KNN: Rozpoznání matematických rovnic

Jakub Málek
Tomáš Milostný
Marek Večeřa

18. května 2025

{xmalek17,xmilos02,xvecer31}@vutbr.fit.cz

Akademický rok 2024/2025

Obsah

1 Úvod

Tento dokument popisuje problematiku a řešení rozpoznání matematických rovnic. Jedná se o velmi aktuální téma, které se v posledních letech stalo předmětem mnoha výzkumů a aplikací. Rozpoznávání matematických vzorců je důležité pro automatizaci různých úloh, jako je převod tištěných nebo ručně psaných vzorců do digitálního formátu, což usnadňuje jejich zpracování a analýzu.

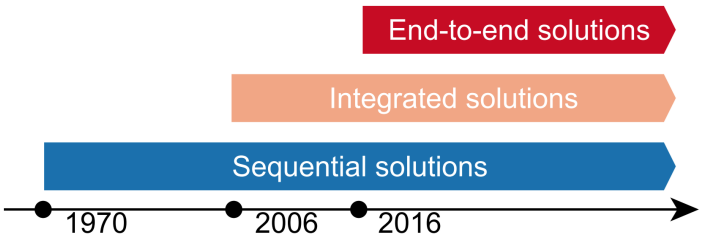
Zdrojový kód je otevřeně dostupný na https://github.com/malekjakub69/knn-math.

1.1 Popis úlohy

Úkolem projektu je vytvoření a natrénování modelu pomocí strojového učení, který bude schopný rozpoznat obrázky matematických výrazů na vstupu a převést je do strojového zápisu. Formát vstupních obrázků jsme si zvolili jako obrázky ručně psaných rovnic, které budou následně převedeny do formátu LaTeX– **offline recognition**.

2 Přehled existujících řešení

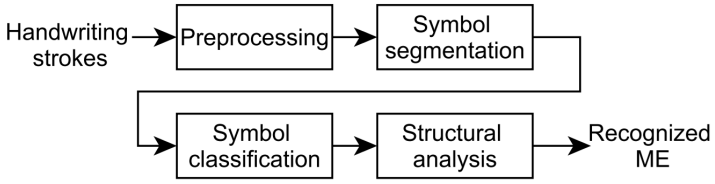
V celé historii problematiky rozpoznávání ručně psaných matematických rovnic se objevily tři odlišné přístupy 1. Na počátku se jednalo o jednodušší více přímočaré metody označované jako sekvenční řešení [1].



Obrázek 1: Časový vývoj různých metod pro rozpoznávání matematických vzorců

2.1 Sekvenční metody

Používají princip dekompozice a tuto metodu lze rozložit do několika částí (obrázek 5). Modely pracující na tomto principu nejprve (1) rozpoznají jednotlivé symboly a následně provedou (2) strukturní analýzu, kde jednotlivé symboly poskládají do celého výrazu. Rozpoznávání symbolů lze pak dále rozložit na segmentaci jednotlivých symbolů a jejich klasifikaci. Tato metoda o několika krocích má ale nevýhodu v akumulaci chyb během výpočtu jednotlivých kroků.



Obrázek 2: Diagram postupu sekvenčního řešení

2.2 Integrované metody

Generují množinu symbolických hypotéz a následně pomocí strukturní analýzy vyberou tu nejlepší na základě znalosti odpovídající gramatiky a sémantiky. Tento přístup je složitější a vyžaduje více výpočetních prostředků, ale může poskytnout lepší výsledky v případě složitějších vzorců.

2.3 End-to-end metody

Zatímco předchozí dvě metody řešení problematiky rozpoznávání matematických rovnic byly v posledních dekádách studovány důkladně, **end-to-end řešení** na principu **encoder–decoder** se začaly objevovat relativně nedávno. Tento přístup se snaží vyřešit problém rozpoznávání vzorců jako celek, bez rozdělení na jednotlivé části. Používá hluboké učení a neuronové sítě bez jakékoli znalosti konkrétní domény pro tvorbu modelu – **data-driven** přístup. Enkodér převede vstup na latentní reprezentaci a dekodér extrahuje skrytou reprezentaci za pomoci attention mechanismu a generuje výstupní sekvenci.

Jeden z prvních návrhů konkrétní architektury se skládal z **VGG** sítě, která sloužila jako enkodér [2]. Model dále využíval **GRU** jako dekodér. Takto navržený systém disponuje přesností okolo 45 % na datasetech **CROHME** (Competition on Recognition of Handwritten Mathematical Expressions (HME)), které jsou standardem evaluace modelů pro rozpoznávání matematických vzorců.

Evaluační metrika **expression recognition rate/accuracy** – vyhodnocuje správnost modelu jak z pohledu jednotlivých symbolů, tak i celkové struktury. Jako správný se výstup považuje v případě, že celkový počet chyb v jednom výstupu není více než 3.

Vylepšení původního návrhu architektury dále zahrnovala nahrazení VGG sítě a použití DenseNet (densely connected conv. network) jako enkodéru [3]. Ve článku [4] představují end-to-end řešení: kaskáda CNN - feature extractor, BLSTM - encoder a LSTM s attention mechanismem jako dekodér.

V posledním ročníku soutěže CROHME 2023 se objevilo několik návrhů. Vítězný tým představil model skládající se z vrstev CNN a BLSTM pro enkodér a dekodér v podobě attention mechanismu (expression recognition rate 86.95 %).

Další účastníci se řešení používá DenseNet jako enkodér a 2 různé řešení dekodéru: bidi-

rectional tree decoder a klasický dekodér. S tímto přístupem vznikají 2 modely, jejichž spojením vzniká hybridní model (expression recognition rate 86.95 %).

Poslední model představený v soutěži, který uvedeme v této kapitole, je složen ze 2 částí: obrazový model – CRNN architektura trénovaná na CTC loss funkci, a jazykový model – LSTM architektura trénovaná pro predikci dalšího LaTeX tokenu (expression recognition rate 75.55 %).

Týmy mají pro tvorbu řešení k dispozici online i offline datasety, které důmyslně používají spolu s augmentací a generováním offline dat z online datasetu pro zlepšení výsledků.

[5]

3 Popis vlastního řešení

Na základě provedené rešerše existujících řešení jsme sestavili vlastní NN model založený na **Vision Transformer** (ViT) enkodér-dekodér architektuře (poprvé představena v roce 2019) spolu s VGG sítí na vstupu pro prvotní extrakci rysů z obrázku.

3.1 Vstupní data

Použili jsme oficiální dataset soutěže ICDAR CROHME 2023 s celkem 30 000 obrázky ručně psaných matematických rovnic (HME) spolu s přiloženými LaTeX sekvencemi, které jsme přímo použili jako ground truth.

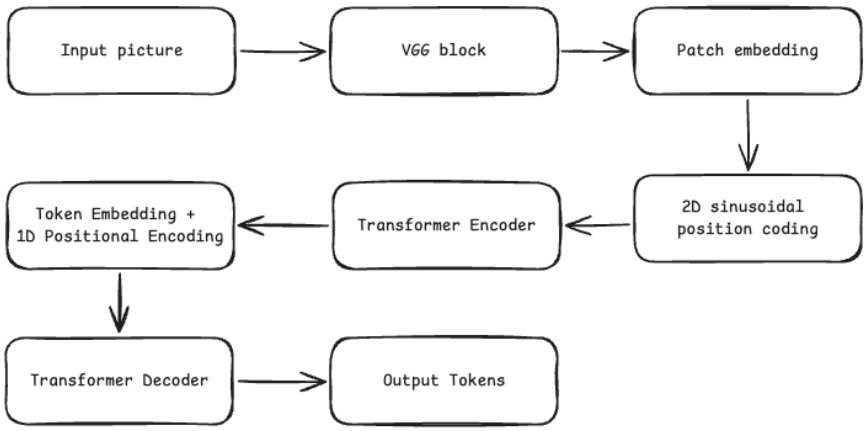
Dále jsme zkusili trénovací sadu expandovat pomocí generování obrázků HME z offline dat, které jsou k dispozici v mnohem větším množství. K tomuto účelu jsme využili největší dostupný dataset HME – MathWriting, který obsahuje 230 000 ručně psaných a 400 000 syntetických offline záznamů ve formátu InkML, který ke každému záznamu rovněž obsahoval LaTeX ground truth. Pomocí souřadnic jednotlivých tahů offline záznamů jsme byli schopni vykreslit obrazovou reprezentaci a takto celkovou trénovací sadu mnohonásobně rozšířit.

3.1.1 Augmentace

Pro zlepšení robustnosti modelu jsme do procesu trénování přidali řadu transformací vstupního obrázku. Konkrétně se aplikují mírné změny výšky a šířky obrázku, změna barev (color jitter), rozmazání dle Gaussova rozdělení, rotace, posuny a přidání náhodného šumu.

3.2 Model

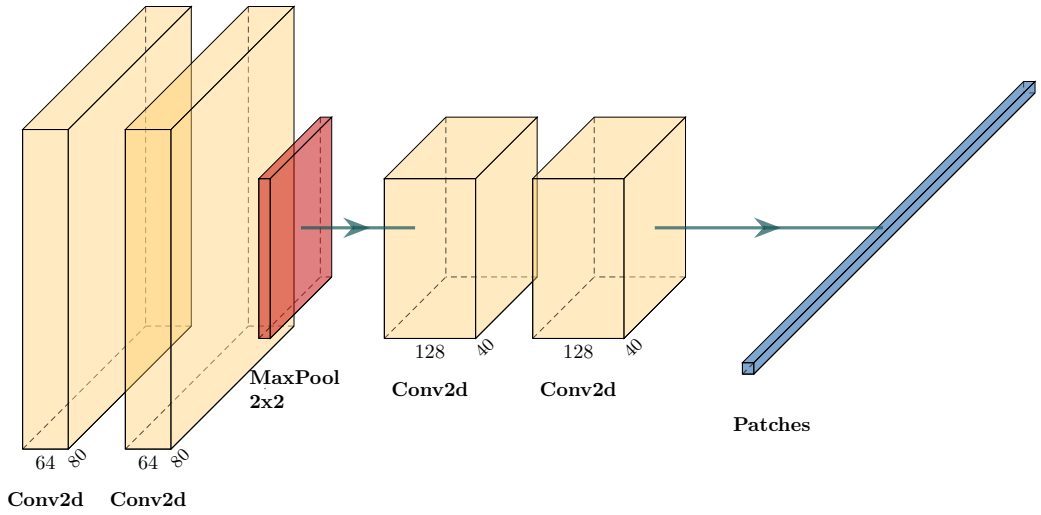
Původní baseline řešení představené v rámci checkpointu a videa obsahovalo řadu problémů, které jsme se pokusili vyřešit konzultací a nalezením dalších dříve zmíněných existujících řešení. Aktuální verze modelu LaTeX OCR převádí vstupní obrázek na sekvenci tokenů reprezentující LaTeX kód s využitím hybridní architektury založené na transformerovém kodéru a dekodéru s přidanými konvolučními vrstvami VGG, patch embeddingem a pozičním kódováním.



Obrázek 3: Architektura modelu LaTeX OCR

3.2.1 Komponenty arhitektury

- **Vstupní obrázek** - pro zachování správného rozměru se vstupní obrázek nejprve pomocí interpolace změní na pevnou výšku 80 pixelů, přičemž je zachován původní poměr stran, a následně se doplňuje padding tak, aby šířka byla dělitelná velikostí patche.
- **VGG blok** - slouží jako první stupeň extrakce vizuálních příznaků ze vstupního obrázku. Inspirovaný architekturou VGG sítě, používá sekvenci konvolučních vrstev následovaných batch normalizací a ReLU aktivací. Tento blok redukuje prostorové rozměry na polovinu (tedy výšku 40 pixelů) pomocí MaxPoolingu a současně zvyšuje počet kanálů na 64, čímž vytváří bohatší reprezentaci vizuálních příznaků před jejich dalším zpracováním patch embedingem (obrázek 4). Zachování prostorových informací je klíčové pro správné rozpoznání matematických symbolů a jejich vzájemných vztahů.
- **Patch Embedding** - klíčová součást architektury, která převádí obrazové příznaky na sekvenci vektorů (patches (oblasti) o velikosti 8x8 pixelů, které jsou transformovány pomocí konvolučních vrstev na vektory s dimenzí 128) pro zpracování transformerem. Tento proces efektivně převádí 2D obrazovou reprezentaci na sekvenci embedingů, kde každý embedding reprezentuje určitou oblast původního obrázku, což umožňuje transformeru zpracovávat obrazové informace jako sekvenci, podobně jako zpracovává textové tokeny.



Obrázek 4: Diagram architektury VGG bloku

- **Poziční kódování** - původní model popsaný v checkpointu obsahoval učící se poziční parametry inicializovaného vektorem s hodnotami 0. Tento přístup byl však neefektivní pro vstupy s různou šířkou. Nyní používáme **Sinusoidové poziční kódování** [6] pro zachycení prostorových vztahů jak v obrazových datech (2D), tak v sekvencích LaTeXtokenů (1D). Toto kódování je klíčové pro transformer architekturu, protože umožňuje modelu rozlišovat pozice jednotlivých prvků bez ohledu na jejich vzdálenost. Pro obrazové příznaky se používá 2D varianta, která kombinuje výškovou a šířkovou složku pomocí funkcí sinus a cosinus s různými frekvencemi. Tento přístup vytváří unikátní poziční signál pro každou pozici v mřížce příznaků, přičemž relativní pozice jsou zachyceny pomocí lineárních kombinací těchto trigonometrických funkcí. Pro dekódování LaTeXsekvence se používá standardní 1D poziční kódování, které následuje stejný princip, ale pouze v jedné dimenzi.

- **Transformer encoder** - zpracovává sekvenci patch embedingů vytvořených z obrázku. V naší implementaci používáme nn.TransformerEncoder z knihovy PyTorch s několika vrstvami (*num_transformer_layers=4*), kde každá vrstva obsahuje multi-head self-attention mechanismus (s *nhead=8* hlavami) následovaný feed-forward sítí. Encoder transformuje prostorově uspořádané příznaky z patch embedingu na kontextově bohatou reprezentaci, kde každý vektor obsahuje informace o globálních vztazích v obraze. Tato reprezentace je klíčová pro následné dekódování LaTeX sekvence, protože zachycuje jak lokální detaily matematických symbolů, tak jejich vzájemné prostorové vztahy.

- **Transformer decoder** - převádí zakódované vizuální příznaky na sekvenci LaTeX tokenů. V původní verzi popsané v checkpointu, jsme využívali **LSTM** s pozorností (attention mechanism) s vrstvou normalizace (LayerNorm). Toto řešení jsme nakonec opustili a jako vhodný dekodér pro transformer kodér jsme zvolili standardní nn.TransformerDecoder s maskovým self-attention mechanismem, který zajišťuje, že při generování každého tokenu může model vidět pouze předchozí tokeny. Dekodér kombinuje attention na zakódované vizuální příznaky (cross-attention) s maskovaným self-attention na již vygenerované tokeny. Toto umožňuje modelu generovat

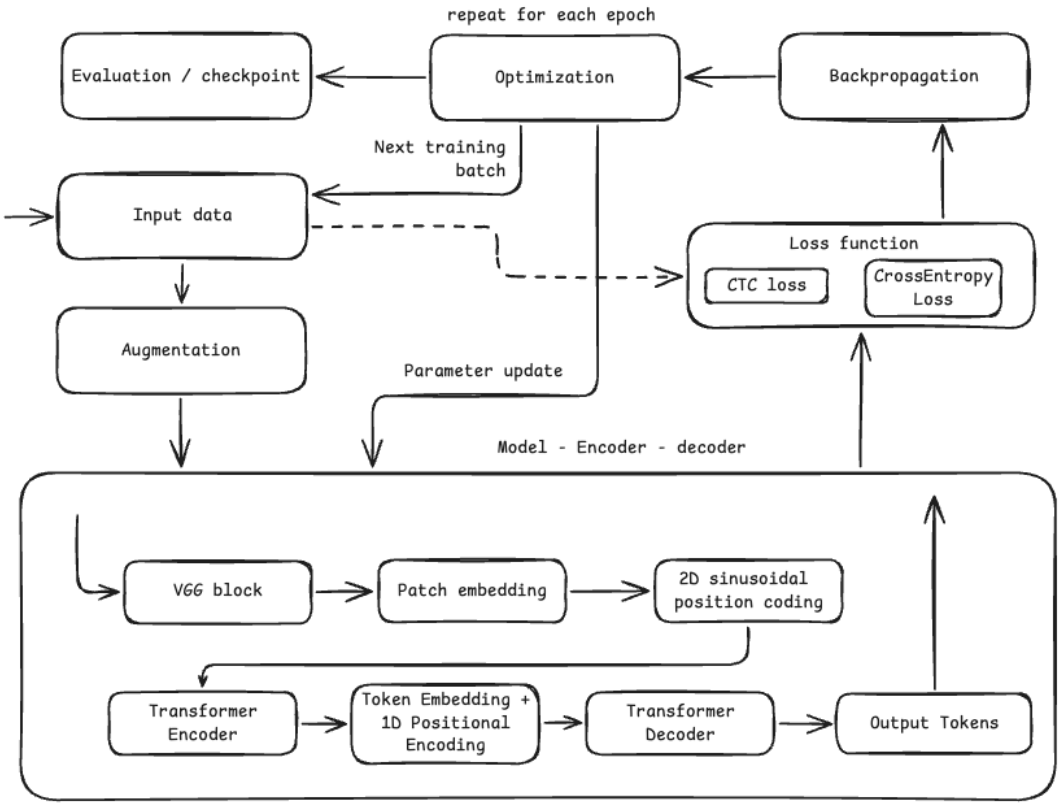
LaTeXvýstup token po tokenu, přičemž každý nový token je generován s ohledem na vizuální kontext i dosud vygenerovaný text.

Model nabízí dva přístupy ke generování výsledné LaTeXsekvence. Základní **greedy decoding** vždy vybírá token s nejvyšší pravděpodobností, což je rychlé, ale může vést k suboptimálním výsledkům. Sofistikovanější **beam search** (výchozí s *beam_size=3*) udržuje několik nejpravděpodobnějších sekvencí současně, což poskytuje lepší výsledky za cenu vyšší výpočetní náročnosti.

Každá sekvence začíná speciálním **<START>** tokenem (1) a končí **<END>** tokenem (2), čímž model ví, kdy začít a ukončit generování. Pro běžné použití je doporučen beam search, který poskytuje lepší kvalitu rozpoznávání matematických výrazů díky uvažování více možných cest při generování sekvence.

3.3 Proces trénování

Trénování začíná inicializací modelu na zvoleném zařízení (CPU/CUDA/MPS) a nastavením optimalizačních komponent. Model využívá AdamW optimizér s váhovým rozpadem pro regularizaci a OneCycleLR scheduler pro adaptivní učení. Pro efektivní využití GPU paměti je implementován mixed precision training pomocí automatického přepínání mezi float16 a float32 přesností.



Obrázek 5: Trénovací smyčka modelu

3.3.1 Kombinovaná loss funkce

Model popsaný v rámci checkpointu používal pouze **Cross Entropy** loss s label smoothingem (0.15), která optimalizuje přesnost na úrovni jednotlivých tokenů a pomáhá s generalizací. Aktuální model jsme se však pokusili rozšířit o využití sofistikované kombinace dvou loss funkcí, každá s váhou 0.5. **Connectionist Temporal Classification** (CTC) loss řeší problém zarovnání sekvencí mezi vstupním obrázkem a výstupním LaTeX kódem, umožňující modelu naučit se mapování bez explicitního zarovnání. Tato hybridní loss funkce umožňuje modelu současně se učit globální strukturu matematického výrazu i přesnou tokenizaci.

3.3.2 Optimalizační strategie

Trénink využívá gradient accumulation (2 kroky) pro simulaci větších batch size a stabilnější trénink. Learning rate je řízen pomocí OneCycleLR scheduleru, který implementuje warmup fázi (30% epochy) následovanou cosinovou annealing strategií. Pro prevenci explodujících gradientů je použit gradient clipping s maximální normou 2.0. Mixed precision training s GradScalerem efektivně využívá GPU paměť a zrychluje trénink při zachování numerické stability.

3.3.3 Monitoring a checkpointing

Dataset HME obrázků jsme rozdělili na trénovací, validační a testovací sadu v poměru 8:1:1. Na konci každé trénovací epochy jsme použili validační sadu ke zjištění aktuální přesnosti modelu. V průběhu trénování používáme 2 metriky přesnosti.

1. **Token-level accuracy** měří poměr správně predikovaných tokenů na výstupu vůči celkovému počtu tokenů na výstupu. Takto vypočítaná hodnota je sice dobrým přibližným ukazatelem přesnosti modelu, zejména v počátečních fázích trénování, o celkové správnosti výsledku ale příliš dobře nevypovídá, neboť nezahrnuje informaci o celkové struktuře.
2. **Sequence accuracy** udává podíl kompletně správně predikovaných sekvencí vůči celkovému počtu sekvencí v sadě – tato metrika naopak model podhodnocuje, v pozdních fázích trénování se ale může více blížit univerzální metrice **Expression recognition rate**, která se používá pro porovnání HME recognition modelů.

Proces trénování implementuje mechanismus *early stopping* s trpělivostí 10 epoch pro předcházení přetrénování. Checkpointy jsou ukládány ve třech režimech: při dosažení nejlepší accuracy, při dosažení nejlepší loss a periodicky každých 5 epoch. Toto umožňuje flexibilní obnovu tréninku a výběr optimálního modelu podle různých kritérií.

3.4 Experimenty

Vyzkoušeli jsme řadu technik, jak výkon modelu zlepšit, včetně změn architektury, úpravy trénovacího procesu, přidání různých augmentací vstupních trénovacích obrázků. Také jsme rozšířili původní datovou sadu ze soutěže o velké množství obrázků převedených z digitálního formátu InkML.

3.4.1 Změny v architektuře modelu

Původní LSTM architekturu s attention mechanismem jsme nahradili plně transformerovou architekturou. Transformery prokázaly výrazně lepší schopnost paralelizace a efektivnější zpracování dlouhých sekvencí. Zavedení mechanismu multi-head attention (8 hlav) umožnilo modelu paralelně zpracovávat různé aspekty vizuálních rysů rovnic a zlepšilo zachycení dlouhodobých závislostí.

Zásadní změnou bylo nahrazení učitelných pozičních embeddingů sinusoidálním pozičním kódováním. Pro vstupní obraz jsme implementovali 2D sinusoidální pozicování (rozložené na vertikální a horizontální komponenty), zatímco pro dekodér bylo použito standardní 1D sinusoidální kódování. Tato změna vedla ke stabilnějšímu tréninku a lepší schopnosti generalizace.

3.4.2 Optimalizátor a Scheduler

- AdamW s parametry (lr=3e-4, weight_decay=0.01, betas=(0.9, 0.98))
- OneCycleLR scheduler s 30% warmup fází
- Gradient accumulation (2 kroky) pro stabilnější trénink
- Mixed precision training pro efektivní využití GPU

3.4.3 Implementace kombinované loss funkce

- CTC Loss (váha 0.5) pro sequence alignment
- Cross Entropy Loss (váha 0.5) s label smoothing 0.15
- Gradient clipping s max_norm=2.0

3.4.4 Hyperparametry a další varianty

- **Počet vrstev TransformerEncoderu**:
  - `layers=2`: nižší kapacita, přesnost stagnovala pod 2 %.
  - `layers=6`: mírné zlepšení, ale delší trénink bez poklesu validační ztráty.
  - `layers=4`: nejlepší kompromis mezi výkonem a paměťovou náročností.

- **Dropout**:
  - `dropout=0.0`: výrazné přeučení po cca 10 epochách.
  - `dropout=0.3`: stabilnější validační ztráta, ale pomalejší trénink.
  - `dropout=0.1`: vyvážené řešení s nejlepším poměrem stability a výkonu.

3.4.5 Analýza výkonnosti

- Trénovací metriky:
  - Počáteční loss: > 5.5
  - Konvergovaná loss: ≈ 0.8
  - Trénovací přesnost: 98%
- Validační metriky:
  - Validační loss: ≈ 6.4284 (bez zlepšení)
  - Validační přesnost: 3.6%
- Testovací metriky:
  - Test loss: 6.4398
  - Testovací přesnost: 3.66%

3.4.6 Identifikované problémy

Pozorujeme efekt overfittingu - model vykazuje klasické známky přetrénování, kdy má téměř perfektní výsledky při trénování a už se není schopen zlepšovat, ale validační a testovací metriky na neviděných datech se nezlepšují

Možné příčiny:

1. Příliš komplexní model pro danou velikost datasetu
2. Špatná datová sada obsahující malý počet realistických fotografií ručně napsaných rovnic ze soutěže a data konvertovaná z formátu InkML (digitální formát z tabletu)
3. Neoptimální hyperparametry, které je pro daný problém nutné jinak nastavit

Tyto výsledky naznačují, že přestože model dokáže efektivně memorovat trénovací data, má významné problémy s generalizací.

4 Závěr

Tato dokumentace popisuje návrh, implementaci a experimentální vyhodnocení systému KNN-Math pro převod ručně psaných matematických rovnic do formátu LaTeX. Systém je postaven na moderní architektuře typu encoder–decoder využívající konvoluční síť (VGG blok) pro extrakci vizuálních rysů a transformerový model pro sekvenční generování LaTeX výrazů.

V průběhu vývoje jsme otestovali řadu přístupů ke zlepšení výkonnosti modelu. Zásadní změnou byla náhrada LSTM dekodéru za plně transformerový dekodér a přechod na sinusoidální poziční kódování. Dále jsme zavedli kombinovanou ztrátovou funkci (CTC + Cross Entropy), pokročilé trénovací techniky (gradient accumulation, mixed precision training) a rozšířili datovou sadu o synteticky vygenerované vzorky z formátu InkML.

Ačkoliv model dosahoval vysoké přesnosti na trénovacích datech ($\approx 98\,\%$), experimenty potvrdily výskyt přeučení – validační a testovací přesnost se stabilně pohybovala pod 4%. To naznačuje omezenou schopnost modelu generalizovat na nové vstupy, pravděpodobně kvůli datové nevyváženosti a převažujícímu podílu syntetických vzorků.

Projekt KNN-Math je modulární a otevřený dalšímu rozšiřování. Do budoucna by bylo vhodné zaměřit se na zlepšení kvality a rozmanitosti trénovacích dat, použití větších a realističtějších datasetů, nebo experimentování s jazykovými modely pro zpřesnění dekódované sekvence.

Zdroje

[1] ZHELEZNIAKOV, Dmytro; ZAYTSEV, Viktor; RADYVONENKO, Olga. Online Handwritten Mathematical Expression Recognition and Applications: A Survey. *IEEE Access*. 2021, roč. 9, s. 38352–38373. Dostupné z DOI: 10.1109/ACCESS.2021.3063413.

[2] ZHANG, Jianshu; DU, Jun; ZHANG, Shiliang; LIU, Dan; HU, Yulong; HU, Jinshui; WEI, Si; DAI, Lirong. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*. 2017, roč. 71, s. 196–206. ISSN 0031-3203. Dostupné z DOI: 10.1016/j.patcog.2017.06.017.

[3] ZHANG, Jianshu; DU, Jun; DAI, Lirong. Multi-Scale Attention with Dense Encoder for Handwritten Mathematical Expression Recognition. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018, s. 2245–2250. Dostupné z DOI: 10.1109/ICPR.2018.8546031.

[4] LE, Anh Duc; NAKAGAWA, Masaki. Training an End-to-End System for Handwritten Mathematical Expression Recognition by Generated Patterns. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. 2017, sv. 01, s. 1056–1061. Dostupné z DOI: 10.1109/ICDAR.2017.175.

[5] XIE, Yejing; MOUCHÈRE, Harold; SIMISTIRA LIWICKI, Foteini; RAKESH, Sumit; SAINI, Rajkumar; NAKAGAWA, Masaki; NGUYEN, Cuong Tuan; TRUONG, Thanh-Nghia. ICDAR 2023 CROHME: Competition on Recognition of Handwritten Mathematical Expressions. In: FINK, Gernot A.; JAIN, Rajiv; KISE, Koichi; ZANIBBI, Richard (ed.). *Document Analysis and Recognition - ICDAR 2023*. Cham: Springer Nature Switzerland, 2023, s. 553–565. ISBN 978-3-031-41679-8.

[6] JANUPALLI, Pranay. Understanding Sinusoidal Positional Encoding in Transformers. *Medium*. 2024. Dostupné také z: https://medium.com/@pranay.janupalli/understanding-sinusoidal-positional-encoding-in-transformers-26c4c161b7cc.

[7] LI, Zhe; JIN, Lianwen; LAI, Songxuan; ZHU, Yecheng. Improving Attention-Based Handwritten Mathematical Expression Recognition with Scale Augmentation and Drop Attention. In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2020, s. 175–180. Dostupné z DOI: 10.1109/ICFHR2020.2020.00041.