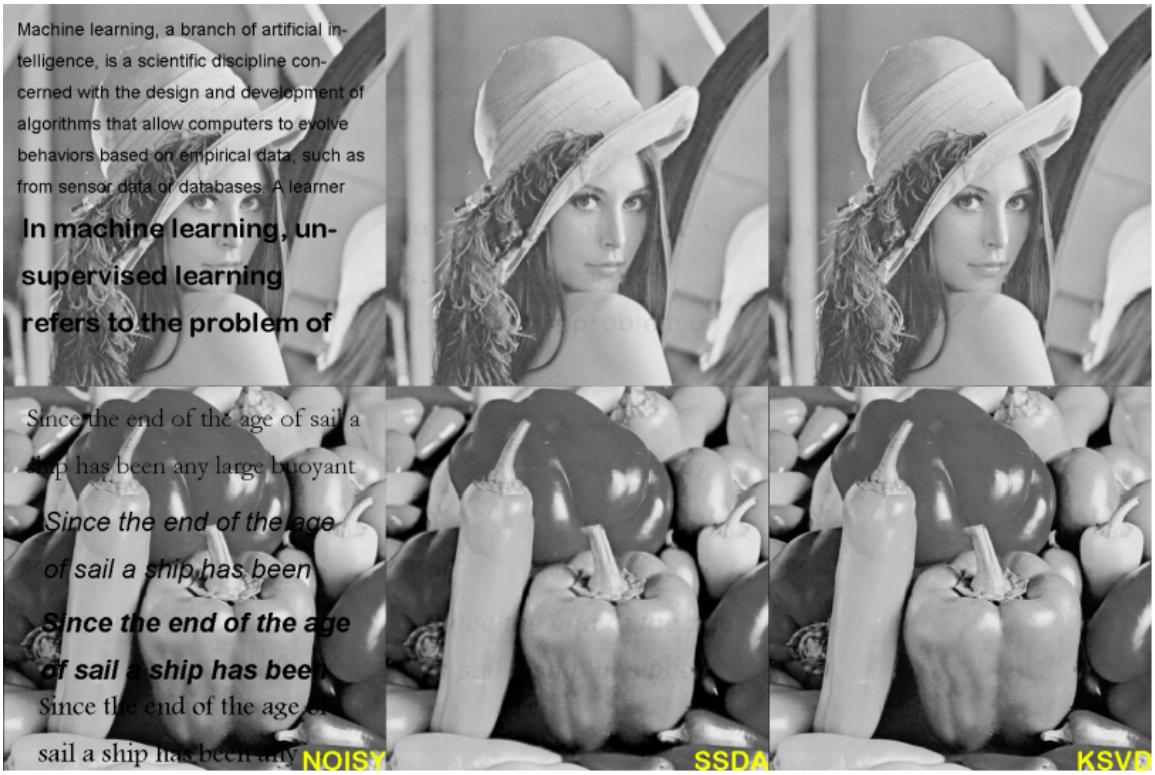


Obrázek 3.3: Doostření rozmazaného obrazu pomocí konvolučních neuronových sítí. Obrázek a popis převzat z [13].

Obrázek 3.4: Výsledky rekonstrukce poznávacích značek. V prvním řádku jsou poznávací značky zachycené kamerovým systémem s prodlouženým časem expozice. Na druhém řádku jsou výstupy sítě CNN-L15 natrénované pro doostření rozmazání s délkou 1–23px. Obrázek a popis převzat z [32].

Problém s odstraněním šumu v obraze vznikne, je-li obraz poškozen aditivním bílým Gaussovským šumem, který se běžně vyskytuje ve snímaných datových kanálech. Naopak, tzv. inpainting problém se vyskytne, chybí-li zcela hodnoty některých pixelů nebo se snažíme o odstranění sofistikovaných vzorů, jako překrývajícího textu nebo jiných objektů z obrazu [33].

K řešení těchto úloh je možno přistoupit několika způsoby. Doplnění překrytých částí obrazu je možné provést informovaně. V této metodě algoritmus obdrží informaci o pozici nechtěných objektů a ty následně odstraní a doplní chybějící strukturu. Takto informované algoritmy dokáží velmi dobře zpracovat obrazy obsahující text, ale i velmi velké objekty. Doplňování bez informací o odstraňovaném objektu je ovšem velmi obtížný problém. Tyto algoritmy si obvykle dokáží poradit pouze s jednoduchými strukturovanými impulsními šumy [33]. Výsledky algoritmů odstraňující text z obrazu jsou vidět na obrázku 3.5.



Obrázek 3.5: Vizuální porovnání výsledků odstraňování textu z obrazu různými metodami. Převzato z [33].

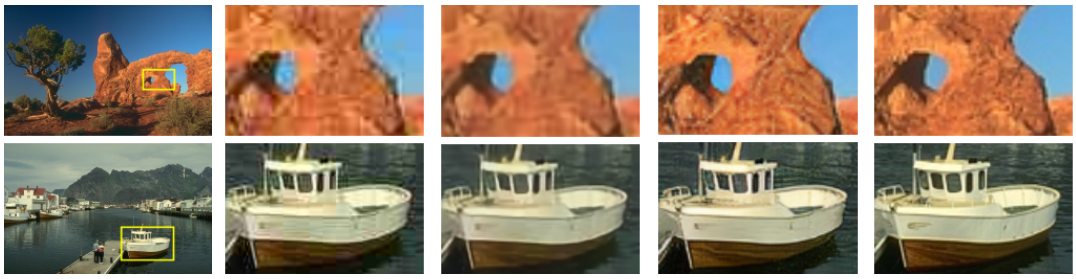
Je vyvíjeno mnoho různých technik zabývajících se odstraněním artefaktů z obrazu a vylepšením jeho komprese. Při redukci artefaktů se velice často používá VGG síť, která pomáhá zachovat charakteristiku obrazu. Sítě, nevyužívající těchto charakteristik, mohou dosahovat horších výsledků. Konktrétně, je-li místo náhodně zvoleného šumu použita sofistikovanější metoda poškození [33].

V článku *BlockCNN: A Deep Network for Artifact Removal and Image Compression* [26] je k odstranění JPEG artefaktu použita kombinace konvolučních a residuálních bloků. Zpracovávaný obraz je rozdělen po blocích o velikosti 8×8, které jsou zpracovávány odděleně a výsledný obraz je z těchto bloků složen zpět. Ke zpracování daného bloku je použito také jeho okolí, pro poskytnutí více informací během zpracování. Výsledky odstranění artefaktů komprimace JPEG jsou ukázány na obrázku 3.6.

(a) Originální obraz (b) Komprimovaný obraz (c) Rekonstruovaný obraz

Obrázek 3.6: Rekonstrukce obrazu komprimovaného metodou JPEG s blokovými artefakty. Převzato z [26].

V článku *Deep Generative Adversarial Compression Artifact Removal* [9] je ke zpracování opět použitá síť sestávající se z konvolučních vrstev a residuálních bloků Avšak oproti výše zmíněnému článku, zde autoři k učení sítě využívají modelu GAN 2.3. K učení sítě pro odstranění šumu a obelhaní diskriminátoru jsou využity charakteristiky obrazu získané pomocí sítě VGG19 [31]. Při porovnání výsledného obrazu je výsledek z vizuálního hlediska téměř nerozpoznatelný od originálního, jak je vidět na obrázku 3.7.



(a) Originál (b) JPEG 20 (c) AR-CNN (d) GAN článku (e) Detail

Obrázek 3.7: Výsledky rekonstrukce ukázané na detailech dvou komplexních struktur. Komprese JPEG vytvoří několik blokových, kruhových a barevných artefaktů. Síť AR-CNN [8] je schopna detail částečně obnovit, ale vytvoří rozmazaný výsledek. Výsledek zpracovaný sítí představenou článku je vizuálně těžce rozlišitelný od originálního obrazu. Převzato z [9].

Jak je na těchto výsledcích vidět, využití modelu GAN hraje roli v ostrosti produkovaných obrazů. Rozdíl v ostrosti je jasně viditelný například v článku *Image-to-Image Translation with Conditional Adversarial Networks* [14], který se zabývá rekonstrukcí obrazu na základě jeho sémantické segmentace. Autoři tohoto článku navrhli model diskriminátoru, který je soustředěn právě na ostrost výsledného obrazu. K rekonstrukci obrazu autoři použili model generátoru založený na architektuře U-Net [29], jenž je složena z kodéru s typickou architekturou konvoluční sítě a dekodéru, který má opačnou strukturu, tedy snižuje počet filtrů a zvětšuje rozměry aktivačních map. Architekturu U-Net doplňuje spojení aktivačních map v dekodéru s příslušně ořezanými aktivačními mapami z kodéru.

Kapitola 4

Návrh řešení

Tato kapitola se zabývá možnostmi řešení zlepšování kvality digitalizovaných textů. V první části jsou rozebrány možné zdroje dat, které budou sloužit jako vstup systému. Následně bude zvolen model implementované sítě na základě poznatků z podkapitoly 3.2. V závěrečné podkapitole budou představeny experimenty pro zhodnocení výsledků a metrika, na jejímž základě se tohoto vyhodnocení dosáhne.

Přestože fáze předzpracování obrazu obsahuje velmi rozsáhlé množství problémů, tato práce se soustřeďuje na problémy spojené s chybnou digitalizací a problémům spojených s poškozením fyzického zdroje textu před jeho digitalizací. Těmito poškozeními jsou myšleny vady tisku, degradace materiálu, na němž se text nachází. Tato poškození mohou vzniknout postupem času, častým používáním, špatným zacházením apod. Tyto vady jsou řešeny při rozpoznávání textu u starých knižních a novinových výtisků, u kterých nemusí existovat digitální přepis a kde se pracuje s velkým množstvím dat pro ruční korekturu. Digitalizované výtisky také mohou být vzácné, proto nelze spoléhat na nalezení výtisku nacházejícího se v lepším stavu pro úspěšnější zpracování textu. Správná digitalizace, předzpracování a kvalitní OCR pomohou redukovat množství chyb ve výsledných přepisech.

4.1 Datové sady

Již existuje velké množství volně dostupných datových sad určených k analýze a rozpoznávání textu. Ovšem většina z nich obsahuje relativně malé množství vzorků, není připravena pro zde řešený typ úlohy nebo neobsahují přepisy textů. V této podkapitole jsou představeny datové sady, jež byly v rámci vývoje této práce využity.

4.1.1 Noviny

Tento datový set je založen na výřezech z novinových stránek Lidových Novin a Národních listů, které jsou k dispozici na internetových stránkách Moravské zemské knihovny (MZK) – Digitální knihovna[1]. Tyto noviny obsahují stovky vydání z let v rozmezí 1863–1945. Každá digitalizovaná stránka je poskytnuta ve velkém rozlišení, ovšem rozlišení a rozměry jednotlivých stránek není konstantní.

Při generování výřezu dat, dochází k náhodnému výběrů stránek a pozici na nich. Na tyto výřezy jsou následně aplikovány jednoduché efekty rozmazání a šumu. Tento proces je podrobněji popsán v podkapitole 5.2. Ukázky výřezů jsou na obrázku 4.1.

[1] http://www.digitalniknihovna.cz/mzk/

Obrázek 4.1: Ukázka výřezů novinových stránek (a) rozmazaný text (b) nepoškozený text (c) text s nežádoucím fragmentem (d) komerční část.

4.1.2 B–MOD

Datová sada B–MOD vznikla v rámci výzkumného projektu, který má za cíl zvýšit dostupnost k digitalizovaným historickým dokumentům [1], a jeho cílem je poskytnout velké množství vhodných dat, které umožní a podpoří výzkum v oblasti metod pro analýzu dokumentů s nízkou kvalitou.

Stejně jako mnoho jiných podobných datových setů, je i tento navrhnut a určen pro vývoj a testování OCR. Celkem se skládá z 19 725 fotografií 2 113 stránek náhodných vědeckých článků různého zaměření. Jednotlivé snímky jsou pořízeny různými lidmi pomocí 23 mobilních zařízení. Tímto se docílilo snímků s různou kvalitou osvětlení, pořízených z různých úhlů a vzdáleností. Hodně fotografií také obsahuje pohybovou neostrost vzniklou pohnutím snímajícího zařízení, či snímaného objektu během jeho expozice.

Takto získané fotografie byly dále zpracovány až do podoby 515 400 řádků textu a jejich odpovídajících přepisů. Tyto dvojice jsou dále rozděleny do devíti skupin. První rozdělení určuje, zda se jedná o trénovací, validační nebo testovací množinu, kde jednotlivé podmnožiny obsahují prvky v poměru 8:1:1. Druhé rozdělení je určeno přesností přepisů pomocí neuronových sítí trénované nad celým datovým setem. Řádky, které byly korektně přepsány, jsou označeny jako jednoduché. Přepisy, obsahující méně jak 20 % chybných znaků, jsou označeny jako středně obtížné, ostatní jsou označeny jako velmi obtížné [20]. Příklady rozdělení dle složitosti jsou zobrazeny na obrázku 4.2.

4.1.3 Generovaná datová sada

Rozhodnutí pro vytvoření umělé datové sady bylo ovlivněno především absencí již vytvořené sady, které by obsahovala všechna potřebná data a stylově se hodila pro řešený typ úlohy. Většina dostupných sad, jako výše zmíněný B–MOD, je tvořena pro trénování a testování OCR sítí. Obsahují tedy poškozené texty a v ideálním případě i jejich přepisy. Zvolená architektura modelu popsaná v kapitole 4.2 však vyžaduje i obrazy nepoškozeného textu. Dostupné datové sady obvykle neobsahují tuto část.

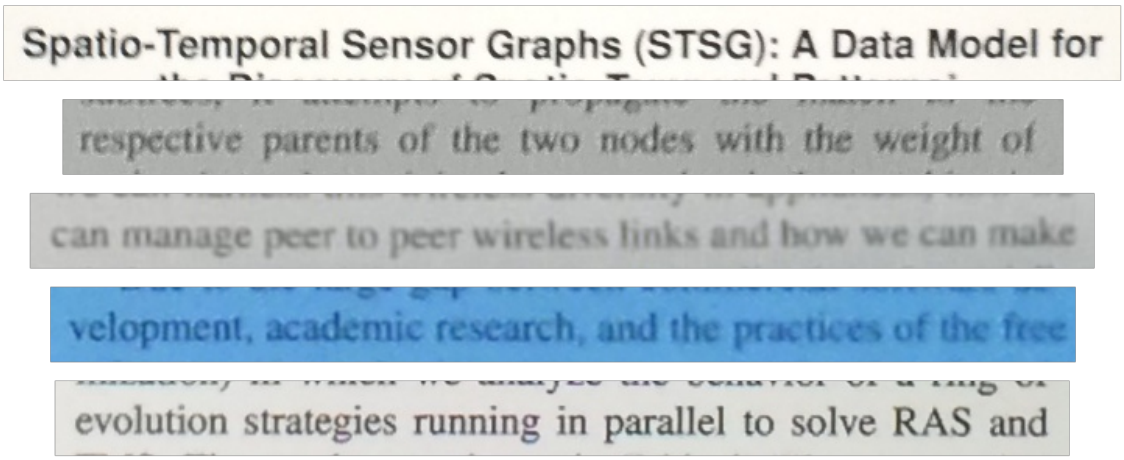
Využitý generátor dat[2] je implementován ke generování historických textů pro trénování OCR sítí. Takto vytvořené stránky se vzhledově snaží co nejvíce přiblížit skutečným naskenovaným historickým dokumentům. K tvorbě využívá historická písma a textury podkladu, které mají vzhled starého papíru [21].

Na takto vysázený text jsou následně aplikovány různé efekty, které mají za cíl co nejvíce přiblížit výsledný obraz podobě skutečného historického textu. Mezi tyto efekty patří nedokonalost tisku, který vytváří nesouvislé mapy imitující nerovnoměrnosti při tisku a defekty samotných písmen. Dalším efektem je doplnění tzv. zadního tisku, jež je velice průsvitný text napodobující situaci, kdy se text z opačné stránky propíjí či prosvítá nebo došlo k otisku textu ze stránky vedlejší. Posledními efekty jsou doplnění další textury a rozmazání písmen [21]. Celý proces generování je uveden na schématu 4.3 a ukázka vygenerovaných dat je na obrázku 4.4.

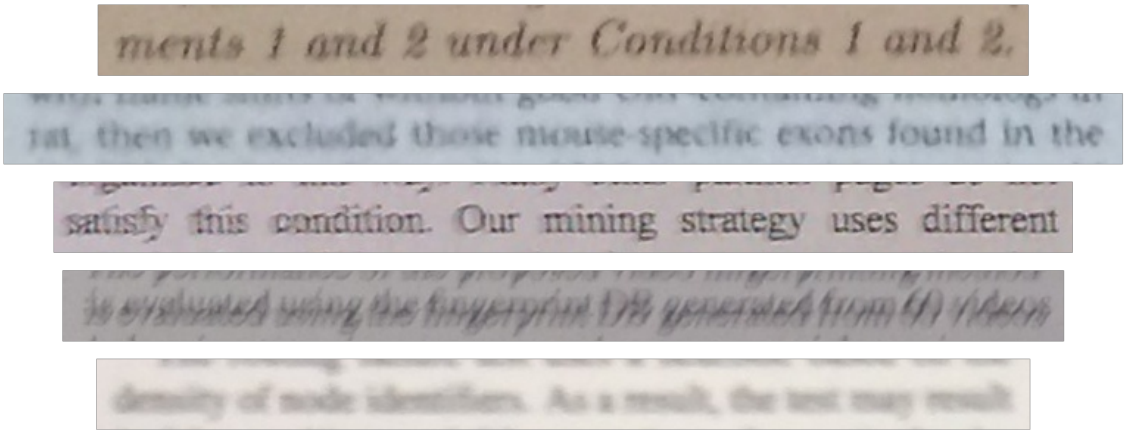
Takto vygenerované stránky historického textu se vizuálně velice podobají novinovým stránkám z podkapitoly 4.1.1. Použitím generátoru lze získat přístup k neomezenému množství trénovacích a testovacích dat. Pro účely této práce bude potřeba doplnit text generovaných dat a novodobá písma.

Tento generátor textů byl implementován s cílem tvoření dat pro učení a testování OCR sítí. Tomu odpovídají i výstupní soubory generování, které obsahují nejen vygene-

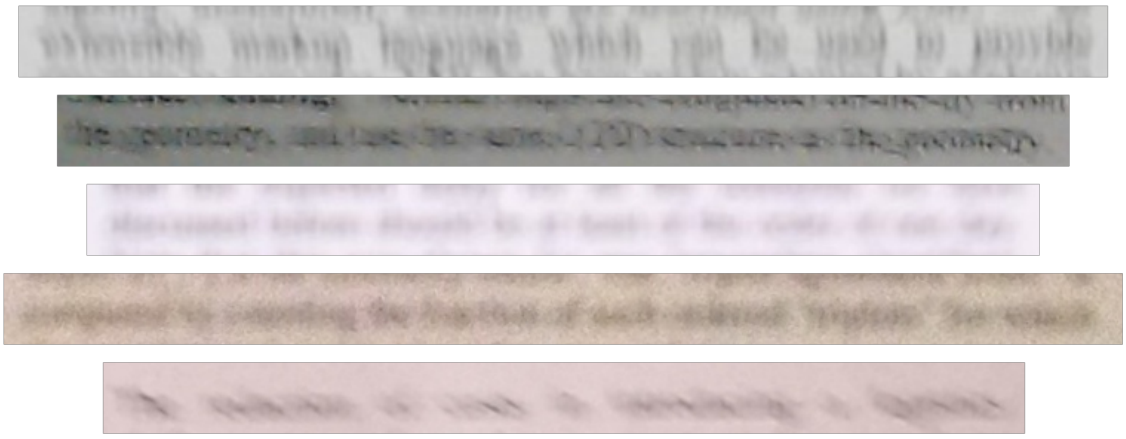
[2] https://github.com/xkissm00/text_dataset_generator



(a)

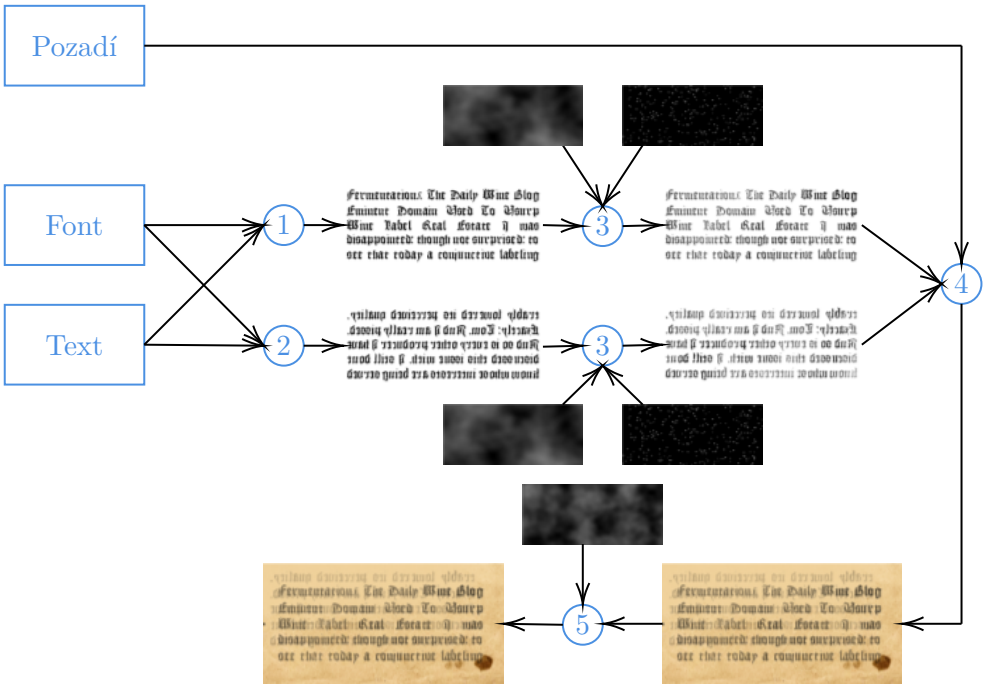


(b)



(c)

Obrázek 4.2: Ukázka řádků testovací sady B–MOD s rozdělením do množin dle složitosti přepisu (a) jednoduché s korektně vytvořenými přepisy textu (b) středně obtížné s maximálně 20 % chybně přepsaných znaků textu (c) velmi obtížné s více jak 20 % chybně přepsaných znaků. Převzato z [1].



Obrázek 4.3: Schéma generování umělých historických stránek. Operace 1 odpovídá vysázení textu, operace 2 vysázení zadního textu. Operace 3 představuje efekt simulující nedokonalosti tisku. Operace 4 nanáší vysázené texty na vybranou texturu a operace 5 přidává rozmazání. Schéma a popis převzat z [21].

rované obrazy, ale také příslušné anotace k jednotlivým znakům a pozici řádků. Výstup však neobsahuje obrazy čistě vysázených textů a samotný čistý text vysázený na jednotlivé řádky. Z tohoto důvodu je potřeba tento generátor upravit.

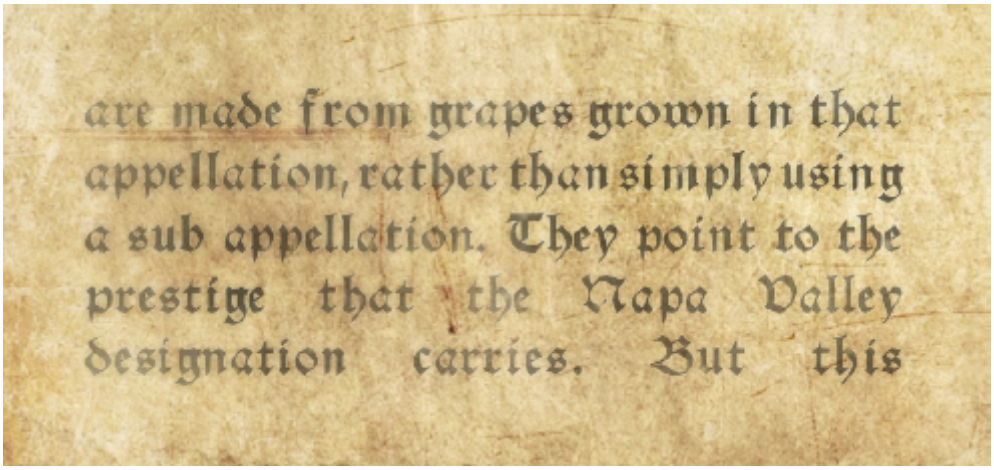
4.2 Sítě pro zlepšení kvality textových obrazů

Navrhnutý model pro zlepšení kvality textových obrazů je založen na poznatcích z článků uvedených v podkapitole 3.2. Většina zmíněných sítí používá pro zpracování kombinací konvolučních a residuálních bloků. Důležitým poznatkem je, že k učení těchto sítí se využívají zdrojové nepoškozené obrazy. Ty nejsou vždy k dispozici a jejich získání nemusí být triviální záležitost. Z tohoto důvodu bude využito architektury GAN, jejíž modely generátoru a diskriminátoru budou vycházet z modelů představených v článku *Image-to-Image Translation with Conditional Adversarial Networks* [14]. Tyto sítě dosáhly vysoké ostrosti výstupního obrazu, která je při práci s textem důležitá.

Vzhledem k problematickému ohodnocení výstupů některých datových sad, jsou zde uvedené informace založeny na použití datové sady vygenerované za pomoci generátoru historických textů. U každého vygenerovaného řádku se pracuje s rozměry 48×512 a se třemi barevnými kanály.

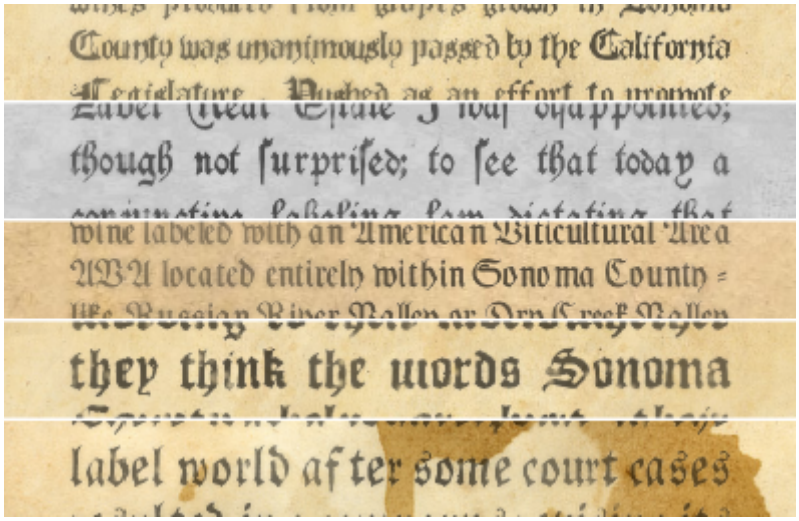
Generátor

Architektura generátoru je založena na modelu U-Net, která se skládá z kodéru a dekodéru. Počet filtrů, použitých v první vrstvě a počet bloků ze kterých se kodér a dekodér skládá, je



(a) Vygenerovaný odstavec historického textu.

(b) Ukázka výstupních dat. První řádek představuje řádek text, druhý zobrazuje ohraničení jednotlivých znaků a třetí ukazuje sémantickou segmentaci textu.



(c) Ukázka možných výstupů.

Obrázek 4.4: Ukázka výstupů generátoru. Obrázky převzaty z [21].

možno ovlivnit vstupními parametry. Ve výchozím nastavení je v počátečním bloku 16 filtrů a celkem se pracuje se čtyřmi bloky kodéru a dekodéru. Každý následující blok kodéru použitý počet filtrů zdvojnásobuje a každý blok dekodéru tento počet snižuje na polovinu. Vstupem generátoru je obraz G_{Input}, u nějž se má zvýšit čitelnost textu jež obsahuje.

Přehled architektury za použití výchozích parametrů určující velikost sítě je v tabulce 4.1. Každý blok kodéru (encoder) je složen z konvoluční vrstvy, normalizační vrstvy v rámci dané dávky (BatchNormalization) a aktivační funkce LeakyReLU. Do prvního bloku kodéru není vložena normalizační vrstva.

Dekodér (decoder) má opačnou úlohu. Pomocí dekonvolučních vrstev se dojde k požadovaným rozměrům aktivačních map, ze kterých je složen výsledný obraz. Každý blok dekodéru je složen z dekonvoluční vrstvy, normalizační vrstvy v rámci dané dávky, tzv. dropout vrstvy a aktivační funkce ReLU. Každý z bloku dekodéru je zakončen spojením výstupních aktivačních map s aktivačními mapami kodéru na stejné úrovni, jež mají totožné rozměry. Tímto se zvýší počet vstupních aktivačních map do dalšího bloku kodéru a doplní se informace ze vstupního obrazu, které mohly být procesem ztraceny. Oproti ostatním blokům dekodéru, poslední blok je složen pouze z dekonvoluční vrstvy.

Blok	Vstup	Filtry / Krok	Vstupní velikost	Výstupní velikost
enc_1	G_{Input}	4×4 / 2	$48 \times 512 \times 3$	$24 \times 256 \times 16$
enc_2	enc_1	4×4 / 2	$24 \times 256 \times 16$	$12 \times 128 \times 32$
enc_3	enc_2	4×4 / 2	$12 \times 128 \times 32$	$6 \times 64 \times 64$
enc_4	enc_3	4×4 / 2	$6 \times 64 \times 64$	$3 \times 32 \times 128$
dec_1	enc_4	4×4 / 2	$3 \times 32 \times 128$	$6 \times 64 \times 128$
dec_2	enc_3, dec_1	4×4 / 2	$6 \times 64 \times 192$	$12 \times 128 \times 64$
dec_3	enc_2, dec_2	4×4 / 2	$12 \times 128 \times 96$	$24 \times 256 \times 32$
dec_4	enc_1, dec_3	4×4 / 2	$24 \times 256 \times 48$	$48 \times 512 \times 3$

Tabulka 4.1: Architektura generátoru.

Diskriminátor

Obdobně jako generátor, architektura diskriminátoru je opět založena na diskriminátoru z uvedeného článku. Jeho základní architektura je i s parametry popsána v tabulce 4.2. Vstupem je zpracovaný text generátorem G_{Output} nebo zdrojový, čistě vysázený textový řádek D_{Input}. Každý z uvedených bloků je složen z konvoluční vrstvy, normalizační vrstvy v rámci dané dávky (BatchNormalization) a aktivační funkce LeakyReLU. Normalizační vrstva není použita v prvním bloku diskriminátoru. Síť je zakončena konvoluční vrstvou, jejíž výstupem je pravděpodobnostní mapa o rozměrech 3×61.

Blok	Vstup	Filtry / Krok	Vstupní velikost	Výstupní velikost
Blok_1	G_{Output}, D_{Input}	4×4 / 2	$48 \times 512 \times 3$	$24 \times 256 \times 32$
Blok_2	Blok_1	4×4 / 2	$24 \times 256 \times 32$	$12 \times 128 \times 64$
Blok_3	Blok_2	4×4 / 2	$12 \times 128 \times 64$	$6 \times 64 \times 128$
Vystup	Blok_3	4×4 / 1	$6 \times 64 \times 128$	$3 \times 61 \times 1$

Tabulka 4.2: Architektura diskriminátoru.

4.3 Metoda vyhodnocení výsledků

Vyhodnocení výstupů sítí je během trénovací a testovací fáze rozdílné. V trénovací fázi se odvíjí od zvolené metody pro výpočet chyby sítě. V testovací fázi je pak testovací datová sada zpracována danou sítí a její výstupy uloženy. Takto předzpracované řádky textu jsou přivedeny na vstup nástroje Tesseract OCR[3], který se v obrázku pokusí rozeznat vysázený text.

V takto získaném textovém řetězci jsou nahrazeny všechny sekvence bílých znaků jednou mezerou. V rámci tohoto vyhodnocení porovnáváme obsah textu a jeho korektní rozdělení po slovech. Nadbytečný počet mezer, odřádkování apod. je pro předání informací nacházejících se v textu nepodstatný. Výstup je za pomoci Levenštejnovy vzdálenosti porovnán s textem, který byl na daný řádek skutečně vysázen. Z takto získaného ohodnocení lze dopočítat procentuální znakovou přesnost přepisu řádků dosazením do rovnice:

$$acc = \frac{l - d}{l} \times 100, \tag{4.1}$$

kde acc označuje vypočítanou přesnost v procentech, l počet znaků v delším z porovnávaných textů a d je Levenštějnova vzdálenost mezi textem vysázeným na řádek a získaným přepisem řádku.

Z takto získaných ohodnocení jednotlivých řádků je proveden průměr, který určuje celkové ohodnocení sítě. Tyto průměry lze použít k porovnání výsledků různých sítí.

[3] https://github.com/tesseract-ocr/tesseract

Kapitola 5

Implementace

Tato kapitola se zabývá implementačními detaily navrženého systému. V úvodní části jsou stručně popsány nástroje, které byly použity pro implementaci, společně s popisem systémů, na kterých bylo provedeno testování a ověřena funkcionalita implementovaného kódu.

V druhé části této kapitoly jsou okrajově představeny implementační detaily pro vytvoření datového setu, jeho použití při trénování a testování neuronových sítí a ohodnocení jejich výsledků. V souboru **Makefile** jsou připraveny náležitě okomentované spustitelné úlohy pro jednotlivou práci s datovými sety, trénováním, testováním a ohodnocením sítí, ale také pro instalaci potřebných Python knihoven.

5.1 Použité nástroje

Tato práce je implementována v jazyce Python, který společně se seznamem použitých knihoven a jejich verzí umožňuje jednoduchou instalaci a funkcionalitu mezi systémy. Pro práci s obrazy byla použita knihovna Pillow a pro práci s matematickými strukturami, daty a výpočty nad nimi knihovna NumPy. K implementaci neuronových sítí, jejich trénování a testování je použit nástroj Keras v rámci knihovny TensorFlow.

TensorFlow

TensorFlow je knihovna pod licencí otevřeného softwaru, sloužící pro vysoce náročné numerické výpočty. Díky flexibilní architektuře je možné ji využít pro výpočty na různých platformách (CPU, GPU, TPU) a zařízeních, ať již stolních počítačích, clusterech nebo mobilních zařízeních. Původně vytvořeno týmem pracovníků Google Brain ve společnosti Google AI. Její součástí je silná podpora pro strojové učení a hluboké učení s dalšími výpočetními technikami pro mnoho dalších vědeckých odvětví [3].

Tento nástroj byl vybrán pro jeho snadné použití v kombinaci s jazykem Python, jeho podpory ze strany uživatelů, ať už vědeckých pracovníků nebo velkých firem, které jej využívají a zajišťují tak další vývoj této knihovny.

Další výhodou TensorFlow je, že implementovaný Python kód, který s touto knihovnou pracuje, je možno spouštět ve volně dostupném online nástroji Colaboratory[1], který je určen pro vzdělávání a výzkum strojového učení.

[1] https://colab.research.google.com/

Implementační prostředí

Práce s neuronovými sítěmi je výkonnostně velmi nákladná, proto se doporučuje k jejich trénování použít knihovny, které umožňují pro výpočty využít GPU daného počítače. Jelikož toto není vždy možné a jelikož mohou nastat problémy s verzí knihoven a daného systému, byla ověřena funkčnost implementovaného řešení v těchto systémech a zařízeních s následujícími parametry.

Hlavní trénování a testování sítí bylo provedeno na herním notebooku s operačním systémem Ubuntu 18.04, procesorem Intel Core i7-7700HQ @ 2,80GHz × 8 a grafickou kartou NVidia GeForce GTX 1050 Ti s pamětí 4096 MB.

Funkčnost implementace byla také ověřena na systému Windows 10 za použití Ubuntu terminálu volně dostupného v Microsoft Store. Dle dostupných informací nelze v tomto prostředí využít prostředků grafické karty pro práci s neuronovými sítěmi, což má za následek navýšení časové náročnosti trénování sítí.

Pro využití externích prostředků je možno využít Google Colab, na němž byly vybrané části implementace testovány. Implementované řešení je však potřeba mírně upravit. Tato služba umožňuje trénování neuronových sítí s využitím GPU, avšak jednotlivá připojení jsou časově omezená. Při využití vlastních dat je možno využít interního úložiště poskytnutého připojení, ale všechna data na tomto úložišti budou ztracena při ukončení spojení. Ztrátě dat lze předejít využitím propojení s Google Drive, ze kterého budou potřebná data načítána, případně na něj zapisována. Tento přístup má několik negativních bodů:

- Všechna potřebná data je potřeba mít na tomto úložišti, které má v neplacené verzi omezenou velikost.
- Prvotní nahrávání na toto úložiště může být časově velice náročné.
- Google Colab má problém při práci se složkami, které obsahují velké množství souborů, a to v řádu několika tisíc.

5.2 Generování datového setu

Generování datového setu lze rozdělit do dvou částí. Generování z dat novinových stránek a pomocí generátoru historických textů.

Datový set novinových výřezů

Před samotným generováním dat je zapotřebí stáhnout stránky sloužící jako zdroj výřezů. K tomuto účelu byl implementován skript **dataset_download.py**, jenž pomocí API poskytnuté Moravskou zemskou knihovnou stáhne množinu stránek v závislosti na zadaných parametrech. Parametry lze určit maximální množství stran, které se mají stáhnout, kořenový uzel a jestli se má zachovat struktura zanoření.

Každý dotaz na tuto API vrátí informace ve formátu JSON, ze kterých jsou následně získány potřebné informace směřující k hledaným stránkám. Data jsou uložena ve stromové struktuře, kde obdržené informace o daném uzlu obsahují:

- **pid** – Jedinečný identifikátor daného uzlu, složen ze 32 znaků šestnáctkové soustavy (0-9, a-f).
- **datanode** – Identifikátor, zda se jedná o listový uzel s daty.

- **root_title** – Jméno kořenového uzlu.
- **model** – Typ modelu reprezentující podstrom daného uzlu.
- **details** – Podrobnější informace o daném uzlu např. rok, číslo,. . . .

Nejedná-li se o konečný uzel, lze opět pomocí API získat potomky tohoto uzlu a proces opakovat. Při dosažení datového uzlu se provede stažení jeho dat, upravení počtu stažených stránek a jejich seznamu, který je v závěru uložen do zvoleného souboru.

Tento soubor je využit ve skriptu **dataset_newspaper_generate.py**, který z těchto stránek vytvoří výřezy o zadaných rozměrech. Na tento výřez se následně aplikuje efekt rozmazání za pomoci Gaussovského filtru a šumu založeného na Poissonově rozdělení pravděpodobnosti. Takto poškozený výřez i jeho nepoškozená verze se uloží do příslušných složek.

V úvodní fázi vývoje byla zamýšlena implementace většího množství efektů, avšak po celkovém upuštění od tohoto datového setu, jehož důvod je popsán v kapitole 6, tento vývoj nepokračoval.

Datový set historických textů

Z důvodu, že nejsem autorem tohoto generátoru, není přiložen ve zdrojových souborech této práce. Pro jeho stažení z GitHub repositáře je v souboru **Makefile** připravena úloha, společně s nakopírováním potřebných souborů do vytvořeného adresáře, vyjma použitých fontů.

Soubor **dataset_generate.py** obsahuje upravené metody tohoto generátoru textů. Do původní implementace byla doplněna logika vrácení vysázeného textu do hlavní části generování, jeho uložení a uložení vygenerovaného řádku společně s jeho čistě vysázenou verzí do příslušných podsložek. Z metody jsou také odstraněny některé části generování dat nepotřebné v této práci.

5.3 Trénování sítí a jejich ohodnocení

V souboru **quality_enhancement.py** se nachází veškerá logika pro trénování a testování implementovaných neuronových sítí. Jak bylo popsáno v návrhu implementace 4.2, struktura sítí je založena na uvedeném článku. Struktura diskriminátoru zůstává neměnná, ale velikost sítě generátoru lze ovlivnit vstupními parametry *Units*, udávající počet filtrů v prvním bloku kodéru, a *Layers*, určující počet bloků kodéru a dekodéru. Maximální počet vrstev je omezen rozměry vstupního obrazu.

Během trénování se zaznamenávají chyby sítí pomocí *TensorBoard*. Použitá metoda pro její výpočet je nastavitelná parametrem *loss*. Změny výpočtu chyby bude využito v experimentech. V průběhu trénování se vytváří a ukládají ukázkové výstupy, a to na základě zvolené frekvence ukládání těchto výstupů, již je možno nastavit parametrem *sample_rate*.

Během trénovací fáze jsou data každé dávky získána pomocí implementované třídy nacházející se v souboru **dataset_provider.py**. Ta na základě informací o datasetu načítá příslušné obrazy, normalizuje je do intervalu <-1;1> a společně s textovými přepisy je vrací do procesu učení sítí.

Testování sítě funguje obdobně, avšak každý vstupní obraz se zpracovává zvlášť, není využita třída **Dataset_provider** a každý zpracovaný obraz je uložen do určené složky.

K ohodnocení testovacích dat je určen skript v souboru **evaluation.py**. Ten pro daný běh načte každý vstupní obraz, získá přepis textu pomocí nástroje Tesseract OCR, jenž pomocí Levenštejnovy vzdálenosti porovná se skutečným textem na daném řádku a vypočte ohodnocení. Získaný přepis a jeho přesnost uloží do výstupního souboru. Souhrnné ohodnocení daného běhu uloží do zvoleného souboru, obsahujícího souhrnné informace všech běhů.

Vytvořené soubory obsahující přepisy testovacích variant je možno dále použít k vyhodnocení pomocí skriptu v souboru **character_counter.py**. Tento skript pro všechny soubory se zadanou příponou vytvoří statistiku výskytů znaků. Ta je užitečná pro porovnání zastoupení jednotlivých znaků mezi sítěmi trénované s různými parametry.

Kapitola 6

Experimenty a vyhodnocení

Cílem uvedených a vyhodnocených experimentů je ověřit funkčnost a požadované chování implementovaných neuronových sítí. Experimenty lze rozdělit do dvou skupin. První má za cíl posoudit vliv parametrů struktury sítě na její výsledné ohodnocení. Druhá skupina obsahuje stejnou strukturu neuronové sítě s měnícími se metodami pro výpočet její chyby během fáze učení. Všechny experimenty jsou vyhodnoceny stejnou metodou a jejich výsledky a porovnání jsou uvedeny v tabulkách na konci jednotlivých sekcí, popisujících danou skupinu experimentů.

Trénovací a testovací datová sada

Od použití datové sady novinových výřezů 4.1.1 bylo v průběhu vývoje upuštěno, a to především z důvodu, že neobsahuje korespondující výstupní data, jež jsou očekávaným výstupem navrhované sítě.

Datová sada novinových výřezů byla použita v úvodní fázi vývoje a seznámením se z neuronovými sítěmi. Vygenerovaný datový set ovšem obsahoval velké množství výřezů okrajů listů a jiných částí, které neobsahovaly žádný text, nebo obsahovaly netextová sdělení v podobě obrázků a nežádoucích komerčních sdělení.

Další problém vyvstal při redukci poškození v datech. Při testování si síť dokázala poradit s vygenerovanými poškozeními, avšak neměla tendenci opravovat poškození, které se vyskytovaly na samotných stránkách. Toto bylo zapříčiněno faktem, že model diskriminátoru rozhodoval mezi výstupy generátoru a výřezy ze stránek, které tyto poškození také obsahovaly. Lze očekávat, že lepších výsledků by se dosáhlo, pokud by diskriminátor rozhodoval mezi výstupy generátoru a výřezy stránek, která obsahují čistě vysázený text bez podkladu a poškození.

Posledním problémem tohoto datového setu byl text samotný. K jednotlivým listům novin jsou poskytnuty textové přepisy, avšak nelze určit, kde se daný text na stránce vyskytuje. U provedených výřezů tak není možno zjistit, jaký text se na nich nachází bez provedení samotné anotace použitých stránek. Z tohoto důvodu nelze výsledky jednoduše ohodnotit zvolenou metodou pro ohodnocení výstupů.

Přestože k datové sadě B–MOD 4.1.2 existují anotované texty, ze kterých je možnost vytvořit chybějící data pro síť diskriminátoru, tato datová sada příliš neodpovídala vytyčenému cíli této práce. Poskytnutá data jsou soustředěna na nekvalitně osvětlený a rozmazaný text, bez poškození textu samotného nebo podkladu na nějž je vytisknut.

K trénování a ohodnocování výsledků neuronových sítí tedy byla použita předvygenerovaná datová sada použitím generátoru popsaného v 4.1.3 s úpravami pro získání potřeb-

ných dat, popsaných v 5.2. Pro sázení textu při generování byla použita písma, dostupná na systému Windows 10. Krok uvedených změn v chování generování, které upravuje nebo nahrazuje některé z nastavení v konfiguračním souboru, a výměně použitých písem, bylo využito zdrojů poskytnutých v rámci tohoto generátoru. Mezi tyto zdroje spadají textury papíru, na které je text vysázen, ale také aplikace poškození textu pro simulaci skutečného tisku, včetně nastavitelných hodnot pro tato poškození.

Jako zdroj vysázeného textu byla využita kolekce děl Johna Bunyana[1], získaná z digitální knihovny projektu Gutenberg [2]. Tato kolekce děl v původní podobě obsahuje 12 041 616 znaků, a to včetně mezer, prázdných řádků a informací o autorském právu k tomuto dílu. Vzhledem k velikosti tohoto dokumentu byla během generování vždy vyjmuta část textu a uložena do samostatného dočasného souboru, který byl následně použit ke generování. Tímto se urychlilo generování datové sady, neboť interní zpracování při generování nemuselo pracovat s celým paměťově objemným textem, ale pouze s jeho momentálně zpracovávanou částí.

Celkově bylo vygenerováno 160 000 trénovacích řádků textu a 9 600 řádků testovacího textu. Všechny vygenerované řádky mají konstantní výšku 48px a šířku 512px. Zachováním rozměrů se předejde deformaci textu, k němuž by docházelo při změně na požadovanou velikost v rámci daného běhu. Každý řádek je uložen ve dvou podobách, a to v podobě čistě vysázeného textu bez pozadí a jeho verze vysázená ve stylu tištěného knižního textu s efekty poškození. V rámci výstupu byl vygenerován soubor s názvy a umístěním jednotlivých řádků a textem, který je na těchto řádcích vysázen.

6.1 Vliv hodnot parametrů sítě

První sada experimentů má za cíl porovnat modely sítí s různými vstupními parametry, jež ovlivňují její strukturu a rychlost učení v trénovací fázi. Pro tento experiment byly zvoleny tyto parametry: Počáteční míra učení (LR – Learning Rate), velikost dávek (BS – Batch Size), počet filtrů (U – Units), počet bloků vrstev (L – Layers) a počet epoch trénování (E – Epochs). Každý parametr je testován pomocí několika variant. Jednotlivé varianty se liší pouze v hodnotě parametru, na který je daná varianta zaměřena. Ostatní parametry mají svou výchozí hodnotu. Přehled hodnot pro dané varianty a parametry je uveden v tabulce 6.1.

Parametr	Míra učení	Velikost dávky	Počet filtrů	Počet vrstev	Počet epoch
Varianta 1	1×10^{-3}	4	4	2	1
Varianta 2	6×10^{-4}	8	8	3	2
Varianta 3	1×10^{-4}	16	16	4	3
Varianta 4	6×10^{-5}	32	32	-	4
Varianta 5	1×10^{-5}	64	64	-	5
Výchozí	1×10^{-4}	16	16	4	1

Tabulka 6.1: Testované parametry a jejich hodnoty v jednotlivých variantách. Pro každý parametr je testováno až 5 různých variant sítí. Hodnoty parametrů těchto sítí jsou ponechány výchozí a je změněna jen hodnota testovaného parametru dle dané varianty. Výchozí hodnoty parametrů jsou uvedeny ve spodní části tabulky.

[1] http://www.gutenberg.org/cache/epub/6049/pg6049.txt

Pro trénování všech variant v této sadě experimentů je zvoleno učení na základě zpětné vazby ze sítě diskriminátoru za pomoci funkce cross entropy pro získání chyby sítě. Natrénované modely, ukázkové výstupy z trénování a výstupy z testování a ohodnocení jsou k dispozici v příslušných složkách na paměťovém médiu této práce. Jejich označení je dáno jako kombinace použité chybové funkce během trénování (GAN), testovaný vstupní parametr (LR, BS, U, L, E) a testovaná varianta (např. Case_1). Pro každý testovaný parametr chybí jeden model s uvedeným pojmenováním. To je dáno tím, že daná hodnota parametru je výchozí, a tedy obsažena v obecném modelu pojmenovaném GAN.

Výsledné ohodnocení jednotlivých variant pro různé parametry jsou zobrazeny v tabulce 6.2, kde tučně zvýrazněné hodnoty zobrazují nejvyšší přesnost pro daný parametr. Je vidět, že různé hodnoty parametrů mohou vést k velmi rozdílným výsledkům. Ovšem je třeba mít na paměti, že uvedené výsledky jsou pro konkrétní sítě. Nelze tedy vzít hodnoty s nejlepším ohodnocením v každé kategorii a očekávat stejné, ne-li lepší výsledky. Jednotlivé parametry jsou vzájemně provázány a obvykle se provádí určitá kompenzace. Například zvolením nižší počáteční hodnoty míry učení lze dosáhnou lepších výsledků, je-li toto pomalejší učení kompenzováno vyšším počtem epoch, což má za následek časově náročnější fázi trénování sítě.

Parametr	Míra učení	Velikost dávky	Počet filtrů	Počet vrstev	Počet epoch
Varianta 1	**93.03 %**	86.91 %	81.91 %	83.13 %	90.75 %
Varianta 2	92.72 %	90.16 %	86.82 %	88.47 %	90.99 %
Varianta 3	90.75 %	**90.75 %**	90.75 %	**90.75 %**	90.86 %
Varianta 4	89.65 %	90.24 %	91.59 %	–	91.72 %
Varianta 5	63.18 %	88.12 %	**93.33 %**	–	**91.91 %**

Tabulka 6.2: Výsledné ohodnocení testovaných parametrů pro jednotlivé varianty. Tučně jsou zvýrazněné hodnoty označující nejvyšší ohodnocení pro daný parametr.

Obvykle se tyto parametry volí s přihlédnutím na úlohu, kterou daná síť řeší. Složitá úloha vyžaduje hlubší síť o velkém množství filtrů, což vyústí v pomalé učení a celkově ve vysokou časovou a paměťovou náročnost.

Optimalizaci sítě pro danou úlohu lze provést obdobně jako probíhalo trénování, testování a ohodnocování v této sadě experimentů. Odlišný přístup bude při přechodu mezi jednotlivými parametry. Po ohodnocení všech variant pro daný parametr se identifikuje nejlépe ohodnocená varianta a její hodnota pro daný parametr se použije při testování všech následujících parametrů. Tímto přístupem lze síť optimalizovat, avšak je opět třeba brát v potaz, že různé parametry mají různý vliv. Pořadí optimalizace jednotlivých parametrů povede k rozdílným výsledným sítím.

Vzhledem k časové náročnosti trénování jednotlivých modelů nebude tento způsob optimalizace testován. V druhé sadě experimentů se bude pracovat se zde získanými ohodnoceními.

6.2 Porovnání chybových funkcí

Druhá sada experimentů je zaměřena na porovnání výsledků sítí při jejichž trénování byly využity různé chybové funkce a jejich kombinace. Hlavní chybová funkce posuzovaná v této práci je cross entropy, která pracuje s výstupem sítě diskriminátoru. Tato chybová funkce bude dále referována pod zkratkou GAN. Ohodnocení tohoto modelu je porovnáno s funk-

cemi pro výpočet MSE (Mean Squared Error) a MAE (Mean Absolute Error), které se často využívají v konvolučních sítích.

Dále byly ohodnoceny modely, které využívají kombinace těchto chyb. Toto je využíváno například v článku, na němž je architektura sítí v této práci založena. Ten využívá kombinaci chyby GAN a MAE jakožto jejich součet. Autoři však poukazují na důležitost hodnoty chyby MAE a to tak, že tuto hodnotu doporučují před součtem vynásobit hodnotou parametru $\lambda = 100$. Celý výpočet této chyby, je pak dán následující rovnicí:

$$chyba = GAN + \lambda \times MAE. \tag{6.1}$$

Tento výpočet je použit jako další testovací varianta v této sadě experimentů pod označením GAN-MAE.

Obdobně je vytvořena varianta využívající chybovou funkci MSE, jejíž výstupy a ohodnocení jsou uloženy pod názvem GAN-MSE a jejíž hodnota násobícího parametru je nastavena $\delta = 100$. Jako poslední je pak zvolena varianta využívající kombinaci všech tří chyb označená GAN-MSE-MAE. Výpočet chyby modelů GAN-MSE a GAN-MSE-MAE je určen následujícími rovnicemi:

$$chyba = GAN + \delta \times MSE \tag{6.2}$$

$$chyba = GAN + \delta \times MSE + \lambda \times MAE. \tag{6.3}$$

Uvedené modely jsou testovány na sítích se třemi variantami vstupních parametrů, určující strukturu sítí a rychlost jejich učení. První z nich se skládá z výchozích hodnot parametrů, jak jsou určeny v kapitole 4.2. Druhá a třetí varianta je odvozena z výsledků experimentů v podkapitole 6.1, kde je pro každý parametr z těchto testů zvolena hodnota produkující nejlepší hodnocení. Rozdíl mezi těmito variantami je počet epoch trénování, kde pro druhou variantu je tento počet nastaven na 1 epochu, kdežto pro třetí variantu je nastavena hodnota 5, která opět vykazovala nejlepší hodnocení pro tento parametr. Jak již bylo zmíněno v závěru podkapitoly 6.1 u modelů, jejichž hodnoty parametrů byly získány tímto způsobem, nelze očekávat nejlepší výsledky, neboť jednotlivé parametry se vzájemně ovlivňují. Přehled testovaných variant a hodnot parametrů sítí je uveden v tabulce 6.3.

Model	Míra učení	Velikost dávky	Počet filtrů	Počet vrstev	Počet epoch
Varianta 1	1×10^{-4}	16	16	4	1
Varianta 2	1×10^{-3}	16	64	4	1
Varianta 3	1×10^{-3}	16	64	4	3

Tabulka 6.3: Parametry sítí testovaných variant.

Natrénované modely a jejich výstupy jsou uloženy pod názvem použité kombinace chybových funkcí. Pro druhou a třetí variantu této sady experimentů jsou názvy doplněny o dodatek OPT (optimalizováno), případně OPT-E, kde E označuje vyšší počet epoch během trénování.

Výsledná ohodnocení pro tuto sadu experimentů jsou uvedena v tabulce 6.4. Z těchto hodnot lze vyčíst, že varianta GAN zdaleka nedosahovala přesnosti jako ostatní modely. V třetí variantě dosáhl výrazně horších výsledků než ve variantách předchozích. Ostatní modely v každé z variant dosáhly lepších výsledků. Ohodnocení kombinace chybových funkcí jsou v první a třetí variantě nižší, než výsledky modelů MSE a MAE. Naopak ve variantě druhé jsou tyto výsledky rovnocenné nebo lepší než při použití samostatných chybových

Ztrátová funkce	Varianta 1	Varianta 2	Varianta 3
GAN	90.75 %	93.23 %	88.66 %
MSE	95.51 %	97.02 %	97.70 %
MAE	94.74 %	97.12 %	97.62 %
GAN-MSE	94.83 %	97.06 %	97.30 %
GAN-MAE	94.45 %	97.09 %	97.40 %
GAN-MSE-MAE	94.99 %	97.13 %	97.54 %

Tabulka 6.4: Výsledné ohodnocení modelů trénovaných za použití různých ztrátových funkcí a jejich kombinací.

funkcí. Je možné, že správnou optimalizací parametrů by se pro tyto modely dosáhlo větších rozdílů.

Při vyhodnocování úspěšnosti je také třeba brát v úvahu ohodnocení vstupních řádků a řádků, kde byl daný text vysázen bez aplikace efektů či pozadí. Oba tyto datové sety řádků byly vyhodnoceny zcela stejným způsobem, jako jsou hodnoceny výsledky zpracované sítěmi. Čistě vysázený text byl ohodnocen průměrnou přesností 98.27 %. Je zřejmé, že použitý OCR nástroj má problém korektně zpracovat i čistě vysázený text. Řádky použité jako vstup neuronových sítí byl ohodnocen průměrnou přesností 65.61 %. Při porovnání těchto přesností s výsledky jednotlivých modelů zjistíme, že žádný z modelů nedosáhl přesnosti čistě vysázeného textu. Pozoruhodné je ovšem zlepšení ohodnocení v porovnání se vstupními řádky, kde u všech sítí bylo dosaženo zlepšení o více jak 20 %.

Využitím výstupních přepisů z testování lze nahlédnout na problematické znaky, které způsobují problémy a negativně tak ovlivňují výsledné hodnocení. Hodnoty v tabulce 6.5 jsou zaměřeny na výstupní přepisy modelů GAN a jeho jednotlivé varianty parametrů. Pro srovnání jsou uvedeny i skutečné počty znaků vysázeného textu na testovacích řádcích. V tabulce jsou uvedeny pouze znaky, jejichž počet je pro variantu GAN-OPT-E oproti skutečnému počtu vysázených znaků menší nebo větší než 100. Řada detekovaných znaků se v původním textu vůbec nevyskytuje. Některé z nich spadají do výběru uvedeném v tabulce, avšak nelze je korektně zobrazit v této práci. Celkový počet těchto znaků je do jednoho tisíce.

V uvedeném přehledu si lze všimnout, že nejvyšší počet chybné detekce je u velmi malých znaků. Počet detekovaných znaků „.“ je přibližně o stejný počet vyšší, jako nedetekovaných znaků „,“. V tomto případě je možné, že při zpracování řádků sítí, nemá síť dostatečné množství informací, aby vyhodnotila, o který ze znaků se má jednat. Jelikož se oba znaky v běžném textu vyskytují, nemusí je diskriminátor považovat za chybné a předat tuto informací zpět v podobě chyby. Při porovnání hodnot těchto znaků mezi druhou a třetí variantou, se lze domnívat, že při delším učení, ať již na stejných nebo rozdílných datech, začal být jeden ze znaků systematicky ignorován ve prospěch znaku druhého. Pevné rozhodnutí, zdali tomu tak skutečně je, by vyžadovalo další testování.

U jiných znaků je obtížnější vyhodnotit, co způsobuje jejich chybnou detekci. Roli můžou hrát poškození vstupních řádků, díky kterým jsou dané znaky zaměněny. Ruční kontrolou přepisů bylo například zjištěno, že použité OCR detekuje znak „I“ jako znak „|“. Na obrázcích 6.1 jsou ukázky problematických výstupů sítí.

Znak	Skutečný výskyt	GAN	GAN-OPT	GAN-OPT-E	Absolutní rozdíl
.	1 594	2 863	2 461	2 983	1 389
,	6 355	5 027	5 459	4 973	1 382
f	5 568	5 102	5 120	4 753	815
i	14 723	14 608	14 688	13 908	815
e	27 879	27 398	27 615	28 596	717
g	3 824	3 706	3 749	3 221	603
o	17 255	16 319	16 791	17 777	522
h	18 047	17 127	17 546	17 561	486
v	2 252	2 536	2 401	2 724	472
p	2 868	2 926	2 821	2 436	432
a	17 024	17 291	17 146	17 427	403
s	14 879	14 748	14 767	14 478	401
r	12 853	12 802	12 683	12 513	340
d	9 893	9 663	9 571	9 593	300
u	6 058	5 832	5 947	5 766	292
c	4 519	4 683	4 755	4 777	258
l	9 004	8 382	8 675	8 842	162
b	3 573	3 528	3 484	3 432	141
’	316	356	320	451	135
m	4 434	4 442	4 403	4 559	125
w	4 631	4 513	4 529	4 516	115
|	0	94	95	110	110

Tabulka 6.5: Celkové výskyty vybraných znaků ve výstupních přepisech modelů GAN pro jednotlivé varianty parametrů a skutečně vysázené počty znaků na testovacích řádcích. Jsou vypuštěny všechny znaky, jejichž počet pro variantu GAN-OPT-E je oproti skutečnému počtu vysázených znaků menší nebo větší než sto. Znaky jsou seřazeny od nejvyššího počtu rozdílného výskytu po nejnižší.

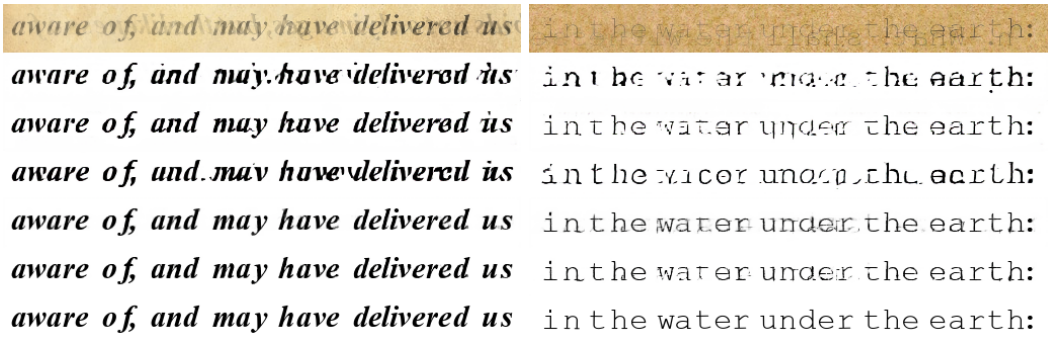
6.3 Shrnutí výsledků

Z výsledků experimentů je patrné, že vhodně zvolené parametry sítě mohou navýšit její úspěšnost, avšak k této optimalizaci je potřeba přistupovat systematicky. Po provedení experimentů v rámci jednoho parametru a zvolení hodnoty produkující nejlepší výsledky je potřeba tuto hodnotu zafixovat a dále neměnit. Přestože toto pravidlo nebylo v rámci první sady experimentů dodrženo, z výsledků uvedených v tabulce 6.2 je patrné, že změnou parametrů v druhé variantě došlo ke zlepšení ohodnocení. Nedodržení zmíněného pravidla se projevila doplněním hodnoty pro počet trénovacích epoch, která vykazovala nejlepší hodnocení v rámci testů pro tento parametr. Lze vidět, že tento přístup není možno považovat za optimalizaci.

Při srovnávání výsledků různých chybových funkcí žádná nedosáhla ohodnocení srovnatelné nebo lepší než ohodnocení čistě vysázeného textu. Hodnocení výstupů některých modelů se k tomuto hodnocení velmi přiblížily, a to s rozdílem menším než 1 %. Dosáhlo se také zajímavých výsledků při kombinaci více chybových metod. Sítě s vhodnými parametry a kombinaci chybových funkcí, dosahovaly výsledků na stejné úrovni jako sítě používající tyto výpočty chyby odděleně. Jednotlivé chybové funkce tedy na sebe vzájemně neměly negativní vliv.

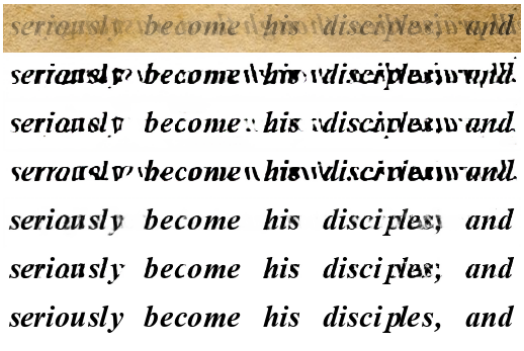
Modely trénované pouze za pomoci diskriminátoru nedosahovaly výsledků jako jiné modely, které pro své učení využívaly i očekávaný výstup. Přesto se u takto zpracovaných řádků dosáhlo hodnocení, které je o 25.14 % vyšší než u při vyhodnocení nezpracovaných řádků.

Na obrázcích 6.2 a 6.3 je ukázáno zpracování výřezu novinové stránky. Tento výřez obsahuje různé prvky, na kterých je možno posoudit kvalitu zpracování. Je vidět, že vizuálně čitelný text nebyl zpracován korektně. Síť si dokázala poradit s různou velikostí textu, a i obrázek si zachoval hrubou strukturu. Naopak v hlavním textu tohoto výřezu místy zcela chybí části písmen. To se projevuje téměř u všech textů psaných kurzívou. U jiných znaků může být příčinou učení sítě na jiném stylu písma a české znaky, které se na tomto výřezu nalézají, ale nebyly použity při trénování sítě. Na zpracování také může hrát roli výška jednotlivých řádků. V trénovací a testovací datové sadě jsou výšky řádků 48px. V tomto textu je řádek vysoký přibližně 38px. Je tedy vidět, že obsah trénovacích dat omezuje aplikaci dané sítě.

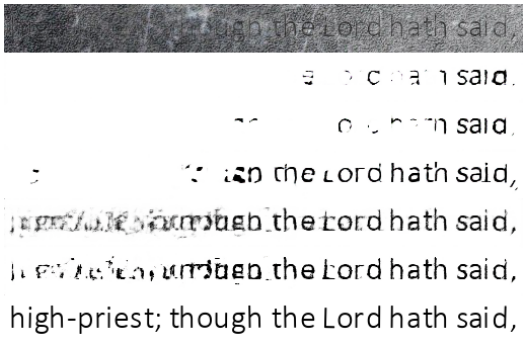


(a) Řádka č. 160 308

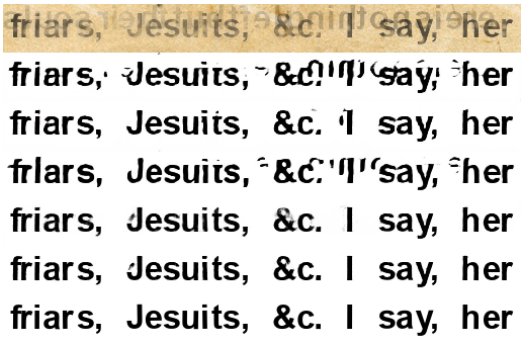
(b) Řádka č. 160 459



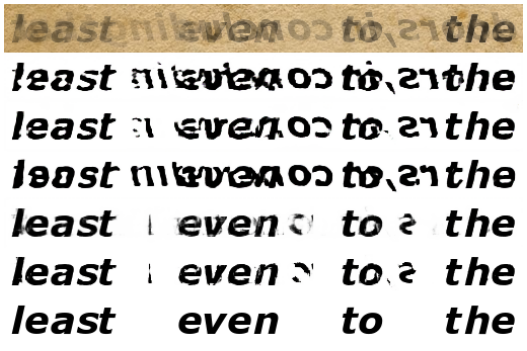
(c) Řádka č. 160 510



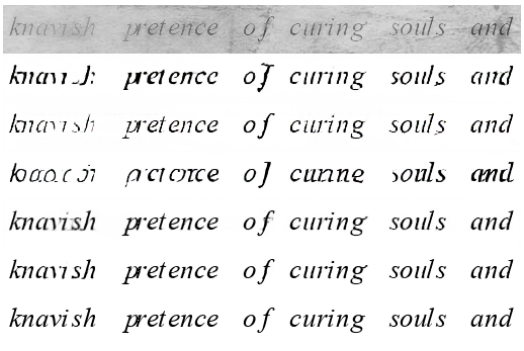
(d) Řádka č. 160 620



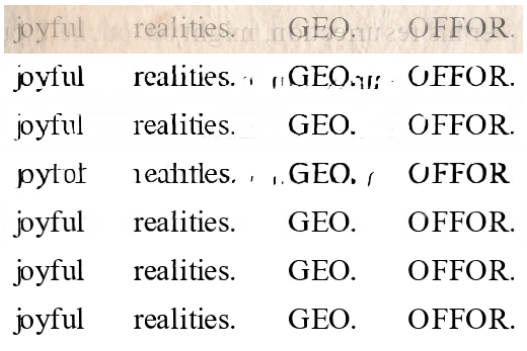
(e) Řádka č. 160 758



(f) Řádka č. 161 073



(g) Řádka č. 161 770



(h) Řádka č. 162 198

Obrázek 6.1: Ukázka výstupu testovacích dat. V každém obrázku jsou od shora dolů tyto řádky: Vstupní řádek, GAN, GAN-OPT, GAN-OPT-E, MSE-OPT, MAE-OPT a vygenerovaný neporušený řádek.

Po účinku zlá rada! Lev, oloupen o vše, co lvem jej činilo, byl konečně rád, že alespoň tak s tím životem vyváznul, i bloudil světem jako zmoklá slepice, nikde ni rady ni pomoci nenalezaje. Nehoda konečně zavedla jej v den sv. Sylvestra; právě když starý rok novému roku tajuplně ruku podává, do redakce „Humoristických listův“, i byl chuděra! nad míru potěšen, že má komu se svěřiti a si postěžovati. Měli jste ho ale vidět, co dělal, když po úplném vyslechnutí jeho žalob a nářku vydavatel »Hum. listů« jemu mezi čtyrma očima, aby to donášeči neslyšeli, sdělil, že umí připravovati mast, jížto když se někdo natře, jemu i hned vše chybící opět naroste. Lev div že se radostí nezbláznil a redaktora »Hum. l.« samou vděčnosti neumačkal, i chtěl stante pede divotvornou mast tu míti. Inu jo! jemu se to snadno křáplo! Divotvorná mast není jen tak leda co a nenajde se všude za plotem. K divotvorné masti je přede vším třeba vzácného, a tudíž drahého koření; k zakoupení drahého koření ale jest opět přede vším jiným peněz zapotřebí. Poněvadž ale čeští redaktoři do posud — jak vůbec známo — spíše hojnost v nedůstatku mají, byl by milý lev náš nepochybně nucen býval, pozdovu bez pazourů, tesáků a ohonu s kokrhelem na hlavě světem se potloukati, kdyby vydávatel »H. l.«, veden jsa známou spanilomyslností svou, 1000 výtisků letošního ročníku „Listův hum.“ nebyl na zakoupení drahého koření toho obětoval. *Kdožkoli edy na venkově III. ročník „Listův humoristických“ odebere, resp. celoročně 5 zlatými 70 kr., půll. 3 zl. 85 kr. a čtvrtletně 1 zl. 45. kr. r. č. se předplatí a předplatné vydavatelstvu „H. l“ do Prahy do Uršulinské ulice čís. 140—II franco zašle*, nezavděčí se pouze našemu milému lvu, nobrž i sobě, neboť obdrží mimo 52 archová, nápady a výpady nabitá čísla ještě i *dvě prœmie zcela zdarma*: obraz totiž a kalendář na stěnu na r. 1861, oboje v ceně 2 zl. 75 kr. r. č. A že nápotom, jak mile potřebné k zakoupení drahého koření peníze pohromadě budou, i léčení lva řádně a rozšafně diti se bude, ručí důstatečně *konsilium doktorův humoru a vtipu*, jejichžto pomoci jsme si k cíli tomu schvalně byli vyprosili, a mezi nimiž jsou mužové v ohledu tom světové pověsti požívající, jako na př. P. T. pánové: *Bělák, Benál, Burgerstein, Drahňovský, Gabriel, Gollat, Grov, Hajniš, Hálek, Košin z Radostova, Kouble, Kulda, Mirohorský, Neruda, Peška, Pfleger, Přerhof, Starý, Svátek, Štrauch, Tonner, Trubelka* a. j. v., kteří co *lékařští assistenti* operaci též přítomni jsou. 2—1

Komu tedy na tom záleží, aby lev náš přestal býti terčem vtipu trpaslíků zlých a aby v krátkosti předešlého opět zdraví a brnění nabyl, *předplať se na III. ročník „Listův humoristických“*, neboť tak jenom umožní se připravování divotvorné masti

vydavatelstvu „humoristických listů“.

v Praze, v Uršulinské ulici, čís. 140—II.

Douška. Do 20. ledna 1861 bere vydavatelstvo „H. l.“ i *staré jednušky*, tak zvané „zlatky stříbra“ v celé jejich hodnotě, t. j. za 1 zl. 5 kr, r. č., a však jen *co předplatné* na „L. h.“ zároveň také oznamujeme, že — dojde-li nás povolení — ještě během tohoto čtvrtletí „H. l.“ *dvakrát týdně* co list *politicko-humoristický bez zvýšení předplatného* vydávati počneme.

Josef Richard Vilímek,

redaktor a nakladatel.

F. DURST,

mechanikus a optikus

v Praze v ovocné ulici č. 953 vedle kavárny Böhmovy odporučuje svůj sklad 8—1

brejlí, lorňetů, dalekohledů, tlakoměrů, teploměrů,

a jiného podobného zboží všeho druhu, též správky i zhotovování optických a mechánických nástrojů.

LITOGRAFIE

FARSKÉHO

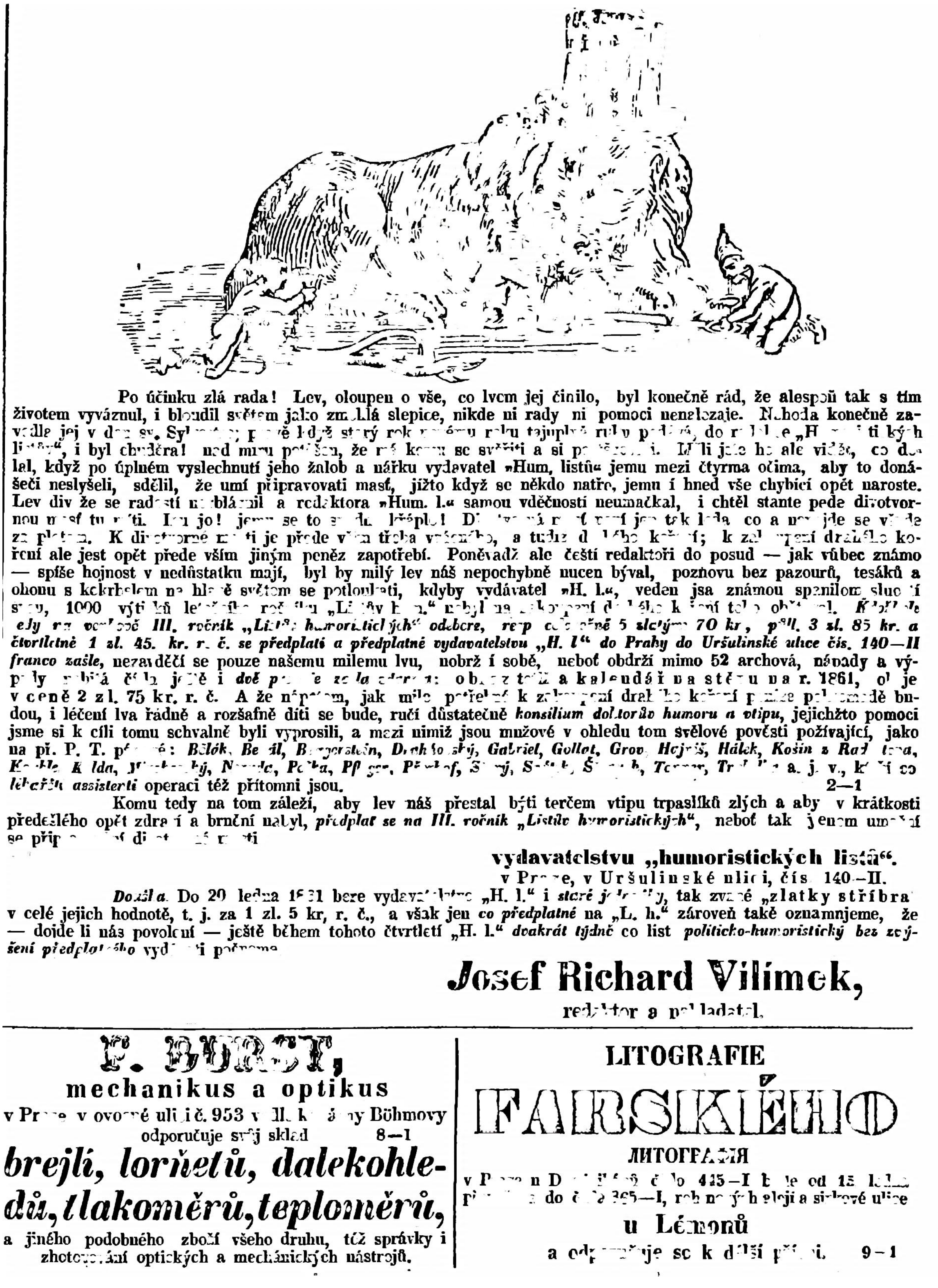
ЛИТОГРАФІЯ

v Praze u Dominikánů číslo 445—I bude od 15. ledna přenešena do čísla 365—I, roh nových alejí a sirkové ulice

u Lémonů

a odporučuje se k další přízni. 9—1

Obrázek 6.2: Testovaný vstupní výřez novinové stránky, jehož zpracovaný výstup je na obrázku 6.3.



Po účinku zlá rada! Lev, oloupen o vše, co lvem jej činilo, byl konečně rád, že alespoň tak s tím životem vyváznul, i bloudil světem jako zmoklá slepice, nikde ni rady ni pomoci nenalezaje. Náhoda konečně zavedla jej v den sv. Sy[illegible] právě když starý rok [illegible] roku tajuplné ruky [illegible], do r[illegible] „H[illegible] listův“, i byl chuděra! nad míru p[illegible], že [illegible] sc svěřiti a si p[illegible]. [illegible] ale viděc, co dělal, když po úplném vyslechnutí jeho žalob a nářku vydavatel „Hum. listů“ jemu mezi čtyrma očima, aby to donášeči neslyšeli, sdělil, že umí připravovati mast, jížto když se někdo natře, jemu i hned vše chybící opět naroste. Lev div že se rad[illegible] a redaktora „Hum. l.“ samou vděčností neumačkal, i chtěl stante pede divotvornou [illegible] jo! [illegible] se to [illegible] k divotvorné [illegible] je přede v[illegible] třeba [illegible], a tudíž d[illegible] k[illegible]; k [illegible] drahého koření ale jest opět přede vším jiným peněz zapotřebí. Poněvadž ale čeští redaktoři do posud — jak vůbec známo — spíše hojnost v nedůstatku mají, byl by milý lev náš nepochybně nucen býval, poznovu bez pazourů, tesáků a ohonu s kekrhelem na hl[illegible] světem se potloukati, kdyby vydávatel „H. l.“, veden jsa známou spanilom[illegible] s[illegible], 1000 výt[illegible] „Li[illegible]“ n[illegible] k[illegible]. *K[illegible] ely na [illegible] III. ročník „Lis[illegible] humoristických“ odebere, resp [illegible] 5 zl. [illegible] 70 kr, p[illegible] 3 zl. 85 kr. a čtvrtletně 1 zl. 45. kr. r. č. se předplatí a předplatné vydavatelstvu „H. l“ do Prahy do Uršulinské ulice čís. 140—II franco zašle,* nezavděčí se pouze našemu milemu lvu, nobrž i sobě, neboť obdrží mimo 52 archová, nápady a výp[illegible] je[illegible] i *[illegible]*: o b[illegible] a k a l e n d á ř n a s t ě [illegible] u n a r. 1861, o[illegible] je v c e n ě 2 z l. 75 kr. r. č. A že n[illegible], jak m[illegible] p[illegible] k [illegible] drah[illegible] koření p[illegible] budou, i léčení lva řádně a rozšafně díti se bude, ručí důstatečně *konsilium doktorův humoru a vtipu,* jejichžto pomoci jsme si k cíli tomu schvalně byli vyprosili, a mezi nimiž jsou mužové v ohledu tom světové pověsti požívající, jako na př. P. T. p[illegible]: *Bělák, Be[illegible], B[illegible], D[illegible], Gabriel, Gollet, Grov[illegible] Hejr[illegible], Hálek, Košin z Rad[illegible], K[illegible], K[illegible]lda, [illegible], N[illegible], Po[illegible]a, Pf[illegible], P[illegible], S[illegible], S[illegible], Š[illegible], T[illegible], Tr[illegible]* a. j. v., k[illegible] co *lékařům assistenti* operaci též přítomni jsou. 2—1

Komu tedy na tom záleží, aby lev náš přestal býti terčem vtipu trpaslíků zlých a aby v krátkosti předešlého opět zdra[illegible] a brnční nabyl, *předplať se na III. ročník „Listův humoristických“*, neboť tak j[illegible] se přip[illegible] di[illegible]

vydavatelstvu „humoristických listů“.
v Pr[illegible]e, v Uršulinské ulici, čís. 140—II.

Dožl a. Do 20 ledna 1861 bere vyd[illegible] „H. l.“ i *staré j[illegible]*, tak zvané „zlatky stříbra“ v celé jejich hodnotě, t. j. za 1 zl. 5 kr, r. č., a však jen *co předplatné* na „L. h.“ zároveň také oznamujeme, že — dojde li nás povolení — ještě během tohoto čtvrtletí „H. l.“ *dvakrát týdně* co list *politicko-humoristický bez zvýšení předpla[illegible]* vyd[illegible] p[illegible]

Josef Richard Vilímek,
red[illegible] a [illegible]

F. B[illegible],
mechanikus a optikus
v Pr[illegible] v ov[illegible] ul. č. 953 [illegible] Böhmovy
odporučuje s[illegible] sklad 8—1
brejlí, lornětů, dalekohledů, tlakoměrů, teploměrů,
a jiného podobného zboží všeho druhu, též správky i zhotov[illegible] optických a mechánických nástrojů.

LITOGRAFIE
FARSKÉHO
ЛИТОГРАФИЯ
v P[illegible] n D[illegible] č. [illegible] 435—I [illegible] od 15 [illegible] do č. [illegible]—I, [illegible] u Lémonů
a od[illegible] se k d[illegible] 9—1

Obrázek 6.3: Zpracovaný výřez novinové stránky neuronovou sítí s názvem GAN-OPT. Vstupní obrázek sítě je na obrázku 6.2.

Kapitola 7

Závěr

V rámci této práce je probrán přístup k rozpoznávání textů v obraze a postup při získávání jeho přepisu. Detailněji byly analyzovány metody pro předzpracování obrazu, jaké techniky a architektury využívají a jakým způsobem se sítě trénují. Na základě získaných informací byla zvolena architektura založena na modelu GAN, která byla následně implementována.

Na implementovaných sítích byly provedeny dvě sady experimentů. První z nich měla za cíl ohodnotil vliv jednotlivých parametrů modelu na jeho zpracování textu. Přesnost přepisu nezpracovaných dat dosáhl přesnosti 65.61 %, řádky textu předzpracované sítí s výchozími parametry dosáhly přesnosti 90.75 %. Dosáhlo se tedy navýšení o 25.14 %. Experimenty bylo zjištěno, že největší vliv na výsledky má počáteční míra učení a počet filtrů sítě. Síť, jejíž hodnoty parametrů byly zvoleny na základě výsledků experimentů, dosáhla navýšení přesnosti o 2.48 % na 93.23 %. Avšak, byla-li síť trénována počtem epoch, který opět vykazoval nejvyšší úspěšnost pro svůj parametr, klesla přesnost na 88.66 %. Je nutno podotknout, že hodnoty parametrů v tomto experimentu nebyly získány korektním postupem, jímž by se získaly optimální parametry.

Druhá sada experimentů porovnává přesnost sítí trénovaných za použití rozdílných výpočtů chyby. Síť s upravenými parametry trénovaná pomocí PatchGAN diskriminátoru dosahovala přesnosti 93.23 %. Síť pro jejíž výpočet chyby během trénovací fáze byla použita chybová funkce MSE, dosáhla přesnosti 97.02 %, a síť s chybovou funkcí MAE 97.12 %. Tato přesnost se při zvýšení počtu trénovacích epoch ještě zvýšila na 97.7 % pro MSE a 97.62 % pro MAE.

V rámci další práce by bylo vhodné nalézt optimální parametry sítě generátoru. Síť s optimálními parametry a učením pomocí diskriminátoru může dosáhnout přesnosti na úrovni jiných testovaných chybových funkcí.

Zajímavé by také bylo rozšířit tuto síť o části snažící se nalézt jednotlivé znaky, které budou separátně extrahovány a opraveny. Okolí znaků by se v tomto případě ignorovalo. Síť by se tedy nesoustředila na zpracování obrazu jako celku, ale pouze na jednotlivé znaky textu. Tento přístup však již pracuje na podobné úrovni jako klasifikační část rozpoznávání textu.

Literatura

[1] Projekt PERO.
URL https://pero.fit.vutbr.cz/

[2] Project Gutenberg. 1971.
URL http://www.gutenberg.org

[3] Abadi, M.; Agarwal, A.; Barham, P.; aj.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015, software available from tensorflow.org.
URL https://www.tensorflow.org

[4] Bahdanau, D.; Cho, K.; Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. 2014, arXiv:1409.0473.

[5] Bartz, C.; Yang, H.; Meinel, C.: STN-OCR: A single Neural Network for Text Detection and Text Recognition. *CoRR*, ročník abs/1707.08831, 2017, 1707.08831.
URL http://arxiv.org/abs/1707.08831

[6] Berthelot, D.; Schumm, T.; Metz, L.: BEGAN: Boundary Equilibrium Generative Adversarial Networks. 2017, arXiv:1703.10717.

[7] Chen, K.; Seuret, M.: Convolutional Neural Networks for Page Segmentation of Historical Document Images. *CoRR*, ročník abs/1704.01474, 2017, 1704.01474.
URL http://arxiv.org/abs/1704.01474

[8] Dong, C.; Deng, Y.; Loy, C. C.; aj.: Compression Artifacts Reduction by a Deep Convolutional Network. *CoRR*, ročník abs/1504.06993, 2015, 1504.06993.
URL http://arxiv.org/abs/1504.06993

[9] Galteri, L.; Seidenari, L.; Bertini, M.; aj.: Deep Generative Adversarial Compression Artifact Removal. *CoRR*, ročník abs/1704.02518, 2017, 1704.02518.
URL http://arxiv.org/abs/1704.02518

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

[11] Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; aj.: Generative Adversarial Networks. 2014, 1406.2661.

[12] Heusel, M.; Ramsauer, H.; Unterthiner, T.; aj.: GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. 2017, arXiv:1706.08500.

[13] Hradiš, M.; Kotera, J.; Zemčík, P.; aj.: Convolutional Neural Networks for Direct Text Deblurring. In *Proceedings of BMVC 2015*, The British Machine Vision Association and Society for Pattern Recognition, 2015, ISBN 1-901725-53-7, s. 1–13, doi:10.5244/C.29.6.
URL https://www.fit.vut.cz/research/publication/10922

[14] Isola, P.; Zhu, J.; Zhou, T.; aj.: Image-to-Image Translation with Conditional Adversarial Networks. *CoRR*, ročník abs/1611.07004, 2016, 1611.07004.
URL http://arxiv.org/abs/1611.07004

[15] Jiří Holčík, M. K. a. k.: Matematická biologie: e-learningová učebnice. 2015.
URL http://portal.matematickabiologie.cz

[16] Kai Wang; Babenko, B.; Belongie, S.: End-to-end scene text recognition. In *2011 International Conference on Computer Vision*, 2011, s. 1457–1464.

[17] Karatzas, D.; Shafait, F.; Uchida, S.; aj.: ICDAR 2013 Robust Reading Competition. In *2013 12th International Conference on Document Analysis and Recognition*, 2013, s. 1484–1493.

[18] Karpathy, A.: CS231n Convolutional Neural Networks for Visual Recognition.
URL http://cs231n.stanford.edu/

[19] Kingma, D. P.; Ba, J.: Adam: A Method for Stochastic Optimization. 2014, arXiv:1412.6980.

[20] Kiss, M.; Hradis, M.; Kodym, O.: Brno Mobile OCR Dataset. *CoRR*, ročník abs/1907.01307, 2019, 1907.01307.
URL http://arxiv.org/abs/1907.01307

[21] Kišš, M.: *Rozpoznávání historických textů pomocí hlubokých neuronových sítí*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2018.
URL https://www.fit.vut.cz/study/thesis/20898/

[22] Kohút, J.: *Aktivní učení pro rozpoznávání textu*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2019.
URL https://www.fit.vut.cz/study/thesis/22021/

[23] Krizhevsky, A.: Learning multiple layers of features from tiny images. Technická zpráva, 2009.

[24] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, editace F. Pereira; C. J. C. Burges; L. Bottou; K. Q. Weinberger, Curran Associates, Inc., 2012, s. 1097–1105.
URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[25] LeCun, Y.; Cortes, C.: MNIST handwritten digit database. 2010.
URL http://yann.lecun.com/exdb/mnist/

[26] Maleki, D.; Nadalian, S.; Derakhshani, M. M.; aj.: BlockCNN: A Deep Network for Artifact Removal and Image Compression. *CoRR*, ročník abs/1805.11091, 2018, 1805.11091.
URL http://arxiv.org/abs/1805.11091

[27] Mishra, A.; Alahari, K.; Jawahar, C.: Scene Text Recognition using Higher Order Language Priors. In *Proceedings of the British Machine Vision Conference*, BMVA Press, 2012, ISBN 1-901725-46-4, s. 127.1–127.11, doi:http://dx.doi.org/10.5244/C.26.127.

[28] Radford, A.; Metz, L.; Chintala, S.: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. 2015.
URL http://arxiv.org/abs/1511.06434

[29] Ronneberger, O.; Fischer, P.; Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*, ročník abs/1505.04597, 2015, 1505.04597.
URL http://arxiv.org/abs/1505.04597

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

[31] Simonyan, K.; Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 1409.1556*, 09 2014.

[32] Svoboda, P.; Hradis, M.; Marsik, L.; aj.: CNN for License Plate Motion Deblurring. *CoRR*, ročník abs/1602.07873, 2016, 1602.07873.
URL http://arxiv.org/abs/1602.07873

[33] Xie, J.; Xu, L.; Chen, E.: Image Denoising and Inpainting with Deep Neural Networks. In *Advances in Neural Information Processing Systems 25*, editace F. Pereira; C. J. C. Burges; L. Bottou; K. Q. Weinberger, Curran Associates, Inc., 2012, s. 341–349.
URL http://papers.nips.cc/paper/4686-image-denoising-and-inpainting-with-deep-neural-networks.pdf

[34] Zhao, Z.-Q.; Zheng, P.; tao Xu, S.; aj.: Object Detection with Deep Learning: A Review. 2018, arXiv:1807.05511.

Příloha A

Obsah SD

datasets/ Datová sada pro trénování a testování neuronových sítí

documentation/ Zdrojové dokumenty k překladu textové části práce

checkpoints/ Záchytné body sítí vytvořené během trénování

models/ Natrénované modely sítí

outputs/ Testovací set a výsledky testování modelů a jejich ohodnocení

page/ Testovací výřez novinové stránky

samples/ Ukázkové výstupy vytvořené v průběhu trénování

scripts/ Implementované zdrojové kódy

slides/ Zdrojové dokumenty k překladu prezentace pro demonstraci výsledků práce

summaries/ Záznamy chybových hodnot z průběhu trénování

video/ Prezentace a video s demonstrací výsledků práce

dp.pdf Elektronická verze této práce

character_count.csv Statistiky výskytů jednotlivých znaků v přepisech testovacích řádků

JohnBunyana.txt Kolekce děl autora Johna Bunyana ke generování dat

Makefile Spustitelné úlohy pro implementované chování

README.md Stručné informace o instalaci, obsahu složek apod.

requirements.txt Seznam knihoven použitých v této práci

run_model.py Hlavní zdrojový soubor spouštějící trénování a testování neuronových sítí

text.txt Seznam anglických slov