KNN checkpoint projektu

Radomír Bábek (xbabek02), Petr Volf (xvolfp00), Přemek Janda (xjanda28)
Brno University of Technology

Úvod

Pro společný vývoj jsme využili prostředí Google Colab, kde je k dispozici námi implementovaný kód pro projekt a github repozitář.

1 Definice úlohy

Naše úloha se zaměřuje na optické rozpoznávání znaků (OCR), konkrétně na rozpoznávání ručně psaného textu. Cílem je převést obrazový vstup na strojově čitelný text. Oproti běžnému přístupu pomocí konvolučních sítí vstupní obrazová data slouží jako podklad pro dekódování textu prostřednictvím encoder–decoder modelu postaveném na attention mechanismu a transformerové architektuře.

Vstupem v našem případě je dataset LAM, který obsahuje naskenované obrázky ručně psaných textů spolu s jejich textovými přepisy. Aktuálně máme TrOCR model, který byl fine-tunován na tomto datasetu, nicméně jeho schopnost rozpoznávat jednotlivé znaky je zatím velmi omezená, což se odráží i na vysoké hodnotě chybovosti (viz výsledky trénování 3. Cílem je proto zlepšit kvalitu výstupu prostřednictvím lepšího trénování dekodéru a vhodných evaluačních metrik.

2 Přehled existujících řešení

Na Hugging Face jsou pro ručně psané texty k dispozici tři modely: `small`, `base` a `large`. Pro jednoduchost a rychlost učení jsme si vybrali variantu `small`.

Model TrOCR je postaven na architektuře typu encoder–decoder, kde vizuální vstup (obraz textu) zpracovává vizuální transformer a výstupní text generuje jazykový transformer. V případě encoderu využívá TrOCR předtrénovaný model **DeiT (Data-efficient Image Transformer)**, který nahrazuje tradiční CNN backbone. Model pracuje na základě rozdělení obrazu do patchů, které zpracovává jako sekvenci.

V naší práci bychom se rádi zaměřili na dekodér, který z vizuální reprezentace vytváří jazykový výstup. Při práci s různými typy rukopisu nebo méně rozšířenými jazyky je zásadní kontext okolo jednotlivých písmen pro správnou interpretaci znaků.

Přehled základních OCR technik

Předpracování

Většina OCR nástrojů zpracovává svůj vstup na bázi několika kroků. Vstupem je většinou naskenovaný dokument, který může být tištěný nebo ručně psaný. Nad stranou je poté provedena analýza, jejíž výsledkem je rozdělení na řádků textu. Před rozpoznáváním textu, který se na řádku nachází, je často potřeba vstupní obrázek předzpracovat.

Jednou z frekventovaně používanou technikou předzpracování vstupů je binarizace pomocí histogramu hodnot pixelů. Pokud existuje obrázek ve stupních šedé, můžeme spočítat zastoupení jednotlivých stupňů, následně zvolit barevný práh, který rozdělí pixely na 2 stejně velké poloviny, z nichž tmavější polovina je převedena na černou, světlejší na bílou.

Rozpoznávání znaků

Existuje několik metod, které lze využít pro rozpoznání znaků. Jednou z nich je využití konvolučních neuronových sítí, které jsou schopny rozpoznat hrany jednotlivých znaků. Dále využívanými sítěmi jsou sítě rekurentní, které jsou

schopny udržovat kontext z předcházejících vstupů. Rekurentní sítě na základě sekvence znaků dokáží například odhadovat fontovou rodinu, ze které znaky řádku textu pocházejí. V praxi se však čím dál více uplatňují neuronové sítě s dlouhodobou a krátkodobou pamětí LSTM, jejich velkou výhodou je, že při vysoké velikosti kontextu nedochází k exponenciálnímu zvětšování či zmenšování gradientu na základě váhy u zpětných smyček. Tomuto nechtěnému jevu se říká vanishing/exploding gradient.

Chybové funkce

Moderní OCR modely se liší nejen použitou architekturou neuronových sítí, ale také ztrátovou funkcí, kterou používají při trénování vnitřních parametrů.

Mezi nejpoužívanější chybové funkce patří například CTC (Connectionist Temporal Classification)[1]. Funkce CTC je specifická v tom, že nevyžaduje zarovnání textu. Vstupní řádek s textem je rozdělen na bloky, pomocí RNN jsou poté vypočítány pravděpodobnosti výskytů jednotlivých znaků. Pokud není detekován žádný znak, nejvyšší pravděpodobnost je přiřazena symbolu ϵ. Tím je vytvořena sekvence predikovaných znaků. Stejné po sobě jdoucí znaky jsou sloučeny do jednoho, symboly ϵ jsou ve výsledku odstraněny.

Tesseract OCR

Tesseract[2] je open source nástroj vydáván pod licencí Apache 2.0. Je vyvíjen od roku 1985 (společnost HP)[3]. Od roku 2006 je primárně spravován společností google. Dokáže zpracovat více než 100 různých jazyků a znaky standardu UTF-8. Nepodporuje vstupy formátu PDF, umožňuje však zpracovat většinu běžných obrázkových formátů - jmenovitě formáty PNG, JPEG, TIFF, GIT, WebP, BMP a PNG. Výstupem jsou poté soubory typu TXT, TSV, hOCR, nebo PDF.

Prvotní verze Tesseractu využívaly primárně klasické algoritmy pro segmentaci a analýzu tvarů písmen. Od verze 4.00 funguje Tesseract na bázi subsystému neuronových sítí LSTM, které lze trénovat na vlastních datech. Neuronové sítě jsou využívány pro detekci řádků. Před spuštěním rozpoznávání textu se provede řada kroků, které souvisí s předzpracováním obrazu (odstranění šumu, binarizace, detekce textových oblastí. Výsledkem této operace je rozdělení dokumentu na bloky, řádky či jednotlivá písmena. Následně je předzpracovaný text podroben extrakci rysů pomocí konvolučních neuronových sítí. Sekvence rysů jsou následně analyzovány pomocí rekurentních vrstev složených z hlubokých obousměrných LSTM.

Tesseract není vhodný pro obrázky s ručně psaným textem, model je navržen pro zpracování tištěných dokumentů. Model lze dotrénovat pomocí souborů se strukturou `.traineddata`.

Google Cloud Vision

Oproti Tesseractu se jedná o cloudovou placenou službu, která se prezentuje vysokou přesností bez velkého množství vynaložené práce. Daleko lépe si poradí s ručně psaným textem a tabulkami. Dokáže rozpoznat i rozložení dokumentu a extrahovat loga nebo detekovat obličeje. OCR je pouze jeden z využitelných nástrojů, je naučený na velkou škálu fontů.

3 Dataset

3.1 LAM Dataset

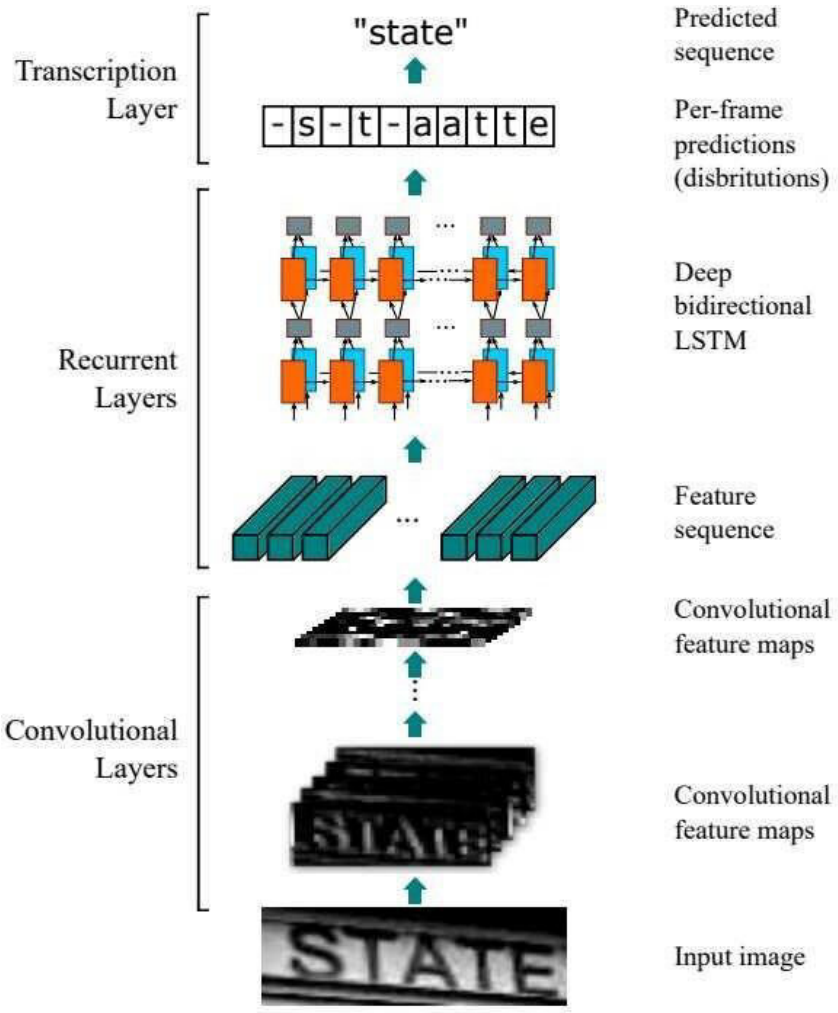
Námi zvolený dataset obsahuje zápisy Italského autora Ludovico Antonio Muratori (LAM), pořízené během 60 let jeho života. Vzhledem ke změnám ve formě je vhodná pro HTR (Hand Text Recognition). [1] Dataset je rozdělen na stránky a dále segmentován do řádků, kterých je celkově 25 823. Dataset je předem rozdělený na trénovací, validační a testovací sady.

3.2 Datasety pro další fine-tuning

V dalším postupu bychom se zaměřili na porovnání s dalšími datasety, případně aplikovali fine-tuning a vytvořili nové dotrénované inidividuálně pro každý další model.

[1] `https://distill.pub/2017/ctc`

[2] `https://github.com/tesseract-ocr/tessdoc/blob/main/tess4/NeuralNetsInTesseract4.00.md`



Obrázek 1: Tesseract OCR rozčleněn na jednotlivé funkcí vrstvy, převzato z [2]

Středověké texty (Zenodo)

Dataset obsahuje středověké rukopisy převážně v němčině se skeny historických dokumentů a jejich přepisy. Je vhodný pro rozšíření schopností modelu TrOCR na historické a netypické rukopisy, které se výrazně liší od moderního písma.

German Handwriting (fhswf/german_handwriting)

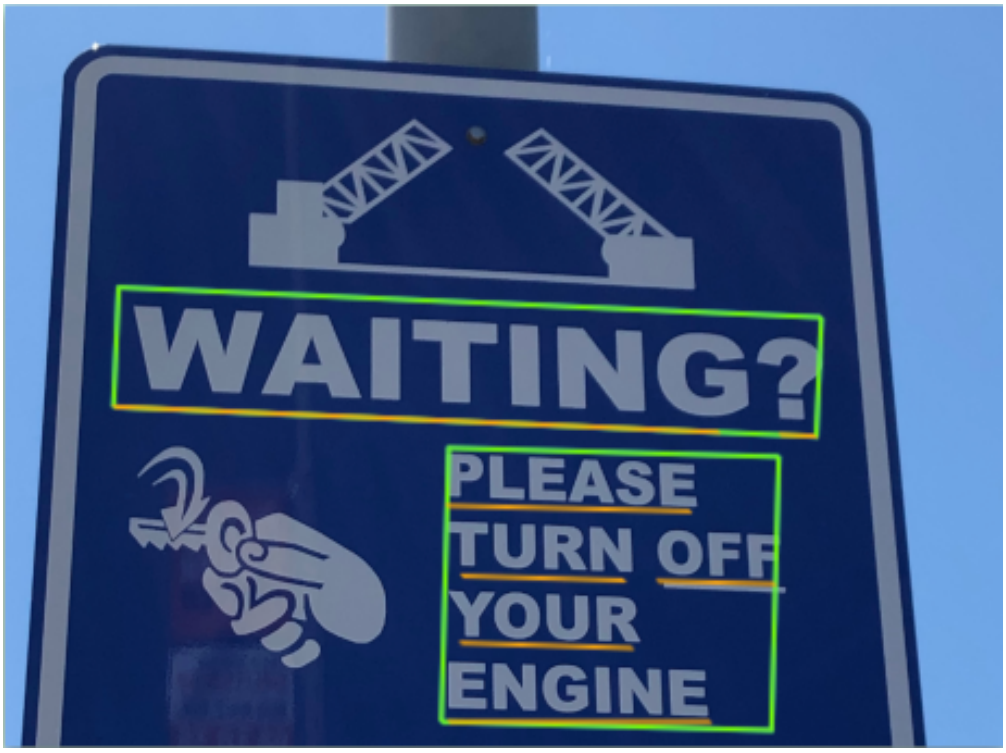
Jedná se o dataset modernějšího ručně psaného textu v němčině, který zachycuje variabilitu rukopisů. Dataset je vhodný pro trénink modelu na OCR v německém jazyce a poskytuje kvalitní a dobře strukturovaná data.

Vietnamese Handwritten Text (BinKhoaLe1812/Vietnamese_Handwritten_Text)

Tento dataset obsahuje moderní ručně psaný vietnamský text s diakritikou. Je ideální pro rozšíření modelu TrOCR na asijské jazyky používající latinku, přičemž učí model pracovat s jinou jazykovou strukturou a bohatou diakritikou.

Handwriting Recognition Dataset (gymprathap)

Dataset nabízí obecné ručně psané texty (pravděpodobně v angličtině) s různými styly písma. Je vhodný pro trénink robustního modelu schopného zvládat různorodé rukopisy napříč jazykovými variantami.



Obrázek 2: Detekce textu na dopravní značce pomocí Google Cloud Vision

Handwriting Recognition (Kaggle - landlord)

Dataset obsahuje znaky a číslice, které jsou často využívány pro klasické OCR. Slouží spíše jako doplněk pro rozpoznávání symbolů a číslic, než pro kompletní větné struktury.

Očekávaný výkon

U datasetů se moderními rukopisy (německý, vietnamský, obecný) lze očekávat chybovost v rozmezí 3–10 % (CER), přičemž jednodušší znaky (např. číslice) z Kaggle datasetu si myslíme, že by chybovost mohla dosahovat pod 2 %. Naopak u středověkých textů bude kvůli náročnosti písma a kvalitě skenů pravděpodobně mnohokrát vyšší chybovost, obdobně jako u současného.

4 Hodnotící metrika

Vyhodnocovací metriky

Pro OCR úlohy se běžně používají dvě základní metriky: chybovost na úrovni písmen (*Character Error Rate*, CER) a chybovost na úrovni slov (*Word Error Rate*, WER). Výstupní (predikovaný) text modelu je porovnáván s manuálně označeným (referenčním/ground truth) textem.

CER měří rozdíl mezi jednotlivými znaky a je vhodnější pro modely zaměřené na rozpoznávání bez znalosti jazykového kontextu, typicky tam, kde modely klasifikují každé písmeno zvlášť. Oproti tomu WER porovnává celá slova a je vhodnější tam, kde model využívá jazykové modelování (např. jedna chyba v písmenu vede k celkové chybě ve slově, což WER oproti CER penalizuje více.

Vzhledem k námi zvolené metodě, konkrétně modelu TrOCR, který kombinuje vizuální enkodér a jazykový dekodér (na bázi Transformer architektury), jsme se rozhodli pro použití metriky **WER**, která lépe odráží skutečnou čitelnost generovaného textu a jeho smysluplnost v rámci konkrétního jazyka. Nicméně pro účely fine-tuningu jsme implementovali seq2seq pipeline, která je vhodnější vyhonotit na úrovni písmen.

Dalším krokem, který jsme při výpočtu metrik aplikovali, je **normalizace textu**, při které odstraňujeme interpunkci a nadbytečné bílé znaky. Tyto znaky nejsou pro naši úlohu zásadní a jejich přítomnost by mohla zkreslit vyhodnocení skutečné chybovosti.

Měříme celkem čtyři metriky:

$$\text{CER} = \frac{S + D + I}{N}$$

$$\text{WER} = \frac{S + D + I}{W}$$

$$\text{Accuracy} = \frac{\text{Počet správně rozpoznaných slov}}{\text{Celkový počet slov}}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

kde:

- S je počet substitucí (špatně rozpoznaných znaků nebo slov),
- D je počet delecí (chybějících znaků nebo slov),
- I je počet insercí (nadbytečných znaků nebo slov),
- N je počet znaků v referenčním textu (pro CER),
- W je počet slov v referenčním textu (pro WER).

5 Vyhodnocení baseline

Pokusili jsme se rozšířit model *TrOCR*, aby byl schopen rozpoznávat psané písmo v *LAM* datasetu. Nejprve bylo třeba zjistit, zda znaková sada, na které byl model předtrénovaný, obsahuje speciální italské znaky. Pokud by neobsahovala, museli bychom pravděpodobně měnit i tokenizer, to ovšem nebyla pravda a slovník znaky obsahoval, tudíž existuje šance, že by bylo možné dekodér dotrénovat.

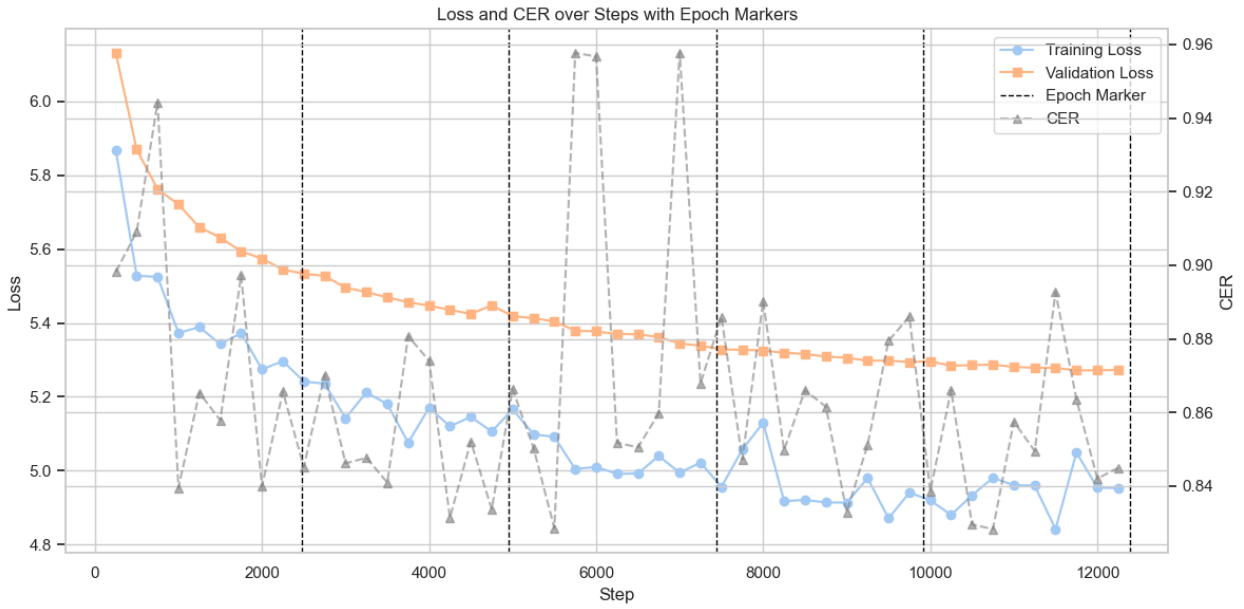
Podle testovacích výsledků bylo zřejmé, že model byl předtrénovaný na angličtině. Pokusy o inferenci testovacích dat z *LAM* vracely špatně rozpoznané řetězce, které reflektovaly spíše naučenou angličtinu než italštinu, tedy chybovost CER i WER se pohybovaly v rozmezí 80-90 %. Pro normalizované výstupy byla úspěšnost nižší, především se odvíjela od množství šumu, překrývajících se písmen apod.

Z grafu výše 3 vyplývá, že i přesto, že trénovací i validační ztráta plynule klesají, Hodnotící metrika se během učení modelu příliš neměnila a ve velké míře kolísala. Nicméně, tendence obou loss funkcí byla stále klesající, takže bychom pravděpodobně mohli trénovat déle. Důvodů proč je však CER vysoká může být mnoho, rádi bychom to v další fázi projektu více prozkoumali a ověřili, zda někde nemáme při dekódování chybu nebo nemáme špatně volené hyper-parametry.

6 Další postup

Jak jsme již psali výše, rádi bychom se zaměřili na dekodér. Například bychom mohli pomocí finetuningu doladit předtrénovaný model, úpravou architektury (např. hloubka nebo počet attention hlav), nebo přizpůsobením tokenizeru cílovému jazyku. Zlepšení dekodéru by tak snad mohlo výrazně snížit chybovost.

- Postup
  1. Zjistit proč je CER tak vysoká
  2. Dotrénovat model na více epochách
  3. Zkusit vyladěný model na dalších datasetech
  4. Vyhodnotit, porovnat
  5. Zkusit pozměnit implementaci dekodéru
  6. Zaměřit se na tokenizer
- Experimenty



Obrázek 3: Graf průbehu trénování TrOCR-small-handwritten.

1. Zvýšení počtu epoch
2. Změna velikosti batch size
3. Úprava architektury

Reference

[1] Silvia Cascianelli, Vittorio Pippi, Maarand Martin, Marcella Cornia, Lorenzo Baraldi, Kermorvant Christopher, and Rita Cucchiara. The lam dataset: A novel benchmark for line-level handwritten text recognition. In *International Conference on Pattern Recognition*, 2022.

[2] Ritika Rai, Srushti Shitole, Pratiksha Sutar, Swapnali Kaldhone, and Jayashree Jadhav. Automatic license plate recognition using yolov4 and tesseract ocr. *International Journal of Innovative Research in Computer and Communication Engineering*, 10:1656, 01 2008.

[3] R. Smith. An overview of the tesseract ocr engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633, 2007.