FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

Klasifikace skenovaných dokumentů

Jakub Fafek

PLZEŇ

2024

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

Klasifikace skenovaných dokumentů

Jakub Fafek

Vedoucí práce
Ing. Jiří Martínek, Ph.D.

PLZEŇ 2024

Citace v seznamu literatury:
FAFEK, Jakub. *Klasifikace skenovaných dokumentů*. Plzeň, 2024. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Ing. Jiří Martínek, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub FAFEK**
Osobní číslo:	**A19B0034P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Klasifikace skenovaných dokumentů**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prozkoumejte datovou sadu obrazových skenovaných dokumentů Tobacco-3482.
2. Navrhněte vhodné metody pro extrakci příznaků.
3. Proveďte analýzu klasifikátorů, které je možné použít pro úlohu klasifikace dokumentů.
4. Vybrané metody a klasifikátory implementujte.
5. Proveďte experimenty s cílem ověřit funkcionalitu a dosažené výsledky kriticky zhodnoťte.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Jiří Martínek, Ph.D.**
Nové technologie pro informační společnost

Datum zadání bakalářské práce: **29. srpna 2023**
Termín odevzdání bakalářské práce: **2. května 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 29. srpna 2023

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 1. května 2024

. . . . . .

Jakub Fafek

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Tato bakalářská práce se zabývá tématem klasifikace skenovaných dokumentů. Cílem této práce je vyvinout klasifikátor, který bude schopen rozlišit naskenované dokumenty do celkem deseti tříd. Jako příznak se použije text, který je rozpoznán pomocí metod optického rozpoznávání znaků (OCR). Dále bude využita hluboká konvoluční neuronová síť, která zohlední vizuální příznaky. Nakonec bude vytvořen klasifikátor, který zohledňuje oba typy příznaků. Klasifikátor bude vyhodnocen na datové sadě dokumentů Tobacco-3482.

Abstract

This bachelor thesis deals with the topic of classification of scanned documents. The aim of this thesis is to develop a classifier that will be able to distinguish scanned documents into a total of ten classes. Text will be used as a feature which is recognized using optical character recognition (OCR) methods. Furthermore, a deep convolutional neural network will be used to take visual features into account. Finally, a classifier will be created that accounts for both types of features. The classifier will be evaluated on the Tobacco-3482 document dataset.

Klíčová slova

klasifikace dokumentu • skenovaný dokument • neuronová síť • klasifikace textu • klasifikace obrázku • OCR

Poděkování

Rád bych poděkoval Ing. Jiřímu Martínkovi, Ph.D. za odborné vedení, za pomoc a za rady při zpracování této práce.

Jakub Fafek

Obsah

Úvod 1

Digitální dokumenty se staly nepostradatelným a běžným zdrojem informací. Samotný web obsahuje velké množství textových dokumentů jako jsou zprávy, časopisy nebo publikace, které jsou dostupné v elektronické podobě. Avšak, přestože jsou dostupné, často tyto dokumenty nejsou v organizované formě. Velkou část představují skenované dokumenty, které nemají textovou formu a jejich automatické zpracování je obtížné. Existují také další problémy, jako například, že velmi málo z těchto dokumentů je nějakým způsobem označeno, takže se neví, co tento dokument obsahuje. Je proto velice užitečné umět tyto typy dokumentů rozlišovat.

V současné době se pro klasifikaci textu, obrázků, dokumentů a podobně výhradně používají neuronové sítě s různými vstupními příznaky. V případě moderních hlubokých neuronových sítí není potřeba takovéto příznaky vytvářet. Vstupem je přímo zakódovaný text či obrázek a síť si příznaky vytvoří/odvodí sama. Oproti jiným jednodušším klasifikátorům potřebuje síť nicméně velké množství vzorků, pomocí kterých se neuronová síť natrénuje. Nevýhodou takového modelu je to, že potřebuje hodně času na trénování a k trénování je potřeba velkého výpočetního výkonu.

Tématem této bakalářské práce je klasifikace dokumentů, které jsou ve formě skenovaných obrázků. Cílem je navrhnout a implementovat takový klasifikátor, který bude tyto dokumenty klasifikovat do celkem deseti tříd a zohlední jak vizuální, tak i textovou informaci, která bude automaticky dodána pomocí metod optického rozpoznávání znaků (OCR).

Následující kapitola popíše, jakým způsobem se provádí klasifikování dokumentů. V kapitole OCR bude nastíněno fungování OCR a zdůvodněno, proč byl vybrán Tesseract pro rozpoznávání znaků a zmíní ostatní možné modely. Další kapitola popíše datovou sadu Tobacco-3482, která byla v této práci použita, včetně její analýzy. V kapitole Experimenty budou postupně popsány experimenty, které se v prováděly v této práci. Na konci práce bude závěr, který zhodnotí práci a výsledky a zároveň popíše možné vylepšení.

Klasifikace dokumentů

2

Klasifikace je metoda strojového učení pro zařazování či třídění dat do specifikovaných skupin či tříd. Úloha má celou řadu aplikací. Jedná se například o detekce spamů, třídění doručených e-mailů, klasifikace obrázků se zvířaty, kategorizace novinových článků či zpráv. Dle zdroje [Kum22] lze základní druhy klasifikací s využitím umělé inteligence dělit na tři typy:

1. Binární klasifikace (*binary class*);
2. Vícetřídní klasifikace (*multiclass*);
3. Vícetřídní klasifikace, přičemž se připouští více tříd najednou (*multilabel*);

Binární klasifikace je typ klasifikace, při níž jeden klasifikovaný prvek může spadat pouze dvou možných tříd. A to buď do jedné nebo do druhé. Jedná se o nejjednodušší formu klasifikace, kde výstup může být buď pozitivní, nebo negativní, pravdivý, nebo nepravdivý, 0 nebo 1. Příklady binární klasifikace zahrnují detekci spamu, diagnostiku nemocí, detekci podvodů a analýzu sentimentu.

Vícetřídní klasifikace je typ klasifikace, při níž jeden klasifikovaný prvek může spadat do více než dvou možných tříd. Jinými slovy, jedná se o klasifikační problém s více než dvěma cílovými třídami. Příklady klasifikace více tříd zahrnují rozpoznávání zvířat, klasifikaci dokumentů nebo kategorizaci výrobků.

Vícetřídní klasifikace s více třídy najednou, je druh klasifikace, při níž jeden klasifikovaný prvek může spadat do více než jedné možné třídy. Je podobná vícetřídní klasifikaci (multiclass), ale místo přiřazení jedné značky každému klasifikovanému prvku se přiřazuje více značek. Příkladem této klasifikace je klasifikace žánru hudby či žánru filmu.

U textových dokumentů se setkáváme se všemi třemi typy (binární – detekce spamu, vícetřídní – kategorizace zpráv). Vstupem do těchto klasifikátorů je většinou samotný text (resp. jeho zakódovaná podoba). Pro klasifikaci jsou v dnešní době nejčastěji používány modely neuronových sítí [Aud+19][Fer+20].

2.1 Extrakce příznaků

Důležitým krokem v procesu klasifikace naskenovaných dokumentů je extrakce příznaků. Extrakce příznaků je proces, při kterém se z dat identifikují a extrahují určité charakteristiky. Účelem extrakce příznaků je identifikovat a extrahovat relevantní informace z naskenovaného dokumentu, které pak mohou být použity k zařazení dokumentu do určité kategorie či třídy. Aby bylo možné klasifikovat nějaký dokument, je potřeba pro daný dokument vytvořit příznakový vektor. Jako jedna z prvních příprav pro extrakci příznaků je potřeba dokument předzpracovat. Literatura [BLK10] uvádí, že předzpracování spočívá v odstranění jazykově závislých faktorů, tokenizaci, odstranění stop slov a převedení slov do jejich základní tvaru. Odstranění jazykově závislých faktorů znamená, že text by měl být převeden na malé písmena a interpunkce by měla být odstraněna. Tokenizace znamená, že text by měl být rozdělen na seznam tokenů, kde každé slovo je jeden token. Odstranění stop slov se provádí tak, že se nejprve vytvoří seznam stop slov, což jsou slova, která neobsahují užitečné informace, a proto se často odstraňují. Tato stop slova jsou poté odstraněna ze seznamu tokenů. Poté lze vytvořit příznakový vektor.

2.2 Neuronová síť

Neuronová síť je výpočetní model, který se učí provádět úlohy zpracováním dat prostřednictvím vrstev propojených uzlů neboli umělých neuronů [Wik23]. Je schopna učit se složité vzory a vztahy v datech a může být natrénována k provádění předpovědí nebo rozhodnutí na základě nových ještě neviděných vstupů.

2.2.1 Vícevrstvý perceptron

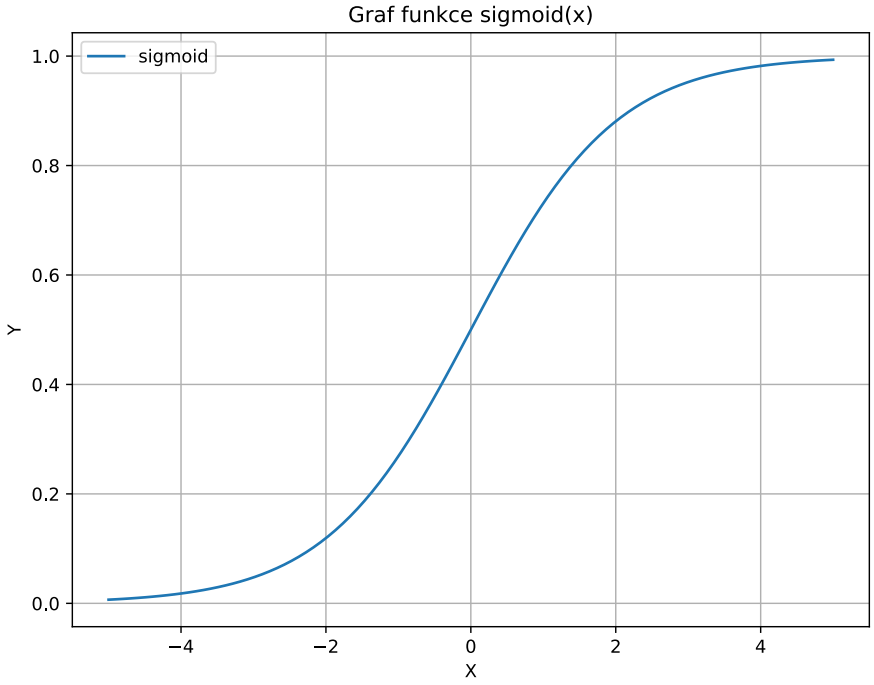
Vícevrstvé perceptrony (MLP – Multilayer perceptron) byl jedním z nejpoužívanějších modelů neuronových sítí, zejména v aplikacích, jako je rozpoznávání obrazu, rozpoznávání řeči a zpracování přirozeného jazyka. S postupným zlepšováním se ale začaly namísto vícevrstvých perceptronů využívat hluboké sítě.

MLP je dopředná neuronová síť, která se skládá z jedné nebo více vrstev uzlů, přičemž každá vrstva je plně propojena s další vrstvou [Pop+09]. První vrstva je vstupní vrstva, do které jsou přiváděna vstupní data. Další vrstvy se nazývají skryté vrstvy. Poslední vrstvou je výstupní vrstva (output layer), která vytváří předpovídaný výstup.

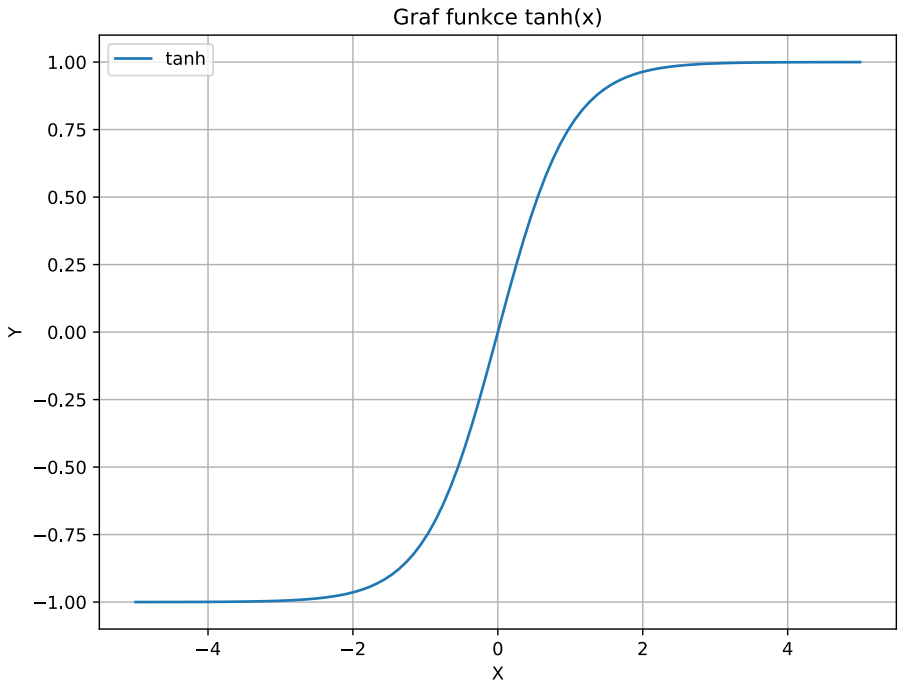
Každý uzel v MLP je neuron, který přijímá vstupy ze všech uzlů v předchozí vrstvě, a používá nelineární aktivační funkci společně s váhami spojení k vytvoření výstupu. Váhy spojení mezi uzly jsou parametry, které se síť učí během trénování.

Aktivační funkce slouží k zavedení nelinearity do modelu, což mu umožňuje zachytit složité vztahy mezi vstupem a výstupem.

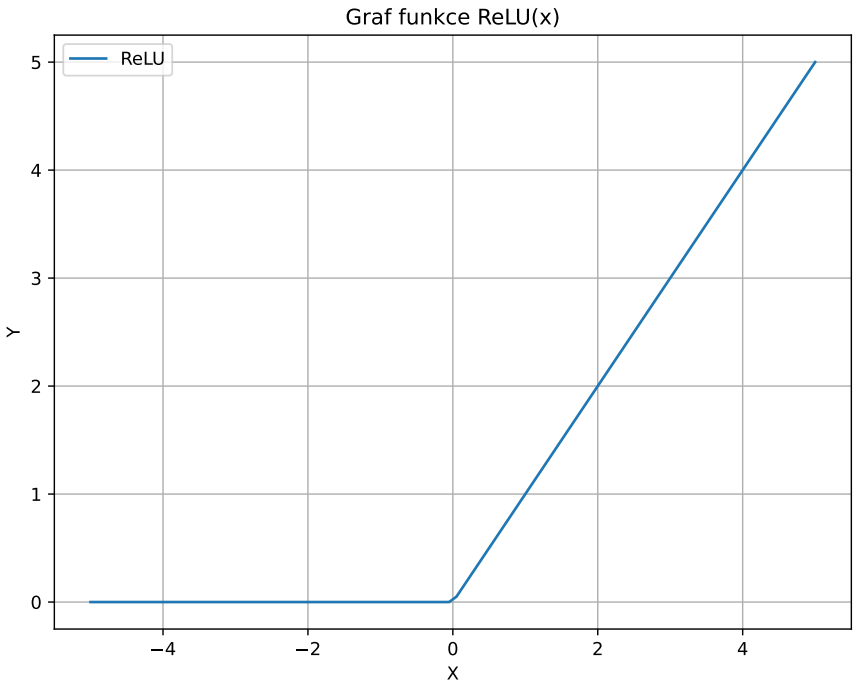
Nejčastěji používanými aktivačními funkcemi v MLP je funkce sigmoid (obrázek 2.1), hyperbolický tangens (obrázek 2.2) a ReLU (obrázek 2.3). Sigmoid a hyperbolický tangens jsou funkce, které vytvářejí hladkou křivku, což jim umožňuje mapovat libovolný vstup na výstup mezi 0 a 1, respektive -1 a 1. Funkce ReLU je naproti tomu po částech lineární funkce, která vytváří nulový výstup pro záporné vstupy a lineární výstup pro kladné vstupy.



Obrázek 2.1: Funkce sigmoid



Obrázek 2.2: Funkce hyperbolický tangens

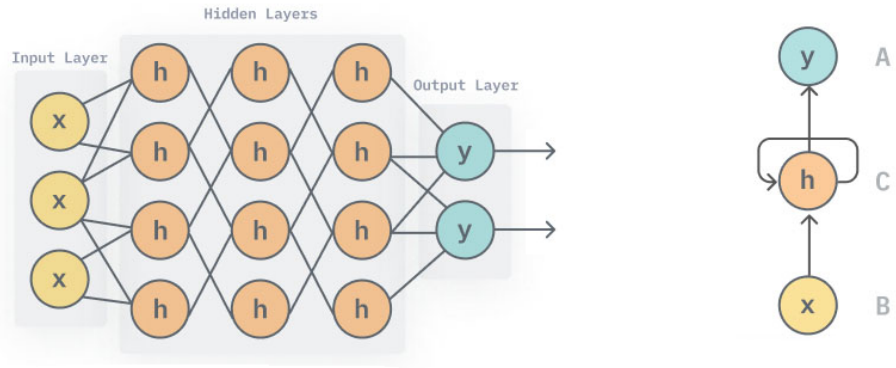


Obrázek 2.3: Funkce ReLU

Trénování MLP zahrnuje techniku zvanou zpětná propagace chyb (back propagation), což je optimalizační algoritmus sestupného gradientu. Zpětné šíření funguje tak, že se vypočítá chyba mezi předpovídaným výstupem sítě a skutečným výstupem a tato chyba se pak zpětně šíří sítí, aby se aktualizovaly váhy mezi neurony. Tento proces se opakuje po několik iterací, dokud není chyba minimalizována a síť není schopna předpovídat výstup na trénovacích datech s co nejmenší chybou.

2.2.2 Rekurentní neuronová síť

Rekurentní neuronová síť (RNN) je typ umělé neuronové sítě, která je určena ke zpracování sekvenčních dat. Na rozdíl od dopředných neuronových sítí, které zpracovávají vstupy pouze v jednom průchodu a neudržují žádnou vnitřní paměť, obsahují RNN dočasný vnitřní stav, který jim umožňuje zpracovávat sekvence vstupů [Sch19]. Na obrázku 2.4 je ukázána architektura rekurentní neuronové sítě.



Obrázek 2.4: Architektura rekurentní neuronové sítě [1]

Vnitřní stav RNN umožňuje síti zachytit informace z minulých vstupů a použít je pro zpracování aktuálního vstupu. Díky tomu jsou sítě RNN zvláště užitečné pro úlohy, kde je důležité zohlednit posloupnost prvků na vstupu, jako je například rozpoznávání řeči nebo zpracování přirozeného jazyka.

RNN jsou obvykle implementovány pomocí speciálního typu neuronu zvaného „memory cell“ (paměťová buňka). Tato paměťová buňka uchovává paměť předchozích vstupů a využívá ji k aktualizaci svého aktuálního stavu. Stav paměťové buňky je pak předán dalšímu neuronu v síti, který jej může použít k aktualizaci svého vlastního stavu na základě aktuálního vstupu.

Jednou z klíčových výhod sítí RNN je jejich schopnost pracovat se vstupními sekvencemi proměnné délky. Díky tomu se používají tam, kde je důležité zohlednit posloupnost prvků na vstupu. Protože si síť uchovává paměť předchozích vstupů, může zpracovávat vstupy různé délky a stále vytvářet smysluplný výstup. Díky tomu jsou sítě RNN zvláště užitečné pro úlohy, jako je rozpoznávání řeči, kde se délka vstupní sekvence může měnit v závislosti na délce mluvené řeči.

Rekurentní síť je příklad architektury, který je vhodný ke zpracování sekvencí. Může tedy být využit pro klasifikaci textové informace z naskenovaného dokumentu.

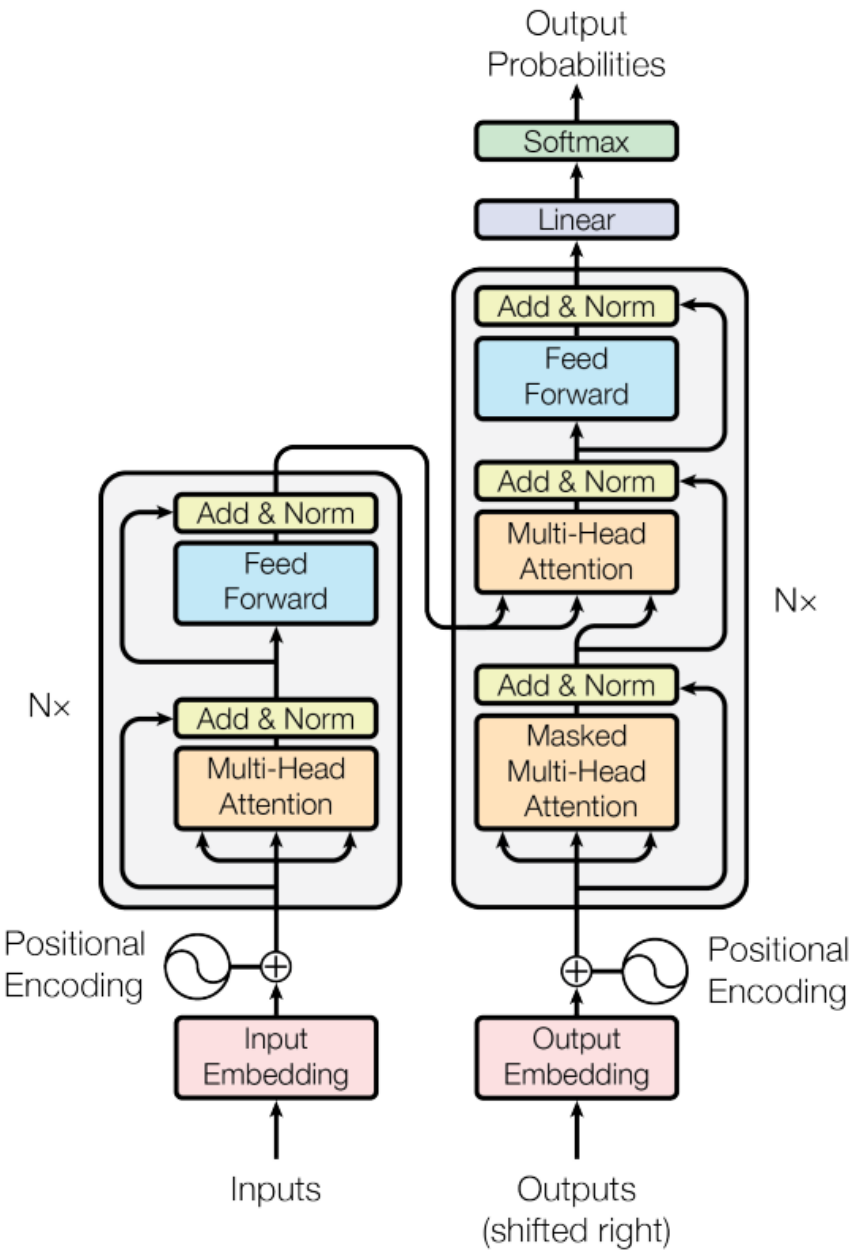
2.2.3 Transformer

Transformer je architektura neuronové sítě, která je speciálně navržena pro sekvenční data, jako je například text či řeč [Vas+17]. Je založena na hlubokém učení, což znamená, že se může naučit komplexní reprezentaci vstupních dat trénováním na velkém množství označených dat.

Architektura Transformeru se skládá ze dvou hlavních komponent, a to kodéru a dekodéru. Tyto komponenty se skládají z několika vrstev neuronových sítí. Ko-

[1] Obrázek použit z `v7labs.com/blog/recurrent-neural-networks-guide`

dér zpracovává vstupní sekvenci a vytváří sekvenci skrytých reprezentací, které předává dál. Dekodér přebírá tyto sekvence vytvořené kodérem a používá je k vytvoření výstupní sekvence. Každá vrstva kodéru i dekodéru obsahuje dvě podvrstvy: self-attention mechanismus [Vas+17], který umožňuje síti zaměřit se na různé části vstupní posloupnosti, a dopřednou síť, která aplikuje nelineární transformaci na výstup mechanismu self-attention, aby vytvořila reprezentaci pro danou vrstvu. Na obrázku 2.5 je zobrazena architektura Transformeru.



Obrázek 2.5: Architektura Transformeru [Vas+17]

Mechanismus self-attention umožňuje Transformeru věnovat pozornost různým částem vstupní sekvence při generování každé skryté reprezentace. Tento self-attention mechanismus se liší od jiných architektur neuronových sítí, protože je založen na celé vstupní posloupnosti, a nikoli pouze na pevném okně kontextu. Self-attention mechanismus umožňuje Transformeru se zaměřit na důležité části vstupní posloupnosti.

Jelikož Transformery postrádají pojetí ohledně pořadí slov, využívá se mechanismu kódování pozic (positional encoding). Jedná se o klíčovou složkou, která má za úkol poskytovat informace o pozici slov nebo tokenů ve zpracovávané sekvenci. Tato informace je vnášena do vstupu Transformeru.

Dopředná síť (feed-forward) v každé vrstvě kodéru a dekodéru přebírá výstup self-attention mechanismu a používá nelineární transformaci k vytvoření konečné skryté reprezentace pro danou vrstvu. Síť typu feed-forward je obvykle vícevrstvý perceptron (MLP), který se skládá z jedné nebo více plně propojených vrstev s nelineárními aktivačními funkcemi.

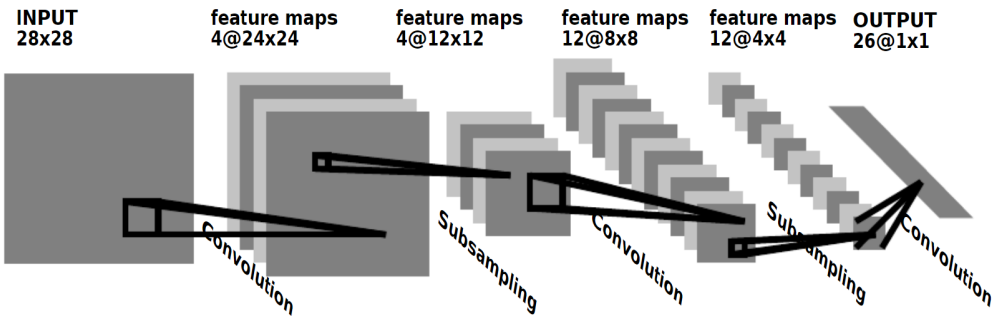
Autoři článku [Vas+17] zmiňují, že Transformery jsou vhodné pro úlohy, kde vstupem a výstupem jsou posloupnosti proměnné délky. Může se jednat o překlad jazyka, shrnutí textu a zodpovídání otázek.

2.2.4 Konvoluční neuronová síť

Konvoluční neuronové sítě (CNN) jsou považovány za nejlepší v oblasti zpracování obrázků. Podle studie Krizhevskyho a dalších [KSH12] představují konvoluční neuronové sítě současnou špičku v oblasti rozpoznávání obrazu, překonávají tradiční metody a dosahují výrazně vyšší přesnosti na srovnávacích souborech dat, jako je ImageNet.

Typicky konvoluční neuronová síť obsahuje několik konvolučních filtrů (anglicky filter nebo kernel), které jsou následovány vrstvami tzv. subsamplingu, např. max-pooling (viz obrázek 2.6). Na konci je výstupní vrstva dle počtu tříd, do kterých se klasifikuje.

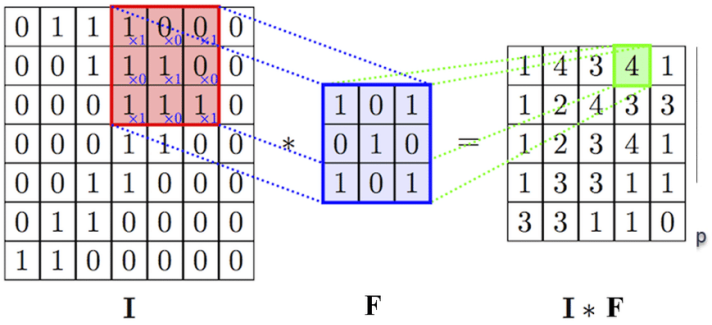
Úkolem konvolučních vrstev je extrahovat důležité elementy v obrázku. Váhy konvolučních filtrů jsou optimalizovány v průběhu trénování sítě a jejich výstupem jsou tzv. příznakové mapy, které vzniknou aplikací operátoru konvoluce (viz Rovnice 2.1 a obrázek 2.7).



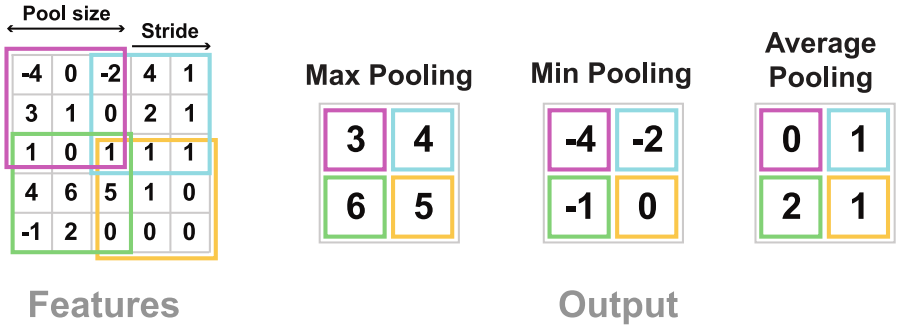
Obrázek 2.6: Příklad architektury CNN [LB+95]

$$[H](I * K)[m,n] = \sum_j \sum_k K[j,k] I[m-j, n-k] \tag{2.1}$$

V rovnici 2.1 $(I * K)[m,n]$ představuje výsledek konvoluce v souřadnicích (m, n). $K[j,k]$ je hodnota na souřadnicích (j,k). $I[m-j, n-k]$ je hodnota na souřadnicích (m - j, n - k).



Obrázek 2.7: Ukázka konvolučního operátoru [2]



Obrázek 2.8: Ukázka techniky pooling [3]

První pokusy s konvoluční sítí byly provedeny už koncem 80. let min. století (rozpoznávání ručně psaných číslic a model LeNet [Le +89]). V současné době se používají obrovské hluboké konvoluční sítě, které mají miliony parametrů pro různé klasifikační úlohy.

Mezi nejvýznamnější zástupce patří AlexNet [KSH12], GoogLeNet [Sze+14], ResNet [He+15] nebo EfficientNet [TL19], který je použit v této práci.

[2] Obrázek použit z `medium.com/analytics-vidhya/implementing-kernel-filter-convolutional-in-your-own-a6146d87d791`

[3] Obrázek použit z `epynn.net/Pooling.html`

2.3 Klasifikace skenovaných dokumentů

V současné době je velká skupina dokumentů pouze ve formě skenovaného obrázku. V takovém případě je nutné pro účinnou klasifikace rozpoznat a extrahovat text. K tomu lze použít metody optického rozpoznávání znaků (optical character recognition – OCR). Problematika OCR včetně populárních knihoven a OCR systémů bude vysvětlena v následující kapitole.

Aby bylo možné provést klasifikaci, tak je potřeba vytvořit model. Tento model je potřeba trénovat. Pro trénování modelu je potřeba mít označenou datovou sadu. Poté, co je model natrénován, může být použit k predikci třídy nových, neoznačených dat. Například, pokud chceme vytvořit klasifikační systém pro třídění e-mailů, potřebovali bychom datovou sadu označených e-mailů. Tuto datovou sadu bychom použili k trénování klasifikačního systému, který by se pak používal k třídění nových, neoznačených e-mailů.

Účelem klasifikace je předpovědět nebo odhadnout skupinu nebo třídu, do které nový datový bod patří. Například při klasifikaci dokumentů lze klasifikační systém použít k předpovědi tématu nového dokumentu, například zda se jedná o sport, politiku nebo historii. To je užitečné, protože dokumenty lze následně automaticky třídit a uspořádat podle jejich témat.

2.4 Trénování

Techniky trénování samotného klasifikátoru lze rozdělit na dvě metody:

1. učení s učitelem (supervised)
2. učení bez učitele (unsupervised)

Při trénování klasifikátoru s učitelem je model trénován na označené množině dat, což znamená, že data jsou již zařazena do různých kategorií. Cílem tréninku je natrénovat klasifikátor tak, aby předpovídal označení nových, dosud nezaznamenaných dat na základě vzorů naučených z trénovacích dat. Trénovací data se skládají ze sady vstupních souborů a jim odpovídajících výstupních tříd. Algoritmus pak tato označená data používá k učení vzorů a vztahů v datech, které může použít ke klasifikaci nových, neviděných dat.

Při trénování klasifikátoru bez učitele je naproti tomu model trénován na neoznačeném souboru dat, což znamená, že data ještě nejsou klasifikována do různých kategorií. Cílem tohoto trénování je najít v datech vzory a vztahy a seskupit je do shluků na základě jejich podobností nebo rozdílů. Příkladem takové klasifikace je kategorizace dokumentů pro organizační účely. V tomto případě by dataset tvořily texty dokumentů a algoritmus by na základě jejich obsahu identifikoval vzory

a podobnosti mezi dokumenty a seskupil je do různých kategorií na základě jejich obsahových charakteristik. To může firmám pomoci lépe organizovat své dokumenty, zlepšit a optimalizovat přístup k informacím.

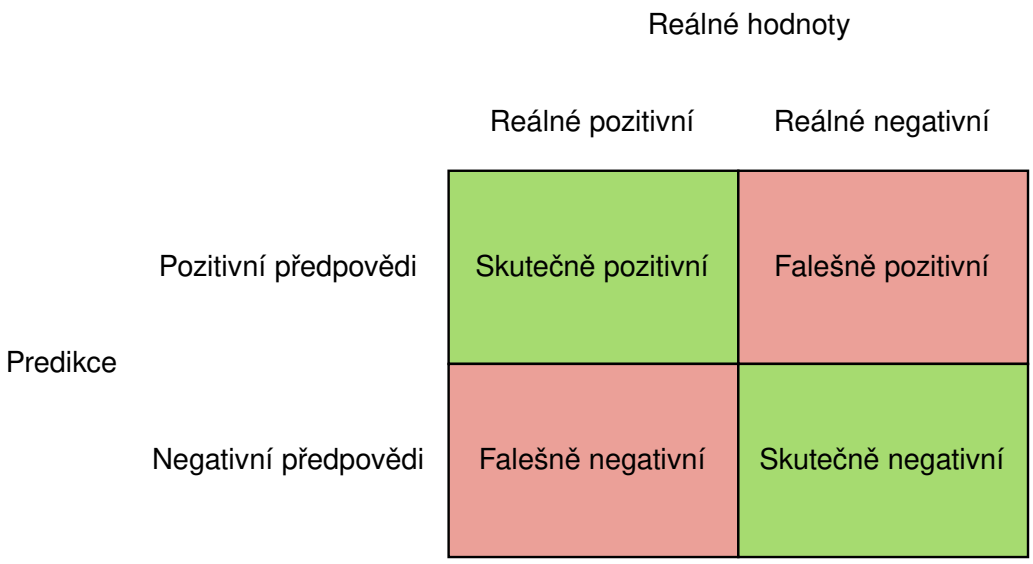
Ve zkratce se učení s učitelem používá tam, kde jsou data označena a cílem je předpovědět označení nových dat. Učení bez učitele se používá tam, kde data nejsou označena a cílem je najít vzory a seskupit data do shluků na základě podobností nebo rozdílů. V této práci bude použita metoda klasifikace s učitelem, protože trénovací data jsou roztříděna a označena.

2.5 Přesnost klasifikátoru

Po trénování je třeba ověřit funkčnost klasifikátoru. Při klasifikaci mohou nastat celkem 4 situace. Předpovězený prvek může být skutečně pozitivní (true positive), skutečně negativní (true negative), falešně pozitivní (false positive) a falešně negativní (false negative).

1. Skutečně pozitivní: Situace, kde předpovídaná hodnota je pravdivá a skutečná hodnota je také pravdivá.
2. Skutečně negativní: Situace, kde předpovídaná hodnota je nepravdivá a skutečná hodnota je také nepravdivá.
3. Falešně pozitivní: Situace, kde předpovídaná hodnota je pravdivá, ale skutečná hodnota je nepravdivá.
4. Falešně negativní: Situace, kde předpovídaná hodnota je nepravdivá, ale skutečná hodnota je pravdivá.

Tyto hodnoty lze vložit do matice záměn (confusion matrix), která je zobrazená na obrázku 2.9.



Obrázek 2.9: Confusion martix (matice záměn) [4]

Co se týče klasifikace dokumentů, tak skutečně pozitivní třída zahrnuje dokumenty, které správně odpovídají danému tématu. Naopak, skutečně negativní třída zahrnuje dokumenty, které oprávněně neodpovídají danému tématu. Falešně pozitivní třída obsahuje dokumenty, které jsou nesprávně klasifikovány jako relevantní pro dané téma. Falešně negativní třída zahrnuje dokumenty, které jsou nesprávně klasifikovány jako nerelevantní pro dané téma.

Jako další příklad můžeme uvést klasifikaci nevyžádané e-mailové pošty. Skutečně pozitivním případem je e-mail, který je správně identifikován jako spam, zatímco skutečně negativním případem je e-mail, který je správně identifikován jako legitimní. Falešně pozitivním případem by byl e-mail, který je nesprávně označen jako spam, zatímco falešně negativním případem by byl e-mail, který je nesprávně identifikován jako nevyžádaný.

2.5.1 Metriky

Přesnost klasifikačního modelu je podíl správných předpovědí provedených modelem. Jinými slovy se jedná o podíl položek, které jsou modelem správně klasifikovány. Obecně se přesnost klasifikačního modelu měří pomocí následujících metrik:

- Precision (Přesnost): Je to počet správných předpovědí dělený celkovým počtem pozitivních předpovědí. Lze spočítat dle vzorce 2.2

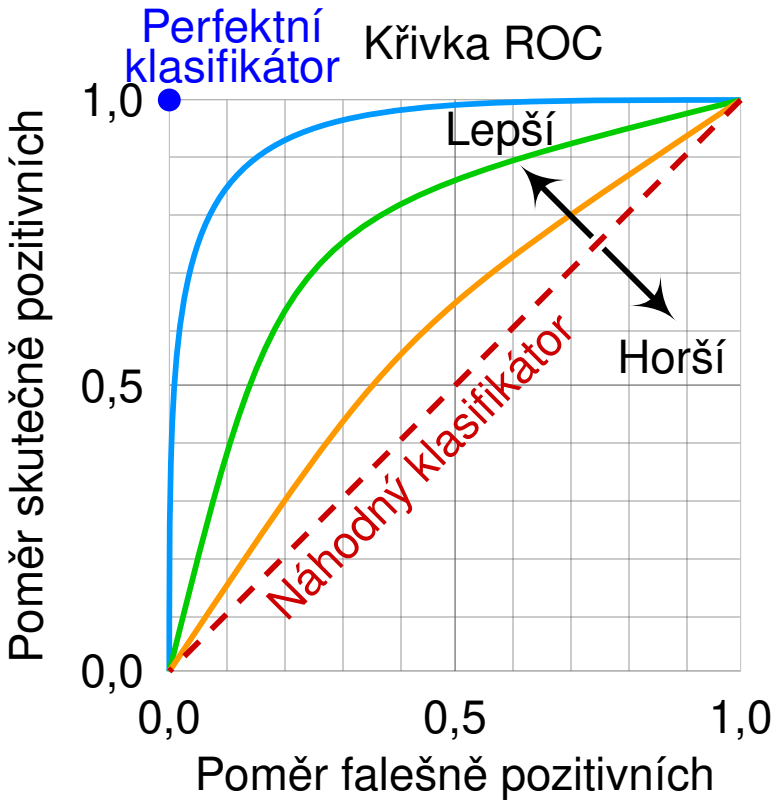
[4] Obrázek použit z `opentrafficcam.org/OTLabels/validation/metrics/`

- Recall: Vyjadřuje podíl skutečně pozitivních vůči celkovému počtu výsledků dané třídy (true positive + false negative). Lze spočítat dle vzorce 2.3
- Skóre F1: Skóre F1 je harmonický průměr přesnosti a recall. Lze spočítat dle vzorce 2.4
- Křivka ROC (Receiver Operating Characteristic): Jedná se o graf poměru skutečně pozitivních výsledků a falešně pozitivních výsledků. Čím větší je plocha pod křivkou, tím lepší je model.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.2}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.3}$$

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$



Obrázek 2.10: Křivka ROC [5]

Na obrázku 2.10 je ukázka ROC křivek. Křivka ROC je vizuální nástroj používaný k měření výkonnosti klasifikace. Na ose y je zobrazena míra skutečně pozitivních výsledků a na ose x míra falešně pozitivních výsledků.

Dokonalý klasifikátor by byl schopen správně rozlišit mezi pozitivním a negativním výsledkem. Na křivce ROC se to odráží v levém horním bodu. To znamená, že model má stoprocentní míru skutečně pozitivních výsledků (správně identifikoval všechny pozitivní případy) a nulovou míru falešně pozitivních (správně identifikoval všechny negativní případy).

Křivka ROC pro náhodný klasifikátor by byla diagonální přímka od levého spodního rohu do pravého horního rohu. Na obrázku je tato křivka označena červenou barvou. Je to proto, že náhodný klasifikátor by nebyl lepší než náhodný odhad. Klasifikátor, který si vede lépe než náhodný, bude mít křivku ROC blíže levému hornímu rohu grafu. Čím blíže je křivka ROC levému hornímu rohu, tím lépe klasifikátor rozlišuje mezi pozitivními a negativními případy.

Přesnost klasifikačního modelu lze měřit také pomocí confusion matrix neboli matice záměn (obrázek 2.9). Matice záměn je tabulka, která zobrazuje počet správných a nesprávných předpovědí pro každou třídu. Řádky tabulky představují skutečné třídy a sloupce představují předpovězené třídy. Pokud je například skutečná třída 1 a předpovězená třída 2, jedná se o chybu. Pokud je skutečná třída 1 a předpovězená třída je 1, jedná se o správnou předpověď. Přesnost modelu lze vypočítat součtem diagonálních prvků matice záměny a vydělením celkovým počtem předpovědí.

2.6 Klasifikace

V současné době se pro klasifikaci skenovaných dokumentů používají rozsáhlé neuronové sítě. Ukázalo se, že konvoluční neuronové sítě jsou pro klasifikaci skenovaných dokumentů obzvláště účinné. Důvodem, proč jsou konvoluční neuronové sítě vhodné pro klasifikaci skenovaných dokumentů je to, že velice dobře extrahují příznaky z obrázků a dokáží se ze vstupních dat naučit velmi komplexní vzory. Je to proto, že konvoluční neuronové sítě byly navrženy pro práci s obrázky a jsou tak velmi účinné při extrakci příznaků z obrázků.

Abychom mohli klasifikovat skenovaný dokument, musíme být schopni extrahovat příznaky z obrazu dokumentu. Konvoluční neuronové sítě jsou velmi účinné při extrakci příznaků z obrázků, takže jsou pro tuto úlohu dobrou volbou. Tyto modely jsou však také velmi rozsáhlé a jejich trénování vyžaduje velký výpočetní výkon. Proto není vždy praktické používat tyto modely pro klasifikaci dokumentů.

[5] Obrázek použit z `en.wikipedia.org/wiki/Receiver_operating_characteristic`

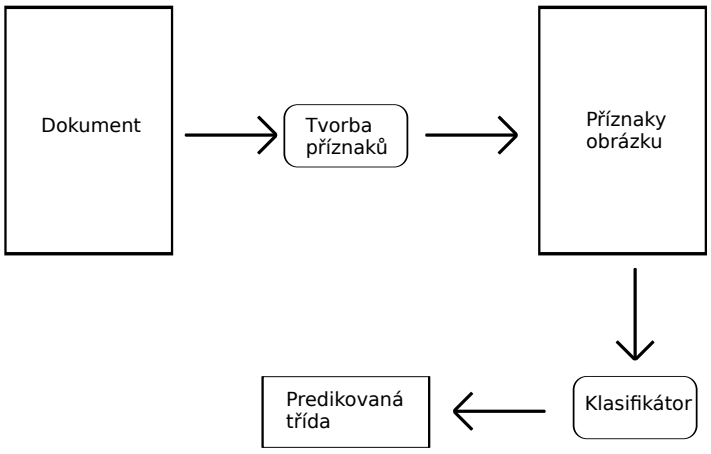
Vstupem do klasifikačního modelu je rastrový obrázek. Výstup modelu je predikovaná třída dokumentu dle kategorií uvedených výše. Na problém lze nahlížet v několika úrovních:

1. klasifikace vizuální podoby dokumentu,
2. klasifikace textu dokumentu,
3. kombinace obou metod.

Všechny úrovně a metody řešení budou dále popsány v jednotlivých sekcích.

2.6.1 Klasifikace vizuální podoby dokumentu

Samotná klasifikace obrazu, kdy se určí třída dokumentu pouze na základě extrahovaných příznaků obrázku pracuje pouze s vizuální podobou dokumentu. Z naskenovaného dokumentu se získají příznaky, jako je například histogram pixelů v obrázku, pozice textu či zda dokument obsahuje obrázky. Natrénovaný klasifikátor na jejich základě určí třídu dokumentu (viz obrázek 2.11).

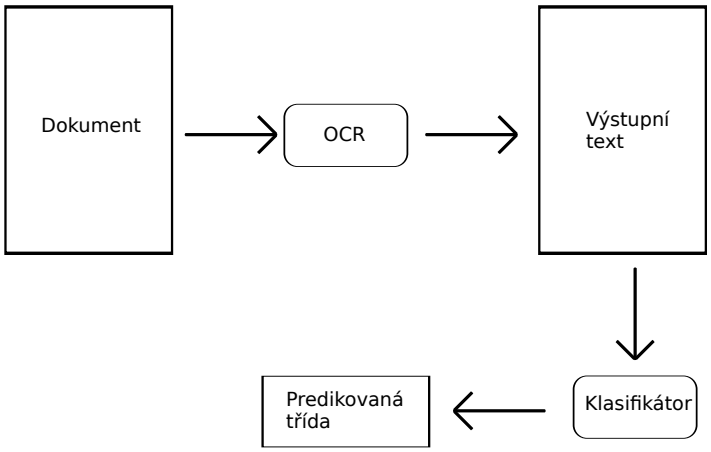


Obrázek 2.11: Ilustrace modelu pro klasifikaci příznaků obrázku

2.6.2 Klasifikace textu dokumentu

Vzhledem k tomu, že textové skeny dokumentů jsou vizuálně velmi podobné, je velmi pravděpodobné to, že klasifikátor, který využívá pouze obrázek dokumentu, bude vykazovat nízkou úspěšnost klasifikace. Proto dochází k rozpoznání textu

pomocí OCR. Text nese sémantickou informaci o dokumentu a pro korektní klasifikaci je důležitý. Rozpoznaný text lze před vložením do klasifikátoru odfiltrovat od slov, které jsou buď špatně rozpoznány (chyba OCR) nebo nepřidávají žádnou další užitečnou informaci, jako jsou například spojky nebo předložky (tzv. stop-slova). Nad textem lze provádět celou řadu technik tzv. předzpracování (např. převedení na malá písmena, opravy chyb, nahrazování čísel různými tokeny apod.) a tím lze zpravidla zvýšit úspěšnost klasifikátoru. Na obrázku 2.12 lze vidět, jak tato klasifikace probíhá.



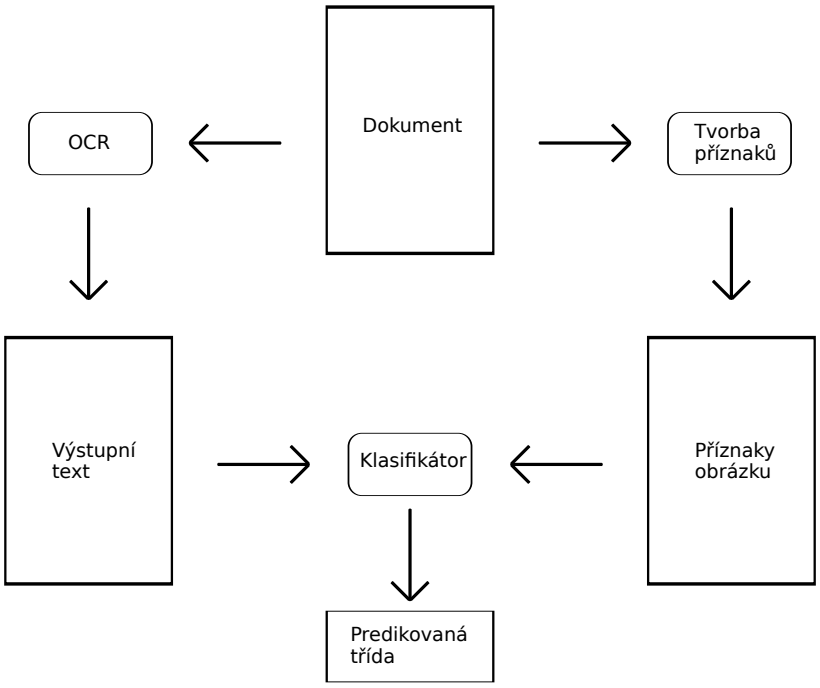
Obrázek 2.12: Ilustrace modelu pro klasifikaci textu z obrázku

2.6.3 Kombinovaná klasifikace textu a vizuální podoby dokumentu

Při klasifikaci dokumentů pouze na základě textu mohou být přehlédnuty některé vizuální podněty, které by mohly pomoci při klasifikaci. Mezi tyto vizuální prvky patří například rozvržení stránky, relativní umístění jednotlivých bloků textu a rozdíly ve stylu písma mezi nadpisy a ostatním textem. Kromě toho může text získaný pomocí optického rozpoznávání znaků (OCR) obsahovat chyby způsobené špatnou kvalitou skenování, což by mohlo dále bránit přesné klasifikaci. Začlenění vizuálních informací vedle textových údajů by proto mohlo potenciálně zlepšit proces klasifikace dokumentů.

Za tímto účelem lze vytvořit model, při němž jsou z obrazu dokumentu identifikovány a extrahovány vizuální prvky. Text je pak zpracován pomocí technologie OCR, aby se získaly textové údaje. Vizuální i textové údaje jsou následně vloženy

do klasifikačního modelu, který pak určí příslušnou třídu dokumentu (viz obrázek 2.13).



Obrázek 2.13: Ilustrace modelu pro kombinovanou klasifikaci

2.7 Modely pro zpracování textu

Zpracování textu je základní úlohou zpracování přirozeného jazyka (NLP), která zahrnuje transformaci nestrukturovaných textových dat do strukturovaného formátu, který lze snadno analyzovat stroji. Pro splnění této úlohy byly v průběhu let vyvinuty různé modely, které umožňují získávat z textových dat užitečné informace. V této části budou některé populárními modely pro zpracování textu popsány.

2.7.1 Bag-of-words

Model bag-of-words (BOW) je jednoduchý způsob reprezentace textových dat ve strojově čitelném formátu. Jedná se o reprezentaci textu, kde je každé slovo reprezentováno celým číslem [Qad19]. Toto celé číslo může být index slova ve slovníku, počet slov v dokumentu nebo jiná transformace slova. Zahrnuje vytvoření vektoru fixní velikosti, který reprezentuje výskyt každého slova v daném dokumentu. Tento vektor lze vytvořit spočítáním četnosti výskytu jednotlivých slov v dokumentu nebo

transformací slov na indexy, které představují jejich pozici v předem definovaném slovníku.

Ačkoli je model bag-of-words jednoduchým a efektivním způsobem reprezentace textových dat, má několik omezení, která mohou ovlivnit kvalitu modelů strojového učení postavených na této reprezentaci. Jednou z hlavních nevýhod tohoto modelu je, že nezachycuje kontextové informace textu. Tím, že model ignoruje pořadí slov v dokumentu, ztrácí důležité informace o tom, jak spolu slova souvisejí. To může vést k problémům, jako je nejednoznačnost a nesprávná interpretace textu. V modelu bag-of-words mohou například existovat dvě věty, které by byly naprosto identické, protože obsahují stejná slova, i když mají zcela odlišný význam. Bag-or-words říká pouze to, jaká slova se v dokumentu vyskytují, ne kde se vyskytla. Nezáleží tedy na pořadí nalezených slov.

Dalším omezením modelu bag-of-words je, že nebere v úvahu význam samotných slov. Model by například považoval slova „koruna“ s významem koruny stromu a „koruna“ s významem měny za naprosto totožná, i přestože mají zcela odlišný význam. To může vést k problémům v aplikacích, jako je analýza sentimentu, kde je význam slov pro přesnou klasifikaci klíčový.

2.7.2 Word2vec

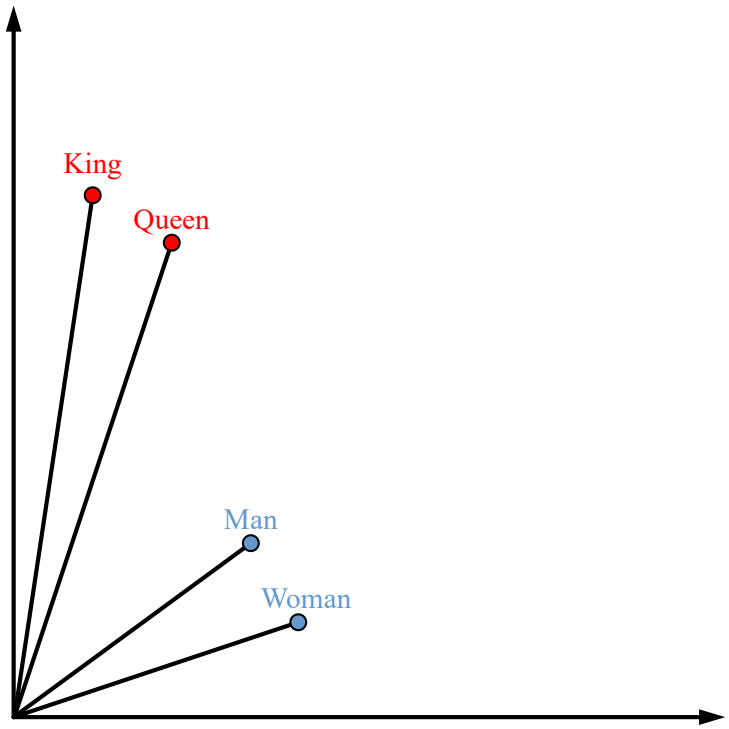
Word2vec je technika používaná zpracování přirozeného jazyka, která se používá oblasti počítačové lingvistiky. Poprvé byla představena v zásadním článku Tomase Mikolova a jeho kolegů ze společnosti Google v roce 2013 [Mik+13]. Základní myšlenkou Word2vec je vytvoření vektorových reprezentací slov, známých také jako word embeddings, které zachycují význam a kontext slov v daném souboru textu.

Klíčovou inovací Word2vec je použití architektury neuronové sítě, která je speciálně navržena tak, aby se tyto vektorové reprezentace učila bez učitele. Model je trénován na velkém objemu textových dat, jako jsou zpravodajské články nebo stránky Wikipedie, a učí se předpovídat kontext, ve kterém se dané slovo vyskytuje. Toho se dosáhne tak, že se model naučí předpovídat pravděpodobnost výskytu slova vzhledem k jeho okolním slovům (známý jako model „skip-gram“) nebo naopak pravděpodobnost výskytu okolních slov vzhledem k středovému slovu (známý jako model „continuous bag of words“ neboli CBOW).

Slova, která jsou si sémanticky podobná, například „auto“ a „vozidlo“ nebo „strom“ a „les“, budou mít v tomto prostoru vektory, které jsou blízko sebe. Naopak slova, která si nejsou sémanticky podobná, jako například „radar“ a „mrakodrap“ nebo „stůl“ a „koloběžka“, budou mít vektory daleko od sebe. Pokud bychom chtěli vektorový prostor vizualizovat, tak to lze vyřešit pomocí technik redukce dimenzionality pomocí například t-SNE [23b]. To může pomoci s odhalením shluků slov,

jež odpovídají různým sémantickým kategoriím, jako jsou zvířata, potraviny nebo profese.

Jednou z nejužitečnějších aplikací Word2vec je vyhledávání podobných slov nebo frází. To se provádí výpočtem kosinové podobnosti mezi vektory dvou slov, která měří úhel mezi oběma vektory. Pokud se úhel blíží nule, jsou si oba vektory podobné, zatímco pokud se úhel blíží 90 stupňům, tak tyto vektory si nejsou podobné. Například vektory pro slova „muž“ a „žena“ budou mít kosinovou podobnost blízkou 1, zatímco vektory pro slova „muž“ a „stůl“ budou mít kosinusovou podobnost blízkou 0. Na obrázku 2.14 je zobrazen vektorový prostor, v kterém jsou slova King (Král), Queen (Královna), Man (Muž) a Woman (Žena). Slova King a Queen jsou ve vektorovém prostoru blízko sebe. Také slova Man a Woman jsou ve vektorovém prostoru blízko sebe.



Obrázek 2.14: Podobná slova [6]

Jak již bylo řečeno, zjištění podobnosti dvou slov lze vypočítat pomocí kosinové podobnosti vektorů. Kosinová podobnost je mírou toho, jak blízko k sobě mají dva vektory, a je definována jako skalární součin dvou vektorů dělený součinem jejich

[6] Obrázek použit z wikidocs.net/166646?fbclid=IwAR1cwoTNXocPtiv-t4hnIX4vB-776aYQmyeeCJBd3eEugtFVmsunHqTQ7QE

délek. Jsou-li dány dva vektory A a B, tak kosinová podobnost $\cos(\Theta)$, kde A_i a B_i jsou složky vektoru A a B, je znázorněna pomocí skalárního součinu a velikosti takto:

$$\cos(\theta) = \frac{AB}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2}\sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{2.5}$$

Word2vec lze také použít k výpočtu podobnosti mezi dvěma frázemi nebo větami. To se provádí zprůměrováním vektorů jednotlivých slov v každé frázi a následným výpočtem kosinové podobnosti mezi výslednými vektory. Tato technika, známá jako „sentence embedding“, je účinná v řadě úloh zpracování přirozeného jazyka, jako je analýza a klasifikace textu.

Kromě vyhledávání podobných slov a frází lze Word2vec použít také ke shlukování sémanticky příbuzných slov. K tomu slouží shlukovací algoritmy, jako je například k-means, které seskupují slova, jež jsou si ve vektorovém prostoru nejblíže. Tyto shluky pak mohou být použity k identifikaci různých sémantických kategorií nebo témat v rámci korpusu textu, jako je sport či politika.

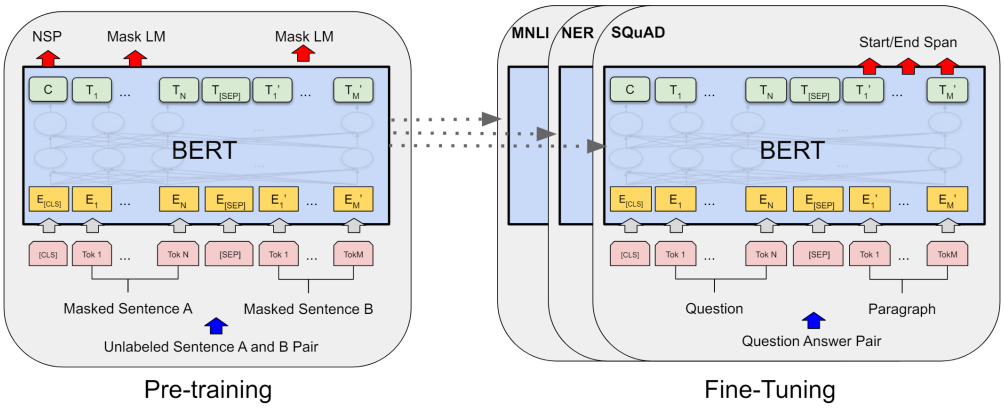
Využití pro klasifikaci dokumentů spočívá v použití modelu Word2vec k vytvoření vektoru pro každé slovo v dokumentu. Tyto slovní vektory lze poté zprůměrovat a vytvořit tak jediný vektor pro daný dokument. Tento vektor pak lze použít jako příznakový vektor pro klasifikovaný dokument.

2.7.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) je state-of-the-art model pro reprezentaci jazyka, který byl vyvinutý společností Google [Dev+19] v roce 2019. BERT patří do modelů typu Transformer. Tyto modely zavedly mechanismy pozornosti pro efektivní zachycení kontextových informací. Konkrétně model BERT zavedl obousměrnost (bidirectionality). BERT je navržen tak, aby předtrénoval hluboké obousměrné reprezentace z neoznačeného textu spojením levého i pravého kontextu ve všech vrstvách. Díky tomu lze předtrénovaný model BERT jemně doladit pomocí jedné další výstupní vrstvy a vytvořit tak modely pro širokou škálu úloh, jako je například zodpovídání otázek, aniž by bylo nutné provádět zásadní úpravy architektury specifické pro danou úlohu.

BERT dokáže zachytit kontext z obou směrů současně. To znamená, že při kódování určitého slova bere v úvahu všechna slova ve větě, což vede k lepšímu porozumění kontextu zpracovávaného textu, což výrazně přispívá k výkonu modelu v různých úlohách zpracování přirozeného jazyka.

Na obrázku 2.15 je ukázáno trénování a ladění modelu BERT.



Obrázek 2.15: Model BERT [7]

2.8 Modely pro zpracování obrázků

Zpracování obrazu je základním úkolem při získávání smysluplných informací. Jednou z nejvýznamnějších aplikací zpracování obrazu je klasifikace obrazu, kdy je cílem přiřadit obrazu jednu nebo více tříd. K dosažení tohoto cíle vyvinuli výzkumníci celou řadu modelů, které jsou speciálně určeny pro zpracování obrazu. V této sekci budou popsány některé modely sloužící pro klasifikaci obrázků.

2.8.1 EfficientNet

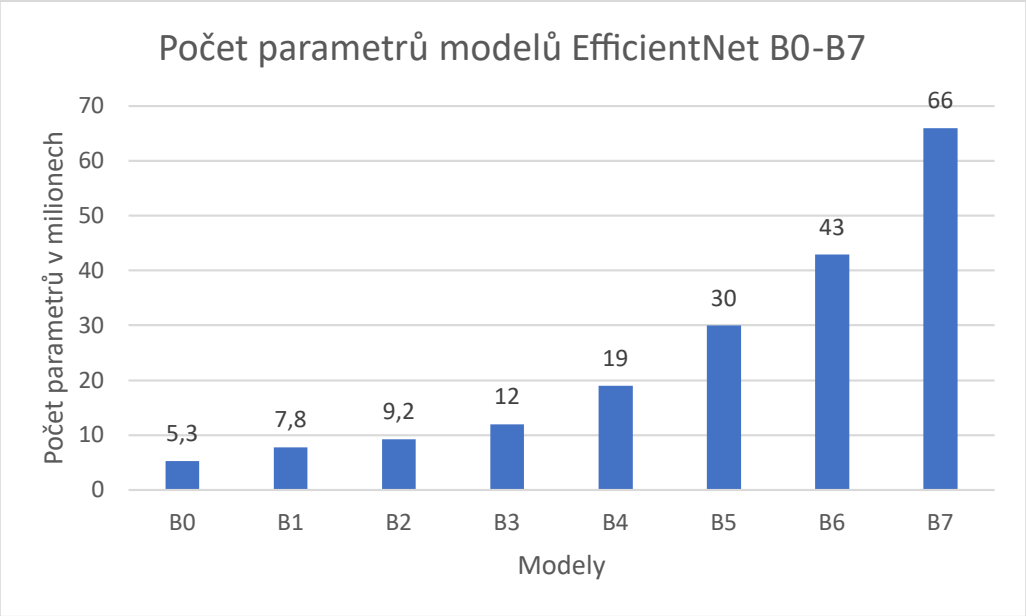
Autoři článku *Improving accuracy and speeding up document image classification through parallel systems* [Fer+20] použili model *EfficientNet* a dosáhli výsledků téměř 90 % úspěšnosti v kombinaci s textovým model BERT na datasetu Tobacco-3482.

EfficientNet je model konvoluční neuronové sítě (CNN). Modely EfficientNet byly představeny ve výzkumné studii, kterou společnost Google zveřejnila v roce 2019 [TL19].

Klíčovou inovací v modelu EfficientNet je použití nového škálování, které rovnoměrně škáluje dimenze hloubky, šířky a rozlišení sítě pomocí „compound scaling“. To umožňuje modelům dosáhnout vyšší přesnosti při zachování relativně nízkého počtu parametrů a výpočetních nákladů celé sítě. Oproti ostatním hlubokým konvolučním sítím (např. ResNet, který má několik desítek milionů parametrů), je EfficientNet několikrát menší (pouze jednotky milionů parametrů) a rychlejší. Síť EfficientNet přesto dosahuje srovnatelných nebo dokonce lepších výsledků než jiné konvoluční sítě.

[7] Obrázek použit z [Dev+19]

EfficientNet zahrnuje několik modelů. Jedná se o EfficientNet B0 až B7, které se liší velikostí a výpočetními náklady. Větší modely, jako je B7, jsou přesnější, ale také výpočetně náročnější a vyžadují delší trénování. Na druhou stranu menší modely, jako je B0, mají méně parametrů a jejich trénování a využívání je rychlejší. Graf 2.16 ukazuje počet parametrů pro modely EfficientNet B0 až B7.



Obrázek 2.16: Porovnání počtu parametrů modelů EfficientNet

2.8.2 AlexNet

AlexNet je architektura hluboké konvoluční neuronové sítě, kterou v roce 2012 vyvinuli Alex Krizhevsky, Ilya Sutskever a Geoffrey Hinton [KSH12]. AlexNet se skládá z celkem osmi vrstev, z nichž prvních pět jsou konvoluční vrstvy a zbývající tři jsou plně propojené vrstvy.

Vstupem do sítě AlexNet je barevný obrázek RGB o rozměrech 256 × 256, který je zpracováván v řadě konvolučních a poolingových vrstev. Konvoluční vrstvy se učí filtry, které ze vstupního obrazu získávají stále složitější rysy, zatímco poolingové vrstvy snižují dimenzionalitu a pomáhají síti zvýšit odolnost vůči malým změnám na vstupu.

Výstupem AlexNet je vektor pevné velikosti 1000. Ve vektoru jsou pravděpodobnosti tříd pro každou z 1000 kategorií objektů v datové sadě ImageNet, což umožňuje určit do které kategorie objekt na obrázku patří.

2.8.3 GoogLeNet

GoogLeNet je architektura hluboké konvoluční neuronové sítě (CNN), kterou v roce 2014 navrhl tým výzkumníků společnosti Google [Sze+14]. Skládá se z celkem 22 vrstev.

Klíčovou inovací GoogLeNet je použití nové architektury nazvané Inception. Tato architektura se skládá z několika vrstev konvolučních filtrů různých velikostí, po nichž následuje pooling a spojování. Tato architektura umožňuje síti efektivně zachytit různé rysy obrazu.

2.8.4 ResNet

ResNet neboli Residual Network, je architektura hluboké neuronové sítě. Jejími autory jsou výzkumníci ze společnosti Microsoft Research Kaiming He, Xiangyu Zhang, Shaoqing Ren a Jian Sun kteří jí představili v roce 2015 [He+15].

Hlavní myšlenkou ResNet je využití zkratkového spojení (residual connection) mezi vrstvami, aby bylo umožněno efektivnější trénování velmi hlubokých sítí. Tyto hluboké sítě se obvykle skládají ze stovek nebo dokonce tisíců vrstev. Tato zkratková spojení jsou navržena tak, aby umožnovala přímé sloučení výstupů vrstev s výstupy hlubších vrstev jednoduchým sčítáním prvků, čímž se efektivně obcházejí některé vrstvy sítě. Jinými slovy jde o přímé spojení vstupu vrstvy s s jejím výstupem. Tento přístup umožňuje sítím ResNet překonat problém mizejícího gradientu, který se může vyskytnout u hlubokých sítí, kdy se gradienty vzhledem k parametrům dřívějších vrstev stávají velmi malými, což ztěžuje efektivní aktualizaci těchto vrstev během trénování. Tím, že sítě ResNet umožňují přímý tok informací z dřívějších vrstev do hlubších vrstev, usnadňují efektivní šíření gradientů v síti a umožňují trénovat velmi hluboké sítě s vyšší přesností a rychlostí konvergence.

Další výhodou sítě ResNet je to, že mohou získat vyšší přesnost s větší hloubkou sítě.

3 OCR

OCR (Optical Character Recognition) je technologie používaná k extrakci textu například z digitálních obrázků, naskenovaných dokumentů, fotografií či snímků obrazovky [SBB12]. OCR lze použít k převodu tištěného nebo ručně psaného textu na upravitelný a prohledávatelný digitální text, který lze ukládat, zpracovávat a analyzovat pomocí různých softwarových aplikací.

Proces OCR obvykle zahrnuje dvě hlavní fáze:

1. Preprocessing (Předzpracování) – filtrování
2. Rozpoznávání – převod znaků z obrázku na text

Ve fázi předzpracování se obraz analyzuje a zpracovává, aby se zlepšila kvalita obrazu a odstranil se šum na pozadí, zkreslení nebo jiné nežádoucí prvky, které by mohly narušit proces rozpoznávání. Fáze rozpoznávání zahrnuje převod obrazu na text pomocí pokročilých algoritmů a technik strojového učení. Software OCR analyzuje obraz, identifikuje jednotlivé znaky nebo slova a poté je mapuje na odpovídající digitální textové ekvivalenty. I přes značný pokrok v technologii OCR, tak tato technologie stále není dokonalá. Software OCR může mít potíže se špatně naskenovanými obrázky, rukopisem a složitým rozvržením. Přesnost OCR může být navíc ovlivněna typem písma, velikostí písma a rozlišením obrazu.

K dispozici je několik programů OCR, od základních až po pokročilé, ale za jeden z nejpřesnějších a nejspolehlivějších OCR je považován Tesseract. Mezi další možnosti patří například Goodle Cloud Vision OCR či Kraken, které jsou popsány níže.

3.1 Tesseract

Pro OCR v této práci byl použit software Tesseract. Tesseract je nástroj pro optické rozpoznávání znaků. Tesseract je k dispozici jako bezplatná knihovna s otevřeným zdrojovým kódem (open source) a patří mezi nejpopulárnější a nejvíce používané

software pro rozpoznávání znaků [23c]. Tesseract je velmi přesný a ve své současné verzi má podporu celé řady jazyků a písem.

Jednou z klíčových vlastností Tesseract je schopnost trénovat vlastní model. To znamená, že uživatelé mohou software trénovat na rozpoznávání konkrétních písem, znaků nebo jazyků, které nemusí být podporovány hned po instalaci. Poskytnutím trénovací sady dat, která obsahuje příklady znaků, jež mají být rozpoznány, může Tesseract přizpůsobit své rozpoznávací algoritmy tak, aby poskytovaly přesnější výsledky. Tesseract je natrénován na různých sadách dat a je neustále aktualizován o nové sady dat a jazyky. Systém je navržen tak, aby pracoval s různými typy vstupů, včetně obrázků, souborů PDF a TIFF. Na výstupu může být text v různých formátech, včetně prostého textu, HTML, XML i PDF.

Pro rozpoznání textu v obrázku Tesseract nejprve obrázek předzpracuje, aby zlepšil kvalitu textu. To zahrnuje odstranění šumu, normalizaci obrazu a binarizaci obrazu. Tyto techniky se používají k zajištění toho, aby byl text jasný a dobře definovaný, než Tesseract zahájí svůj proces rozpoznávání.

Jakmile je obraz předzpracován, použije Tesseract sadu pravidel pro identifikaci znaků v obraze. Může například hledat znaky, které jsou blízko sebe, nebo znaky, které mají stejný tvar. Tesseract bere v úvahu také kontext textu, například použitý jazyk nebo typ zpracovávaného dokumentu.

K určení nejpravděpodobnějšího pořadí znaků používá Tesseract jazykový model, který je založen na trénovací sadě dat. Tato datová sada obsahuje správný pravopis slov a tvary jednotlivých znaků a slouží k vytvoření statistického modelu jazykových vzorů. Tesseract používá tento jazykový model k nalezení posloupnosti znaků, která je s největší pravděpodobností správná, a rozpoznaný text vypíše jako řetězec znaků ve zvoleném tvaru.

Existuje několik důvodů, proč byl pro klasifikaci dokumentů vybrán Tesseract. Za prvé, jedná se o nástroj s otevřeným zdrojovým kódem (open source), což znamená, že je zdarma k použití a nejsou s ním spojeny žádné licenční náklady. Za druhé, Tesseract je velmi přesný a zvládá řadu různých jazyků. Další jeho výhoda je v jednoduchosti jeho použití. A nakonec, Tesseract je velmi rychlý, což je důležité při práci s velkými objemy dat.

3.1.1 Nastavení Tesseract OCR

Tesseract pro rozpoznávání textu pomocí OCR má různá nastavení. Jedno z nejdůležitějších nastavení je způsob segmentace stránky (PSM - Page segmentation mode), která se OCR předá. Nastavení segmentace stránky změní to, jakým druh textu bude Tesseract předpokládat na jeho vstupu. Toto je seznam nastavení PSM:

- PSM 0 – Pouze OSD (Orientation and script detection)

- PSM 1 – Automatická segmentace stránky s OSD
- PSM 2 – Automatická segmentace stránky bez OSD a OCR
- PSM 3 – Plně automatická segmentace stránky bez OSD
- PSM 4 – Předpoklad jednoho sloupce textu s různou velikostí
- PSM 5 – Předpoklad jednoho bloku textu s vodorovným textem
- PSM 6 – Předpoklad jednoho bloku textu
- PSM 7 – Zacházení s obrázkem jako jeden řádek textu
- PSM 8 – Zacházení s obrázkem jako s jedním slovem
- PSM 9 – Zacházení s obrázkem jako s jedním slovem v kroužku
- PSM 10 – Zacházení s obrázkem jako s jedním znakem
- PSM 11 – Řídký text. nalezení co nejvíce textu bez žádného určeného pořadí
- PSM 12 – Řídký text s OSD
- PSM 13 – Předpoklad, že obrázek je jeden řádek textu bez použití speciálních vlastností Tesseractu

OSD pouze prozkoumá obraz a namísto textu vrátí úhel, pod kterým je text napsaný a spolehlivost výsledku (confidence)

3.2 Google Cloud Vision OCR

Google Cloud Vision OCR je výkonná cloudová služba OCR (optické rozpoznávání znaků), která dokáže rozpoznávat text na obrázcích. Tuto službu nabízí společnost Google a je postavena na pokročilých algoritmech počítačového vidění a technologiích strojového učení společnosti Google. Podporuje více jazyků a dokáže rozpoznat text v obrázcích bez ohledu na velikost nebo styl písma, včetně ručně psaného textu.

Tato služba má také schopnost identifikovat objekty, obličeje a další prvky na obrázcích. To může být velmi užitečné pro podniky a organizace, které potřebují analyzovat obrázky z hlediska obsahu nebo identifikovat konkrétní objekty na obrázcích. Jednou z klíčových výhod Google Cloud Vision OCR je snadná integrace s dalšími službami Google Cloud. Poskytuje rozhraní API (Application Programming Interfaces) pro použití v různých programovacích jazycích, jako jsou Python, Java a Ruby. Díky tomu mohou vývojáři snadno začlenit Google Cloud Vision OCR do svých aplikací a pracovních postupů.

Další výhodou Google Cloud Vision OCR je jeho přesnost. K rozpoznávání textu a objektů na obrázcích využívá pokročilé algoritmy počítačového vidění a technologie strojového učení společnosti Google. To znamená, že dokáže přesně identifikovat i malé detaily v obrázcích, což může být v mnoha aplikacích důležité.

3.3 Kraken

Kraken je open source systém OCR, který byl navržen tak, aby poskytoval efektivní a účinné řešení pro rozpoznávání tištěných a ručně psaných textů. Tento software je volně k použití pro komerční i nekomerční účely pod licencí Apache, verze 2.0. Z githubu pro Kraken lze říci, že software je neustále vyvíjen a aktualizován o nové funkce [23a].

Webová stránka tohoto softwaru [22] zmiňuje jako jednu z hlavních funkcí systému Kraken plně trénovatelnou analýzu rozložení a funkce rozpoznávání znaků, která umožňuje rozpoznávat text v široké škále formátů. Navíc podporuje písmo zprava doleva, BiDi (obousměrný) a shora dolů, takže je ideální volbou pro vícejazyčné aplikace OCR, kde se tyto druhy textů mohou vyskytovat.

Software dále dokáže vytvořit výstup rozpoznaného textu v několika různých formátech, jako je ALTO, PageXML, abbyyXML a hOCR. Tato flexibilita zajišťuje, že výstup lze snadno integrovat do jiných softwarových aplikací.

Jako jedna z dalších funkcí v softwaru Kraken je ohraničení slov a znaků, která umí identifikovat počátky a konce slov a znaků, což umožňuje efektivnější extrakci textu. Podporuje také rozpoznávání více písem, což znamená, že dokáže rozpoznat text v různých písmech, včetně ručně psaných.

Další užitečnou funkcí aplikace Kraken je veřejné úložiště modelů Kraken. To znamená, že uživatelé mohou sdílet a stahovat předtrénované modely, což značně usnadňuje práci s OCR.

Kraken také nabízí variabilní architekturu rozpoznávací sítě, která umožňuje přizpůsobit systém OCR jejich specifickým požadavkům. Díky této flexibilitě je ideální volbou pro širokou škálu aplikací OCR.

Dataset 4

K experimentům bude použit dataset Tobacco-3482. Dataset obsahuje 10 tříd, rozdělených do složek s obrázky. Každá složka představuje jiný typ dokumentu, například ADVE pro inzeráty, Email pro e-maily atd. Dokumenty jsou naskenované, což znamená, že jsou v obrazovém formátu a nelze je prohledávat ani zpracovávat pomocí tradičních nástrojů pro analýzu textu. Každý obrázek je černobílý sken dokumentu. Každý obrázek dokumentu je v jiném rozlišení. Dataset obsahuje celkem 3482 souborů (skenů dokumentů v podobě obrázků). Rozdělení dokumentů mezi třídy se pohybuje od 120 obrázků pro kategorii Resume po 620 obrázků pro kategorii Memo. Každý dokument patří do právě jedné třídy. Třída každého dokumentu je označena umístěním dokumentu ve složce. Rozložení kategorií lze vidět na obrázku 4.1.

Dataset je nevyvážený, což znamená, že existuje výrazný rozdíl v počtu vzorků pro každou třídu. Jinými slovy každá složka obsahuje jiný počet obrázků. Nevyváženost datasetu může vést k problémům při trénování klasifikátoru. Nevyvážená datová sada může způsobit, že klasifikátor bude zaujatý vůči převažující třídě a nemusí být schopen se tím pádem dobře naučit vlastnosti méně početné třídy. To může vést ke špatnému výkonu v klasifikačních úlohách pro méně početnou třídu. Není rovněž k dispozici rozdělení na trénovací, testovací a ani na validační část. Proto musela být vytvořena vlastní testovací, trénovací a validační část. Trénovací sada obrázků byla vytvořena náhodným výběrem 70 % dostupných skenů. Následně do testovací části bylo náhodně vybráno 20 % obrázků. Nakonec všechny zbylé obrázky (10 %) byly použity na validační část.

Skenované dokumenty jsou převážně tištěným písmem, nicméně kategorie Note (poznámky) je z velké části tvořena ručně psaným textem. Všechny skenované dokumenty z datasetu jsou v anglickém jazyce. Navzdory těmto menším problémům poskytuje dataset cenný zdroj dat pro trénování či testování a pro tvorbu potřebného klasifikátoru pro skenované dokumenty.

Autor práce *Light-Weighted CNN for Text Classification* s tímto datasetem dokázal za pomocí pouze textového klasifikátoru mít úspěšnost 46 % [Yad20]. Oproti tomu autoři práce *Efficient Document Image Classification Using Region-Based Graph Neural Network* dokázali mít větší úspěšnost. A to celkem 82,3 % [Man+21] s využitím

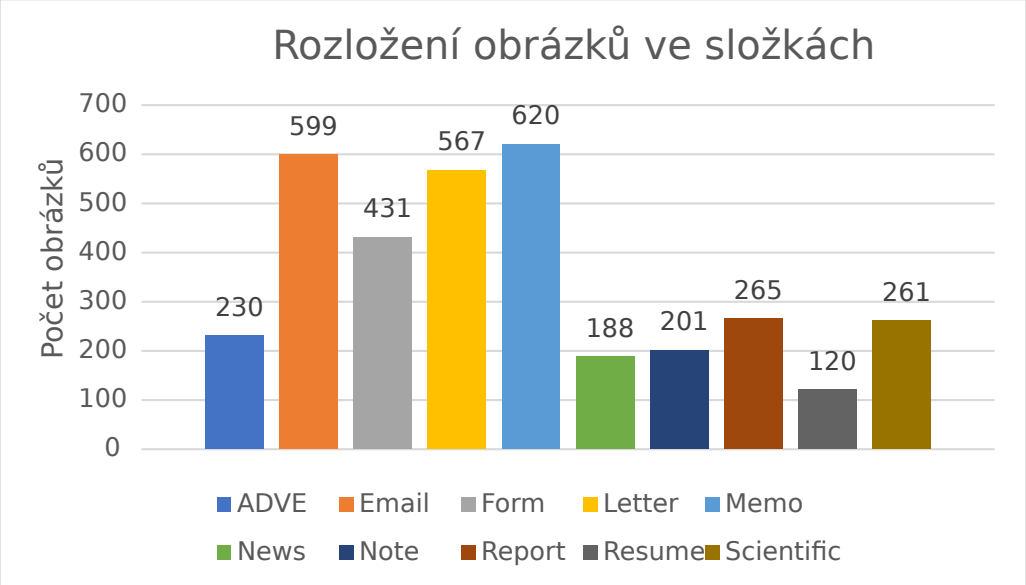
obrázkové a textové informace na datasetu Tobacco-3482.

4.1 Kategorie dokumentů

Jak již bylo zmíněno dříve, tak dataset, který byl použit, obsahuje celkem 10 složek s celkem 3482 dokumenty. Seznam, který je níže, ukazuje jména složek a jejich obsah. Počty souborů v jednotlivých složkách jsou následně zobrazeny v grafu 4.1. Popis jednotlivých složek je v tabulce 4.1

Label	**Popis**
ADVE	Složka obsahuje skeny reklam.
Email	Složka obsahuje skeny e-mailů.
Form	Složka obsahuje skeny formulářů.
Letter	Složka obsahuje skeny dopisů.
Memo	Složka obsahuje skeny memorandum.
News	Složka obsahuje skeny novinových stran.
Note	Složka obsahuje skeny poznámek.
Report	Složka obsahuje dokumenty různých zpráv.
Resume	Složka obsahuje skeny životopisů.
Scientific	Složka obsahuje skeny vědeckých zpráv.

Tabulka 4.1: Tabulka složek



Obrázek 4.1: Rozložení kategorií dokumentů v datasetu Tobacco-3482

Výše uvedený graf 4.1 vizuálně znázorňuje rozložení naskenovaných obrazů dokumentů v různých složkách. Graf zobrazuje počet obrázků v jednotlivých složkách, přičemž svislá osa představuje počet obrázků a vodorovná osa jednotlivé složky.

Dataset je nevyvážený, protože některé složky obsahují výrazně více obrázků dokumentů než jiné. Celková nevyváženost datasetu použitého v této práci může mít důsledky pro přesnost a spolehlivost jakékoli analýzy nebo modelu, který je na něm postaven. Nevyvážený dataset může vést ke zkresleným výsledkům, protože model bude trénován na neúměrném počtu obrázků z některých složek ve srovnání s jinými. To může také ztížit identifikaci vzorců nebo trendů v datech, zejména pokud je počet snímků v některých složkách příliš nízký na to, aby bylo možné učinit smysluplné závěry. Nevyváženost datasetu může ovlivnit některé metriky jako je například presicion, recall a F1.

V případě, kdy máme v datasetu výraznou nerovnováhu mezi třídami, může model dosáhnout zdánlivě vysoké úrovně přesnosti pouhým opakováním nejčastější třídy, přičemž menšinové třídy model zanedbává. Takový přístup může vést ke zvýšení metriky precision a vytvářet zdánlivě vysoké schopnosti modelu, zatímco ve skutečnosti zanedbává identifikaci menšinových tříd. Tím pádem může být obtížné posoudit skutečnou úroveň přesnosti a efektivity modelu při správném identifikování všech tříd v datasetu, což je zásadní pro jeho spolehlivost a využitelnost v reálných aplikacích.

Experimenty 5

V této části jsou popsány experimenty, které se prováděly v této práci. Cílem experimentů provedených v této práci bylo prozkoumat efektivitu použití textových i obrazových vstupů pro klasifikaci dokumentů. V první části této kapitoly je vysvětleno, proč byl využit model BERT a model EfficientNet. V následující části této kapitoly jsou popsané experimenty s textem dokumentu. První sada experimentů se zaměřila na použití textu dokumentu ke klasifikaci dokumentů do různých kategorií. Další sada experimentů se zaměřila na vizuální prvek dokumentu. K tomu byla využita sít EfficientNet. Zde jsou popsány experimenty s obrázkovým vstupem do klasifikátoru. Poslední sada experimentů se zaměřila na kombinaci textových a obrazových vstupů pro klasifikaci dokumentů. Jsou zde popsané experimenty, které kombinují jak textovou část, tak i vizuální část dokumentu.

5.1 Finální výběr modelů pro experimenty

Pro experimety byl použit model BERT a síť EfficientNet. Tyto nástroje měly dle předběžných testů nejlepší výsledky. Model BERT byl vybrán kvůli jeho přesnosti a schopnosti zachycovat kontext a význam textu. Síť EfficientNet je state-of-the-art hluboká neuronová síť a byla využita hlavně kvůli její přesnosti a nízkému počtu parametrů, které je potřeba trénovat.

Díky těmto nástrojům lze dosáhnout vysoké úspěšnosti klasifikace a výpočetně nenáročného použití klasifikace.

5.2 Textový Preprocessing

V této části je uvedený popis různých experimentů provedených s využitím modelu BERT pro klasifikaci dokumentů. Zahrnuje to použití OCR Tesseract pro extrakci textu z obrázků dokumentů. Ačkoliv se tento přístup ukázal jako účinný při získávání textu potřebného pro klasifikaci, je důležité poznamenat, že text extrahovaný pomocí OCR může obsahovat chyby, které mohou negativně ovlivnit přesnost klasifikace.

Aby se chybně rozpoznaný text zohlednil, bylo vyzkoušeno odfiltrování chyb a filtrování slov z extrahovaného textu. Naše experimenty však ukázaly, že takové filtrování nevedlo k nejlepšímu možnému výsledku pro klasifikaci.

Stojí za zmínku, že použití OCR představuje pro úlohy klasifikace dokumentů spoustu výzev. Kvalita extrahovaného textu je do značné míry závislá na kvalitě vstupních obrázků a chyby v extrakci textu mohou mít významný dopad na výkonnost finálního modelu. Experimenty popsané v sekcích 5.2.2, 5.2.3 a 5.2.4 nicméně ukazují, že BERT při zvládání těchto výzev je robustní a spolehlivý nástroj pro úlohy klasifikace dokumentů pomocí textové informace.

5.2.1 Nastavení PSM

Tesseract nabízí řadu nastavení režimu segmentace stránky (PSM), která umožňují definovat rozložení a strukturu textu, který se bude zpracovávat. Možnosti PSM sahají od jednoduchých předpokladů o textu, například zda se jedná o jedno slovo nebo jednotný blok textu, až po složitější předpoklady o rozložení textu. Toto nastavení je velmi důležité. Se špatným nastavením může Tesseract vracet naprosto nesprávná a nepoužitelná data. Je tedy důležité toto nastavení vyzkoušet a nastavit ho na správnou hodnotu.

Po provádění testů různých nastavení parametru PSM se pro klasifikaci textu ukázalo, že nejlepší volbou je nastavení PSM 3. V tomto případě Tesseract je plně automatizovaný a s textem si tak poradí sám. Tesseract s jiným nastavením PSM vracel text se spoustou chyb a v určitých případech nedokázal z textu dokumentu rozpoznat nic. S nastavením PSM 3 lze přesněji klasifikovat a kategorizovat text.

5.2.2 Filtrace textu pomocí slovníku

V těchto experimentech byl použit slovník slov, pomocí kterého se filtroval text, který byl získán pomocí OCR. Z textu se vymazala veškerá interpunkce. Text se následně filtroval tak, že každé slovo, získané z dokumentu, se porovnalo se slovníkem. Pokud porovnávané slovo ve slovníku existovalo, bylo přidáno do seznamu rozpoznaných slov. Pokud hledané slovo ve slovníku neexistovalo, rozpoznané slovo bylo zahozeno. Nalezená slova byla následně poskládaná za sebe a byly odděleny mezerou. Takto připravený text byl předán modelu BERT.

Tento způsob filtrace pouze zhoršoval úspěšnost klasifikace a proto se dále s tímto experimentem nepokračovalo.

5.2.3 Filtrace znaků

V tomto experimentu nebyl použit žádný slovník slov. Text, který se získal pomocí OCR, byl filtrovaný tak, že všechny znaky, které nejsou písmena byly z textu vyma-

zány. Mezery mezi slovy se ponechaly. Nalezená slova byla následně poskládaná za sebe a byly odděleny mezerou. Tyto slova z dokumentů se následně předaly modelu BERT, který se následně pomocí těchto slov natrénoval. Stejným principem se text filtroval tak, že namísto filtrování všech znaků, co nejsou písmena, byly ponechány i číslice.

Toto filtrování nepomohlo ke zvýšení úspěšsnoti klasifikace.

5.2.4 Filnální filtrace

Jelikož předchozí metody pro zlepšení textu, který byl získán z OCR, pouze snižovaly úspěšnost, tak tyto metody nebyly použity.

Nejlepší úspěšnost dosáhla jednoduchá úprava textu. A to tak, že se všechny veklá písmena převedla na malá písmena. Následně byly znaky pro novou řádku a pro tabulátor nahrazeny mezerou. Tento text se předal modelu BERT. Tento experiment měl nejlepší výsledky, a proto taky byl v práci použit při extrakci textu z dokumentu za pomocí OCR.

5.2.5 Model BERT

Model BERT, který je popsán v sekci 2.7.3, se trénoval na textu z OCR, který byl jemně upraven. Způsob, jakým byl text upraven je popsán v sekci 5.2.4. Tento text byl předán modelu BERT, na kterém se trénoval.

Tabulka 5.1 představuje výsledky modelu BERT při různých délkách vstupního textu. Nejlepší výsledky dosáhl model BERT při použití maximální délky vstupu 512 tokenů. Model nedosahuje nejlepších výsledků při použití délky vstupu nižší než 512, jako je například 127 či 255. Úspěšnost klasifikace při menší délce vstupu je nižší kvůlu tomu, že se do modelu dostane jen malá část textu dokumentu. Jelikož se nedostane do modelu celý text, tak model může více chybovat, protože nemá k dispozici všechny potřebné informace pro správné určení typu dokumentu. Velikost 512 je na tom nejlépe, protože se do modelu dostane nejvíce informace z textu dokumentu.

Délka vstupu	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
127	78,5	78,7	74,3	75,5
255	81,7	78,9	78,8	78,2
512	**84,9**	**83,2**	**83,0**	**82,9**

Tabulka 5.1: Porovnání výsledků textového klasifikátoru na testovací datové sadě

5.3 Obrázkový vstup

Jak už bylo řečeno, pro klasifikaci skenovaných dokumentů na základě obrázkového vstupu byla použita konvoluční neuronová síť *EfficientNet*. V sekci 5.1 je uveden důvod, proč byla síť EfficientNet použita v této práci pro klasifikaci.

Obrázky byly upraveny na tři velikosti. První úpravou obrázku bylo jeho zmenšení z původní velikosti na velikost 550 × 550 a poté byly odříznuty okraje obrázku na velikost 500 × 500.

Druhou velikostí byl rozměr 400 × 400, na který se obrázek zmenšil. Poté byly jeho okraje odříznuty na velikost 360 × 360.

Třetí velikostí byl rozměr 256 × 256. Na tento rozměr se obrázek zmenšil a následně byly jeho okraje uříznuty na finální rozměr 224 × 224.

Většina obrázků v datasetu obsahuje bíle okraje. Toto je nepotřebná informace pro klasifikátor, proto taky byly tyto okraje odříznuty. Odstraněním těchto bílých okrajů z obrázku byla zvýšena úspěšnost klasifikace.

V této úloze se trénovala celá síť z předem předtrénovaného modelu, který použitá knihovna poskytovala. I když model EfficientNet byl trénován na datasetu `ImageNet`, který je od datasetu Tobacco-3482 odlišný, tak využitím již předtrénovaného modelu se výrazně urychlilo jeho trénování.

Sítě byly trénovány 20 epochy, a jako trénovací dataset bylo použito 2437 obrázků dokumentů z datasetu Tobacco-3482. Byly vyzkoušeny všechny konfigurace modelů (B0 – B7). Na základě pozorování bylo v průměru 20 epoch dostačující, poté již model nevykazoval výrazné změny v průběhu trénování.

Ze všech vyzkoušených konfigurací dosáhla nejlepších výsledků varianta B3 o velikosti obrázku 224 × 224 (viz Tabulka 5.2). Experimenty byly zopakovány i pro variantu 360 × 360 (viz Tabulka 5.3) a variantu 360 × 360 (viz Tabulka 5.4)

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 (%)**
224 × 224				
B0	86,3	84,0	85,1	84,4
B1	86,6	86,7	85,4	85,8
B2	86,7	86,3	85,6	85,6
B3	**87,7**	**88,4**	**85,9**	**86,7**
B4	86,2	84,9	85,4	85,1
B5	81,7	79,3	81,1	80,0
B6	69,9	67,3	66,1	66,1
B7	70,1	67,2	64,8	64,7

Tabulka 5.2: Porovnání výsledků různých konfigurací sítě EfficientNet na testovací datové sadě (rozlišení obrázků (224 × 224))

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
360 × 360				
B0	83,1	81,1	81,9	81,3
B1	81,6	81,2	79,5	79,8
B2	81,3	78,5	87,2	77,4
B3	82,7	75,1	73,6	74,1
B4	80,7	72,2	70,2	71,0
B5	**86,9**	**84,8**	**85,4**	**84,9**
B6	64,5	63,4	60,2	60,0
B7	70,5	67,7	68,0	67,0

Tabulka 5.3: Porovnání výsledků různých konfigurací sítě EfficientNet na testovací datové sadě (rozlišení obrázků (360 × 360))

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
500 × 500				
B0	81,0	81,0	78,9	79,5
B1	82,4	72,4	72,7	72,4
B2	82,6	83,0	80,7	**81,3**
B3	**83,7**	**84,8**	**81,1**	81,1
B4	79,5	80,8	77,9	78,1
B5	81,0	80,9	79,5	78,6
B6	67,9	66,5	63,7	63,9
B7	68,8	69,3	68,9	66,6

Tabulka 5.4: Porovnání výsledků různých konfigurací sítě EfficientNet na testovací datové sadě (rozlišení obrázků (500 × 500))

Zmenšením obrázku dojde k určité ztrátě informace. Z předchozích výsledků je vidět že i po zmenšení obrázku na menší velikost je klasifikace tímto způsobem je na dobré úrovni.

5.4 Kombinovaný vstup

Pro obrazový vstup do kombinovaného modelu byly vybrány pouze konfigurace B3 při velikosti 224 × 224, B5 při velikosti 360 × 360 a model B2 při velikosti obrázků 500 × 500, protože dosahovaly nejlepších výsledků.

Pro textový vstup do kombinovaného modelu byl vybrán model BERT s velikostí vstupu 512, protože měl nejlepší výsleky při testování.

Pro kombinovaný vstup byly vytvořeny celkem dva rozdílné modely, které kombinují vizuální informaci a textovou informaci ze skenu dokumentu. Oba modely se

od sebe funkčně liší. V práci byly pojmenovány model *V1* a model *V2* a jsou popsány v následující sekcích.

5.4.1 Model V1

Jedná se o neuronovou síť, která byla trénována na výstupech textového a obrazového modelu. Model byl trénován na výstupech již natrénovaného textového a natrénovaného obrazového modelu, do kterých se během trénování modelu V1 nijak nezasahovalo. BERT poskytl textovou informaci jako výstupní vektor, který se umístil za výstupní vektor ze sítě EfficientNet, která poskytla informaci vizuální. Vznikl tedy jeden dlouhý vektor, který byl následně předán neuronové síti, na kterých se tato neuronová síť učila.

Počet neuronů skryté vrstvy tohoto modelu byl nastaven na hodnotu 1000. Vyšší hodota nijak nezlepšovala výslednou úspěšnost klasifikace. Jako optimalizátor při trénování modelu byl použit algoritmus Adam, který dodala knihovna torch.

Výsledky tohoto modelu jsou ukázány v tabulce 5.5.

5.4.2 Model V2

Jedná se o model, který kombinuje textovou a obrazovou informaci z dokumentu. Zásadním rozdílem mezi modelem V1 a V2 je jeho rozdílný přístup k textovému a obrazovému modelu. Model V2 nevyužívá již vytvořené a natrénované klasifikátory, ale trénuje se společně s textovým a vizuálním modelem. Během tréninku se změny sítě propagují i do modelu BERT a EfficientNet. Model BERT poskytl informaci o textu dokumentu a model EfficientNet poskytl tomuto modelu vizuální informaci ohledně dokumentu.

Počet neuronů skryté vrstvy modelu V2 byl nastaven na hodnotu 1000. Vyšší hodota nijak nezlepšovala výslednou úspěšnost klasifikace. Jako optimalizátor při trénování modelu V2 byl použit algoritmus Adam, který dodala knihovna torch.

Výsledky modelu V2 jsou ukázány v tabulce 5.6

5.4.3 Výsledky

Model V1	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 (%)**
B3 224 × 224 BERT	**86,7**	84,1	**84,0**	83,8
B5 360 × 360 BERT	86,6	**86,6**	83,4	**84,5**
B2 500 × 500 BERT	84,9	82,7	81,5	81,8

Tabulka 5.5: Porovnání výsledků modelu V1 na testovací datové sadě

Model V2	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
B3 224 × 224 BERT	**88,1**	**87,8**	**87,8**	**87,7**
B5 360 × 360 BERT	85,6	85,1	85,8	84,8
B2 500 × 500 BERT	83,9	81,4	82,2	80,5

Tabulka 5.6: Porovnání výsledků modelu V2 na testovací datové sadě

Z výsledků je patrné, že zlepšení u modelů V1 a V2 přidáním textového vstupu není příliš významné. Z výsledků je také vidět, že modely V1 a V2 jsou na tom v rámi klasifikace podobně. Z modelů typu V1 dosáhl nejlepších výsledků model s konfigurací B3 224 × 224. Ten dosáhl přesnosti 86,7 %. Z modelů typu V2 dosáhl nejlepších výsledků model s konfigurací B3 224 × 224. Ten dosáhl přesností 88,1 % a tím porazil model V1.

Nejhůře byl na tom model V2 s konfigurací B2 500 × 500 s přesností 83,9 %.

Zajímavé je, že přesnost samotného modelu EfficientNet B3 při velikosti vstupního obrázku 224 × 224 byla také poměrně dobrá s celkovou přesností 87,7 %. Toto naznačuje, že EfficientNet je sám o sobě silným modelem.

5.5 Srovnání výsledků s dostupnou literaturou

V práci *Light-Weighted CNN for Text Classification* dosáhli autoři za využití textového klasifikátoru s datasetem Tobacco-3482 úspěšnosti celkem 46 % [Yad20]. V práci *Multimodal deep networks for text and image-based document classification* využili autoři textový model Fastext a MobileNetV2 a dosáhli s datasetem Tobacco-3482 úspěšnosti celkem 87,8 % [Aud+19]. V práci *Efficient Document Image Classification Using Region-Based Graph Neural Network* dosáhli autoři s využitím datasetu Tobacco-3482 úspěšnosti celkem 82,3 % [Man+21]. V práci *Improving accuracy and speeding up Document Image Classification through parallel systems* využili autoři stejný dataset s modely BERT a EfficientNet a dosáhli úspěšnosti 89,4 % [Fer+20]. Klasifikátor vytvořený v této práci dosáhl úspěšnosti celkem 88,1 %. Všechny tyto výsledky jsou uvedeny v tabulce 5.7.

Práce	Úspěšnost (%)
Base Text CNN [Yad20]	46,0
DocBERT [Man+21]	82,3
FasText + MobileNetV2 [Aud+19]	87,8
BERT + EfficientNet [Fer+20]	89,4
Tato práce BERT + EfficientNet	88,1

Tabulka 5.7: Srovnání výsledků různých prací

Závěr 6

Tématem této bakalářské práce byla klasifikace dokumentů. Cílem práce bylo navrhnout a implementovat takový klasifikátor, který bude tyto dokumenty klasifikovat do celkem deseti tříd. Během klasifikace zohlední jak vizuální, tak i textovou informaci. Tato textová informace byla dodána pomocí nástroje Tesseract, která byla následně předána modelu BERT. Vizuální informace byla dodána pomocí sítě EfficientNet.

Všechny techniky zpracování obrazu a zpracování textu použité pro klasifikátor byly vysvětleny v teoretické části práce. Jako dataset pro trénování klasifikátoru byl využit Tobacco-3482. Tento dataset je po vytvoření účtu ke stažení zdarma na stránce `kaggle.com/datasets/patrickaudriaz/tobacco3482jpg`.

Pro ověření funkcionality vytvořeného klasifikátoru byla využita testovací část datasetu, která byla v průběhu práce vytvořena.

Na celou práci a pro veškeré experimenty byl použit Python a knihovna torch.

Finální kombinovaný klasifikátor dosáhl úspěšnosti celkem 88,1 % a to za využití jak obrazového, tak i textového vstupu využitím EfficientNet modelu B3 při velikosti obrázků 224×224 a modelu BERT.

Práce dosáhla výsledné úspěšnosti na datasetu Tobacco-3482 celkem 88,1 %, což hodnotím jako úspěch. Nicméně stále tento výsledek překonávají se svojí úspěšností různé jiné práce. Pravděpodobně hlavním důvodem jejich lepšího výsledku je využití větších neuronových sítí, které dokážou po delším trénování lépe určit danou třídu a jiný způsob extrakce textové a vizuální informace ze skenu dokumentu. Bohužel takovéto trénování je velice výpočetně náročné a velmi zdlouhavé, tudíž je skoro nemožné takovéto modely trénovat na běžném domácím počítači. Proto také byl zvolen model BERT a EfficientNet, které nejsou tak velké a náročné na trénování. Tyto modely lze trénovat i na domácím počítači.

Vylepšení klasifikátoru by mohlo být dosaženo zavedením alternativních metodik pro integraci textových a vizuálních dat získaných z dokumentu. Kromě toho by bylo možné dosáhnout zlepšení přesnosti klasifikátoru využitím rozsáhlejšího a vyváženějšího datasetu naskenovaných dokumentů. Začlenění těchto změn má potenciál přinést lepší výsledky v procesu klasifikace skenovaných dokumentů.

Uživatelská dokumentace

A.1 Příprava

Program je napsaný v programovacím jazyce Python ve verzi 3.7.9. Pro spuštění je potřeba mít na zařízení tento programovací jazyk k dispozici. Programovací jazyk Python 3.7.9 je k dispozici ke stažení na odkaze

`python.org/downloads/release/python-379/`.

Ze staženého souboru práce je potřeba extrahovat složku *Aplikace_a_knihovny*. Toto je kořenová složka programu, z které lze spouštět všechny skripty práce. Následně je potřeba do této složky vložit složky *data* a *models*, které se nacházejí ve složce *Vstupni_data*.

Dále je vyžadováno nainstalovat potřebné knihovny, které jsou v práci využité. K práci je dodán soubor *requirements.txt* ve kterém jsou všechny potřebné knihovny s požadovanou verzí uvedené. Tyto knihovny lze nainstalovat pomocí příkazu

```
pip install -r requirements.txt
```

Dále je potřeba mít na zařízení stáhnutý dataset se skeny dokumentů. Po stažení datasetu je potřeba dataset rozbalit do skožky *dataset* uvnitř kořenová složky programu. Výsledkem tedy bude složka *dataset*, v které bude složka *images*, v které bude 10 složek s obrázky dokumentů. Dataset je k dispozici ke stažení na adrese

`kaggle.com/datasets/patrickaudriaz/tobacco3482jpg`

Nakonec je potřeba upravit soubor s nastavením programu. Tento soubor se jmenuje *settings.py* a nachází se v kořenové složce programu. V tomto souboru se nachází parametr *tesseract_path*, který je hned na počátku v *settings.py*. Tento parametr je potřeba nastavit na cestu k nainstalovanému Tesseractu.

A.2 Tvorba dat

Aby mohly být modely trénovány, tak je potřeba rozdělit dataset na testovací, trénovací a validační sadu. Spuštěním skriptu pro rozdělení datasetu *splitDataset.py* se vytvoří celkem tři soubory. Tyto soubory budou vytvořeny ve složce *data*. Názvy a

cesty k těmto souborům lze změnit v *settings.py* upravením parametrů *train_dataset_path*, *test_dataset_path* a *val_dataset_path*. Dataset lze rozdělit příkazem

```
python .\splitDataset.py
```

Dále je potřeba vytvořit soubor s OCR daty dokumentů. Spuštěním skriptu *doOCR.py* se vytvoří ve složce *data* soubor *OCROutput.tsv*. V *settings.py* lze cestu a název vytvořeného souboru upravit změnou parametru *output_ocr_file_path*. Dále lze v nastavení změnit parametr *ocr_configuration*, se kterým se OCR nad dokumenty spouští. Příprava OCR dat lze provést příkazem

```
python .\doOCR.py
```

Nicméně skripty na tvorbu OCR a pro rozdělování datasetu není potřeba spouštět, protože jejich výstupy jsou již v práci vytvořené.

A.3 Trénování textového modelu

Pro trénování textového modelu lze použít příkaz

```
python .\transformerTrain.py
```

Použitím tohoto příkazu se natrénuje textový model, který bude následně uložen do složky *models* s názvem *transformer.pth*.

A.4 Trénování obrazového modelu

Pro trénování obrazového modelu lze použít příkaz

```
python .\efficientNetTrain.py
```

Použitím tohoto příkazu se natrénuje obrazový model, který bude následně uložen do složky *models* s názvem *efficientModel.pth*.

A.5 Trénování kombinovaného modelu V1

Pro trénování kombinovaného modelu V1 je jako první potřeba použít skript, který vytvoří potřebná data. Jedná se o skript *prepareDataForCombNN.py*. Tento skript vytvoří dva soubory. Skript vytvoří soubor *vect.pth* a soubor *label.pth* ve složce *data*, k nimž je nutné předat cesty skriptu pro trénink. Je nutné dodržet podmínku, že těmito soubory je možné trénovat a testovat pouze jeden typ modelu. Pokud by tento skript využil model B0 a transformer s maximálním vstupem 512 tokemů, tak tyto vytvořené soubory nelze použít pro trénování a testování modelu, který by například obsahoval EfficientNet B5. Skript pro přípravu dat lze spustit pomocí příkazu

```
python .\prepareDataForCombNN.py --text_model cesta k textovému modelu
--image_model cesta k obrazovému modelu
```

Po vytvoření potřebných dat lze natrénovat kombinovaný model V1 pomocí příkazu

`python .\combModelV1Train.py`

kterému lze předat parametrem `--vect_file` *`cesta k souboru vect.pth`* a parametrem `--label_file` *`cesta k souboru label.pth`*. Po dokončení trénování bude vytvořen soubor *combV1.pth* ve složce *models*.

A.6 Trénování kombinovaného modelu V2

Pro trénování kombinovaného modelu V2 lze použít příkaz

`python .\combModelV2Train.py`

Ten po dokončení trénování vytvoří soubor *combV2.pth* ve složce *models*.

A.7 Testování textového modelu

Tato sekce obsahuje příklady příkazů pro testování natrénovaných modelů. Pro testování textových modelů lze použít příkaz

`python .\transformerTest.py --model` *`cesta k Transformer modelu`*

A.8 Testování obrazového modelu

Pro testování obrazových modelů lze použít příkaz

`python .\efficientNetTest.py --model` *`cesta k EfficientNet modelu`*

A.9 Testování kombinovaného modelu V1

Pro testování kombinovaného modelu V1 lze použít příkaz

`python .\combModelV1Test.py --model` *`cesta k V1 modelu`* `--vect` *`cesta k vect.pth`* `--label` *`cesta k label.pth`*

U tohoto příkazu je potřeba předat cestu ke správným souborům *vect.pth* a *label.pth*, které byly vytvořeny skriptem *prepareDataForCombNN.py*.

Pro otestování již vytvořených klasifikátorů V1, které byly s prací předány, lze využít již připravené soubory *label* a *vect* ve složce *data*. Soubory jsou pojmenovány dle použitých modelů, s kterými byly tyto soubory vytvořeny. Například soubory *label_b2_500_512.pth* a *vect_b2_500_512.pth* byly vytvořeny za pomocí modelu EfficientNet B2, která zmenšila obrázek na velikost 500×500 a modelu BERT, který byl trénován s maximální délkou textu 512.

A.10 Testování kombinovaného modelu V2

Pro testování kombinovaného modelu V2 lze použít příkaz

```
python .\combModelV2Test.py --model cesta k V2 modelu
```

A.11 Skripty

Každý ze spustitelných skriptů má svůj seznam parametrů, kterými lze skript ovlivňovat. Pro výpis seznamu parametrů skriptu lze spustit daný skript s parametrem *-h*. Tento parametr zajistí, že skript vypíše do konzole nápovědu s popisem parametrů. Například u skriptů pro trénování je k dispozici parametr například pro změnu názvu souboru, pod jakým bude výsledný model uložen (parametr *--model_path*), nebo změnu počtu epoch trénování (parametr *--epochs*).

U testování modelů je potřeba vždy přidat parametr *--model*, který bude určovat cestu k souboru, který obsahuje natrénovaný model.

A.12 Názvy uložených modelů

K práci jsou přiložené předtrénované modely. Modely jsou rozdělené uvnitř složky *models* do složek

- image – obrazové modely
- text – textové modely
- combV1 – kombinované modely V1
- combV2 – kombinované modely V2

Ve složce *image* jsou modely pojmenovány dle druhu modelu EfficientNet a velikosti obrázku. Například model B3 s velikostí obrázku 360 × 360 se jmenuje *efficientnet_b3_360.pth*.

Ve složce *text* jsou modely pojmenovány dle a velikosti vstupního textu, na kterých byl model trénován. Například model se vstupní velikostí textu 255 se jmenuje transformer_255.pth

Obdobně jsou na tom složky *combV1* (obsahuje modely V1) a *combV2* (obsahuje modely V2), kde na konci názvu souboru obsahují název modelu EfficientNet a velikost obrázku, s kterou model EfficientNet pracuje.

Popis adresářové struktury

B

- Text_prace
  - A19B0034P_text_prace.pdf: Soubor PDF písemné části bakalářské práce
  - src: Složka se soubory k textu práce
- Aplikace_a_knihovny: adresář se skripty
  - Složky a soubory .py které program využívá
  - requirements.txt: soubor obsahující seznam všech potřebných knihoven pro správné fungování programu
  - dataset_tobacco_3482: je složka se soubory, které popisují trénovací testovací a validační část datasetu
- Vstupni_data:
  - Obhajue složku models, která obsahuje složky s natrénovanými modely
  - Obsahuje složku data, která obsahuje data potřebná ke spuštění trénování a testování modelů
- Readme.txt: Detailní popis aktuální adresářové struktury

Bibliografie

[Aud+19] AUDEBERT, Nicolas; HEROLD, Catherine; SLIMANI, Kuider; VIDAL, Cédric. *Multimodal deep networks for text and image-based document classification*. 2019. Dostupné z arXiv: 1907.06370 [cs.CV].

[BLK10] BAHARUDIN, Baharum; LEE, Lam Hong; KHAN, Khairullah. A Review of Machine Learning Algorithms for Text-Documents Classification. *Journal of Advances in Information Technology*. 2010.

[Dev+19] DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. Dostupné z arXiv: 1810.04805 [cs.CL].

[Fer+20] FERRANDO, Javier et al. Improving accuracy and speeding up document image classification through parallel systems. In: *International Conference on Computational Science*. Springer, 2020, s. 387–400.

[He+15] HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. *Deep Residual Learning for Image Recognition*. 2015. Dostupné z arXiv: 1512.03385 [cs.CV].

[22] *Kraken*. 2022. Dostupné také z: kraken.re/main/index.html.

[23a] *Kraken GIT*. 2023. Dostupné také z: github.com/mittagessen/kraken.

[KSH12] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In: PEREIRA, F.; BURGES, C.J.; BOTTOU, L.; WEINBERGER, K.Q. (ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012, sv. 25. Dostupné také z: proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[Kum22] KUMAR, Ajitesh. *Difference: Binary, Multiclass & Multi-label Classification*. 2022. Dostupné také z: vitalflux.com/difference-binary-multi-class-multi-label-classification/.

[Le +89] LE CUN, Yann et al. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*. 1989, roč. 27, č. 11, s. 41–46.

[LB+95] LECUN, Yann; BENGIO, Yoshua et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*. 1995, roč. 3361, č. 10, s. 1995.

[Man+21] MANDIVARAPU, Jaya Krishna; BUNCH, Eric; YOU, Qian; FUNG, Glenn. *Efficient Document Image Classification Using Region-Based Graph Neural Network*. 2021. Dostupné z arXiv: 2106.13802 [cs.CV].

[Mik+13] MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. *Efficient Estimation of Word Representations in Vector Space*. 2013. Dostupné z arXiv: 1301.3781 [cs.CL].

[Pop+09] POPESCU, Marius-Constantin; BALAS, Valentina Emilia; PERESCU-POPESCU, Liliana; MASTORAKIS, Nikos E. *Multilayer perceptron and neural networks*. 2009.

[Qad19] QADER, Wisam Abdulazeez. *An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges*. 2019.

[Sch19] SCHMIDT, Robin M. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. 2019. Dostupné z arXiv: 1912.05911 [cs.LG].

[SBB12] SINGH, Amarjot; BACCHUWAR, Ketan; BHASIN, Akshay. A Survey of OCR Applications. *International Journal of Machine Learning and Computing, Vol. 2, No. 3, June 2012*. 2012.

[Sze+14] SZEGEDY, Christian et al. *Going Deeper with Convolutions*. 2014. Dostupné z arXiv: 1409.4842 [cs.CV].

[23b] *T-distributed stochastic neighbor embedding — Wikipedia, The Free Encyclopedia* [en.wikipedia.org/w/index.php?title=T-distributed_stochastic_neighbor_embedding&oldid=1150604607]. 2023.

[TL19] TAN, Mingxing; LE, Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks. In: *International conference on machine learning*. PMLR, 2019, s. 6105–6114.

[23c] *Tesseract GIT*. 2023. Dostupné také z: github.com/tesseract-ocr/tesseract.

[Vas+17] VASWANI, Ashish et al. *Attention Is All You Need*. 2017. Dostupné z arXiv: 1706.03762 [cs.CL].

[Wik23] WIKIPEDIA CONTRIBUTORS. *Artificial neural network — Wikipedia, The Free Encyclopedia* [en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=1152351649]. 2023.

[Yad20] YADAV, Ritu. *Light-Weighted CNN for Text Classification*. 2020. Dostupné z arXiv: 2004.07922 [cs.LG].

Seznam obrázků

Seznam tabulek

Přehled zkratek

BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag-of-words
CBOW	Continuous bag-of-words
CNN	Konvoluční neuronová síť
MLP	Multilayer perceptron
NLP	Natural language processing
OCR	Optical character recognition
OSD	Orientation and script detection
PSM	Page segmentation mode
RNN	Rekurentní neuronová síť
ROC	Receiver Operating Characteristic