



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

**AUTOMATICKÉ ROZPOZNÁVÁNÍ MATEMATICKÝCH
VÝRAZŮ POMOCÍ NEURONOVÝCH SÍTÍ**

NEURAL NETWORKS FOR AUTOMATIC EQUATION RECOGNITION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. VLADISLAV HALVA

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. MICHAL HRADIŠ, Ph.D.

BRNO 2022

Zadání diplomové práce



Student: **Halva Vladislav, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Strojové učení
Název: **Automatické rozpoznávání matematických výrazů pomocí neuronových sítí
Neural Networks for Automatic Equation Recognition**
Kategorie: Zpracování obrazu
Zadání:

1. Prostudujte základy konvolučních sítí, sítí typu BERT a reprezentace matematických výrazů.
2. Vytvořte si přehled o současných metodách automatického přepisu matematických výrazů z obrazu.
3. Vyberte nebo navrhněte metodu schopnou automaticky extrahovat matematické výrazy z obrazu.
4. Obstarejte si databázi vhodnou pro experimenty. Můžete rozšířit existující databázi.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Wenqi Zhao, Liangcai Gao, Zuoyu Yan, Shuai Peng, Lin Du and Ziyin Zhang: Handwritten Mathematical Expression Recognition with Bidirectionally Trained Transformer, ICDAR, 2021.
- Shuai Peng, Liangcai Gao, Ke Yuan and Zhi Tang: Image to LaTeX with Graph Neural Network for Mathematical Formula Recognition, ICDAR, 2021.
- Dan Anitei, Joan Andreu Sanchez, Jose Manuel Fuentes, Roberto Paredes, Jose Miguel Bened: ICDAR 2021 Competition on Mathematical Formula Detection, ICDAR, 2021.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz <https://www.fit.vut.cz/study/theses/>

Vedoucí práce: **Hradiš Michal, Ing., Ph.D.**

Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.

Datum zadání: 1. listopadu 2021

Datum odevzdání: 18. května 2022

Datum schválení: 1. listopadu 2021

Abstrakt

Tato práce se zabývá problematikou automatického rozpoznávání matematických výrazů pomocí neuronových sítí. Obsahuje přehled existujících přístupů pro tuto úlohu a zaměřuje se zejména na rozpoznávání ručně psaných matematických výrazů a využití grafových neuronových sítí. Jádrem navrženého systému pro rozpoznávání ručně psaných matematických výrazů je model neuronové sítě typu kodér-dekodér využívající grafové neuronové sítě pro přirozenou práci s hierarchickou strukturou matematických výrazů. Úspěšnost systému je vyhodnocena na datové sadě CROHME, která byla publikována v rámci stejnojmenné soutěže v rozpoznávání matematických výrazů. Součástí práce jsou také experimenty, které blíže studují navržený model. Navržené řešení dosahuje úspěšnosti 13.34% ExpRate, tedy přesného rozpoznání matematického výrazu na testovacích datech sady CROHME 2019. Přínosem této práce je zejména návrh metody použití grafových neuronových sítí pro rozpoznávání matematických výrazů z obrázků a obecně jejich zpracování v grafové doméně.

Abstract

This thesis deals with automatic mathematical expressions recognition using deep neural networks. It contains an overview of existing approaches and focuses mainly on handwritten mathematical expressions and the use of graph neural networks. The core of the proposed system for handwritten mathematical expressions recognition is an encoder-decoder neural network model using graph neural networks to exploit the hierarchical structure of mathematical expressions. The designed system is evaluated on the CROHME dataset, which was published within the competition of the same name on mathematical expression recognition. The work also includes description of experiments performed with the designed model. The proposed solution achieves an exact expression recognition rate of 13.34% on the CROHME 2019 test dataset. The contribution of this work is mainly a method of using graph neural networks for mathematical expression recognition from images and their processing in the graph domain.

Klíčová slova

Grafové neuronové sítě, rozpoznávání matematických výrazů, OCR

Keywords

Graph neural networks, mathematical expression recognition, equation recognition, OCR

Citace

HALVA, Vladislav. *Automatické rozpoznávání matematických výrazů pomocí neuronových sítí*. Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Automatické rozpoznávání matematických výrazů pomocí neuronových sítí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Vladislav Halva
18. května 2022

Poděkování

Chtěl bych poděkovat panu Ing. Michalu Hradišovi, Ph.D. za odborné vedení a cenné připomínky při psaní této práce.

Obsah

1	Úvod	2
2	Rozpoznávání matematických výrazů	3
2.1	Současné metody založené na neuronových sítích	4
2.2	Datové sady	6
3	Metody založené na grafových neuronových sítích	9
3.1	Grafové neuronové sítě	10
3.2	Graph-to-graph přístup	15
4	Návrh řešení	19
4.1	Konstrukce vstupního LoS grafu	21
4.2	Výstupní SLT graf	23
4.3	Grafový kodér	23
4.4	Grafový dekodér	24
5	Implementace	27
5.1	Použité nástroje	27
5.2	Načítání a příprava dat	27
5.3	Neuronová síť	28
5.4	Trénování a sledování úspěšnosti	30
6	Experimenty a vyhodnocení	32
6.1	Velikost sítě	33
6.2	Dodatečná supervize	34
6.3	Prohledávací algoritmus	34
6.4	Kvalitativní vyhodnocení	36
6.5	Shrnutí výsledků	37
7	Závěr	39
	Literatura	40
A	Obsah DVD	44

Kapitola 1

Úvod

Rozpoznávání matematických výrazů (anglicky mathematical expression recognition - MER) je problematika převodu tištěných, či ručně psaných matematických výrazů na jejich reprezentaci strukturovaným jazykem, který umožňuje jejich další zpracování. Matematické výrazy jsou standardním vědeckým dorozumívacím prostředkem a jsou důležitou součástí většiny vědeckých a technických dokumentů. Jejich rozpoznávání je důležité pro řadu úloh, například analýzu dokumentů, získávání dat (anglicky data-mining) nebo převod tištěných dokumentů do elektronické podoby. Vzhledem ke komplexní struktuře matematických výrazů a používání nestandardních symbolů je také komplikované jejich zadávání do počítače bez použití značkovacího, či jiného specializovaného jazyka jako je například LaTeX. Ruční psaní na papír nebo digitálním perem je pro nás značně jednodušší a přirozenější. Metody rozpoznávání ručně psaných matematických výrazů nám potom mohou umožnit tyto ručně psané výrazy převést elektronické podoby, například do některého z používaných strukturovaných jazyků a jejich vložení do digitálních dokumentů. V porovnání s rozpoznáváním přirozeného jazyka přináší matematické výrazy řadu komplikací vyplývajících z jejich dvou-dimenzionální struktury, široké škály používaných symbolů a podobně.

Tato práce se zabývá problematikou automatického rozpoznávání matematických výrazů pomocí neuronových sítí se zaměřením na ručně psané matematické výrazy a využití grafových neuronových sítí.

Následující kapitola 2 se věnuje detailnějšímu popisu problematiky rozpoznávání matematických výrazů a současným metodám založených na neuronových sítích. Blíže jsou tu popsány zejména metody používající konvoluční a rekurentní neuronové sítě. Obsahuje také souhrn dostupných datových sad pro trénování neuronových sítí pro tuto úlohu a diskutuje jejich výhody a nedostatky.

Kapitola 3 rozšiřuje přehled současných metod o přístupy využívající grafové neuronové sítě a hlouběji rozebírá vybranou metodu pracující se vstupní i výstupní reprezentací v podobě grafu, kterou je následně inspirována metoda navržená v této práci. Obsahuje také sekci věnovanou grafovým neuronovým sítím, jejich principu a klasifikaci.

V kapitole 4 je popsán přístup navržený v rámci této práce pro rozpoznávání ručně psaných matematických výrazů z jejich obrázků. Jádrem metody je model typu kodér-dekódér používající grafové neuronové sítě.

Experimenty s navrženým řešením a jejich výsledky jsou popsány v kapitole 6.

Kapitola 2

Rozpoznávání matematických výrazů

Rozpoznávání matematických výrazů [2] je běžně používáno k rozeznání matematických výrazů z obrázkové formy tištěného či ručně psaného textu. V případě ručně psaných matematických výrazů se často pracuje také s tzv. online vstupem, tedy sekvencí tahů z například digitálního pera.

Jedná se o typickou úlohu rozpoznávání strukturovaného textu [26]. Ve srovnání s rozpoznáváním přirozeného jazyka, tedy nejběžnější doménou optického rozpoznávání znaků (anglicky optical character recognition - OCR), je situace komplikovanější kvůli dvou-dimenzionální struktuře výrazů, ta vyplývá z výskytu zlomků, horních a dolních indexů a podobně [6]. Dalším úskalím je proměnlivý význam symbolů, ten může být ovlivněn jejich velikostí či okolními znaky, z toho důvodů je u matematických výrazů také silnější vazba na kontext, než u běžného textu. Z těchto důvodů je třeba specializovaného systému pro optické rozpoznávání matematických výrazů, který je schopen spolehlivého přepisu do strukturovaného, resp. značkovacího jazyka jako LaTeX nebo MathML [2].

Proces rozpoznávání matematických výrazů [8] lze rozdělit na dvě části: (i) segmentace a rozpoznání symbolů a (ii) analýza struktury výrazu. Tradiční metody obvykle tyto části řeší sekvencně nebo globálně (současně). V případě sekvencního provedení je nejprve vstupní obrázek rozdělen, tj. segmentován, na jednotlivé matematické symboly, které jsou následně rozpoznány. V další fázi je provedena analýza struktury výrazu. Globální přístupy provádí oba kroky současně, což umožňuje sdílení informací. Obě metody však vyžadují dobrou znalost domény a metodiku návrhu podobných komplexních systémů.

K popisu vztahů mezi symboly v rámci analýzy struktury výrazu využívá řada zavedených metod gramatik, například grafových, či jiných rozsáhlých skupin pravidel [6][26][27]. Systém INFTY [3] je jeden z nejvýznamnějších takových systémů. Slouží k převodu tištěných matematických výrazů do strukturovaného jazyka LaTeX a dalších značkovacích jazyků.

K problematice rozpoznávání matematických výrazů lze však přistupovat jako ke speciálnímu případu převodu sekvence na sekvenci, respektive obrázku na sekvenci. Tímto způsobem na úlohu nahlíží řada navržených přístupů využívajících neuronové sítě [26].

V rámci této kapitoly jsou dále popsány metody založené na použití neuronových sítích pro rozpoznávání matematických výrazů, kterým se věnuje sekce 2.1.

2.1 Současné metody založené na neuronových sítích

V současné době se vývoj soustředí zejména na rozpoznávání ručně psaných matematických výrazů. Některé přístupy se ale zaměřují i na tištěné (strojově psané) matematické výrazy. U tištěné podoby se vždy pracuje s tzv. *offline vstupem*, tedy například obrázky naskenovaných matematických výrazů. U ručně psaných matematických výrazů pak řada přístupů uvažuje tzv. *online vstup*, kdy se zpracovává vstupní sekvence tahů, získaných například z elektronického pera.

Pro řešení problematiky rozpoznávání matematických výrazů byla navržena řada přístupů používajících neuronové sítě. Vzhledem k tomu, že k problematice nejčastěji přistupují jako k úloze převodu sekvence na sekvenci se běžně používají architektury typu *kodér-dekodér* [5], resp. *sequence-to-sequence* [31]. V rámci této architektury je proces rozdělen do dvou částí: (i) část kodér se učí zakódovat vstupní data do vnitřní reprezentace příznakových vektorů a (ii) část dekodér, která z vnitřní reprezentace produkuje výstupní sekvenci. Trénovací proces modelu je potom řízený čistě daty, tedy *end-to-end*. Architektura je typicky rozšířena *attention* mechanismem [33], který zajišťuje flexibilnější převod mezi obrazovou a sekvenční doménou a umožňuje modelu soustředit se vždy na relevantní části vstupních dat, respektive jejich vnitřní reprezentace. Architektura typu kodér-dekodér s *attention* mechanismem byla úspěšně použita například i pro úlohu automatického generování popisků obrázků [38].

Zhang a spol. [42] představili model typu kodér-dekodér s *attention* mechanismem založený na rekurentních neuronových sítích s GRU (Gated Recurrent Unit) [4] buňkami. Model pracuje s online vstupem, tedy trajektorií z elektronického pera reprezentující matematický výraz, která je nejprve zpracována dvousměrnou GRU rekurentní sítí a tedy zakódována do vnitřní reprezentace modelu v podobě příznakových vektorů. Následně je generována výstupní sekvence symbolů jazyka LaTeX opět GRU rekurentní sítí s *attention* mechanismem.

Použití kodér-dekodér architektury pro rozpoznávání ručně psaných matematických výrazů z jejich obrázků bylo prezentováno v práci *Zhang a spol.* [45] jako tzv. *Watch, Attend and Parse (WAP)* přístup. V rámci tohoto modelu byla použita konvoluční neuronová síť založená na VGG architektuře [29] jako kodér pro transformaci vstupních obrázků na vysoko dimenzionální příznakové mapy a GRU rekurentní síť s *attention* mechanismem jako dekodér pro generování výstupní sekvence jazyka LaTeX. *Attention* mechanismus modelu opět umožňuje nahlížet pro každé generované slovo na celý vstupní obrázek, resp. mapu příznaků nad kterou pracuje, a soustředit se na relevantní regiony. Model s *attention* mechanismem v této základní podobě má problém s pokrytím celého vstupního obrázku (anglicky *lack of coverage*) [32], jak uvádí autoři. Proto byl rozšířen o tzv. *coverage vector*, který modeluje na které části mapy příznaků se model v rámci *attention* mechanismu zaměřoval v předchozích krocích překladu. Tímto způsobem se model pokouší předejít nedostatečnému (*under-parsing*), nebo nadbytečnému (*over-parsing*) zaměření na některé části obrázku. Výsledky modelu značně předčily konvenční modely na datasetu CROHME 2014 [19] s 46.55% úspěšností přesného rozpoznání výrazu (*ExpRate*).

Ding a spol. [7] představili model velice podobný přístupu *WAP*, který následně rozšířili o hierarchický *attention* mechanismus označený *coarse-to-fine attention* [6]. Model pro zakódování do vnitřní reprezentace používá konvoluční neuronovou síť následovanou rekurentní neuronovou sítí s LSTM (Long Short-Term Memory) buňkami [13]. Tato dodatečná rekurentní síť pracuje nad jednotlivými řádky mapy příznaků, které jsou výstupem předcházející konvoluční sítě a jejím účelem je jejich překódování a obohacení o kontext

v horizontálním směru. Pro generování výstupní sekvence je pak použita LSTM rekurentní síť s již zmíněným *coarse-to-fine attention* mechanismem. Jejím principem je dvouúrovňový výběr oblastí, na které se model zaměřuje. Obrázek je rozdělen na oblasti, resp. buňky hrubou (anglicky *coarse*) mřížkou, která je následně dále dělena na menší, jemnější (anglicky *fine*) buňky. V první fázi jsou v rámci attention vybrány hrubé buňky a až následně se model soustředí na jemné buňky, které jsou však volené pouze z hrubých buněk vybraných v předchozí fázi. Důvodem tohoto rozšíření je snížení výpočetní složitosti attention redukcí oblastí, nad kterou se počítá. Motivací tohoto konkrétního řešení je skutečnost, že odvodit hrubou pozici symbolu v obrázku by mělo být možné s vysokou pravděpodobností z předchozího symbolu. V rámci této práce byl také představen veřejně dostupný dataset *IM2LATEX-100K*, který obsahuje přes 100000 reálných matematických výrazů v jazyce LaTeX. Model dosáhl úspěšnosti 76.15% přesného rozpoznání výrazu (ExpRate) a 87% BLEU skóre [23] na tomto datasetu obrázků tištěných výrazů. Na datasetu ručně psaných matematických výrazů CROHME 2014 dosáhl úspěšnosti 36.4% přesného rozpoznání výrazu (ExpRate) a 68% BLEU skóre.

Singh a spol. [30] úspěšně použil modifikovanou variantu modelu pro automatické generování popisků obrázků *Show, Attend and Tell* [38] na úlohu převodu obrázků tištěných matematických výrazů do jejich reprezentace v jazyce LaTeX. V práci dále analyzoval schopnost attention mechanismu zaměřit se při transformaci na oblast obrázku, ve které se aktuálně zpracovává symbol vyskytuje.

Výše uvedené přístupy ukazují, že modely typu kodér-dekodér s využitím attention mechanismu jsou schopné řešit úlohu rozpoznávání matematických výrazů bez využití předchozí znalosti jazyka jako například u přístupů založených na gramatikách. Řada výzkumů se následně zaměřila na vylepšení tohoto paradigmatu z různých úhlů pohledu, například úpravy architektury, trénovací strategie apod.

Zhang a spol. [43] rozšířili výše zmíněnou architekturu *WAP* o attention mechanismus pracující ve více rozlišeních označovaný jako *multi-scale attention* a nahradili původní architekturu konvoluční sítě extrahující příznaky z vstupního obrázku za architekturu typu *DenseNet* [15]. Pro funkčnost upraveného attention mechanismu je do části kodér přidána nová výstupní větev, která vedle původní mapy příznaků s nízkým rozlišením produkuje mapu příznaků s vyšším rozlišením. V části dekodér je následně provedena nezávisle operace attention nad oběma mapami příznaků a výstupy jsou pro generování výstupní sekvence spojeny. Attention operace nad příznaky s vyšším rozlišením kompenzuje informace ztracené podvzorkovacími (pooling) operacemi, které provádí konvoluční síť, při kterých se může ztratit informace o menších symbolech výrazu. Tento problém je u matematických výrazů způsobem velice variabilní velikostí symbolů. Model dosáhl přesnosti rozpoznání výrazu (ExpRate) 52.8% na datasetu CROHME 2014.

Další existující rozšíření [8] architektury *WAP* je inspirované modelem *transformer* [33]. Model aplikuje více-hlavový attention mechanismus a celou část dekodér duplikuje. Na datasetu CROHME 2014 bylo dosaženo o 3.66% lepších výsledků přesnosti rozpoznání výrazu (Exp Rate) než u původního modelu.

Zhang a spol. [46] představili model velice podobný výše zmíněnému řešení [6] používající *coarse-to-fine attention* mechanismus a konvoluční síť následovanou rekurentní sítí pro vytvoření vnitřní reprezentace. Autoři ale navrhují před extrakcí příznaků zdvojnásobit velikosti vstupních obrázků a rozšířit konvoluční síť o další vrstvu pro přesnější extrakci příznaků. Dále je v architektuře zdvojen attention mechanismus pracující nad totožnou mapou příznaků ale disponující vlastními vahami. Jejich výstupy jsou následně zkombinovány

dle empiricky určených vah. V části dekodér je také zaveden *dropout* mechanismus. Model dosáhl na datasetu IM2LATEX-100K BLEU skóre 88.42%.

Pang a spol. [22] použili pro extrakci příznaků z obrázků matematických výrazů konvoluční síť typu *ResNet* [12] rozšířenou o vrstvu pro zachycení globálního kontextu za použití attention podvzorkování. Konvoluční síť je také následována modulem kodéru inspirovaným architekturou *transformer* [33]. Modul rozšiřuje vstupní mapu příznaků o zakódování pozice, dle formule definované pro architekturu *transformer*, a více-hlavový attention mechanismus (*multi-head attention*) pro zachycení závislostí mezi symboly výrazu. Ke generování výstupní sekvence je pak použita GRU rekurentní neuronová síť s *maskovaným attention* mechanismem, který zabraňuje zaměření modelu na již zpracované oblasti. Model byl testován na datasetu obrázků tišených matematických výrazů IM2LATEX-100K a dosáhl BLEU skóre 89.72%, což je o jednotky procent zlepšení oproti výše uvedeným modelům [6][7][43]. Z analýzy výsledků vyplývá, že má model lepší schopnost zpracovat delší výrazy, než zmíněné architektury.

Výše uvedené metody generují výstupní reprezentaci v podobě sekvence znaků značkovacího jazyka LaTeX. Vzhledem k tomu, že matematické výrazy mají hierarchickou grafovou strukturu, resp. stromovou, je komplikované se tímto přístupem vypořádat s touto strukturální vlastností.

Zhang a spol. [44] navrhl řešení v podobě architektury, která používá konvoluční síť typu *DenseNet* pro vytvoření příznakové mapy a dekodér, který generuje výstupní reprezentaci mající stromovou strukturu. Vzhledem k tomu, že je použita GRU rekurentní síť tak je strom reprezentující výraz dekomponován na podstromy skládající se vždy z páru rodičovského uzlu a jeho potomka. Výstupní reprezentace je pak sekvence těchto podstromů. Model vykazuje vyšší schopnost generalizace na výrazy vyšší komplexity, než které mu byly předloženy v trénovací fázi, v porovnání s modely, které pracují s čistě sekvenčním výstupem. Na datasetu CROHME 2014 bylo dosaženo přesnosti rozpoznání výrazu (ExpRate) 49.1%, což je srovnatelné s výsledky varianty s sémanticky sekvenční variantou části dekodér [43].

Peng a spol. [26] prezentovali architekturu, která v rámci enkodéru kromě konvoluční neuronové sítě používá grafovou neuronovou síť pro obohacení vnitřní reprezentace modelu o informaci o struktuře matematického výrazu. *Wu a spol.* [36] přistupuje k úloze rozpoznávání matematických výrazů jako problému převodu grafu na graf a navrhl model používající grafovou neuronovou síť pro zakódování vstupní reprezentace i generování výstupního popisu matematického výrazu. Oba přístupy jsou blíže popsány v kapitole 3.1.

2.2 Datové sady

Tato kapitola popisuje nalezené datové sady, které by mohly být použité pro trénování a vyhodnocování neuronových sítí pro rozpoznávání matematických výrazů. Ukázky elementů jednotlivých datových sad jsou na obrázcích 2.1. a tabulka 2.1 obsahuje stručné shrnutí parametrů těchto sad.

CROHME dataset [21] je datová sada, která byla zveřejněna v rámci soutěže *ICDAR2019-CROHME (Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection)* z roku 2019. Dataset obsahuje celkem 12 178 matematických výrazů rozdělených do trénovací, validační a testovací sady. Průměrná délka výrazů je 9,5 znaků. Zdrojové výrazy jsou k dispozici v podobě jednotlivých tahů pro rozpoznávání *online* vstupů, ale také v podobě obrázků pro úlohu rozpoznávání *offline* dat. Anotace jednotlivých

výrazů jsou k dispozici prostřednictvím InkML dokumentů, které pro každý matematický výraz obsahují anotaci v podobě zdrojového kódu v jazyce LaTeX a Presentation MathML, které odpovídají struktuře výrazu v dané *Symbol Layout Tree (SLT)* [41] grafem. Přiložené anotace také obsahují pozice jednotlivých tahů a jejich seskupení včetně referencí na příslušné symboly výrazu. Tyto reference jsou realizované pomocí identifikátorů, které jsou přítomné pouze pro skupiny tahů a elementy MathML reprezentace. Přímé mapování na anotaci v jazyce LaTeX tedy není možné. Díky této příslušnosti je u této datové sady možné také sestavení ohraničujících obdélníků symbolů výrazu a v důsledku vytvoření anotací pro segmentaci symbolů z obrázku. K datové sadě jsou také k dispozici nástroje pro vyhodnocení a detailní rozbor rozpoznávání výrazů. Hlavní metrikou, kterou nástroje používají je míra přesného rozpoznání výrazů *expression recognition rate (ExpRate)*, která není prováděna přímo nad popisem v jazyce LaTeX či MathML, ale obdobnou reprezentací tzv. *Symbol Label Graph*. Přiložené nástroje obsahují skripty pro automatické převedení jazyka LaTeX do tohoto formátu. Datová sada je použita k trénování a vyhodnocení velké části nedávno publikovaných přístupů pro rozpoznávání ručně psaných matematických výrazů.

IM2LATEX-100k je datová sada tištěných matematických výrazů prezentovaná v [7]. Skládá se z celkem 103556 reálných matematických výrazů extrahovaných ze zdrojových článků dostupných v rámci soutěže *2003 KDD cup*¹. Výrazy jsou tedy dostupné v podobě obrázků generovaných ze zdrojového kódu v jazyce LaTeX. Anotace jsou k dispozici pouze v jazyce LaTeX. Počet znaků v rámci jednotlivých výrazů se pohybuje v rozmezí od 38 do 997 se střední hodnotou 118 a mediánem 98. K dispozici je také varianta tohoto datasetu v podobě synteticky vytvořených ručně psaných výrazů [6]. Data byla vytvořena nahrazením každého symbolu v tištěné variantě korespondujícím ručně psaným symbolem, který byl náhodně vybrán z testovacích dat pro nástroj *Detexify*².

Aida Calculus Math Handwriting Recognition Dataset [25] je datová sada obsahující 100000 synteticky generovaných obrázků ručně psaných matematických rovnic. Anotace datové sady zahrnují zdrojové kódy v jazyce LaTeX a ohraničující obdélníky pro všechny symboly výrazu. Datová sada je tématicky zaměřena na limity, takže není příliš vhodná pro trénování modelu, mohla by ale být použita pro rozšíření testovací sady.

InftyMCCDB-2 je dataset, který byl vytvořena v rámci práce [18] pro rozpoznávání tištěných matematických výrazů. Obsahuje 19381 matematických výrazů, naskenovaných z řady článků, délky v rozmezí 1 až 75 symbolů, s průměrem 7.33. Anotace jsou k dispozici v podobě *Label Graph (LG)* grafů. Tedy matice sousednosti na úrovni propojených komponent včetně ohraničujících obdélníků a symbolů, které komponenty reprezentují, resp. ke kterým patří. Symboly jsou také rozděleny do 213 tříd. Dataset je rozdělen na trénovací, validační a testovací sadu s přibližně stejně zastoupenými třídami symbolů a podobnou průměrnou délkou výrazů.

¹<https://www.cs.cornell.edu/projects/kddcup/>

²<https://detexify.kirelabs.org/classify.html>

Název sady	Typ	Anotace	Počet prvků	μ
CROHME	ručně psané online i offline	LaTeX, MathML tahy	12 178	9.5
IM2LATEX-100k	tištěné, ručně psané offline	LaTeX	103 556	118
Aida Calculus	ručně psané offline	LaTeX	100 000	-
InftyMCCDB-2	tištěné	Label Graph	19 381	7.33

Tabulka 2.1: Přehled datových sad. Sloupec s označením μ označuje průměrnou délku výrazů v datové sadě.

$$x = r \cos \theta$$

$$q = \sqrt{\left(\frac{B-A}{C}\right)^2 + 1} + \frac{B-A}{C}$$

$$\int_C^{d(B)} \tan^\alpha q \, ds$$

(a) CROHME

$$\frac{l}{f} = \pm \frac{1}{\sqrt{d}}, \frac{p}{d}$$

$$F_{B,F}(\beta) = \pm \frac{1}{\beta} \int \frac{d^{d-1}k}{(2\pi)^{d-1}} \ln(1 \mp e^{-\beta w_k})$$

$$W(x) = \frac{x^3}{3} - a^2 x,$$

(b) IM2LATEX-100k

$$\lim_{y \rightarrow 2} \frac{\tan(4y)}{\tan(6y)}$$

$$\lim_{x \rightarrow 6^+} \sin x + 7 \frac{1}{x}$$

$$\lim_{y \rightarrow 9} \frac{\frac{1 \sin(2y)}{3y}}{\frac{3 \sin(7y)}{8y}}$$

(c) Aida Calculus Math Handwriting Recognition Dataset

$$h + cg$$

$$\zeta' = h(\xi, \zeta) = \sum h_{ij} \xi^i \zeta^j$$

$$p = \sum_{r=0}^k x_n^r p_r$$

(c) InftyMCCDB-2

Obrázek 2.1: Ukázka datových sad. U datové sady IM2LATEX-100k jsou první dva výrazy ze základní verze datové sady a poslední ze synteticky generované, ručně psané varianty.

Kapitola 3

Metody založené na grafových neuronových sítích

Matematické výrazy mají hierarchickou strukturu jak v samotné formuli, tak v jejich reprezentaci značkovacím jazykem typu LaTeX. Dosud prezentované metody, popsané výše v sekci 2.1, ale k problematice jejich rozpoznávání přistupují jako k úloze převodu obrázku na sekvenci znaků. Důsledkem je, že implicitně nezohledňují hierarchickou strukturu výrazů a nedokážou tedy plně využít informace z ní vyplývající. Pro pokrytí tohoto nedostatku byly navrženy architektury využívající grafové neuronové sítě, jejichž doménou jsou právě strukturovaná data. Grafové neuronové sítě byly úspěšně použity například na úlohu generování popisků obrázků [40].

Prvním krokem k použití grafových neuronových sítí je například architektura [26], která rozšiřuje standardní model typu kódér-dekódér transformující vstupní obrázku tištěných matematických výrazů do jazyka LaTeX o grafovou neuronovou síť pro zahrnutí prostorových vztahů symbolů výrazu do vnitřní reprezentace a tím pádem zlepšení schopnosti jejich rozpoznávání. Zakódování vstupního obrázku do vnitřní reprezentace je rozděleno do dvou částí. První část je realizována konvoluční sítí typu *ResNet* [12], která pracuje nad celým vstupním obrázkem a jejímž účelem je extrakce vizuálních příznaků z celého výrazu. Stejně jako v případě některých modelů popsaných v sekci 2.1 je následována rekurentní sítí pracující nad jednotlivými řádky mapy příznaků pro zahrnutí pozičního kontextu.

Druhá část je tvořena grafovou neuronovou sítí, konkrétně *Gated Graph Neural Network* [17], která distribuuje prostorové vazby mezi symboly na základě grafu, který jí je předložen na vstup. Pro vytvoření grafu je nejprve vstupní obrázek segmentován na jednotlivé symboly pomocí analýzy propojených komponent (connected components analysis - CCA). Nad segmentovanými komponentami je následně sestaven tzv. *Line of Sight (LOS)* graf [14], ve kterém jsou propojeny komponenty, které jsou navzájem viditelné, tzn. není mezi nimi žádná jiná komponenta, která by je navzájem zastínila. Pro získání příznakových vektorů jednotlivých komponent je použita další konvoluční síť pracující nad těmito jednotlivými oblastmi symbolů. Před předložením na vstup grafové sítě jsou příznakové vektory komponent, resp. uzlů grafu rozšířeny o poziční kódování, tzv. *coordinate embedding*, pro vyjádření globální pozice v rámci výrazu. Vytvořené příznakové vektory jsou použity jako počáteční stavy jednotlivých uzlů grafu v rámci zpracování grafovou sítí.

Po zpracování grafovou sítí jsou výstupní stavy uzlů sjednoceny do mapy příznaků a následně sjednoceny, resp. konkatenovány s příznakovou mapou získanou z konvoluční sítě pracující nad celým vstupním obrázkem. Tento sjednocený výstup je následně zpracován

LSTM rekurentní sítě s attention mechanismem a je generována výstupní sekvence jazyka LaTeX.

Experimenty s modelem byly prováděny nad datovou sadou IM2LATEX-100K [7]. Dle analýzy výsledků model dosahuje lepších výsledků, než modely podobné architektury, které nepoužívají grafové neuronové sítě. Jedná se o zlepšení o 2.46% přesného rozpoznání (ExpRate), resp. 1.77% BLEU skóre. Toto zlepšení indikuje vhodnost rozšíření modelu o mechanismus extrakce prostorových vztahů symbolů pro zvýšení přesnosti. Z dalších provedených testů robustnosti modelu, kdy byl do obrázků testovací sady přidán šum lze také vidět, že model je odolnější vůči tomuto šumu než výše zmíněné modely, se kterými byl porovnáván. Autoři testovali také přínos jednotlivých elementů modelu na jeho úspěšnost a při odebrání větve používající grafovou síť dosáhli o 13.77% horšího BLEU skóre.

Do návrhu této architektury je již zapojena skutečnost, že doménou nad kterou pracuje jsou data mající hierarchickou strukturu. Výstupní reprezentace v jazyce LaTeX je ale stále generována sekvenčně rekurentní neuronovou sítí. Model tedy nevyužívá hierarchické struktury této výstupní reprezentace.

Wu a spol. představil architekturu, které využívá znalosti o hierarchické struktuře vstupní i výstupní sekvence. Tento přístup je detailně popsán v sekci 3.2.

3.1 Grafové neuronové sítě

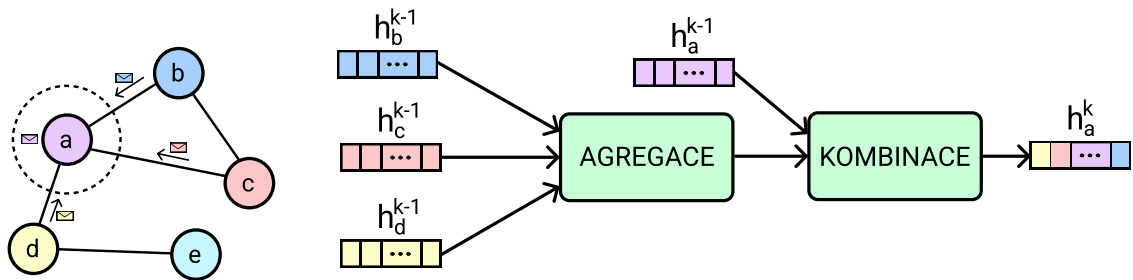
Grafové neuronové sítě (anglicky Graph Neural Networks - GNN) [28] jsou relativně novým typem neuronových sítí navržených pro práci nad obecnými grafovými daty. Vznikly jako důsledek vzrůstající pozornosti věnované analýze grafových dat, kvůli jejich expresivní schopnosti a širokému spektru aplikace ve strojovém učení. Data, která mají strukturu grafu se vyskytují v mnoha doménách, např. chemii, zpracování obrazu, zpracování přirozeného jazyka apod.

Učení nad grafovými daty vyžaduje efektivní reprezentaci jejich struktury. Principiálně grafové neuronové sítě rekurzivně agregují informace ze sousedních uzlů každého uzlu grafu, tzv. zasílání zpráv (anglicky *message passing*), pro výpočet nové hodnoty příznakového vektoru daného uzlu. Po určitém počtu opakování pak příznakový vektor uzlu reprezentuje strukturální informaci o jeho okolí [39]. Proces zasílání zpráv pro jeden konkrétní uzel je znázorněn na obrázku 3.1a, kde symbol obálek reprezentuje příznakové vektory sousedních uzlů uzlu a , které jsou zasílány pro agregaci jejich hodnot a následnou kombinaci s příznakovým vektorem uzlu a pro úpravu jeho hodnoty.

Výstupy grafové neuronové sítě pracující nad daty mající grafovou strukturu, s informacemi uloženými v uzlech, mohou být použity k různým úlohám analýzy, které se dají rozčlenit do následujících kategorií: [37][47]

Úlohy na úrovni uzlů (node-level) se zaměřují na samotné uzly grafu. Do této kategorie spadá *klasifikace uzlů*, *regrese nad uzly* apod. Cílem klasifikace uzlů je kategorizovat uzly do předem definovaného počtu tříd. Regrese nad uzly odvozuje spojitou hodnotu pro každý uzel.

Úlohy na úrovni hran (edge-level) se vztahují k *klasifikaci hran* a *predikci spojení*. Cílem těchto metod je klasifikovat hrany grafu do několika definovaných tříd, resp. predikovat, zda existuje hrana mezi dvěma danými uzly grafu.



(a) Abstraktní pohled na zaslání zpráv při výpočtu nové reprezentace uzlu.

(b) Aktualizace příznakového vektoru konkrétního uzlu, daná agregací příznakových vektorů sousedících uzlů a jejich kombinace s příznakovým vektorem daného vybraného uzlu. Převzato z [35] a upraveno.

Obrázek 3.1: Znázornění procesu zaslání zpráv realizovaného grafovými neuronovými sítěmi a postup aktualizace příznakových vektorů uzlu.

Úlohy na úrovni grafu (graph-level) vychází z grafu jako celku. Pro dosažení vypovídající reprezentace grafu často realizují *pooling* (podvzorkovací) a *readout* (výstupní) operace nad grafem. Do kategorie spadají úlohy klasifikace či regrese grafu.

3.1.1 Teorie grafů

Pro další popis je nutné představit základní pojmy teorie grafů a použitou notaci: [37]

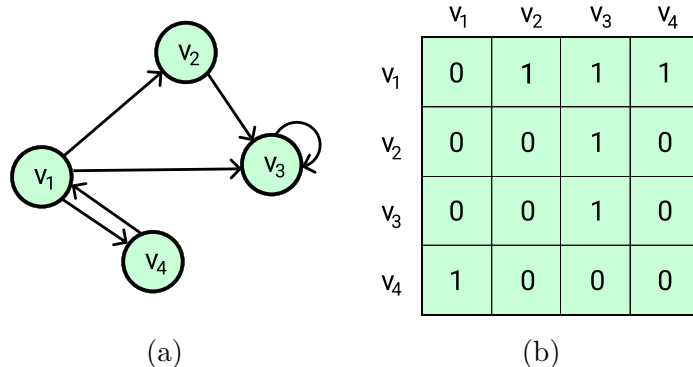
Graf G je uspořádaná dvojice (V, E) , kde V je množina vrcholů (resp. uzlů) $v_i \in V$ a E je množina hran grafu $e_{ij} = (v_i, v_j) \in E$.

Orientovaný graf je takový graf, kdy je E množina *uspořádaných* dvojic, tj. hrany mají směr. V opačném případě se jedná o *neorientovaný graf*.

Vážený graf je graf, jehož hrany e_{ij} jsou ohodnoceny váhami. Pokud ne, jedná se o *nevážený graf*.

Cyklický graf umožňuje vznik cyklických posloupností hran grafu. Opačným případem je *acyklický graf*.

Maticе sousednosti je běžně používána pro reprezentaci struktury grafu u grafových neuronových sítí. Jedná se o matici A rozměru $n \times n$, kde n je počet uzlů grafu. Hodnota $a_{ij} \in A$ je rovna 1 pokud existuje hrana z uzlu v_i do uzlu v_j , tedy $e_{ij} \in E$. V opačném případě je $a_{ij} = 0$. Matici sousednosti vytvořenou pro ukázkový orientovaný, cyklický graf ukazuje obrázek 3.2.



Obrázek 3.2: Vytvoření matice sousednosti. (a) Orientovaný, cyklický graf. (b) Matice sousednosti odpovídající grafu na obrázku (a).

3.1.2 Formalizace grafových neuronových sítí

Grafové neuronové sítě uvažují reprezentaci grafu $G = (V, E)$, jak byla definována v sekci 3.1.1 rozšířenou o příznakové vektory X_v pro $v \in V$ [39]. Učením se pokouší získat příznakové vektory h_v pro $v \in V$ reprezentující jednotlivé uzly nebo celý graf h_G . Úvodní reprezentace, resp. příznakový vektor uzlu v je inicializován jako $h_v^0 = X_v$. Za tímto účelem iterativně upravují, resp. aktualizují hodnoty příznakových vektorů uzlů agregací hodnot uzlů v jejich sousedství. Provedením k iterací příznakové vektory uzlů zachycují informaci o jejich k -okolí, tedy o podgrafu tvořeném uzly, jejichž cesta k vybranému uzlu je kratší nebo rovna k . Obecně lze k -tou vrstvou, respektive iteraci grafové sítě zapsat jako:

$$a_v^k = f^k(\{h_u^{k-1} : u \in N(v)\}) \quad (3.1)$$

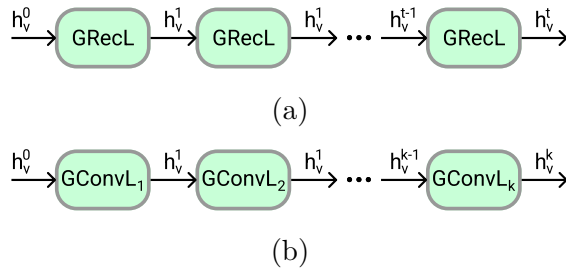
$$h_v^k = g^k(h_v^{k-1}, a_v^k) \quad (3.2)$$

Zde h_v^k označuje příznakový vektor uzlu v v k -té vrstvě sítě, $N(v)$ označuje uzly náležící sousedství uzlu v , tedy $\{u : u \in V, \exists e_{uv} = (u, v) \in E\}$. f^k a g^k označují funkci agregace, resp. kombinace, jejichž volba je klíčovým parametrem definující funkčnost architektury.

Výše popsaná definice vrstvy grafové sítě je pro aktualizaci hodnoty jednoho uzlu a se sousedními uzly b, c, d znázorněna na obrázku 3.1b.

Pokud uvažujeme úlohu klasifikace uzlů definujeme třídu uzlu v jako y_v a cílem učení je získat příznakový vektor h_v^k takový, že platí $y_v = f(h_v^k)$. V případě úlohy klasifikace grafu analogicky uvažujeme množinu grafů $\{G_1, \dots, G_N\} \in \mathcal{G}$ a jejich třídy $\{y_1, \dots, y_n\} \in \mathcal{Y}$ a sít se pokouší naučit získat reprezentaci grafu h_G takovou, že je možné predikovat třídu grafu G jako $y_G = g(h_G)$. U úloh na úrovni grafu, například právě klasifikace grafu, uvažujeme funkci o agregující finální reprezentace uzlů h_v^k do příznakového vektorů reprezentujícího graf h_G dle:

$$h_G = o(\{h_v^k : v \in G\}) \quad (3.3)$$



Obrázek 3.3: Demonstrace rozdílu reprezentace uzlů grafu rekurentními a konvolučními grafovými sítěmi. (a) Zpracování rekurentní grafovou neuronovou sítí. (b) Zpracování konvoluční grafovou neuronovou sítí. Převzato z [37] a upraveno.

3.1.3 Klasifikace grafových neuronových sítí

Grafové neuronové sítě lze v rámci strojového učení rozdělit do čtyř kategorií dle následující taxonomie [37]:

Rekurentní grafové neuronové sítě

Rekurentní grafové neuronové sítě zahrnují zejména rané modely, které se učí extrahovat příznakové vektory reprezentující uzly rekurzivní aplikací funkce pro jejich agregaci a kombinaci, tzv. *zasílání zpráv* (message passing).

V rámci prvotního modelu [11] tento proces probíhá dokud nedojde do stavu konvergence, po které jsou příznaky zpracovány výstupní funkcí. Hlavním problémem tohoto přístupu, je skutečnost, že vliv jednotlivých uzlů s každým krokem exponenciálně klesá. Důsledkem je řada modelů, které tuto problematickou vlastnost upravují a zavádí různá omezení pro přenášení informací na delší vzdálenosti v rámci grafu, např. model *Gated Graph Neural Network (GGNN)* [17] zavádějící buňky GRU. Rekurentní neuronové sítě používají stejnou vrstvu v každém kroku iterace, tzn. kroky sdílí matici vah.

Konvoluční grafové neuronové sítě

Konvoluční grafové neuronové sítě jsou inspirovány úspěchem konvolučních neuronových sítí v doméně dat s mřížkovou topologií. Jsou velice blízce spjaté s rekurentními grafovými sítěmi. Místo aplikace usměrňujících omezení iterace ale řeší problém zavedením předem definovaného počtu vrstev, z nichž každá reprezentuje jednu iteraci zasílání zpráv. Každá vrstva také disponuje vlastními váhami. Díky této architektuře umožňují návrh komplexnějších modelů. Tento rozdíl je schématicky zobrazen na obrázku 3.3. Zde lze vidět, že při zpracování rekurentní grafovou sítí je v každém kroku iterace použita totožná vrstva sítě *GRecL*, tedy jediná matice vah. U konvoluční grafové sítě je naopak každá iterace realizována jinou vrstvou *GConvL_k* s vlastní maticí vah. Výraz h_v^t , resp. h_v^k reprezentuje příznakový vektor uzlu v v kroku t , resp. k .

Kategorie konvolučních grafových neuronových sítí se dále rozpadá na dva typy: (i) *spektrální* (spectral-based) a (ii) *prostorové* (spatial-based).

Spektrální konvoluční grafové sítě vychází ze spektrální teorie grafů, resp. grafového zpracování signálu. Definují operaci konvoluce vykládanou jako odstraňování šumu z grafo-

vých signálů pomocí filtrů, z perspektivy zpracování signálu. Jedná se tedy o konvoluci ve spektrální doméně. Uvažují pouze neorientované grafy, resp. jejich reprezentaci v podobě normalizované Laplaceovy matice pro analýzu vlastních vektorů a následný převod do spektrální domény, konvoluci a zpětný převod do prostorové domény. Pro převod mezi prostorovou a spektrální doménou je použita Fourierova transformace, resp. zpětná Fourierova transformace [37].

V základní podobě spektrální konvoluční grafové sítě pracují korektně pouze pro grafové struktury, které jim byly předloženy ve fázi trénování a mají vysokou výpočetní složitost $O(N^3)$, kde N je počet uzlů grafu [35]. Oba problémy vychází z analýzy systému vlastních vektorů. Zmíněné problémy byly zmírněny u následujících modelů, například *ChebNet* a *Graph Convolutional Network (GCN)*. U modelu GCN se podařilo pomocí aproximačních metod a dalších úprav docílit výpočetní složitosti systému vlastních vektorů $O(M)$, kde M je počet hran grafu. Na model GCN lze nahlížet také z pohledu prostorové grafové konvoluční sítě.

Prostorové konvoluční grafové sítě využívají k definici konvoluce přímo topologii grafu, tedy prostorové propojení. K odvození úpravy reprezentace uzlu konvolují jeho hodnotu s hodnotami sousedních uzlů, tj. uzlů, se kterými sdílí hranu. Jedná se tedy o stejný princip zasílání zpráv jako u rekursivních grafových neuronových sítí. Jak již bylo zmíněno výše, rozdílem je skutečnost, že uvažují předem definovaný počet iterací reprezentovaný v podobě vrstev sítě.

Grafové autoenkodéry

Grafové autoenkodéry jsou architektury, které realizují autoenkoding nad daty s grafovou topologií. Komprimují, resp. kódují reprezentaci vstupního grafu do relativně nízkodimenzionálního prostoru a z něho dekódují výstupní reprezentaci. Cílem procesu je získání reprezentace grafu, resp. reprezentace příznakových vektorů jeho uzlů, v nízkodimenzionálním prostoru, tzv. *embedding space*, který zahrnuje informaci o topologii celého grafu. Tato reprezentace grafu získaná nad neoznačenými daty, může dále značně zvýšit úspěšnost jiné úlohy. Typicky autoenkodéry používají konvoluční grafové neuronové sítě jako základní stavební architekturu.

Časově proměnné grafové neuronové sítě

Časově proměnné grafové neuronové sítě pracují nad dynamickými grafy, kdy se uvažují časově proměnné vstupní hodnoty uzlů a jejich vzájemná závislost. Je tedy nutné zachytit prostorové a časově proměnné závislosti v rámci grafu. Cílem tedy může být například predikce budoucích hodnot uzlů, či jejich časově proměnná klasifikace. Časově proměnné grafové neuronové sítě často kombinují přístupy rekurentních a konvolučních grafových neuronových sítí. Typickou aplikací může být například predikce provozu v dopravní síti.

3.1.4 Graph Attention Network

Graph Attention Network (GAT) [34] je architektura spadající do kategorie konvolučních grafových neuronových sítí. Představením vrstev sítě, které využívají *self-attention* mechanismus umožňují modelu soustředit se při konvoluci, tedy při aktualizaci hodnoty příznakového vektoru uzlů, na okolní uzly s různou váhou. Operace je také nezávislá na topologii grafu a hodnotě uzlů. Model je tedy schopen pracovat i nad grafem s topologií, která mu nebyla předložena ve fázi trénování.

Uvažujeme-li graf s množinou N uzlů a příznakovými vektory $h = \{h_i | 1 \leq i \leq N\}$, který je vstupem vrstvy sítě, pak tato vrstva produkuje množinu nových příznakových vektorů $h' = \{h'_i | 1 \leq i \leq N\}$ stejné, či různé dimenze, dle formule:

$$h'_i = \sigma \left(\sum_{j \in N(i)} \alpha_{ij} W h_j \right), \quad (3.4)$$

kde σ je sigmoidální nelinearita, W je učená matice vah, tedy lineární transformace, $N(i)$ označuje množinu sousedních uzlů uzlu i a α_{ij} je normalizovaný *attention* koeficient, který vyjadřuje váhu příznaků uzlu j pro uzel i . Tyto koeficienty jsou získány aplikací *self-attention* operace na příznakové vektory uzlů a následnou normalizací operací *softmax* přes všechny uzly $j \in N(i)$, tedy v okolí uzlu i dle:

$$a_{ij} = \text{attention}(W h_i, W h_j) \quad (3.5)$$

$$\alpha_{ij} = \text{softmax}(a_{ij}) = \frac{\exp(a_{ij})}{\sum_{k \in N(i)} \exp(a_{ik})} \quad (3.6)$$

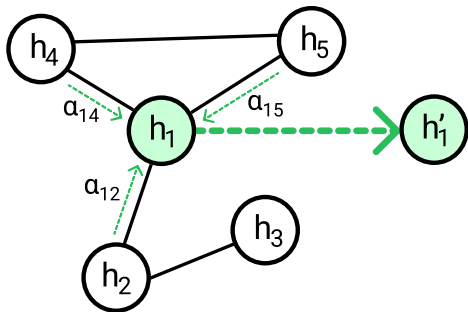
Ve standardní podobě modelu je funkce *attention* realizována plně propojenou vrstvou neuronové sítě parametrizovanou učeným vektorem v následovaná nelineární funkcí *LeakyReLU*. Výpočet normalizovaných koeficientů tedy odpovídá rovnici:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(v^T [W h_i || W h_j]))}{\sum_{k \in N(i)} \exp(\text{LeakyReLU}(v^T [W h_i || W h_k]))} \quad (3.7)$$

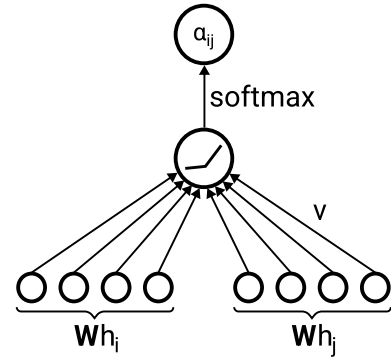
Zde $||$ značí konkatenci dvou vektorů a v^T transpozici vektoru v . Výpočet výstupních příznakových vektorů tímto způsobem probíhá paralelně pro všechny uzly grafu a operace je tedy výpočetně velice efektivní. Operace *self-attention* může být také rozšířena na vícehlavový mechanismus, kdy se uvažuje K nezávislých *attention* mechanismů. Takto získané výsledné příznakové vektory mohou být následně konkaténovány nebo průměrovány. Toto rozšíření vede ke stabilnějším výsledkům při učení.

3.2 Graph-to-graph přístup

Graph-to-graph přístup [36] definuje rozpoznávání matematických výrazů jako úlohu převodu grafu na graf. Navržená architektura pracuje se vstupní reprezentací v podobě sekvence tahů ručně psaných matematických výrazů a cílovou formou je reprezentace výrazu v jazyce LaTeX. Motivací pro reprezentaci v podobě grafu je skutečnost, že matematické výrazy mají hierarchickou prostorovou strukturu. Tento vztah můžeme pozorovat například u zlomků, či exponentů. Stejně tak na výstupní reprezentaci v podobě značkovacího jazyka LaTeX lze nahlížet jako na stromovou strukturu odpovídající stromu rozložení symbolů (Symbol Layout Tree - SLT) [41]. Tato korespondence je znázorněna na obrázku 3.5.

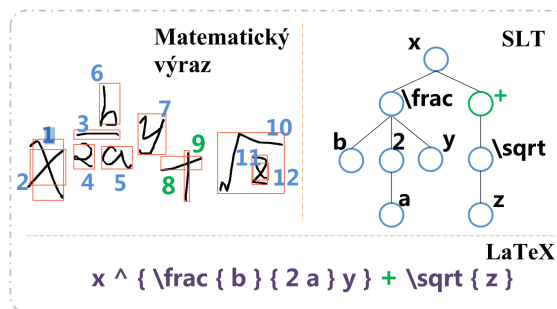


(a) Znázornění výpočtu *attention* koeficientů, tedy vynásobení maticí vah W , následně váhovým vektorem v , aplikace nelinearity *LeakyReLU* a normalizace funkcí *softmax*.



(b) Ukázka výběru okolí a aplikace *attention* koeficientů pro výpočet výstupního příznakového vektoru h'_1 uzlu 1.

Obrázek 3.4: Schématické znázornění funkce vrstvy modelu *Graph Attention Network*. Převezato z [34] a upraveno.



Obrázek 3.5: Znázornění korespondence ručně psaného matematického výrazu, Symbol Layout Tree (SLT) grafu a reprezentace v jazyce LaTeX. Převezato z [36] a upraveno.

Základním blokem prezentované architektury je grafová neuronová síť, konkrétně modifikovaná *Graph Attention Network (GAT)* [34] pracující jako kodér, která vytváří vnitřní reprezentaci vstupního výrazu. A dekodér využívající druhou grafovou neuronovou síť, jehož účelem je generovat výstupní reprezentaci v podobě SLT grafu. Grafové sítě umožňují pracovat s matematickými výrazy explicitně v podobě grafů a tím pádem prozkoumat a využít informací o jejich struktuře. Model navíc realizuje *attention* mechanismus na úrovni podgrafů mezi zmíněnými grafovými sítěmi, tzv. *sub-graph attention*, který umožňuje prozkoumat vazby mezi nimi.

Formálně je zde rozpoznávání matematických výrazů formulováno jako problém učení převodu grafu na graf, tedy $G_x \rightarrow G_y$, kde $G_x = (V_x, E_x)$ je orientovaný graf sestavený nad tahy vstupního výrazu x s uzly V_x odpovídajícími tahům a hranami E_x . V grafu jsou definovány dva typy hran. Prostorové, které odpovídají *Line of Sight (LOS)* grafu [41] vytvořenému nad tahy. A časové hrany, které reflektují časovou posloupnost v jaké byly tahy vytvořeny. $G_y = (V_y, E_y)$ je orientovaný graf odpovídající SLT grafu výrazu x . Pro vyjádření hierarchické struktury jsou použity tři typy hran směřující k uzlu od jeho (i) prarodiče (ii) rodiče a (iii) levého bratra. Cílem modelu je pak odvodit výstupní grafovou reprezentaci G_y podle:

$$p(G_y|G_x) = \prod_t p((v_y^t, E_y^t)|G_y^{<t}; G_x^{s,t}), \quad (3.8)$$

kde v_y^t je generovaný uzel v časovém kroku t , E_y^t je množina orientovaných hran generovaná v kroku t z uzlů v již existujících části výstupního grafu $G_y^{<t}$ do uzlu v_y^t , a $G_x^{s,t}$ je podgraf grafu G_x korespondující s uzlem v_y^t dle *sub-graph attention*.

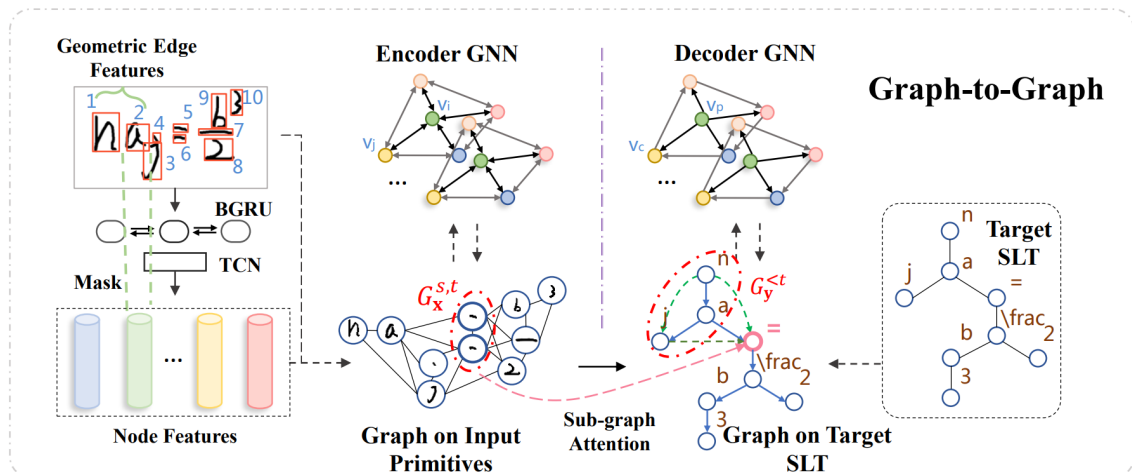
Vstupní příznakové vektory, které reprezentují tahy a tedy uzly grafu G_x jsou vytvořeny pomocí neuronové sítě skládající se z dvousměrné GRU rekurentní sítě následované konvoluční sítí pro zpracování sekvencí (Temporal Convolutional Network - TCN). Příznakové vektory hran grafu jsou inicializovány na hodnotu vypočtenou na základě vzájemné pozice a dalších geometrických a časových vlastností jí korespondujících uzlů, resp. tahů.

Grafová síť pracující jako kodér následně zpracovává reprezentaci vstupního grafu v každé vrstvě následující sekvencí kroků. Nejprve jsou aktualizovány hodnoty příznakových vektorů hran v závislosti na aktuálních vizuálních příznakových vektorech korespondujících uzlů. A následně jsou vypočteny nové hodnoty příznakových vektorů uzlů s využitím *attention* mechanismu pracujícím nad sousedními uzly.

Část dekodér opět používá grafovou neuronovou síť. V každém kroku je generován uzel a množina hran směřující z již vygenerované části výstupního grafu do právě tohoto uzlu. Generování nového uzlu pracuje s částí výstupního grafu z předchozího časového kroku a také příznakovými vektory korespondujících uzlů grafu G_x . Korespondence mezi uzly grafů G_x a G_y je dána zmiňovanou *sub-graph attention*.

Funkce celého modelu je znázorněna na obrázku 3.6. V levé části lze vidět transformaci vstupního matematického výrazu v podobě jednotlivých tahů na vstupní graf. Dále je zde pro uzel výstupního grafu znázorněna již existující část výstupního grafu a korespondující podgraf vstupního grafu pro aktualizaci jeho hodnoty. V pravé části je naznačena konstrukce výstupní reprezentace výrazu v podobě SLT grafu.

Optimalizace modelu je prováděna jak nad výstupem části kodér, tak nad výstupním grafem části dekodér. Výstupní příznakové vektory uzlů grafu G_y jsou přímo použity pro predikci symbolů jazyka LaTeX a cílem optimalizace je maximalizovat pravděpodobnost predikce cílového symbolu daného anotací. Pro každou hranu grafu G_y , která reprezentuje vztah rodičovského a synovského uzlu, je pak vytvořen příznakový vektor v závislosti na



Obrázek 3.6: Přehled procesu rozpoznání matematického výrazu modelem *Graph-to-Graph*. V levé části zpracování vstupního matematického výrazu reprezentovaného jednotlivými tahy pro vytvoření vstupního grafu enkodéru. V pravé části zpracování grafovým enkodérem pro vytvoření vnitřní reprezentace a následné generování výstupního grafu dekodérem s attention mechanismem. Převzato z [36].

příznakových vektorech korespondujících uzlů a je použit pro predikci typu hrany výsledného SLT grafu. Tento ty definuje vzájemnou prostorovou pozici symbolů. Supervize nad grafem G_x , tedy výstupem části kodér se pak snaží maximalizovat pravděpodobnost, že každý uzel náleží konkrétní kategorii matematického symbolu. Provádí se také klasifikace hran toho grafu, jelikož jejich příznakové vektory reflektují vzájemné prostorové uspořádání tahů a tedy jejich pozici v rámci matematického výrazu. V rámci trénování navíc probíhá supervize také nad *attention* mechanismem, který volí podgraf grafu G_x na který je model zaměřen při zpracování uzlu výstupního grafu G_y tak, aby se soustředil právě na podgraf korespondující s tímto uzlem, resp. symbolem. Pro umocnění tohoto efektu je navíc kromě již zmíněné predikce symbolu pro uzel grafu G_y dle jeho vlastního příznakového vektoru, symbol predikován i dle příznakových vektorů jeho podstromu v grafu G_x . Všechny nyní zmíněné složky se pak podílí na výsledné chybové funkci.

Model byl trénován a vyhodnocován na datových sadách dostupných pro soutěž CROHME [20], konkrétně CROHME 2013/2014/2016. V metrice přesného rozpoznání výrazu (ExpRate) na datasetu CROHME 2014 navržený model dosáhl o 4,61% lepších výsledků než nejúspěšnější z modelů, které k problematice přistupují jako k úloze převodu sekvence na sekvenci. Model je navíc schopen velice přesně klasifikovat tahy a hrany ve vstupních grafu G_x . Jde tedy o klasifikaci příslušnosti tahů ke konkrétním symbolům, resp. klasifikaci příslušnosti hran ke stejnému či jinému symbolu a jejich vzájemné pozice. Díky tomu je model schopen také úspěšné segmentace symbolů.

Kapitola 4

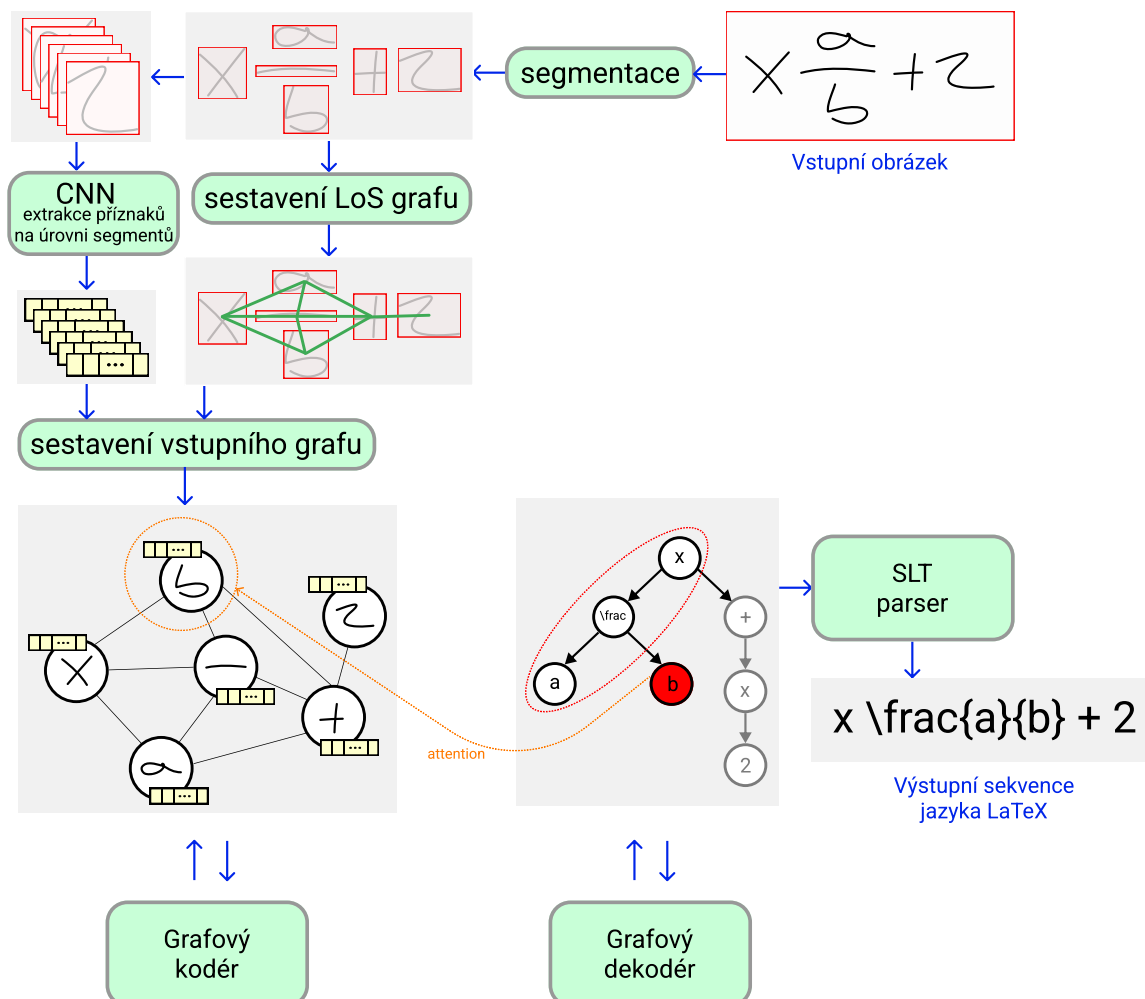
Návrh řešení

V rámci této kapitoly je popsáno navržené řešení pro rozpoznávání ručně psaných matematických výrazů z obrázků navržené na základě informací nastudovaných o problematice rozpoznávání matematických výrazů, které jsou shrnuty v předchozích kapitolách. Z přehledu existujících metod popsanych v kapitole 2.1 je zřejmé, že v rámci výzkumu rozpoznávání matematických výrazů je velké úsilí věnováno zdokonalování přístupu, který na problematiku nahlíží jako na přímý převod sekvence tahů, resp. obrázku matematického výrazu na sekvenci znaků výstupního jazyka. Nevýhodou tohoto přístupu je skutečnost, že zanedbává hierarchickou strukturu matematických výrazů a jejich reprezentace strukturovaným jazykem a nedokáže tak tuto informaci zužitkovat. V sekci 3.2 je následně popsána metoda *Graph-to-graph* [36] pro rozpoznávání ručně psaných matematických výrazů ze vstupní sekvence tahů, tedy tzv. online vstupních dat, která s hierarchickou strukturou výrazů explicitně pracuje a využívá grafové neuronové sítě jako model, který přirozeně nad touto doménou dat operuje. Navržené řešení vychází z tohoto přístupu a modifikuje je jej pro zpracování vstupů v podobě obrázků naskenovaných ručně psaných matematických výrazů.

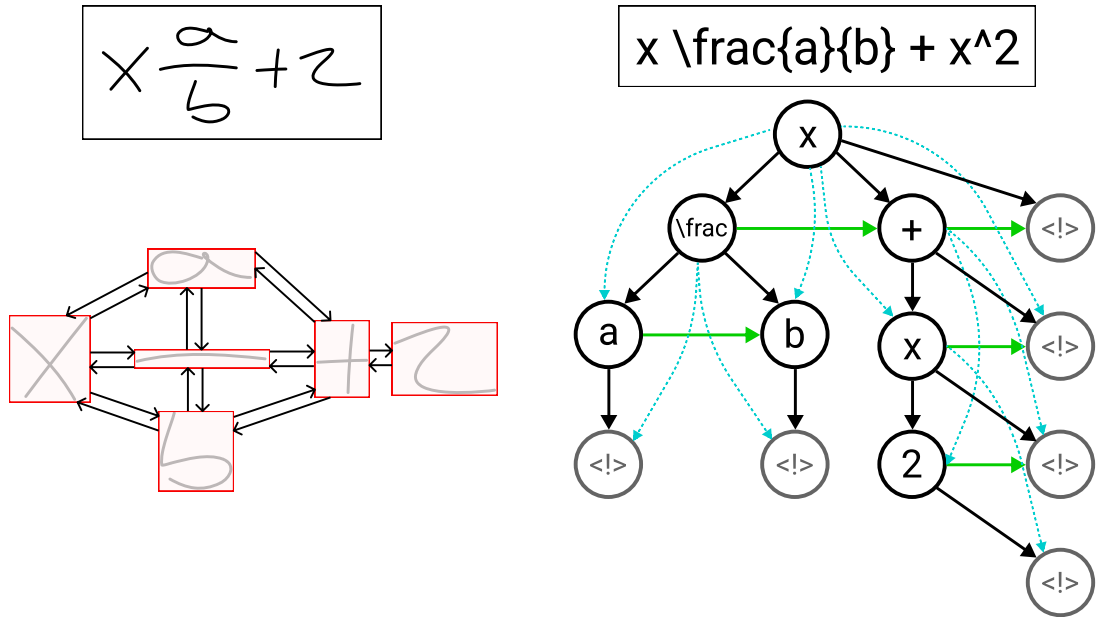
Navržený model tedy pracuje se vstupním obrázkem ručně psaného matematického výrazu. Ten je nejprve rozdělen pomocí analýzy propojeným komponent na jednotlivé symboly, nebo části symbolů. Nad těmito komponentami je následně sestaven Line of Sight (LoS) graf, který vyjadřuje vzájemnou viditelnost komponent v rámci výrazu. Z obrázků jednotlivých komponent, tedy výřezů obrázku odpovídajících minimálním ohraničujícím obdélníkům jsou extrahovány vizuální příznaky pomocí konvoluční neuronové sítě. Následně je sestaven vstupní graf G_x , jehož uzly reprezentují jednotlivé komponenty a hrany odpovídají sestrojenému LoS grafu.

Sestrojený graf je v další fázi zpracován kodérem v podobě grafové neuronové sítě, jehož výstupem je vnitřní reprezentace výrazu. Grafová neuronová síť tak do reprezentace zakomponuje informaci o sousedství jednotlivých komponent.

Dekodér využívající další grafovou konvoluční síť a attention mechanismus poté generuje výstupní reprezentaci výrazu v podobě rozšířeného Symbol Layout Tree (SLT) grafu G_y , kterou je možné transformovat na popis značkovacím jazykem LaTeX či MathML, nebo jinou podobnou reprezentaci. Schématické znázornění přístupu je na obrázku 4.1.



Obrázek 4.1: Schématické znázornění navrženého přístupu. Modré šipky značí tok dat během procesu rozpoznání výrazu matematického výrazu. Tento proces začíná segmentací, tedy analýzou propojených komponent vstupního obrázku. Nad komponentami je následně sestaven *Line of Sight (LoS)* graf spojující vzájemně viditelné komponenty a jsou extrahovány vizuální příznaky komponent pomocí konvoluční neuronové sítě. Tyto příznaky jsou následně použity jako příznakové vektory uzlů vstupního grafu, jehož hrany odpovídají LoS grafu. Graf je zpracován vrstvami grafové konvoluční sítě, modifikované *Graph Attention Network* pro získání vnitřní reprezentace. Následně je grafovým dekodérem postupně, po jednu uzlu, generován výstupní *Symbol Layout Tree (SLT)* graf, který je v poslední fázi transformován na reprezentaci v jazyce LaTeX.



(a) Line of Sight graf. Červená barva označuje oddělené komponenty výrazu a černé šipky orientované hrany mezi vzájemně viditelnými komponentami.

(b) Rozšířený Symbol Layout Tree graf. Černé šipky označují hrany z rodičovského uzlu dle základního SLT grafu. Zelené a modré šipky označují hrany z nejpravějšího bratrského, resp. prarodičovského uzlu a šedou barvou se symbolem <!--> jsou označeny koncové listové uzly přidávané v rámci rozšíření grafu pro řešenou úlohu.

Obrázek 4.2: Schématické znázornění sestaveného Line of Sight (LoS) grafu nad segmentovanými komponentami vstupního obrázku matematického výrazu a rozšířeného Symbol Layout Tree (SLT) grafu z anotace v jazyce LaTeX.

4.1 Konstrukce vstupního LoS grafu

Konstrukce vstupního grafu z obrázku vyžaduje segmentaci na jednotlivá primitiva, tedy části symbolů nebo celé symboly, extrakci příznaků jednotlivých primitiv a získání struktury grafu reflektující prostorové uspořádání primitiv v rámci výrazu. Pro segmentaci symbolů byla navržena řada přístupů [27] z nichž řada používá analýzu propojených komponent (CCA) nebo iterativní x-y řezy (recursive X-Y cut). Často jsou tyto přístupy ale kombinovány se supervizí v podobě modulu pro rozpoznávání segmentovaných symbolů. V tomto řešení je segmentace provedena pomocí analýzy propojených komponent, jelikož jsou jednotlivé symboly matematických výrazů ve vstupních obrázcích vždy dobře oddělené pozadím a výsledkem tak budou oddělené celé symboly, případně části symbolů, pokud se skládají z více částí jako například písmeno *i*, kde bude tečka oddělena a reprezentována samostatnou komponentou. Nemělo by tedy často nastat, že budou v jedné komponentě více symbolů najednou.

Po provedení segmentace je nad jednotlivými primitivy sestrojen neorientovaný *Line-Of-Sight (LOS)* graf, který reprezentuje jejich prostorové uspořádání a vzájemnou viditelnost. V tomto grafu existují hrany mezi uzly symbolů, které jsou ve výrazu vzájemně viditelné a neexistuje tedy jiný symbol výrazů, který by je navzájem zastínil. K jeho sestavení byl použit existující algoritmus [14], vytvářející totožný graf nad křivkami v upravené variantě

pro práci s obrázky komponent. Modifikovaný proces je popsán algoritmem 1. Principem tohoto algoritmu je výpočet zastínění zorného pole každé komponenty. Algoritmus iterativně pro každou komponentu inicializuje volné zorné pole na 360° . Následně iteruje přes všechny ostatní komponenty výrazu v pořadí daném rostoucí vzdáleností centroidu zdrojové komponenty od ohraničujících obdélníků těchto cílových komponent. Pro každou cílovou komponentu je následně vypočten rozsah úhlů zorného pole zdrojové komponenty, které ohraničující obdélník této cílové komponenty zastíňuje. Pokud je dle aktuálního stavu zorného pole zdrojové komponenty cílová komponenta plně viditelná, tak je přidána hrana reprezentující tuto viditelnost. Poté je zorné pole zdrojové komponenty zastíněno nově zpracovanou cílovou komponentou a pokračuje se zpracováním další komponenty.

Algoritmus 1 Algoritmus sestavení Line of Sight grafu

Vstup: množina propojených komponent výrazu $k \in K$

Výstup: Množina hran $E = \{(s, t) : \text{komponenta } t \text{ je viditelná z } s\}$

$E = \emptyset$ je prázdná množina hran

for každou komponentu $k \in K$ se středem ohranič. obdélníku $c_k = (x_0, y_0)$ **do**

$V = \{[0^\circ, 360^\circ]\}$ je volné zorné pole komponenty k

for každou další komponentu $l \in K \setminus \{k\}$ ve vzestupné vzdálenosti od c_k **do**

$\theta_{min} = +\infty, \theta_{max} = -\infty$

for každý vrchol $b_l = (x_b, y_b)$ ohraničujícího obdélníku l **do**

$d = b_l - c_k = (x_b - x_0, y_b - y_0)$ je směrový vektor

$\theta \leftarrow$ úhel vektoru b_l od horizontálního vektoru $h = (1, 0)$

$$\theta = \begin{cases} \arccos\left(\frac{b_l \cdot h}{\|b_l\| \|h\|}\right) & \text{if } y_b \geq y_0 \\ 360 - \arccos\left(\frac{b_l \cdot h}{\|b_l\| \|h\|}\right) & \text{if } y_b < y_0 \end{cases}$$

$\theta_{min} = \min(\theta_{min}, \theta), \theta_{max} = \max(\theta_{max}, \theta)$

$s = [\theta_{min}, \theta_{max}]$ je úhel zastínění zorného pole

for všechny $u \in U$ intervaly volného zorného pole **do**

if $s \in u \cap s$ **then**

$E = E \cup (k, l) \cup (l, k)$

$\triangleright k$ plně vidí l

end if

end for

$U = \bigcup_{u \in U} (u \setminus u \cap s)$

\triangleright aktualizace zastínění zorného pole

end for

end for

end for

Takto sestavený graf viditelnosti je znázorněn na obrázku 4.2a, kde jsou červeně zvýrazněné minimální ohraničující obdélníky jednotlivých primitiv a černě orientované hrany grafu vzájemné viditelnosti. Z implementačních důvodů jsou neorientované hrany reprezentovány kombinací dopředné a zpětné orientované hrany. Pro exaktní vyjádření prostorového rozložení primitiv jsou následně pro každou hranu, tedy dvojici primitiv extrahovány geometrické příznaky popisující jejich relativní velikost a polohu. Výčet těchto příznaků je v tabulce 4.1. Ze zmíněných komponent je následně sestaven vstupní graf G_x , kde příznakové vektory uzlů tvoří obrázky segmentovaných primitiv a příznakové vektory hran extrahované geometrické vlastnosti.

#	Popis
1	Minimální vzdálenost mezi komponentami
2	Maximální vzdálenost mezi komponentami
3	Vzdálenost mezi středy ohraničujících obdélníků komponent
4	Horizontální vzdálenost mezi středy ohraničujících obdélníků komponent
5	Vertikální vzdálenost mezi středy ohraničujících obdélníků komponent
6	Poměr plochy ohraničujících obdélníků komponent
7	Poměr větší z ploch ohraničujících obdélníků ku jejich sjednocení
8	Poměr šířky ohraničujících obdélníků
9	Poměr výšky ohraničujících obdélníků
10	Poměr diagonál ohraničujících obdélníků

Tabulka 4.1: Výčet geometrických vlastností komponent vstupního LoS grafu, použitých pro sestavení příznakových vektorů hran tohoto grafu.

4.2 Výstupní SLT graf

Výstupní Symbol Layout Tree (SLT) graf reprezentující matematický výraz popisuje jednotlivé symboly a prostorové rozložení výrazu. Lze jej sestavit ze zápisu výrazu značkovacím jazykem jako je LaTeX nebo MathML, jelikož tyto zápisy právě tento graf výrazu popisují. Základní forma acyklického orientovaného SLT grafu se skládá z uzlů reprezentujících symboly a hran reprezentujících prezentační strukturu. Každá z těchto hran směřuje z rodičovského do synovského uzlu a nese příznak vzájemné polohy symbolů, tedy *vpravo*, *horní index*, *dolní index*, *uvnitř*, *pod* nebo *nad*. Pro potřeby generování tohoto grafu neuronovou sítí je tato základní forma rozšířena pro každý uzel o hranu z prarodičovského uzlu, hranu z posledního levého bratra a hranu odpovídající smyčce. Toto rozšíření jasně definuje pořadí uzlů grafu při jeho procházení a přináší tak deterministický výsledek nezávislý na permutaci matice sousednosti grafu. Při generování výstupního grafu je také nutné zajistit, že dekodér dokáže rozhodnout kdy přestat generovat potomky uzlu. Pro tento účel je ke každému uzlu grafu připojen speciální koncový listový symbol jako jeho nejpravější potomek, který právě tuto situaci označuje. Takto zkonstruovaný rozšířený SLT graf G_y lze vidět na obrázku 4.2b. Zde jsou hrany směřující z rodičovských do synovských uzlů označeny černou barvou, hrany z prarodičovského uzlu modrou barvou a hrany z levého bratra zelenou barvou. Koncové listové uzly jsou označeny šedou barvou a symbolem $\langle ! \rangle$. Ze schématu je patrné, že pokud uvažujeme pouze hrany typu rodičovský-synovský uzel, tak se stále jedná o strom, což je pro reprezentaci zásadní.

4.3 Grafový kodér

Grafový kodér je prvním blokem navrženého modelu a skládá se z konvoluční neuronové sítě pracující s obrázky jednotlivých primitiv, resp. uzlů grafu následované modifikovanou grafovou sítí typu *Graph Attention Network (GAT)*.

Konvoluční neuronová síť slouží pro extrakci vizuálních příznaků primitiv. Vzhledem ke skutečnosti, že obrázky budou vždy obsahovat jeden symbol, resp. část symbolu v relativně malém rozlišení se může jednat o relativně malou síť typu VGG [29], například VGG11, nebo síť typu ResNet [12], která byla použita pro stejný problém u modelu představeného v [26]. Toho řešení používá konvoluční síť VGG11.

Samotné kódování vstupního grafu G_x do vnitřní reprezentace provádí následující grafová neuronová síť, konkrétně modifikovaná *Graph Attention Network (GAT)*, jejíž architektura byla inspirována modelem *graph2graph* [36]. Modifikace oproti standardní architektuře *GAT*, která je popsána v sekci 3.1.4 spočívá v rozšíření o příznakové vektory hran, jejichž aktualizace v každé vrstvě předchází úpravě příznakových vektorů uzlů. V rámci výpočtu jedné vrstvy sítě jsou tedy vždy nejprve aktualizovány příznakové vektory hran $b_q^{i,j}$ dle:

$$b_q^{i,j} = \text{LeakyReLU}(W_b[h_{q-1}^i \parallel b_{q-1}^{i,j} \parallel h_{q-1}^j]), \quad (4.1)$$

kde q označuje index, resp. pořadí vrstvy grafové sítě, W_b je učená matice vah, h_q^i je příznakový vektor uzlu i na výstupu vrstvy q a \parallel značí konkatenci. Z tohoto výrazu je tedy patrné, že příznakové vektory hran se aktualizují dle jejich předchozích hodnot, ale také příznakových vektorů jí náležících uzlů.

Následně jsou aktualizovány příznakové vektory uzlů grafu dle:

$$h_q^i = \text{LeakyReLU}\left(\sum_{j \in N(i) \cup \{i\}} \alpha_{i,j} W_h h_{q-1}^j\right), \quad (4.2)$$

kde h_q^i je příznakový vektor uzlu i , $N(i)$ označuje množinu sousedních uzlů uzlu i , W_h je matice vah a $\alpha_{i,j}$ je *attention* koeficient, tedy míra relevantnosti uzlu j pro uzel i , dána následujícím výrazem:

$$\alpha_{i,j} = \frac{\exp(u^T \text{LeakyReLU}((W'_c b_q^{j,i} \parallel W''_h h_{q-1}^i \parallel W'_h h_{q-1}^j)))}{\sum_{k \in N(i)} \exp(u^T \text{LeakyReLU}((W'_c b_q^{k,i} \parallel W''_h h_{q-1}^i \parallel W'_h h_{q-1}^k)))}, \quad (4.3)$$

kde u značí vektor vah, a W'_c , W'_h a W''_h jsou matice vah. Použití grafové neuronové sítě místo enkodéru v podobě samostatné konvoluční sítě by mělo do vnitřní reprezentace promítnout uspořádání a sousednost symbolů ve výrazu.

4.4 Grafový dekodér

Grafový dekodér je druhým blokem navrženého modelu, který sekvenčně generuje výstupní matematický výraz v podobě SLT grafu G_y . Tento blok opět vychází z metody *graph2graph* [36] a je inspirován architekturou využívající konvoluční neuronové sítě pro učení nad sekvenčními daty prezentovanou v [10]. V každém kroku je vygenerován jeden uzel a příslušné hrany, které jej spojují s již vygenerovanou částí výstupního grafu.

Základní komponentou dekodéru jsou bloky, z nichž každý disponuje vrstvou modifikované grafové konvoluční sítě *GCN* [16] následovanou *attention* mechanismem. Na tyto bloky lze nahlížet podobně jako na vrstvy samostatné grafové konvoluční sítě a je možné je i stejným způsobem vrstvit za sebe. V rámci každého takového bloku m jsou příznakové vektory z_m^i uzlů $i \in \{1, \dots, I = |V_y|\}$, aktualizovány na základě výstupu předchozího bloku, resp. stavu výstupního grafu G_y po zpracování předchozí vrstvou, a také dle vnitřní reprezentace vstupního grafu G_x zprostředkované prostřednictvím *attention* mechanismu. Formálně je tedy aktualizace hodnoty příznakového vektoru uzlu z_m^i dána dle:

$$z_m^i = W_z e_m^i + c_m^i, \quad (4.4)$$

kde W_z je matice vah, e_m^i příznakový vektor produkovaný vrstvou grafové konvoluční sítě bloku a c_m^i je kontextový vektor získaný pomocí *attention* mechanismu.

Výpočet příznakového vektoru e_m^i grafovou konvoluční sítí je definován jako:

$$e_m^i = \sum_{j \in N(i)} W_{t(j,i)} z_{m-1}^j, \quad (4.5)$$

kde $W_{t(j,i)}$ je skupina matic vah, která je volena dle typu hrany $t(j,i)$ z uzlu j do uzlu i . Konkrétně je tedy volena jiná matice vah pro každý z typů hran, tedy z rodičovského uzlu, prarodičovského uzlu, bratrského uzlu a sebe sama, resp. pro smyčku. Tato skutečnost modelu umožňuje rozlišovat mezi jednotlivými typy hran a reagovat tak na strukturu grafu.

Kontextový vektor c_m^i je, jak již bylo uvedeno, získán pomocí *attention* mechanismu nad vnitřní reprezentací vstupního grafu dle:

$$c_m^i = \sum_{n=1}^{|V_x|} \alpha^{i,n} W_a h^n, \quad (4.6)$$

kde W_a je matice vah, h^n je příznakový vektor uzlu v^n vstupního grafu G_x a $\alpha^{i,n}$ je *attention* koeficient vyjadřující významnost příznakového vektoru h^n pro uzel i výstupního grafu G_y určený dle:

$$\alpha^{i,n} = \frac{\exp(q^i \cdot (W_b h^n))}{\sum_{j=1}^{|V_x|} \exp(q^i \cdot (W_b h^j))} \quad (4.7)$$

$$q^i = \sigma(W_e e_i + W_{bro} z_{m-1}^{bro(i)} + W_{pa} z_{m-1}^{pa(i)}) \quad (4.8)$$

Zde W_b , W_e , W_{bro} a W_{pa} jsou matice vah, σ je sigmoidální aktivační funkce, $bro(i)$ je funkce pro získání příznakového vektoru levého bratra uzlu i a $pa(i)$ slouží pro získání příznakového vektoru rodičovského uzlu i ve výstupním grafu i . Pokud tento uzel v grafu neexistuje tak funkce vracejí nulový vektor, jehož rozměry odpovídají příznakovým vektorům.

Při generování výstupního grafu dekodérem je v každém kroku vložen nový uzel, jehož příznakový vektor z_0^i je inicializován na nulový a je napojen hranami na existující část grafu. Následuje zpracování jednotlivými bloky grafového dekodéru. Výstupní příznakový vektor z_m^i je následně použit v predikci symbolu. Pokud se jedná o ukončovací listový uzel, tak tímto končí generování sourozenců aktuálního uzlu, v opačném případě se pokračuje dalším uzlem.

Pro sestavení výstupního grafu je po zpracování dekodérem predikováno pravděpodobnostního rozložení symbolů \hat{p}_{sym} pro každý uzel dle jeho příznakového vektoru. Pro každou hranu typu rodič-potomek je pak predikováno pravděpodobnostní rozložení jejího typu, resp. strukturálního vztahu \hat{p}_{rel} , určené z příznakových vektorů obou korespondujících uzlů dle:

$$\hat{p}_{sym}^i = \text{softmax}(W_{sym} z^i) \quad (4.9)$$

$$\hat{p}_{rel}^{pa(i),i} = \text{softmax}(W_{rel}[z^{pa(i)}, z^i]) \quad (4.10)$$

Zde W_{sym} a W_{rel} jsou matice vah a z^i je příznakový vektor uzlu i na výstupu posledního bloku dekodéru.

Takto definovaná architektura dekodéru umožňuje ve fázi trénování zpracovat všechny uzly výstupního grafu paralelně, tedy současně, a tím markantně urychlit trénování modelu. Pro tento účel jsou vstupní příznakové vektory uzlů z_o^i určeny pomocí *embedding* vrstvy a jako nulový je pro každý uzel uvažován pouze jeho vlastní příznakový vektor.

Jelikož dekodování v evaluační fázi probíhá sekvenčně po jednom uzlu, tak je zde vysoká citlivost na správnou klasifikaci každého ze symbolů. Pokud se v počáteční fázi relativně blízko kořene stromu, respektive výstupního grafu dekodéru nepovede správně klasifikovat symbol uzlu bude tato chyba propagována do zbytku výstupního grafu a pravděpodobně to poveden na vysoce chybné rozpoznání výrazu. Tento přístup odpovídá hladovému algoritmu, tzv. *Greedy search*. K potlačení nedostatku byla navržena druhá varianta generování stromu realizující paprskové prohledávání, tzv. *Beam search*. Při tomto přístupu je generováno několik výstupních grafů zároveň, jejichž počet je definován šířkou paprsku. V každém kroku je přidán nový uzel do každého z těchto grafů a je zpracován bloky dekodéru. Z jeho příznakového vektoru je následně klasifikováno několik nejpravděpodobnějších symbolů, opět dle šířky paprsku a je pro ně určena apriorní pravděpodobnost dle počtu stromů, ve kterých patří do této n -tice nejpravděpodobnějších. Pro každý z těchto vybraných symbolů je následně určena posteriorní pravděpodobnost pro všechny generované stromy. Do každého ze stromů je pak vložen ten z uvedené skupiny symbolů, který má nejvyšší posteriorní pravděpodobnost právě pro tento strom. Po dokončení generování všech stromů je vybrán jeden s nejvyšší společnou pravděpodobností všech jeho symbolů a hran a je zvolen za výsledek.

Supervize nad dílčími bloky modelu je prováděna pro zrychlení trénování a potenciálně vylepšení úspěšnosti trénovaného modelu, její vliv je předmětem jednoho z prováděných experimentů popsanych níže. Supervize je prováděna kromě standardního výstupů pro klasifikaci uzlů a hran výstupního grafu také nad uzly vstupního grafu po zpracování grafovou sítí a dle jejich příznakových vektorů jsou predikovány symboly, kterým uzly náležejí. V rámci provedeného experimentu je také testován vliv obdobného rozšíření, ale v momentě před zpracováním grafovou sítí, jedná se tedy o supervizi nad výstupy konvoluční VGG sítě extrahující vizuální příznaky. Posledním rozšířením výpočtu chyby modelu je kontrola nad samotným attention mechanismem. Ideální koeficienty attention koeficientů pro uzly výstupního grafu jsou sestrojeny tak, aby jejich hodnoty odpovídaly rovnoměrnému rozložení pro uzly vstupního grafu, které náležejí uzlu výstupního grafu a jinak byly nulové, tedy $\forall y^i \in G_y$:

$$\alpha^{i,n} = \begin{cases} 1/|c(i)| & \text{pokud } n \in c(i) \\ 0 & \text{jinak} \end{cases}, \quad (4.11)$$

kde $c(i)$ je množina indexů n všech uzlů vstupního grafu, které odpovídají uzlu i výstupního grafu a $|c(i)|$ je mohutnost této množiny.

Kapitola 5

Implementace

Tato kapitola se zabývá vybranými implementačními detaily této práce. První sekce popisuje nástroje použité pro zpracování datové sady a trénování neuronových sítí. Následující části se zabývají implementací jednotlivých modulů, tedy grafového kodéru, respektive dekodéru, které jsou základními bloky implementovaného modelu. Poté je popsán proces zpracování dat a modul sloužící pro trénování a vyhodnocování modelu.

5.1 Použité nástroje

Implementační část této práce je napsána v programovacím jazyce Python. Pro trénování neuronových sítí je použita knihovna *Pytorch* [24] a nad ní sestavená knihovna *Pytorch Geometric* [9]. Pytorch je knihovna cílená pro strojové učení, zejména trénování a vyhodnocování neuronových sítí. Poskytuje vysokoúrovňové funkce akcelerované pro výpočet na GPU. Pytorch Geometric je knihovna zaměřená na práci se strukturovanými a grafovými neuronovými sítěmi.

Trénování modelů bylo prováděno v cloudové platformě Gradient Paperspace¹ na stroji s grafickou kartou NVIDIA Tesla V100 32GB.

Zdrojový kód nástroje je členěn do několika Python modulů dle funkčnosti, kterou zajišťuje. Mimo tyto moduly je implementována třída **Trainer**, která zastřešuje inicializaci modelu, dle předané konfigurace a jeho trénování, resp. vyhodnocení.

5.2 Načítání a příprava dat

Načítání a příprava dat je realizována modulem **data**. V rámci tohoto modulu je implementována třída **GMathDataset**, která reprezentuje použitý *dataset* a realizuje vytvoření jednotlivých datových elementů včetně metadat potřebných pro trénování modelu. Při dotazu na prvek datasetu jsou vrácená data zapouzdřena do objektu třídy **GMathData**. Součástí této třídy jsou také definované instrukce pro úpravu datových položek za účelem sestavení trénovací dávky (tzv. *mini-batch*). Při tvorbě mini-batch se datové položky, tedy výčet uzlů jejich grafů apod. sjednotí a dále se na ně nahlíží jako na jediný graf. Jelikož grafy jednotlivých datových položek nesdílí žádné hrany, tak se dále při zpracování neuronovou sítí nijak neovlivňují. Instrukce pro sjednocení třídy **GMathData** pak určují v jakých dimenzích tenzory dílčích datových položek sjednocovat, jakým způsobem inkrementovat indexy sloužící jako reference do jiných tenzorů a pro jaké tenzory generovat mapovací vektory

¹<https://gradient.run/>

identifikující které datové položce náleží jejich hodnoty. Samotné načítání, resp. dotazování dat pro zpracování modelem provádí třída `DataLoader`, která je součástí knihovny Pytorch Geometric a automaticky interpretuje definované instrukce pro mini-batch načítání.

Použitá datová sada CROHME poskytuje ke každému datovému prvku vstupní obrázek matematického výrazu a anotaci v podobě souboru InkML, jehož součástí je zápis výrazu v jazyce LaTeX a MathML, a také definice trajektorie jednotlivých tahů elektronickým perem včetně reference na symbol, ke kterému náleží. Oba zmíněné soubory jsou použité k sestavení datového elementu.

Sestavení vstupního grafu je provedeno z obrázku ručně psaného matematického výrazu. Jsou použity vstupní obrázky v rozlišení 310×310 pixelů. V první fázi je provedena analýza propojených komponent pomocí funkcí poskytovaných knihovnou *OpenCV*², jejíž výstupem jsou jednotlivé komponenty, tedy části symbolů nebo celé symboly včetně centroidů a minimálních ohraničujících obdélníků. Následně je sestaven Line of Sight (LoS) graf viditelnosti podle algoritmu definovaného v sekci 4. Pro hrany v takto sestaveném grafu jsou poté určeny příznaky, které reflektují vzájemnou pozici komponent náležících každé hraně.

Konstrukce cílového SLT grafu je provedena z anotace v InkML souboru. Jelikož anotace v jazyce LaTeX často obsahuje nadbytečné formátovací znaky, pro úpravu například řezu písma, je cílový graf sestaven transformací popisu výrazu v jazyce MathML. Převod z tohoto formátu je také přímočařejší, jelikož má popis v tomto jazyce přímo požadovanou stromovou strukturu. Rekurzivním průchodem do hloubky stromem MathML zápisu výrazu jsou tedy získány jednotlivé symboly, které odpovídají uzlům SLT grafu a jejich vzájemné strukturální vztahy, které odpovídají hranám grafu. Pro dokončení konstrukce rozšířeného SLT grafu jsou následně připojeny koncové listové uzly, sloužící jako značka posledního, nejpravějšího přímého potomka každého uzlu, a hrany směřující do každého uzlu z jeho prarodiče, levého bratra a sebe sama.

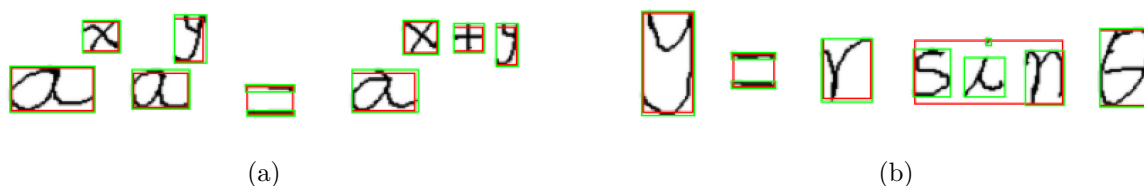
Následně je zpracován popis jednotlivých tahů elektronického pera a ty jsou dle identifikátorů sdruženy do skupin, pro které jsou vypočteny ohraničující obdélníky. Každá z těchto skupin náleží jednomu ze symbolů matematického výrazu a dle identifikátorů v anotaci je mu přiřazena. Získané ohraničující obdélníky jednotlivých symbolů výrazu určují jejich pozici a umožňují tak sestavení *ground-truth* pro attention mechanismus do vstupního LoS grafu a také pro klasifikaci komponent vstupního grafu. Pro každou komponentu vstupního grafu je tedy zvolen korespondující symbol výstupního grafu dle nejvyššího překrytí jeho ohraničujícího obdélníku s ohraničujícím obdélníkem skupiny křivek symbolu, jak je znázorněno na obrázku 5.1.

5.3 Neuronová síť

Implementace modelu se nachází v modulu `model`. Modul obsahuje třídu `Model` reprezentující celou neuronovou síť a implementace všech jeho dílčích bloků. Nachází se zde implementace konvoluční neuronové sítě VGG11 [29], popis vrstev této sítě je v tabulce 5.1.

Třída `GATConv` představuje implementaci modifikované *Graph Attention Network (GAT)* použité v kodéru modelu. Implementace je inspirovaná modelem standardní sítě GAT dostupným v knihovně Pytorch Geometric.

²<https://opencv.org/>



Obrázek 5.1: Ukázka přiřazení komponent vstupního grafu jednotlivým symbolům výstupního grafu. Zelená barva označuje ohraničující obdélníky komponent vstupního grafu. Červenou barvou jsou vyznačeny přiřazené korespondující symboly výstupního grafu.

Bloky	VGG11
	1, 64, $3 \times 3, 2 \times 2$
Konvoluční vrstvy a max-pooling	1, 128, $3 \times 3, 2 \times 2$
	2, 256, $3 \times 3, 2 \times 2$
	2, 512, $3 \times 3, 2 \times 2$
	2, 512, $3 \times 3, 2 \times 2$
	512, 0.5
Plně propojené vrstvy	512, 0.5
	256, 0.5

Tabulka 5.1: Bloky implementované konvoluční sítí VGG11, použité v implementovaném modelu pro extrakci vizuálních příznaků ručně psaných symbolů.

Dekodér sítě jako nejrozsáhlejší část modelu reprezentuje třída `Decoder`. Metoda `forward` realizující dopředný průchod je zde rozdělena na dvě větve, z nichž je jedna volena v trénovací fázi a druhá při vyhodnocování modelu. Při trénování je dekodéru předložena struktura výstupního grafu dle anotace a pro všechny uzly grafu jsou vytvořeny *embedding* vektory. Poté jsou všechny uzly paralelně zpracovány grafovou neuronovou sítí s attention mechanismem a jsou vytvořeny logity pro predikci symbolů a tříd hran. Ve fázi vyhodnocování je zpracování přepojeno do druhé větve, kde je metodou `gen_graph` generován výstupní graf sekvenčně po jednom uzlu bez dodané znalosti požadované struktury grafu. Generování uzlů je prováděno způsobem, který simuluje procházení výstupního grafu do hloubky – s uvážením pouze hran typu rodičovský-synovský uzel, tím je zaručeno, že při přidání každého nového uzlu již existuje kompletní část výstupního grafu, která mu předchází, ve směru od kořene daném orientací hran, a tudíž ovlivňuje jeho hodnotu.

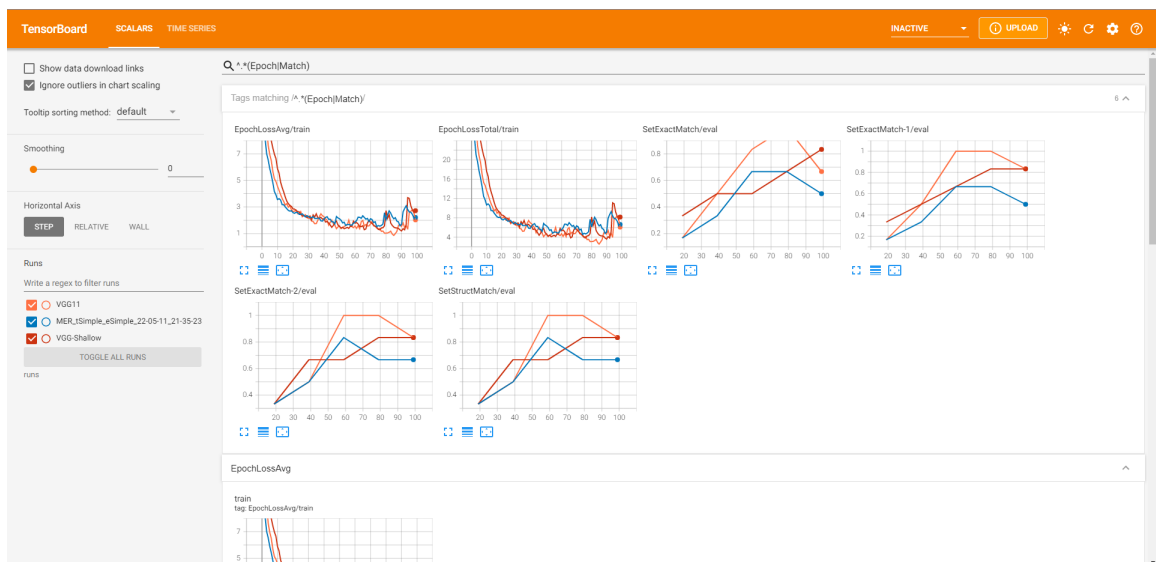
Pro dekódování v evaluačním módu byly implementovány dva přístupy. První přístup realizuje tzv. *Greedy search*, tedy hladový algoritmus. Při generování výstupního grafu je tedy po vložení nového uzlu s inicializovanou hodnotou a zpracování bloky dekodéru vybrán pouze jeden nejpravděpodobnější symbol, jehož embedding vektor je použit jako počáteční příznakový vektor daného uzlu pro generování zbývajících částí výstupního grafu. Nevýhodou tohoto přístupu je vyšší pravděpodobnost chybně klasifikovaného uzlu, což negativně ovlivní generování zbytku výstupního grafu. Díky tvorbě pouze jednoho výstupního grafu tato metoda podporuje testovací dávky (mini-batch) při vyhodnocování. Modul `utils` pro tuto situaci obsahuje funkci `split_databatch` pro rozdělení výstupu sítě na jednotlivé datové elementy dle vygenerovaných map příslušnosti pro transformaci výstupního grafu do jazyka LaTeX a vyhodnocení modelu. Třída `SLTParser`, která slouží k převodu výstupního grafu sítě do jazyka LaTeX se nachází v modulu `data`.

Pro minimalizaci výše zmíněného problému hladového algoritmu byl implementován druhý přístup tzv. *Beam search*, tedy paprskové prohledávání s konfigurovatelnou šířkou paprsku. Tento algoritmus vytváří několik výstupních grafů zároveň, jejichž počet je dán šířkou paprsku. Při generování nového uzlu je vložena jeho inicializovaná hodnota do každého z těchto grafů a jsou zpracovány bloky dekodéru. Následně je z každého grafu vybráno několik nejpravděpodobnějších symbolů, opět dle šířky paprsku. Pro každý z těchto symbolů je následně vypočtena posteriorní pravděpodobnost pro každý z výstupních grafů a do každého je vložen embedding vektor symbolu s nejvyšší posteriorní pravděpodobností podmíněnou zbytkem již vygenerovaného grafu. Implementace této metody nepodporuje použití větší testovací dávky než jeden datový prvek, jelikož je pro jeden datový prvek vytvářeno několik výstupních grafů.

5.4 Trénování a sledování úspěšnosti

K trénování a testování modelů, včetně jejich vyhodnocování slouží třída `Trainer`. Nastavení běhu je možné pomocí konfiguračního JSON souboru, který je nejprve zpracován třídou `Config`, následně předávající zpracovanou konfiguraci třídě `Trainer` při její inicializaci. Tímto způsobem je možné jednoduše nastavit požadované bloky modelu, včetně jejich parametrů a podobně. Uložení v externím souboru také umožňuje přehlednější práci s modely a jejich výsledky. Kromě specifikace požadovaného modelu se v souboru specifikuje také požadovaná akce s modelem včetně parametrů pro trénování, vyhodnocování a případné cesty, například k datové sadě nebo adresáři pro ukládání dílčích výsledků.

K sledování průběhu trénování a vyhodnocování je použitý nástroj *Tensorboard* knihovny *Tensorflow* [1], který umožňuje jednoduchou vizualizaci výsledků vyhodnocování a dalších příznaků sledovaných při trénování modelů, včetně jejich porovnání napříč modely. Na obrázku 5.2 je ukázka uživatelského rozhraní tohoto nástroje a několika sledovaných metrik. Výhodou nástroje je jednoduchá integrace s knihovnou Pytorch a jeho možnost použití na lokálním počítači, nebo například cloudové platformě Gradient Paperspace, která byla využita pro trénování modelů v rámci této práce.



Obrázek 5.2: Ukázka vizualizace výsledků trénovaných modelů nástrojem Tensorboard.

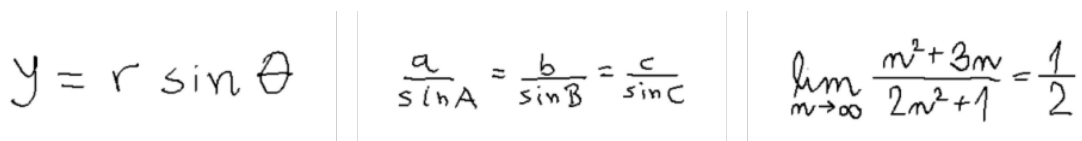
Kapitola 6

Experimenty a vyhodnocení

Tato kapitola se zabývá experimenty, které byly v rámci práce provedeny. V první části je krátce představena použitá trénovací a testovací datová sada. Následně je popsán způsob, jakým byly modely vyhodnocovány a použité metriky. Dále se zde nachází popis a výsledky jednotlivých provedených experimentů. První experiment sleduje vliv velikosti neuronové sítě na její úspěšnost. V rámci experimentu jsou porovnány výsledky tří modelů různé velikosti. Druhý experiment se zaměřuje na vliv rozšíření chybové funkce o supervizi nad výstupy dílčích bloků modelu. Třetí experiment sleduje vliv prohledávacího algoritmu při dekódování výstupního grafu na úspěšnost rozpoznání. Porovnáván je základní hladový algoritmus *Greedy search* a paprskové prohledávání *Beam search* s různou šířkou paprsku.

Datová sada Pro všechny experimenty byla použita datová sada CROHME 2019 obsahující celkem 12178 matematických výrazů, předem rozdělených do trénovací, validační a testovací sady, která je blíže popsána v kapitole 2.2. Datová sada byla zvolena, jelikož poskytované anotace obsahují potřebné informace k supervizi nejen nad výstupními grafem, resp. výrazem v jazyce LaTeX, ale také pro dohled na konvoluční sítě pro extrakci příznaků, enkodérem a attention mechanismem. Druhým důvodem je také skutečnost, že tato datová sada byla použita pro vyhodnocení řady podobných existujících modelů, její použití tak umožňuje porovnání s těmito řešeními. Postup zpracování datové sady a příprava potřebných anotací je popsána v předchozí kapitole. Ukázka obrázků matematických výrazů této použité sady je na obrázku 6.1

Vyhodnocení úspěšnosti modelu je prováděno metrikou míra rozpoznání výrazů *expression recognition rate (ExpRate)*, která je také hlavní metrikou při vyhodnocování řešení v rámci soutěže CROHME. ExpRate vyjadřuje procentuální úspěšnost přesného rozpoznání výrazu a je vyhodnocována nad *Symbol Layout Tree (SLT)* grafem matematického výrazu. $\text{ExpRate} \leq 1$, ≤ 2 a ≤ 3 označují míry rozpoznání výrazu, kde povolena jedna, dvě nebo tři chyby na úrovni rozpoznání symbolu. V rámci této metriky je také uváděna procentuální


$$y = r \sin \theta \quad \left| \quad \frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} \quad \left| \quad \lim_{m \rightarrow \infty} \frac{m^2 + 3m}{2m^2 + 1} = \frac{1}{2}$$

Obrázek 6.1: Ukázka použité datové sady CROHME.

Varianta modelu	ExpRate	ExpRate ≤ 1	ExpRate ≤ 2	Struktura
Základní model 3, 256, 400	13.34	18.04	24.68	34.18
Méně vrstev 2, 256, 400	9.24	12.69	25.31	31.93
Menší model 3, 128, 256	4.74	10.21	20.85	24.38

Tabulka 6.1: Úspěšnosti modelů v rámci experimentů s velikostí sítě. Jednotlivá čísla charakterizující model popisují zleva: počet bloků dekodéru, délku příznakových vektorů dekodéru, délku attention vektoru. Všechny ExpRate hodnoty jsou uvedeny v procentech (%). Sloupec struktura označuje procentuální zastoupení výrazů se správnou strukturou generovaného SLT grafu.

úspěšnost rozpoznání struktury výrazu, tedy SLT grafu. Vyhodnocení ExpRate je prováděno přímo v rámci implementovaného řešení. Pro bližší analýzu výsledku byly použity také nástroje dostupné v knihovně *CROHMElib*, poskytované v rámci soutěže CROHME, které kromě ExpRate vyhodnocují další ukazatele na úrovni jednotlivých výrazů.

Nástroj vyhodnocuje také Levenštejnovu editační vzdálenost nad výstupní sekvencí symbolů jazyka LaTeX. Levenštejnova vzdálenost odpovídá nejmenšímu nutnému počtu úprav výstupního seznamu takových, aby vznikl cílový seznam. Tato metrika slouží spíše jako podpůrný ukazatel úspěšnosti rozpoznání, jelikož nereflektuje sémantický význam matematického výrazu a není tedy uváděna v této práci.

6.1 Velikost sítě

První experiment se zaměřuje na vliv velikosti sítě na úspěšnost rozpoznání matematických výrazů. Při experimentu byly porovnány výsledky základní varianty modelu se třemi vrstvami dekodérových bloků nejprve s totožným modelem s pouze dvěma takovými vrstvami. Snížením počtu vrstev grafové neuronové sítě se pro každý uzel sníží velikost okolí, které se podílí na aktualizaci jeho hodnoty. Předpokladem je, že by snížení na dvě vrstvy nemělo mít zásadní vliv na úspěšnost modelu vezmeme-li v potaz, že dle definované struktury rozšířeného SLT stromu je každý uzel přímo spojen se svým rodičovským i prarodičovským uzlem a už při použití jedné vrstvy je z hlediska SLT stromu aktualizován dle k-skokového okolí.

Následně je základní varianta modelu porovnána s menší variantou z hlediska délky příznakových vektorů a je pozorován vliv na úspěšnost.

Výsledky úspěšnosti rozpoznávání výše zmíněných modelů a přesnější charakteristika jejich parametrů je v tabulce 6.1. Z úspěšností jednotlivých variant modelu je patrné, že neuronová síť s více vrstvami a delšími vektory příznaků dosahovala nejlepších výsledků. Zároveň má odebrání jedné vrstvy dekodéru menší dopad na snížení úspěšnosti, než zmenšení příznakových vektorů. Menší vliv snížení počtu vrstev dekodéru může být důsledkem skutečnosti, že definice výstupního grafu obsahuje do každého uzlu hrany z rodiče, levého bratra a prarodiče a tím pádem i při použití pouze dvou vrstev agregována informace z relativně širokého okolí uzlu, vztaženo k základní struktuře Symbol Layout Tree (SLT) grafu. Snížením velikosti příznakových vektorů také klesá kapacita modelu učit se a je tedy možné, že snížený počet parametrů není pro model dostačující.

Model	ExpRate	ExpRate ≤ 1	ExpRate ≤ 2	Struktura
Základní model	12.18	14.54	27.12	31.29
+ sup VGG	13.34	18.04	24.68	34.18
+ sup enkodér GAT	10.84	14.26	20.33	27.50

Tabulka 6.2: Porovnání úspěšnosti základního modelu po přidání dodatečné supervize. Všechny ExpRate hodnoty jsou uvedeny v procentech (%). Sloupec struktura označuje procentuální zastoupení výrazů se správnou strukturou generovaného SLT grafu.

6.2 Dodatečná supervize

Tento experiment zkoumá vliv doplnění supervize na výstupy jednotlivých bloků modelu, tedy přidání dalších výstupů neuronové sítě a tím pádem rozšíření chybové funkce. Ve fázi načítání a přípravy dat je kromě anotací pro třídy uzlů a hran výstupního grafu sestrojena mapa korespondence uzlů vstupního a výstupního grafu, jak bylo popsáno v sekci 5.2. Díky tomu je možné sestrojít anotace pro vstupní graf, které jsou pro tento experiment zásadní.

V rámci tohoto experimentu je jako referenční uvažován základní model bez dodatečné supervize. Následně je porovnána úspěšnost a rychlost učení totožného modelu po přidání supervize na výstupy konvoluční sítě pro extrakci příznaků pro klasifikaci symbolů uzlů vstupního grafu. Ve druhé části je přidána obdobná supervize na výstupy grafové neuronové sítě enkodéru, opět pro klasifikaci symbolů vstupního grafu.

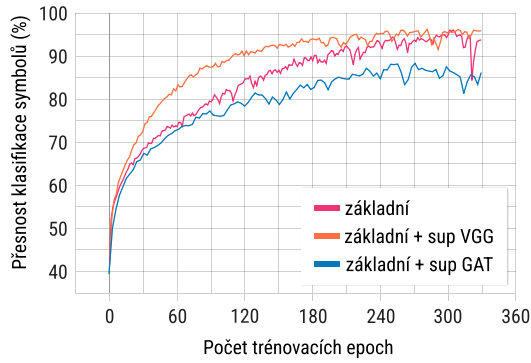
Výsledky dosažených úspěšností modelu, který je objektem experimentu jsou vypsaný v tabulce 6.2 a porovnání průběhů změny chybové funkce během trénování je vizualizováno na grafu 6.3. *Sup VGG* zde označuje přidání výstupu sítě na uzly vstupního grafu po zpracování VGG konvoluční sítí pro extrakci vizuálních příznaků a *sup enkodér* značí přidání výstupu sítě na uzly vstupního grafu po zpracování grafovou sítí enkodéru. Obdobně je na grafech 6.2 zobrazeno srovnání změny úspěšnosti klasifikace uzlů a hran výstupního grafu v trénovací fázi.

Z výsledků uvedených v tabulce 6.2 lze vidět, že dodatečná klasifikace uzlů vstupního grafu po zpracování konvoluční sítí má pozitivní vliv na úspěšnost sítě. V případě supervize nad výstupy grafové konvoluční sítě se jedná naopak o zhoršení úspěšnosti. Tuto skutečnost lze pozorovat také na grafech 6.2, kde je patrné, že po přidání této dodatečné klasifikace nad vstupním grafem dochází také ke snížení přesnosti klasifikace uzlů a hran výstupního grafu v trénovací fázi.

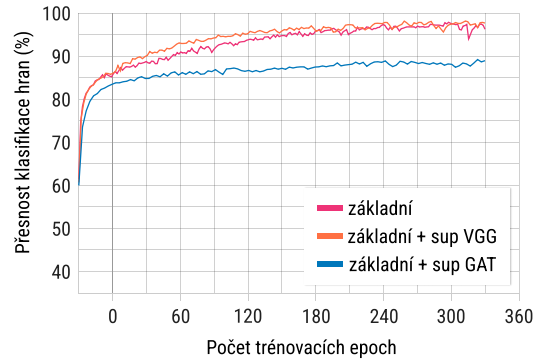
6.3 Prohledávací algoritmus

Cílem tohoto experimentu je posoudit přínos *Beam search* algoritmu při generování výstupních grafů oproti hladovému algoritmu, tedy *Greedy search*. V rámci experimentu jsou použity dva různé natrénované modely a je porovnána jejich míra rozpoznání výrazů ExpRate pro Greedy search a Beam search s několika různými šířkami paprsku. Předpokladem je, že úspěšnost rozpoznání při použití Beam search algoritmu bude vyšší, než u Greedy search algoritmu. Při příliš vysoké šířce paprsku už by však mohl tento algoritmus začít brát v poraz jako relevantní i méně pravděpodobné symboly, které by mohly naopak proces generování svést od správného řešení.

Výsledky porovnání prohledávacích algoritmů jsou v tabulce 6.3. V této tabulce lze vidět, že u obou testovaných modelů při použití paprskového prohledávání dochází k poklesu

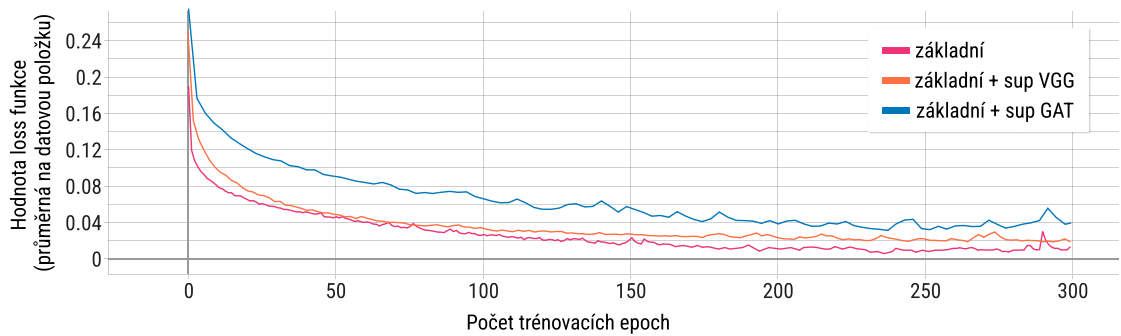


(a) Průběh přesnosti klasifikace uzlů.



(b) Průběh přesnosti klasifikace hran.

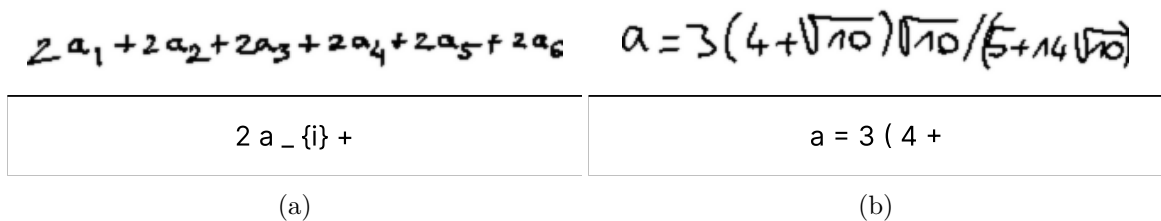
Obrázek 6.2: Vývoj přesnosti klasifikace symbolů, tedy uzlů, a hran výstupního grafu v trénovací fázi při použití dodatečné supervize.



Obrázek 6.3: Srovnání průběhu změny chybové funkce při trénování modelu za použití dodatečných výstupů sítě.

Model	Algoritmus	ExpRate	ExpRate ≤ 1	ExpRate ≤ 2	Struktura
Základní model 3, 256, 400	Greedy	13.34	18.04	24.68	34.18
	Beam 3	10.34	13.57	16.78	19.94
	Beam 5	7.52	11.25	13.75	17.38
Méně vrstev 2, 256, 400	Greedy	9.24	12.69	25.31	31.93
	Beam 3	4.64	6.83	20.41	29.03
	Beam 5	4.64	6.83	16.34	26.12

Tabulka 6.3: Výsledky experimentu porovnávajícího úspěšnost generování výstupního grafu při použití hladového algoritmu Greedy search a paprskového prohledávání Beam search s různými šířkami paprsku na dvou natrénovaných modelech. Číslo uvedené v tabulce vedle označení Beam označuje právě šířku paprsku. Všechny ExpRate hodnoty jsou uvedeny v procentech (%). Sloupec struktura označuje procentuální zastoupení výrazů se správnou strukturou generovaného SLT grafu.



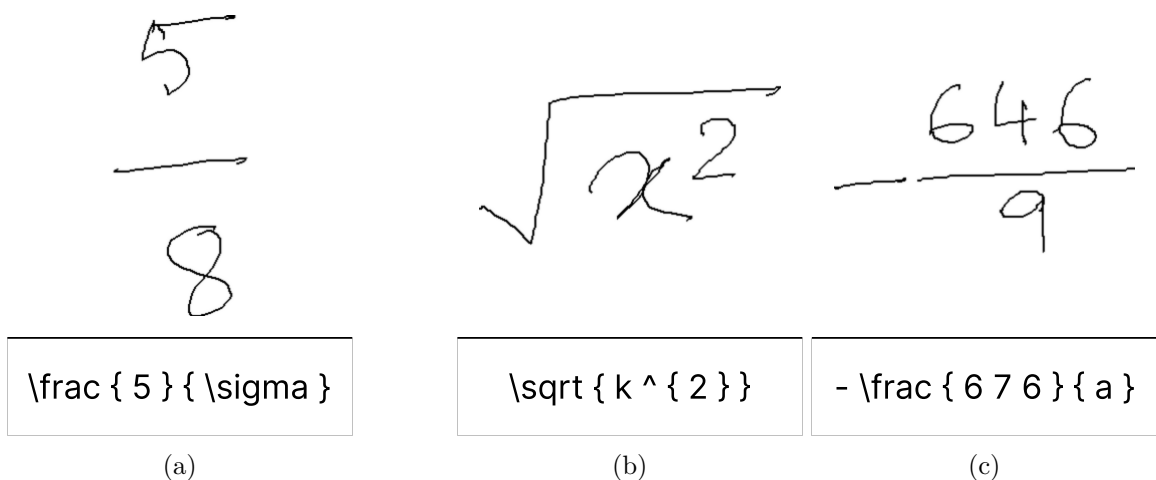
Obrázek 6.4: Ukázka chybného rozpoznání výrazu. Demonstrace problému brzkého ukončení generování u delších matematických výrazů. V horní části zobrazen výřez vstupního obrázku obsahujícího formuli, v dolní části predikovaná sekvence jazyka LaTeX.

míry rozpoznání výrazů ExpRate a tento pokles je výraznější u použití paprsku šířky 5, než paprsku šířky 3. Z další analýzy tohoto experimentu bylo zjištěno, že při použití Beam search algoritmu je v řadě případů přesnější, či přesný strom odpovídající zpracovávanému výrazu prozkoumán, ale je v poslední fázi výběru nejpravděpodobnějšího řešení, respektive řešení, kterým si je model nejjistější zastíněn jiným stromem výrazu.

6.4 Kvalitativní vyhodnocení

V rámci kvalitativního vyhodnocení úspěšnosti řešení byly ručně kontrolovány predikované výrazy modelu a bylo identifikováno několik problémů řešení.

Nejčastěji se projevujícím problémem je brzké ukončení generování výstupního stromu, zejména u delších výrazů. Tato skutečnost byla potvrzena také nástrojem pro vyhodnocování výsledků rozpoznávání poskytovaným společně s datovou sadou CROHME 2019. Nástroj tedy často predikuje kratší výstupní sekvence, což může být důsledkem například také relativně nízkého rozlišení vstupních obrázků. Jelikož jsou vstupní obrázky v rozlišení 310×310 , tak jsou znaky delších sekvencí výrazně menší než u kratších výrazů a po extrakci komponent mohou být ve výrazu těžko rozeznatelné. Ze stejného důvodu může také dojít k více chybám při výpočtu zastínění úhlů zorného pole při sestavování vstupního Line of Sight grafu a pro komponenty tak může být sestaven do jisté míry nepřesný graf viditelnosti. Ukázka takto chybně rozpoznávaných výrazů je na obrázcích 6.4.



Obrázek 6.5: Ukázka chybného rozpoznání výrazu. Demonstrace chybně klasifikovaných podobných symbolů. V horní části zobrazen vstupní obrázek výrazu, v dolní části predikovaná sekvence jazyka LaTeX.

Další frekventovaně se vyskytující chybou je samotné chybné rozpoznání symbolů, které si jsou podobné. Pro minimalizaci tohoto problému byla pro načítání trénovací datové sady přidána náhodná augmentace obrázků jednotlivých komponent a také supervize nad částí enkodér, která by měla pomoci vést učení modelu extrahovat vizuální příznaky. Obě tyto rozšíření přináší zvýšení úspěšnosti rozpoznání výrazů, jak bylo pro supervizi nad konvoluční sítí části enkodér zjištěno v rámci experimentu popsaného výše. Ukázky chybně rozpoznávaných podobných symbolů jsou na obrázcích 6.5.

Při ruční kontrole rozpoznávání bylo také zjištěno, že problematickým místem je klasifikace prvního uzlu generovaného výstupního grafu. Při vytvoření tohoto uzlu jsou jeho hodnoty inicializovány nulovým vektorem a neexistují žádné sousední uzly grafu, které by mohly ovlivnit attention koeficienty do výstupního grafu. Často je tak tento první uzel chybně klasifikován jako jeden z nejfrekventovanějších symbolů v datové sadě, například zlomek $\frac{}{}$.

6.5 Shrnutí výsledků

V provedených experimentech bylo ukázáno, že dodatečná supervize nad konvoluční sítí enkodéru modelu vede ke zlepšení výsledků rozpoznání. Supervize nad grafovou sítí enkodéru naopak v provedených experimentech vedlo ke zhoršení. Dále bylo zjištěno, že použití Beam search prohledávacího algoritmu při generování výstupního grafu vede opět ke snížení úspěšnosti rozpoznávací schopnosti modelu. Byly také testovány tři varianty dekodéru neuronové sítě a z experimentů vyplývá, že vyšších výsledků dosahuje větší neuronová síť z hlediska počtu vrstev grafové sítě i velikosti příznakových vektorů. Následně byly diskutovány některé z problémů navrženého modelu při rozpoznávání matematických výrazů, konkrétně chybná klasifikace symbolů, předčasné ukončení generování výstupních stromů a problematická klasifikace kořenového uzlu výstupního grafu výrazu.

Model	ExpRate	ExpRate <1	ExpRate <2	Struktura
USTC-iFLYTEK	77.15	86.82	88.99	89.49
PAL	71.23	80.31	82.65	83.82
TUAT	24.10	35.53	43.12	43.70
Zde navržený model	13.34	18.04	24.68	34.18

Tabulka 6.4: Porovnání úspěšnosti zde navrženého modelu s některými přístupy, které se účastnili soutěže CROHME 2019. Všechny výsledky byly získány na testovací sadě datového souboru CROHME 2019. Všechny uvedené hodnoty jsou v procentech (%). Sloupec struktura označuje procentuální zastoupení výrazů se správnou strukturou generovaného SLT grafu. Zdroj [21].

Srovnání s existujícími modely Soutěže CROHME 2019 [21] se zúčastnila řada přístupů, které dosahují výrazně lepších výsledků, než zde navržený model. Přehled úspěšností některých z těchto modelů v porovnání se zde navrženým je obsažen v tabulce 6.4. Modely *USTC-iFLYTEK* a *PAL* dosáhly celkově nejlepších výsledků. V případě *USTC-iFLYTEK* se jedná o enkodér-dekodér architekturu používající konvoluční síť typu *DenseNet* pro zakódování do vnitřní reprezentace a rekurentní dekodér. *PAL* používá tzv. Adversariální učení pro natrénování enkodér-dekodér s konvolučním attention mechanismem. Model označovaný *TUAT* dosahuje nejhorší úspěšnosti ze zúčastněných řešení. K řešení používá enkodér-dekodér architekturu obsahující konvoluční síť a dvousměrnou LSTM vrstvu v části enkodér a rekurentní LSTM dekodér s attention mechanismem.

V porovnání se všemi výše uvedenými řešeními dosahuje zde navržený přístup horších výsledků. Vliv na tento rozdíl může mít například i vyšší použité rozlišení vstupních obrázků, které je u modelu *USTC-iFLYTEK* a *PAL* 1000 × 1000 pixelů. U všech uvedených řešení byla také prováděna rozsáhlá úprava vstupních dat a rozšíření trénovací daty. U řešení *PAL* autoři uvádí výslednou trénovací datovou sadu obsahující 310 000 matematických výrazů.

Kapitola 7

Závěr

Cílem této práce je vytvoření nástroje pro automatické rozpoznávání ručně psaných matematických výrazů z jejich obrázků. Na základě nastudované literatury byla zvolena architektura typu kodér-dekodér využívající grafové neuronové sítě k rozpoznávání ručně psaných matematických výrazů z online dat, tedy sekvence tahů elektronického pera, jako základ pro následně navržené řešení pro rozpoznávání matematických výrazů z jejich obrázků. Toto řešení používá grafovou konvoluční neuronovou síť jako enkodér pro vytvoření vnitřní reprezentace matematického výrazu z grafu, sestaveného nad segmentovanými komponentami vstupního obrázku. Struktura tohoto grafu odráží vzájemnou viditelnost extrahovaných komponent z obrázku. Následně pomocí dekodéru využívajícího grafovou neuronovou síť a attention mechanismu generuje výstupní reprezentaci výrazu v podobě Symbol Layout Tree grafu, který je následně transformován do zápisu v jazyce LaTeX.

Navržený model byl následně implementován. K jeho trénování a vyhodnocení byla použita datová sada CROHME 2019, publikovaná v rámci stejnojmenné soutěže v rozpoznávání matematických výrazů.

Během experimentů s neuronovou sítí byly porovnány tři varianty s rozdílným počtem vrstev grafové neuronové sítě v části dekodér a různou velikostí příznakových vektorů. Nejvyšší úspěšnosti rozpoznání matematických výrazů bylo dosaženo s největší testovanou variantou sítě, se třemi vrstvami grafové sítě. Poté byl prozkoumán vliv dodatečné supervize nad výstupy enkodéru sítě, konkrétně nad výstupy konvoluční neuronové sítě použité pro extrakci vizuálních příznaků z obrázků vstupních komponent a grafové sítě použité pro vytvoření vnitřní reprezentace. Supervizí nad výstupy konvoluční sítě bylo dosaženo vyšší úspěšnosti rozpoznávání výrazů. U výstupů grafové sítě naopak došlo ke snížení. V posledním experimentu byl vyhodnocen přínos použití paprskového prohledávání, tedy algoritmu Beam search při generování výstupní reprezentace výrazu v podobě Symbol Layout Tree grafu. Zavedení tohoto algoritmu ovšem vedlo ke snížení úrovně přesného rozpoznání výrazů o 3%, tedy ExpRate.

Výsledky navrženého řešení byly následně porovnány s existujícími metodami pro rozpoznávání ručně psaných matematických výrazů z obrázků. V porovnání s těmito řešeními dosahuje navržené řešení o 11% přesného rozpoznání výrazu horší úspěšnosti. Úspěšnost zde navrženého řešení je 13.34% ExpRate, tedy přesného rozpoznání výrazu.

V rámci další práce by mohla být architektura doplněna o oddělený modul pro klasifikaci symbolů vstupního výrazu. Stávající část modelu by poté generovala výstupní graf se substituovanými symboly, které by do výsledku následně byly injektovány.

Literatura

- [1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Dostupné z: <https://www.tensorflow.org/>.
- [2] Aggarwal, R., Pandey, S., Tiwari, A. K. a Harit, G. Survey of Mathematical Expression Recognition for Printed and Handwritten Documents. *IETE Technical Review*. Taylor & Francis. 2021, s. 1–9. DOI: 10.1080/02564602.2021.2008277.
- [3] Anderson, R. H. Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics. In: New York, NY, USA: Association for Computing Machinery, 1967, s. 436–459. ISBN 9781450373098.
- [4] Cho, K., Merriënboer, B., Bahdanau, D. a Bengio, Y. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. Zář 2014. DOI: 10.3115/v1/W14-4012.
- [5] Cho, K., Merriënboer, B. van, Gulcehre, C., Bahdanau, D., Bougares, F. et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, ř 2014, s. 1724–1734. DOI: 10.3115/v1/D14-1179.
- [6] Deng, Y., Kanervisto, A., Ling, J. a Rush, A. M. Image-to-Markup Generation with Coarse-to-Fine Attention. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. JMLR.org, 2017, s. 980–989. ICML'17.
- [7] Deng, Y., Kanervisto, A. a Rush, A. M. What You Get Is What You See: A Visual Markup Decompiler. *ArXiv*. 2016, abs/1609.04938.
- [8] Ding, H., Chen, K. a Huo, Q. An Encoder-Decoder Approach to Handwritten Mathematical Expression Recognition with Multi-head Attention and Stacked Decoder. In: Lladós, J., Lopresti, D. a Uchida, S., ed. *Document Analysis and Recognition ICDAR 2021*. Cham: Springer International Publishing, 2021, s. 602–616. ISBN 978-3-030-86331-9.
- [9] Fey, M. a Lenssen, J. E. Fast Graph Representation Learning with PyTorch Geometric. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.
- [10] Gehring, J., Auli, M., Grangier, D., Yarats, D. a Dauphin, Y. N. Convolutional Sequence to Sequence Learning. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. JMLR.org, 2017, s. 1243–1252. ICML'17.

- [11] Gori, M., Monfardini, G. a Scarselli, F. A new model for learning in graph domains. In: *IEEE International Joint Conference on Neural Networks*. 2005, sv. 2, s. 729–734. DOI: 10.1109/IJCNN.2005.1555942.
- [12] He, K., Zhang, X., Ren, S. a Sun, J. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, s. 770–778. DOI: 10.1109/CVPR.2016.90.
- [13] Hochreiter, S. a Schmidhuber, J. Long Short-term Memory. *Neural computation*. Prosinec 1997, sv. 9, s. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.
- [14] Hu, L. a Zanibbi, R. Line-of-Sight Stroke Graphs and Parzen Shape Context Features for Handwritten Math Formula Representation and Symbol Segmentation. In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, s. 180–186. DOI: 10.1109/ICFHR.2016.0044.
- [15] Huang, G., Liu, Z., Maaten, L. van der a Weinberger, K. Densely Connected Convolutional Networks. In: *červenec 2017*, s. 2261–2269. DOI: 10.1109/CVPR.2017.243.
- [16] Kipf, T. N. a Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In: *International Conference on Learning Representations (ICLR)*. 2017.
- [17] Li, Y., Tarlow, D., Brockschmidt, M. a Zemel, R. Gated graph sequence neural networks. *International Conference on Learning Representations (ICLR)*. 2016.
- [18] Mahdavi, M., Condon, M., Davila, K. a Zanibbi, R. LPGA: Line-of-Sight Parsing with Graph-Based Attention for Math Formula Recognition. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019, s. 647–654. DOI: 10.1109/ICDAR.2019.00109.
- [19] Mouchère, H., Viard Gaudin, C., Zanibbi, R. a Garain, U. ICFHR 2014 Competition on Recognition of On-Line Handwritten Mathematical Expressions (CROHME 2014). In: *2014 14th International Conference on Frontiers in Handwriting Recognition*. 2014, s. 791–796. DOI: 10.1109/ICFHR.2014.138.
- [20] Mouchère, H., Viard Gaudin, C., Zanibbi, R. a Garain, U. ICFHR2016 CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions. In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, s. 607–612. DOI: 10.1109/ICFHR.2016.0116.
- [21] Mouchère, H., Viard Gaudin, C., Zanibbi, R. a Garain, U. ICDAR 2019 CROHME + TFD: Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection. In: *15th IAPR International Conference on Document Analysis and Recognition (ICDAR 2019)*. 2019, s. 1533–1538.
- [22] Pang, N., Yang, C., Zhu, X., Li, J. a Yin, X.-C. Global Context-Based Network with Transformer for Image2latex. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, s. 4650–4656. DOI: 10.1109/ICPR48806.2021.9412072.
- [23] Papineni, K., Roukos, S., Ward, T. a Zhu, W. J. BLEU: a Method for Automatic Evaluation of Machine Translation. *Řijen 2002*, s. 311–318. Association for Computational Linguistics. DOI: 10.3115/1073083.1073135.

- [24] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alché Buc, F. d', Fox, E. et al., ed. *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, s. 8024–8035.
- [25] Pearson. *Aida Calculus Math Handwriting Recognition Dataset* [<https://www.kaggle.com/aidapearson/ocr-data>]. Srpen 2020. [Online; navštíveno 26.1.2022].
- [26] Peng, S., Gao, L., Yuan, K. a Tang, Z. Image to LaTeX with Graph Neural Network for Mathematical Formula Recognition. In: Lladós, J., Lopresti, D. a Uchida, S., ed. *Document Analysis and Recognition ICDAR 2021*. Cham: Springer International Publishing, 2021, s. 648–663. ISBN 978-3-030-86331-9.
- [27] Sakshi a Kukreja, V. A retrospective study on handwritten mathematical symbols and expressions: Classification and recognition. *Engineering Applications of Artificial Intelligence*. 2021, sv. 103. ISSN 0952-1976.
- [28] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. a Monfardini, G. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*. 2009, sv. 20, č. 1, s. 61–80. DOI: 10.1109/TNN.2008.2005605.
- [29] Simonyan, K. a Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: Bengio, Y. a LeCun, Y., ed. *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*. 2015.
- [30] Singh, S. S. Teaching Machines to Code: Neural Markup Generation with Visual Attention. *ArXiv*. 2018, abs/1802.05415.
- [31] Sutskever, I., Vinyals, O. a Le, Q. V. Sequence to Sequence Learning with Neural Networks. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. Cambridge, MA, USA: MIT Press, 2014, s. 3104–3112. NIPS'14.
- [32] Tu, Z., Lu, Z., Liu, Y., Liu, X. a Li, H. Modeling Coverage for Neural Machine Translation. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany: [b.n.], Srpen 2016, s. 76–85. DOI: 10.18653/v1/P16-1008.
- [33] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is All You Need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: [b.n.], 2017, s. 6000–6010. NIPS'17. ISBN 9781510860964.
- [34] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. et al. Graph Attention Networks. *International Conference on Learning Representations*. 2018.
- [35] Ward, I. R., Joyner, J., Lickfold, C., Rowe, S., Guo, Y. et al. A Practical Guide to Graph Neural Networks. *ArXiv*. 2020, arXiv:2010.05234.

- [36] Wu, J.-W., Yin, F., Zhang, Y.-M., Zhang, X.-Y. a Liu, C.-L. Graph-to-Graph: Towards Accurate and Interpretable Online Handwritten Mathematical Expression Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*. May 2021, sv. 35, č. 4, s. 2925–2933.
- [37] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. et al. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*. 2021, sv. 32, č. 1, s. 4–24. DOI: 10.1109/TNNLS.2020.2978386.
- [38] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In: Bach, F. a Blei, D., ed. *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: PMLR, 2015, sv. 37, s. 2048–2057. Proceedings of Machine Learning Research.
- [39] Xu, K., Hu, W., Leskovec, J., a Jegelka, S. How powerful are graph neural networks? *ArXiv e-prints*. 2019.
- [40] Yao, T., Pan, Y., Li, Y. a Mei, T. Exploring Visual Relationship for Image Captioning. In: Ferrari, V., Hebert, M., Sminchisescu, C. a Weiss, Y., ed. *Computer Vision – ECCV 2018*. Cham: Springer International Publishing, 2018, s. 711–727. ISBN 978-3-030-01264-9.
- [41] Zanibbi, R., Blostein, D., Zanibbi, R. a Blostein, D. Recognition and Retrieval of Mathematical Expressions. *International Journal on Document Analysis and Recognition (IJDAR)*. Prosinec 2011, sv. 15. DOI: 10.1007/s10032-011-0174-4.
- [42] Zhang, J., Du, J. a Dai, L. A GRU-Based Encoder-Decoder Approach with Attention for Online Handwritten Mathematical Expression Recognition. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. 2017, sv. 01, s. 902–907.
- [43] Zhang, J., Du, J. a Dai, L. Multi-Scale Attention with Dense Encoder for Handwritten Mathematical Expression Recognition. In: Srpen 2018, s. 2245–2250. DOI: 10.1109/ICPR.2018.8546031.
- [44] Zhang, J., Du, J., Yang, Y., Song, Y.-Z., Wei, S. et al. A Tree-Structured Decoder for Image-to-Markup Generation. In: III, H. D. a Singh, A., ed. *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 13–18 Jul 2020, sv. 119, s. 11076–11085. Proceedings of Machine Learning Research.
- [45] Zhang, J., Du, J., Zhang, S., Liu, D., Hu, Y. et al. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*. 2017, sv. 71, s. 196–206. ISSN 0031-3203.
- [46] Zhang, W., Bai, Z. a Zhu, Y. An Improved Approach Based on CNN-RNNs for Mathematical Expression Recognition. In: *Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing*. 2019, s. 57–61. ICMSSP 2019. ISBN 9781450371711.
- [47] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C. et al. Graph neural networks: A review of methods and applications. *AI Open*. 2020, sv. 1, s. 57–81. ISSN 2666-6510.

Příloha A

Obsah DVD

Obsah DVD, které je součástí této práce:

- **dataset/** - adresář s datovou sadou a souvisejícími soubory
 - **data/** - obsahuje komprimovanou datovou sadu CROHME 2019 [21]
 - **tools/** - obsahuje oficiální vyhodnocovací nástroje soutěže CROHME 2019 [21]
- **source_codes/** - adresář se zdrojovými kódy
 - **configuration/** - vzorový konfigurační soubor aplikace
 - **src/** zdrojové kódy aplikace
- **text_source/** - adresář obsahující soubory potřebné k sestavení tohoto dokumentu
- **video/** - adresář s videem stručně shrnujícím tuto práci