UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



NEURÓNOVÉ SIETE PRE IDENTIFIKÁCIU DÔLEŽITÝCH ČASTÍ DOKUMENTU

DIPLOMOVÁ PRÁCA

2024 Bc. Alexandra BEHUNČÍKOVÁ

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

NEURÓNOVÉ SIETE PRE IDENTIFIKÁCIU DÔLEŽITÝCH ČASTÍ DOKUMENTU

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomicko-finančná matematika a modelovanie
Študijný odbor:	Matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	Mgr. Martin Chudjak, PhD.

Bratislava 2024 **Bc. Alexandra BEHUNČÍKOVÁ**





Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Alexandra Behunčíková
Študijný program: ekonomicko-finančná matematika a modelovanie (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: matematika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Neurónové siete pre identifikáciu významných častí dokumentu
Neural networks for identification of important document sections

Anotácia: Narastajúca miera komunikácie medzi klientmi a bankou (resp. poisťovňou) v elektronickej podobe prináša so sebou potrebu zefektívnenia spracovania dokumentov. V poslednom desaťročí nastal výrazný prielom vo vývoji konvolučných neurónových sietí pre spracovanie obrazu. Tieto algoritmy poskytujú vhodný prostriedok na zefektívnenie spracovania dokumentov v komerčnom prostredí.

Cieľ: V práci vyvinieme model, ktorý prostredníctvom pokročilých štatistických metód (konvolučných neurónových sietí) identifikuje časti dokumentu obsahujúce relevantné informácie.

Vedúci: Mgr. Martin Chudjak, PhD.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Marek Fila, DrSc.

Dátum zadania: 10.01.2023

Dátum schválenia: 11.01.2023 prof. RNDr. Daniel Ševčovič, DrSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie Touto cestou by som rada vyjadrila úprimnú vďaku vedúcemu tejto diplomovej práce, Mgr. Martinovi Chudjakovi, PhD., za jeho neúnavnú ochotu, trpezlivosť, mnohé odborné a podnetné rady a povzbudenia pri vedení tejto práce. Taktiež by som sa rada poďakovala svojej rodine a priateľom za podporu v optimalizačnej úlohe štúdia aj života.

Abstrakt

BEHUNČÍKOVÁ, Alexandra: Neurónové siete pre identifikáciu dôležitých častí dokumentu [Diplomová práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; vedúci práce: Mgr. Martin Chudjak, PhD., Bratislava, 2024, 79 s.

Táto diplomová práca sa zaoberá praktickým využitím moderných štatistických metód pri spracovaní dokumentov. Špeciálna pozornosť je venovaná konvolučným neurónovým sieťam. V práci sú najskôr predstavené základné teoretické koncepty potrebné pri spracovaní dokumentov a práci s konvolučnými neurónovými sieťami. Je sformulovaná príslušná klasifikačná úloha, demonštrovaná počítačová reprezentácia obrazu a popísaná štruktúra konvolučnej neurónovej siete. Následne práca nadväzuje praktickými problémami spracovania dokumentov, pričom sa sústredí na automatizáciu spracovania e-mailovej komunikácie poisťovne pri likvidácii poistných udalostí s osobitným dôrazom na vizuálnu klasifikáciu dokumentov a využitie segmentácie na identifikáciu dôležitých častí dokumentu.

Kľúčové slová: konvolučné neurónové siete, spracovanie dokumentov, klasifikácia obrázkov, segmentácia obrázkov.

Abstract

BEHUNČÍKOVÁ, Alexandra: Neural networks for identification of important document sections [Master Thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: Mgr. Martin Chudjak, PhD., Bratislava, 2024, 79 p.

This master thesis examines practical use of modern statistical methods in document processing with emphasis on convolutional neural networks. Foremostly it introduces cardinal theoretic concepts of document processing and convolutional neural networks such as classification, demonstration of computer image representation and convolutional neural network structure. Consequently it presents practical problems in the area of document processing, specifically the automatization of e-mail communication processing regarding clients' insurance claims. In this particular problem the use of visual classification and image segmentation of important document sections are of distinct interest.

Keywords: convolutional neural networks, document processing, image classification, image segmentation.

Obsah

Úvod **10**

1 Klasifikácia dokumentov **12**

1.1 Počítačová reprezentácia obrazu 12

1.2 Klasifikácia 14

1.2.1 Klasifikačná úloha 14

1.2.2 Funkcia skóre 15

1.2.3 Loss funkcia 16

1.3 Tréning klasifikátora 17

1.3.1 Optimalizácia loss funkcie 17

1.3.2 Príklad backpropagation algoritmu pre lineárny klasifikátor 20

1.3.3 Príklad backpropagation algoritmu pre jednoduchú neurónovú sieť . 24

2 Konvolučné neurónové siete **31**

2.1 Konvolúcia 31

2.1.1 Operácia konvolúcie 31

2.1.2 Konvolúcia v spracovaní obrazu 32

2.2 Konvolučná vrstva neurónovej siete 35

2.2.1 Charakteristika konvolučnej vrstvy 35

2.2.2 Hyperparametre konvolučnej vrstvy 35

2.2.3 Príklad jednoduchej konvolučnej neurónovej siete 36

2.3 Ostatné vrstvy konvolučných neurónových sietí 37

2.3.1 Prehľad základnej architektúry konvolučnej neurónovej siete 37

2.3.2 Input/Vstup 37

2.3.3 Konvolučná vrstva 37

2.3.4 RELU vrstva 38

2.3.5 Pooling vrstva 38

2.3.6 Dropout vrstva 39

2.3.7 Fully connected (FC) vrstva 40

3 Region based konvolučné neurónové siete **41**

3.1 R-CNN . . . 41
3.2 Fast R-CNN . . . 42
3.3 Faster R-CNN . . . 42
3.3.1 Trénovanie Faster R-CNN . . . 44
3.4 Mask R-CNN . . . 44

4 Praktické problémy - Likvidácia poistných udalostí 47
4.1 Extrakcia eml súborov . . . 48
4.1.1 Vyťažovanie čísel poistných udalostí . . . 49
4.1.2 Hľadanie otázok . . . 50
4.2 Klasifikácia príloh pri likvidácii poistných udalostí pomocou konvolučnej neurónovej siete . . . 51
4.2.1 Dáta . . . 51
4.2.2 Model . . . 55
4.2.3 Augmentácie . . . 56
4.2.4 Metrika úspešnosti . . . 58
4.2.5 Výsledky . . . 58
4.3 Využitie OCR pri klasifikácii dokumentov a vyťažovaní konkrétnych informácií z niektorých druhov dokumentov . . . 64
4.3.1 Využitie OCR a kľúčových slov pri triedení dokumentov . . . 65
4.3.2 Využitie OCR pri vyťažovaní konkrétnych informácií z dokumentov 67
4.4 Segmentácia obrázkov . . . 68
4.4.1 Príprava dát na segmentáciu . . . 69
4.4.2 Segmentačný model . . . 70
4.4.3 Výsledky modelu a úprava výstupov . . . 70
4.4.4 Augmentácie pre segmentáciu . . . 71
4.4.5 Metriky úspešnosti . . . 71
4.5 Finálna agregácia dát . . . 72

Záver 73

Zoznam použitých zdrojov 75

Príloha A

Príloha B

Úvod

Súčasný nevídaný technologický pokrok úzko prepletený s vývojom a inovatívnym použitím štatistických metód a matematických modelov je nepochybne formatívnou zložkou dnešnej spoločnosti. Pojmy ako umelá inteligencia, strojové učenie či neurónové siete vyvolávajú veľa vzrušenia naprieč všetkými odvetviami. Neurónové siete, ktorým sa v tejto práci venujeme, už dávno nie sú viazané len na technologických gigantov, ale práve naopak, neustále viac sa preukazuje silný potenciál pre ich široké využitie.

Táto diplomová práca sa zameriava na štúdium a implementáciu konvolučných neurónových sietí pri spracovaní dokumentov s cieľom automatizovať extrakciu informácií a zlepšiť celkovú efektivitu dokumentačných procesov. Konkrétne je predstavená problematika praktického využitia neurónových sietí vo finančnom sektore – pri spracovaní dokumentov poisťovne. Analyzujeme, ako môžu konvolučné neurónové siete efektívne identifikovať vzory v štruktúre jednotlivých dokumentov a ako môžu byť optimalizované pre špecifické úlohy ich spracovania.

Prvé tri kapitoly práce sú štruktúrované ako efektívny súhrn teoretických poznatkov z viacerých oblastí. Našou snahou je, aby slúžili na vybudovanie porozumenia základným teoretickým konceptom, ktoré považujeme za potrebné pre prácu s konvolučnými neurónovými sieťami. Finálna kapitola sa venuje detailnému popisu procesu využitia neurónových sietí vo vybraných prípadoch spracovania dát poisťovne.

V prvej kapitole popíšeme počítačovú reprezentáciu obrazu a najpoužívanejšie farebné modely. Na základe obsahu kurzu [29] a teoretického základu v [22] si sformulujeme klasifikačnú úlohu a uvedieme základné pojmy a princípy tvorby klasifikátora. Predstavíme zásady tréningu klasifikátora pomocou backpropagation algoritmu, ktoré si ilustrujeme aj konkrétnymi príkladmi.

Druhá kapitola predstaví matematickú operáciu konvolúcie, jej význam pri práci s obrazom a najpoužívanejšie konvolučné masky v súlade s [7], [22], [29], [37] a [38]. Charakterizujeme a ilustrujeme v nej princíp konvolučnej vrstvy. A pomocou [4], [21] a [41] predstavíme najdôležitejšie vrstvy prítomné v konvolučných neurónových sieťach.

V tretej kapitole približujeme postupný vývoj konkrétnej štruktúry v rámci konvolučných neurónových sietí – Mask R-CNN, dokumentovaný v [1], [9], [10], [12], [13] a [14]. Mask R-CNN je nami použitá pri segmentácii obrazu.

Následne v kapitole štvrtej pristúpime k problematike spracovania dokumentov a využitiu neurónových sietí pri automatizácii spracovania e-mailovej komunikácie oddelenia spracovania žiadostí. Nakoľko poisťovne majú zo zákona obmedzenú časovú lehotu na likvidáciu poistných udalostí, prínosy automatizácie tohto procesu v jeho rýchlosti, presnosti a celkovej efektivite predstavujú dôležité zlepšenie. Ďalšou výhodou automatizácie je eliminácia monotónnej a zdĺhavej práce, a teda aj lepšia alokácia času a zdrojov zamestnancov. V rámci automatizácie riešime praktický problém klasifikácie dokumentov do vizuálnych tried pomocou konvolučných neurónových sietí. Predstavíme vizuálne triedy a klasifikačný problém v kontexte klasifikácie sprievodných dokumentov likvidácie poistných udalostí, v rámci ktorého natrénujeme klasifikátor v podobe konvolučnej neurónovej siete Resnet50. V závere práce pristupujeme k využitiu segmentácie obrazu ako prostriedku pre zlepšenie klasifikácie a automatizácie. Pomocou architektúry Mask R-CNN vytvoríme segmentačný model pre identifikáciu dôležitých častí dokumentu.

1 Klasifikácia dokumentov

Pri likvidácii poistných udalostí patria medzi najdôležitejšie časti mailovej komunikácie medzi klientom a poisťovňou obrázkové prílohy potrebných dokumentov. Tieto prílohy potrebuje poisťovňa spracovať, nahrať do svojej internej databázy a tiež prípadne využiť informácie v nich obsiahnuté pre úkony potrebné pri likvidácii poistnej udalosti. Pri automatizácii likvidácie poistných udalostí preto bude jednou z najdôležitejších úloh práve prístup k dokumentom v prílohách, ich spracovanie a správna klasifikácia. V prvej kapitole si predstavíme základne princípy počítačovej reprezentácie obrazu, klasifikácie príloh, ktoré reprezentujeme ako obrázky, a tvorby (t.j. tréningu) klasifikátora. Tak vytvoríme teoretický základ, ktorý neskôr použijeme pri tvorbe a aplikácii klasifikátora v automatizácii likvidácie poistných udalostí.

1.1 Počítačová reprezentácia obrazu

Obrázok je digitálne reprezentovaný ako obdĺžniková mriežka farebných štvorcov. Takto zorganizované štvorce tvoria dvojrozmernú maticu, ktorej prvkami sú štvorce rôznych farieb. V digitálnom ponímaní tieto štvorce resp. základné farebné prvky tvoriace obrázkovú maticu nazývame pixelmi. Tvar a rozmery obrázkov (predovšetkým pomer výšky a šírky obrázka všeobecne známy aj pod anglickým názvom *aspect ratio*) boli v minulosti do veľkej miery dané technologickými možnosťami obrazového záznamu. V súčasnosti s technologickým pokrokom pribúdajú mnohé nové možnosti, ale do značnej miery sa stále využívajú historicky zaužívané pomery strán (*aspect ratios*).

Obrazové médiá sú teda reprezentované ako matica, ktorej prvé dva rozmery sú dané šírkou a výškou obrázku. Táto matica môže byť v závislosti od použitého farebného modelu dvoj- alebo trojrozmerná. Tretí rozmer, často označovaný aj ako farebná hĺbka obrázkovej matice, je daný počtom farebných kanálov. Počet farebných kanálov určuje, na koľkých primárnych farbách je založený farebný model určujúci farby v obrázku alebo inými slovami, určuje k pre $k - ticu$ čísel, ktorá je potrebná pre zakódovanie farby na obrázku pri použití určitého farebného modelu.

Medzi základné, často využívané farebné modely patria:

1. **binárny model** - čierno-biela reprezentácia obrazu, používa výhradne iba čiernu (0) a bielu farbu(1)

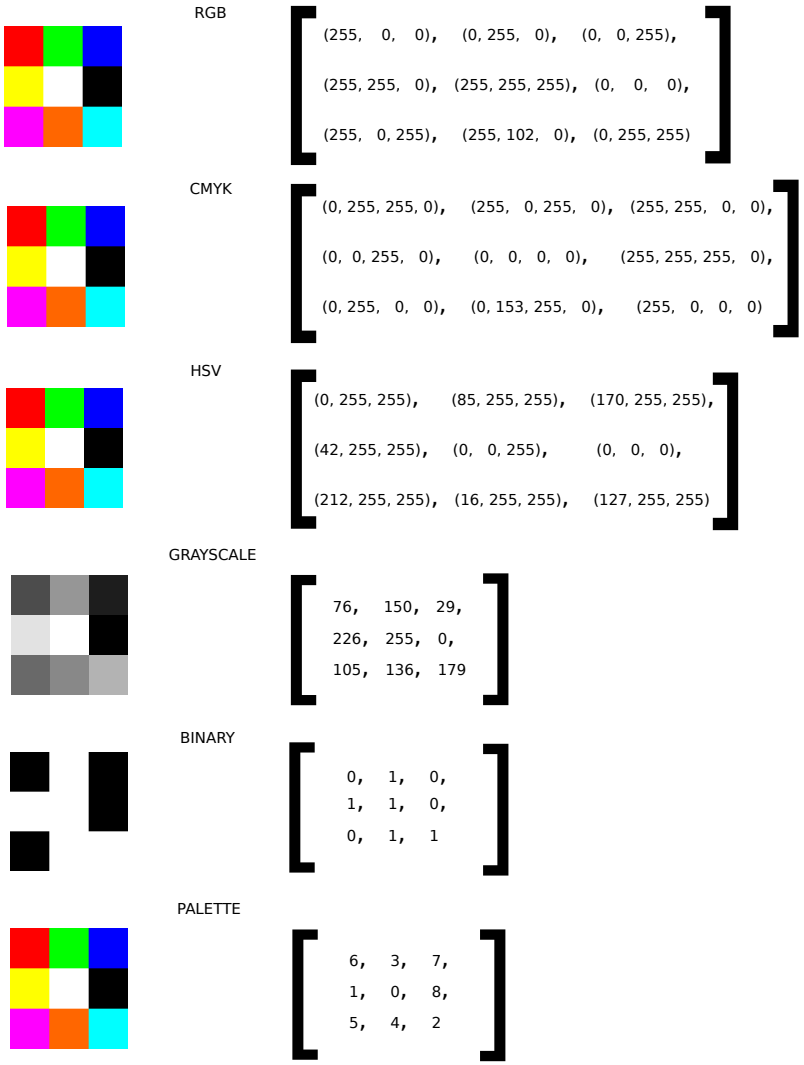
2. **odtiene sivej (grayscale)** - založený na 1 farebnom kanáli, ktorý určuje intenzitu čiernej

3. **RGB (Red, Green, Blue)** - 3 farebné kanály, ktoré určujú intezity pre primárne farby červenú, zelenú a modrú

4. **CMYK (Cyan, Magenta, Yellow, Black)** - 4 farebné kanály, ktoré určujú intezity pre primárne farby cyan, magenta, žltú a čiernu

5. **HSV (Hue, Saturation, Value/Brightness)** - 3 farebné kanály, ktoré určujú odtieň (daný farebným kruhom), sýtosť a jas.

Pre túto prácu budú dôležité predovšetkým RGB a grayscale modely. Farebný kanál ukladá informáciu o intenzite primárnej farby [27], [7].
Špeciálnym spôsobom reprezentácie farieb obrázka, ktorý takisto využívame, je **farebná paleta (palette)**. Farebná paleta je množinou farieb potrebných pre vytvorenie nášho obrázka, pričom každú farbu obrázka reprezentujeme priradeným jednorozmerným indexom. Každý z týchto indexov je pomocou *color lookup table (CLUT)* prepojený s farbami nejakého farebného modelu. Reprezentácia farbenou paletou je vhodným spôsobom
pre zredukovanie veľkosti a dimenzií obrázku. Týmto spôsobom vieme však obsiahnuť maximálne 256 farieb [15].
Obrázok je teda reprezentovaný objektom s rozmermi *šírka* $\times$ *výška* $\times$ *počet farebných kanálov*.



Obr. 1: Počítačová reprezentácia obrázku s rozmermi 3x3 pixely pomocou rôznych farebných modelov a farebnej palety.

1.2 Klasifikácia

1.2.1 Klasifikačná úloha

Uvažujme N obrázkov. Nech každý obrázok $x_i \in R^D, \quad i = 1, \ldots, N$, kde D predstavuje dimenzionalitu obrázku danú ako $D = \textit{šírka} \times \textit{výška} \times \textit{počet farebných kanálov}$.
V mnohých, predovšetkým tradičných klasifikátoroch je obrázok reprezentovaný ako vektor získaný vektorizáciou obrázku v pôvodnej podobe matice.
Každému obrázku nech prislúcha práve jedna z K tried dokumentov $y_i \in \{1, ..., K\}$,

$i = 1, \ldots, N$. Budeme hľadať takú funkciu

$$f : R^D \rightarrow R^K, \tag{1}$$

ktorá každému obrázku priradí reálny K-zložkový vektor skóre, ktorého zložky budú reprezentovať hodnoty skóre, ktoré obrázok dosiahne pre jednotlivé triedy. Táto funkcia predstavuje klasifikátor a budeme ju nazývať funkcia skóre (score function). Trieda s najvyššou hodnotou skóre bude označovať triedu, do ktorej bude obrázok klasifikovaný.

1.2.2 Funkcia skóre

Funkcia skóre je funkcia, ktorej vstupom je obrázok x_i a výstupom vektor skóre pre jednotlivé triedy $f(x_i)$. Najjednoduchšou verziou funkcie skóre je lineárna funkcia

$$f(x_i) = Wx_i + b. \tag{2}$$

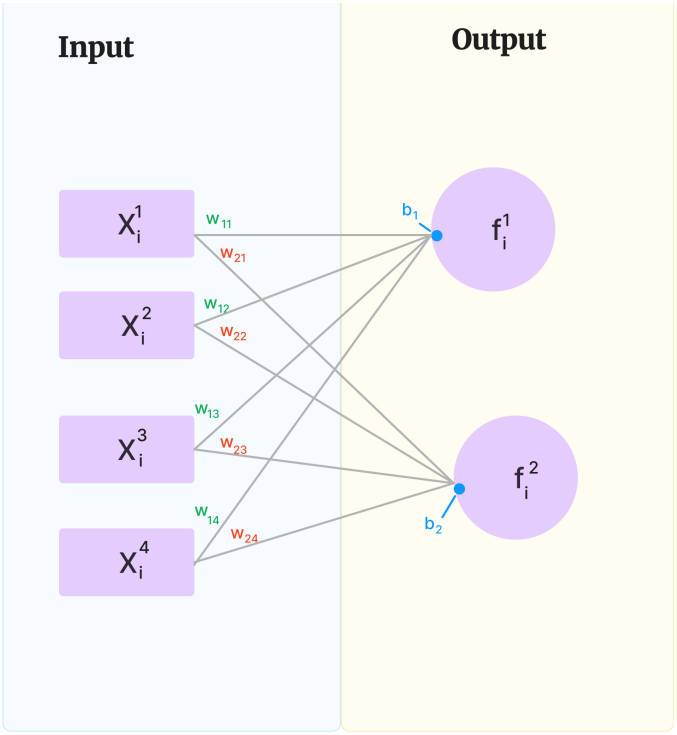
Parametrami funkcie skóre sú

- W - matica váh (weight matrix), pričom pre lineárnu funkciu skóre $W \in \mathbf{R}^{K \times D}$
- b - vektor odchýliek (bias vector), pričom pre lineárnu funkciu skóre $b \in \mathbf{R}^D$ [29].

Parametrami funkcie skóre W a b bude parametrizovaný lineárny klasifikátor. Výstup klasifikátora tvorí informácia o tom, aké skóre pre jednotlivé triedy náš klasifikátor priradí obrazovému vstupu. Vstupný obrázok x_i bude priradený do triedy s najvyšším skóre.
V prípade, že požadujeme výstup v intuitívnejšej a informatívnejšej forme, tak výstup vo forme skóre pre danú triedu vieme transformovať na pravdepodobnosť, že vstup patrí do tej-ktorej triedy. Využijeme na to transformáciu výstupov klasifikátora s názvom **softmax**.
Softmax je označením pre funkciu, ktorá transformuje K-zložkový vektor skóre s reálnymi hodnotami na K-zložkový vektor pravdepodobností s reálnymi hodnotami z intervalu $\langle 0, 1\rangle$, t.j. $s(f(x_i)) : R^K \rightarrow \langle 0, 1\rangle^K$. Je definovaná ako

$$s(f(x_i))^j = \frac{e^{f(x_i)^j}}{\sum_k e^{f(x_i)^k}} \quad \text{pre} \quad j = 1, ..., K, \tag{3}$$

kde značenie $s(f(x_i))^j$ označuje pravdepodobnosť j-tej triedy pre i-ty obrázok, $f(x_i)^j$ skóre j-tej triedy pre i-ty obrázok [22].



Obr. 2: Príklad lineárneho klasifikátora so vstupom x_i v tvare 4-rozmerného vektora a výstupom $f(x_i)$ v tvare 2-rozmerného vektora.

Pod tréningom klasifikátora budeme rozumieť systematické hľadanie, teda optimalizáciu parametrov funkcie skóre. Preto budeme ďalej uvažovať funkciu skóre f ako funkciu vstupného obrázka a zároveň aj jej parametrov W a b, teda v podobe $f(x_i, W, b)$. Pre jednoduchosť zápisu sa často parametre W a b reprezentujú ako jeden. Zvyčajne sa pristupuje k rozšíreniu matice W o jeden riadok váh a pridaniu jednotkovej zložky k vstupu x_i. Pracujeme potom už len so zápisom $f(x_i, W)$. Cieľom optimalizácie funkcie skóre bude nájsť také parametre, pri ktorých budú výsledkom funkcie skóre také hodnoty skóre pre jednotlivé triedy, ktoré budú zodpovedať čo najpresnejšiemu klasifikátoru vstupných obrázkov. Pre účel tréningu klasifikátora si zavedieme pojem **loss funkcie**.

1.2.3 Loss funkcia

Loss funkcia je funkcia, ktorá slúži na vyjadrenie rozdielu medzi predikovanou a očakávanou hodnotou výstupu alebo, inými slovami, straty klasifikátora.
Ako dva najpoužívanejšie druhy loss funkcie [29] predstavuje

1. **Hinge loss**

$$L_i = \sum_{j \neq \text{argmax}(y_i)} max\left(0, f_i^j - f_i^{\text{argmax}(y_i)} + \Delta\right) \quad \text{pre} \quad i = 1, ..., N, \tag{4}$$

kde Δ predstavuje hyperparameter, f_i^j značí $f(x_i, W)^j$, čo predstavuje skóre j-tej triedy pre i-ty obrázok pri parametroch W, $f_i^{\text{argmax}(y_i)}$ značí $f(x_i, W)^{\text{argmax}(y_i)}$, čo predstavuje skóre skutočnej triedy i-teho obrázku pri parametroch W, a

2. **Cross-entropy loss**

$$L_i = -log\left(\frac{e^{f_i^{\arg\max(y_i)}}}{\sum_j e^{f_i^j}}\right) \quad \text{pre} \quad i = 1, ..., N. \tag{5}$$

Pri cross-entropy loss sa môžeme stretnúť s logaritmom so základom 2 ako aj so základom e. Rozdiel medzi hodnotami loss funkcie získanými pre rôzne základy logaritmu spočíva len v prenásobení konštantou.

Celkovú stratu, t.j. hodnotu loss funkcie charakterizujúcu klasifikátor pri daných parametroch W vypočítame ako priemernú stratu (hodnotu loss funkcie) jedného vstupu

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \lambda R(W), \tag{6}$$

kde λ - hyperparameter, $R(W)$ - regularizačná penalta. Ako regularizačná penalta sa veľmi často používa L_1 (t.j. $R(W) = \sum_k \sum_l |w_{k,l}|$) a L_2 norma (t.j. $R(W) = \sum_k \sum_l w_{k,l}^2$, kde $w_{k,l}$ sú prvky matice W).

Minimalizácia loss funkcie podľa parametrov W bude tvoriť tréning klasifikátora.

1.3 Tréning klasifikátora

1.3.1 Optimalizácia loss funkcie

Optimalizáciu loss funkcie podľa jej parametrov vo všeobecnosti zapíšeme ako

$$\min_W \quad L(W, x, y), \tag{7}$$

kde x bude označovať podmnožinu objektov x_i, pre $i = 1, ..., N$, využitú pre výpočet loss funkcie a y zasa množinu príslušných priradení y_i do tried pre jednotlivé objekty x_i.

Gradientná metóda

Na optimalizáciu loss funkcie sa (obzvlášť vo sfére neurónových sietí) zvykne používať

optimalizačná metóda prvého rádu (t.j. metóda využívajúca prvú deriváciu optimalizovanej funkcie) – gradientná metóda. Hodnoty jednotlivých zložiek gradientu nám pomôžu určiť ako zmeniť resp. aktualizovať parametre loss funkcie (váhy uložené v matici W) tak, aby hodnota loss funkcie čo najviac klesala v danom bode parametrického priestoru.
Gradient nám určuje smer optimalizácie funkcie. Potrebujeme však poznať ešte veľkosť optimalizačného kroku. V kontexte klasifikácie a neurónových sietí ide o dôležitý hyperparameter nazývaný vo všeobecnosti aj anglickým názvom *learning rate*. Pre vhodné zvolenie hodnoty optimalizačného kroku (*learning rate*) existuje množstvo optimalizačných algoritmov. Pre *learning rate* je zaužívané označenie α.
Podľa [6], [8], [39] si stručne zhrnieme použitie gradientej metódy pri optimalizácii loss funkcie. Všeobecný predpis gradientnej metódy je

$$W' = W - \alpha \nabla_W L(W, x, y), \tag{8}$$

kde $'$ označuje aktualizáciu váhy a $\nabla_W L(W, x, y)$ predstavuje gradient[1] loss funkcie $L(W, x, y)$ derivovanej podľa jej parametrov uložených v matici W, pre $W \in \mathbf{R}^{K \times D}$

$$\nabla_W L(W, x, y) = \begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} & . & \frac{\partial L}{\partial w_{1D}} \\ \frac{\partial L}{\partial w_{21}} & . & . & \frac{\partial L}{\partial w_{2D}} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial L}{\partial w_{K1}} & . & . & \frac{\partial L}{\partial w_{KD}} \end{bmatrix}. \tag{9}$$

Podľa toho, akú časť trénovacieho datasetu využívame pre jednu aktualizáciu parametrov (jeden iteračný krok), rozlišujeme tri varianty gradientnej metódy.

1. Batch gradient descent - predstavuje klasickú gradientnú metódu, pri ktorej sa použije celý trénovací dataset

$$W' = W - \alpha \nabla_W L(W, x, y). \tag{10}$$

2. Mini batch - je úpravou klasickej gradientnej metódy, ktorá spočíva v tom, že sa pre aktualizáciu parametrov nepoužije celý trénovací dataset. Namiesto toho sa trénovací

[1] Keďže parametre (váhy) loss funkcie bývajú často usporiadané v štruktúre matice, používame aj my túto štruktúru namiesto tradičného zápisu gradientu v tvare vektora.

dataset rozdelí na niekoľko častí (nazývaných *batch*) s určeným počtom objektov (tento počet býva nazývaný aj *batch size*). Ak trénovací dataset nie je deliteľný daným počtom častí (*batches*), posledná *batch* ich bude obsahovať menej. Pri tejto variante pristupujeme ešte pred rozdelením trénovacieho datasetu k náhodnému premiešaniu dát. Aktualizácia váh je daná vzťahom

$$W' = W - \alpha \nabla_W L(W, x_{i:i+b}, y_{i:i+b}), \tag{11}$$

kde $x_{i:i+b}$ označuje množinu objektov prisluchajúcu do jednej *batch* a $y_{i:i+b}$ príslušné priradenia do skutočných tried.

3. Stochastic gradient descent (SGD) - je formou gradientnej metódy, v ktorej sa aktualizujú parametre pre každý objekt trénovacieho datasetu

$$W' = W - \alpha \nabla_W L(W, x_i, y_i). \tag{12}$$

Osobitým problémom je výpočet gradientu loss funkcie. Veľké množstvo parametrov prítomných v klasifikátoroch v podobe konvolučných neurónových sietí sa pre numerický výpočet gradientu ukazuje ako výpočtovo náročné. V praxi sa preto pristupuje k využitiu analytického výpočtu gradientu použitej loss funkcie. Tento analytický predpis sa potom ďalej používa pri optimalizácii loss funkcie. [2]

Backpropagation algoritmus

Okrem použitia samotnej gradientnej metódy bolo, menovite pri tréningu klasifikátorov v podobe neurónových sietí, potrebné navrhnúť algoritmus, ktorý by slúžil na efektívnu aktualizáciu váh počas tréningu. Tento účel plní tzv. backpropagation algoritmus (prvotne navrhnutý fínskym vedcom Seppom Linnainmaanom) [34].

Hodnota loss funkcie vypovedá o kvalite predikcií klasifikátora a využíva sa na výpočet chyby klasifikátora pri aplikovaní backpropagation algoritmu. Backpropagation algoritmus je založený na iterovanej aktuálizácií parametrov klasifikátora a to prostredníctvom opakovania dvoch navzájom prepojených krokov - forward a backward propagation (alebo spätná a dopredná propagácia).

Existuje niekoľko dôležitých pojmov figurujúcich pri tréningu ako klasifikátorov tak aj neurónových sietí. **Iterácia** - je jeden krok optimalizačného algoritmu. **Batch** - je časť

[2] Aby sa predišlo chybám pri analytickom výpočte gradientu, vykonáva sa pred jeho použitím tzv. gradient check - správnosť analytického predpisu gradientu sa porovná s numerickým výpočtom gradientu. Až keď prejde analytický predpis touto skúškou, môžeme ho použiť [29].

objektov trénovacieho datasetu, ktorá sa použije pri jednej iterácii optimalizačného algoritmu, pri čom **batch size** udáva počet objektov pre jednu takúto časť. **Epocha** - popisuje jeden cyklus, počas ktorého optimalizačný algoritmus prejde celým trénovacím datasetom. Tento pojem väčšinou figuruje pri tréningu neurónových sietí.
Fungovanie backpropagation algoritmu si ilustrujeme na príklade lineárneho klasifikátora a následne aj na príklade jednoduchej neurónovej siete s jednou fully connected skrytou vrstvou. Budeme uvažovať dataset s dvoma následovnými vektorizovanými obrázkovými vstupmi

$$\vec{x}_1 = \begin{bmatrix} 0,99 \\ 0,05 \\ 0,6 \\ 1,4 \end{bmatrix}, \quad \vec{x}_2 = \begin{bmatrix} 0,15 \\ 1,15 \\ 0,15 \\ 1,15 \end{bmatrix}. \tag{13}$$

V príkladoch budeme používať cross-entropy loss funkciu danú (5). Skutočné priradenie do tried bude

$$\vec{y}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \vec{y}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{14}$$

Kvôli priehľadnosti budeme pracovať s výpočtami zaokrúhlenými na štyri desatinné miesta.

1.3.2 Príklad backpropagation algoritmu pre lineárny klasifikátor

1. Forward propagation

Forward pass alebo dopredná propagácia spočíva vo výpočte predikcií klasifikátora za použitia aktualizovaných parametrov (váh) a následnom výpočte miery rozdielnosti (hodnoty loss funkcie) predikovaného výstupu oproti požadovanému výstupu.
V našom príklade to bude vyzerať nasledovne. Najskôr si pomocou počiatočných váh vypočítame výstup klasifikátora $\vec{f}_i$ pre $i-tý$ obrázkový vstup. V našom príklade pracujeme s klasifikátorom, ktorý klasifikuje dáta do dvoch tried, preto budeme pre vektor $\vec{f}_i$ počítať jeho dve zložky f_i^1 a f_i^2. Nech hodnoty počiatočných váh sú

$$\begin{aligned} &w_{11} = 0,15, w_{12} = 0,1, w_{13} = 0,25, w_{14} = 0,1, \\ &w_{21} = 0,3, w_{22} = 0,15, w_{23} = 0,2, w_{24} = 0,35. \end{aligned} \tag{15}$$

Dosadením konkrétnych hodnôt prvého objektu (13) a počiatočných váh (15) do vzťahu (2) takto získame

$$f^1(\vec{x}_1) = f_1^1 = 0,15 \times 0,99 + 0,1 \times 0,05 + 0,25 \times 0,6 + 0,1 \times 1,4 + 0,2 = 0,6435, \tag{16}$$

$$f^2(\vec{x}_1) = f_1^2 = 0,3 \times 0,99 + 0,15 \times 0,05 + 0,2 \times 0,6 + 0,35 \times 1,4 + 0,2 = 1,1145. \tag{17}$$

Podobne aj pre druhý zo vstupných objektov (13)

$$f^1(\vec{x}_2) = f_2^1 = 0,15 \times 0,15 + 0,1 \times 1,15 + 0,25 \times 0,15 + 0,1 \times 1,15 + 0,2 = 0,49, \tag{18}$$

$$f^2(\vec{x}_2) = f_2^2 = 0,3 \times 0,15 + 0,15 \times 1,15 + 0,2 \times 0,15 + 0,35 \times 1,15 + 0,2 = 0,85. \tag{19}$$

Výsledne skóre (16)–(19) následne využijeme pre vypočítanie hodnoty loss funkcie pre jednotlivé objekty podľa vzťahu (5)

$$L_1 = -log\left(\frac{e^{f_1^2}}{e^{f_1^1} + e^{f_1^2}}\right) = -log\left(\frac{e^{1,1145}}{e^{0,6435} + e^{1,1145}}\right) = 0,2107, \tag{20}$$

$$L_2 = -log\left(\frac{e^{f_2^1}}{e^{f_2^1} + e^{f_2^2}}\right) = -log\left(\frac{e^{0,49}}{e^{0,49} + e^{0,85}}\right) = 0,3862. \tag{21}$$

Výslednú hodnotu loss funkcie pre dané hodnoty parametrov (váh) nášho lineárneho klasifikátora získame podľa vzťahu (6) pre $\lambda = 0$

$$L = \frac{1}{2}\left(-log\left(\frac{e^{1,1145}}{e^{0,6435} + e^{1,1145}}\right) - log\left(\frac{e^{0,49}}{e^{0,49} + e^{0,85}}\right)\right) = \frac{0,2107 + 0,3862}{2} = 0,29845. \tag{22}$$

2. Backward propagation

Backward pass alebo spätná propagácia je založená na spätnom prenose chyby od výstupu k vstupu. Tento prenos spočíva vo výpočte gradientu loss funkcie podľa parametrov (váh) klasifikátora. Pre klasifikátor v podobe neurónových sietí s niekoľkými vrstvami sa počíta gradient podľa váh pre neuróny v každej vrstve. Na začiatku kapitoly sme predstavili lineárny klasifikátor – ten si môžeme predstaviť ako veľmi jednoduchú neurónovú sieť, ktorá obsahuje iba jednu fully-connected vrstvu.

Derivácia loss funkcie pre jednotlivé parametre bude zloženou deriváciou. V našom príklade

$$\frac{\partial L}{\partial w_{kl}} = \sum_{i=1}^{2} \frac{\partial L}{\partial f_i^k} \frac{\partial f_i^k}{\partial w_{kl}}, \tag{23}$$

čo vieme úpravou na zložky zapísať aj ako

$$\frac{\partial L}{\partial w_{1l}} = \sum_{i=1}^{2} \frac{\partial L}{\partial f_i^1} \frac{\partial f_i^1}{\partial w_{1l}}, \tag{24}$$

$$\frac{\partial L}{\partial w_{2l}} = \sum_{i=1}^{2} \frac{\partial L}{\partial f_i^2} \frac{\partial f_i^2}{\partial w_{2l}}. \tag{25}$$

Pre learning rate $\alpha = 0,05$ bude aktualizácia váh vyzerať nasledovne

$$w'_{kl} = w_{kl} - \alpha \frac{\partial L}{\partial w_{kl}}, \quad k = 1,2 \quad \text{a} \quad l = 1,2,3,4. \tag{26}$$

Podľa vzťahu (6) bude pre parciálnu deriváciu $\frac{\partial L}{\partial f_i^k}$ platiť

$$\frac{\partial L}{\partial f_i^k} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i}{\partial f_i^k}, \tag{27}$$

a pre jednotlivé zložky sumy v (27) bude podľa vzťahu (5) platiť

$$\frac{\partial L_i}{\partial f_i^k} = \begin{cases} \frac{e^{f_i^k}}{\sum_j e^{f_i^j}} - 1, & \text{ak} \quad k = \arg\max(\vec{y_i}), \\ \frac{e^{f_i^k}}{\sum_j e^{f_i^j}}, & \text{ak} \quad k \neq \arg\max(\vec{y_i}). \end{cases} \tag{28}$$

Pre prípad $N = 2$ objektov však tento predpis ešte môžeme upraviť nasledovne

$$\frac{\partial L_i}{\partial f_i^k} = \begin{cases} -\frac{e^{f_i^k}}{\sum_j e^{f_i^j}}, & \text{ak} \quad k = \arg\max(\vec{y_i}), \\ \frac{e^{f_i^k}}{\sum_j e^{f_i^j}}, & \text{ak} \quad k \neq \arg\max(\vec{y_i}). \end{cases} \tag{29}$$

Z (2) ďalej vyplýva

$$\frac{\partial f_i^k}{\partial w_{kl}} = x_i^k, \tag{30}$$

kde x_i^k je $k-$tá zložka x_i. Po vyčíslení hodnôt pre $\frac{\partial L}{\partial f_i^k}$ pomocou programu priloženého v prílohe A a ich dosadení do (24)–(25) spolu s hodnotami objektov $\vec{x_1}$ a $\vec{x_2}$ podľa (13)

získavame

$$\frac{\partial L}{\partial w_{11}} = \frac{1}{2}\left(\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} - \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times x_1^1 + \frac{1}{2}\left(-\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} + \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times x_2^1 \tag{31}$$
$$= \frac{0,4882 \times 0,99 - 0,5090 \times 0,15}{2} = 0,2035,$$

$$\frac{\partial L}{\partial w_{12}} = \frac{1}{2}\left(\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} - \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times x_1^2 + \frac{1}{2}\left(-\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} + \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times x_2^2 \tag{32}$$
$$= \frac{0,4882 \times 0,05 - 0,5090 \times 1,15}{2} = -0,2805,$$

$$\frac{\partial L}{\partial w_{13}} = \frac{1}{2}\left(\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} - \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times x_1^3 + \frac{1}{2}\left(-\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} + \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times x_2^3 \tag{33}$$
$$= \frac{0,4882 \times 0,6 - 0,5090 \times 0,15}{2} = 0,1083,$$

$$\frac{\partial L}{\partial w_{14}} = \frac{1}{2}\left(\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} - \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times x_1^4 + \frac{1}{2}\left(-\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} + \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times x_2^4 \tag{34}$$
$$= \frac{0,4882 \times 1,4 - 0,5090 \times 1,15}{2} = 0,0491,$$

$$\frac{\partial L}{\partial w_{21}} = \frac{1}{2}\left(-\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} + \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times x_1^1 + \frac{1}{2}\left(\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} - \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times x_2^1 \tag{35}$$
$$= \frac{-0,4882 \times 0,99 + 0,5090 \times 0,15}{2} = -0,2035,$$

$$\frac{\partial L}{\partial w_{22}} = \frac{1}{2}\left(-\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} + \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times x_1^2 + \frac{1}{2}\left(\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} - \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times x_2^2 \tag{36}$$
$$= \frac{-0,4882 \times 0,05 + 0,5090 \times 1,15}{2} = 0,2805,$$

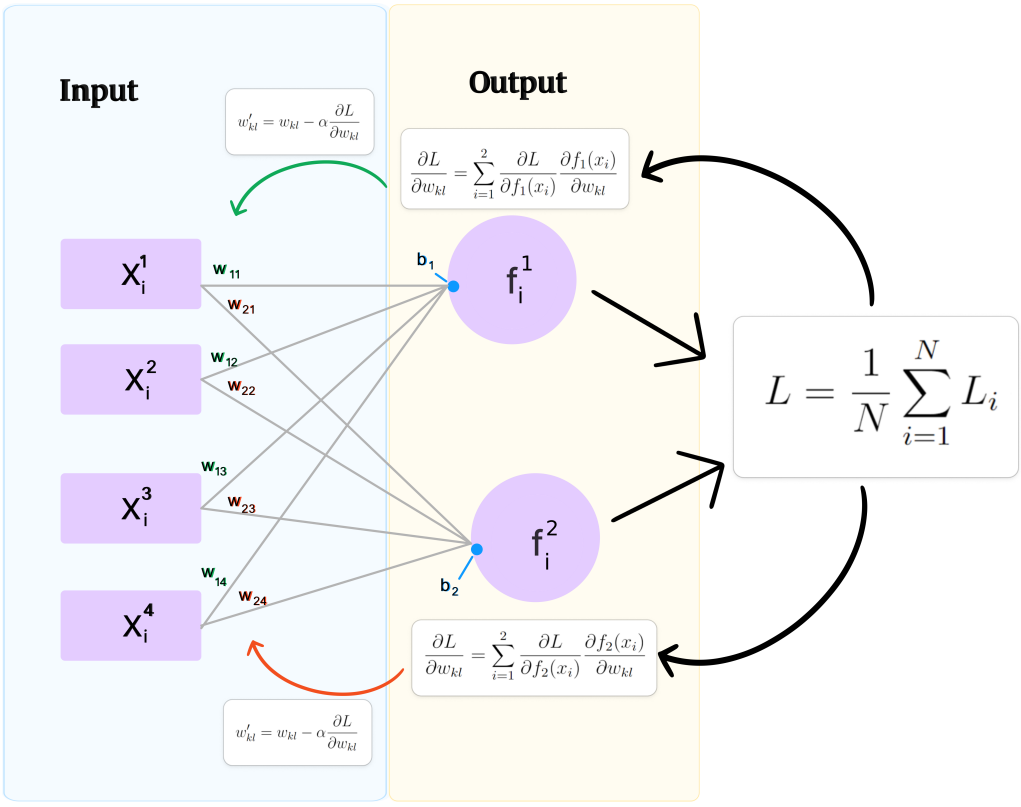
$$\frac{\partial L}{\partial w_{23}} = \frac{1}{2}\left(-\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} + \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times x_1^3 + \frac{1}{2}\left(\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} - \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times x_2^3 \tag{37}$$
$$= \frac{-0,4882 \times 0,6 + 0,5090 \times 0,15}{2} = -0,1083,$$

$$\frac{\partial L}{\partial w_{24}} = \frac{1}{2}\left(-\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} + \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times x_1^4 + \frac{1}{2}\left(\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} - \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times x_2^4 \tag{38}$$
$$= \frac{-0,4882 \times 1,4 + 0,5090 \times 1,15}{2} = -0,0491,$$

a následne aj pre aktualizáciu váh (26)

$$w'_{11} = 0,15 - 0,05 \times 0,2035 = 0,1398, w'_{12} = 0,1 + 0,05 \times 0,2805 = 0,1140,$$
$$w'_{13} = 0,25 - 0,05 \times 0,1083 = 0,2446, w'_{14} = 0,1 - 0,05 \times 0,0491 = 0,0975,$$
$$w'_{21} = 0,3 + 0,05 \times 0,2035 = 0,3101, w'_{22} = 0,15 - 0,05 \times 0,2805 = 0,1360,$$
$$w'_{23} = 0,2 + 0,05 \times 0,1083 = 0,2054, w'_{24} = 0,35 + 0,05 \times 0,0491 = 0,3525.$$

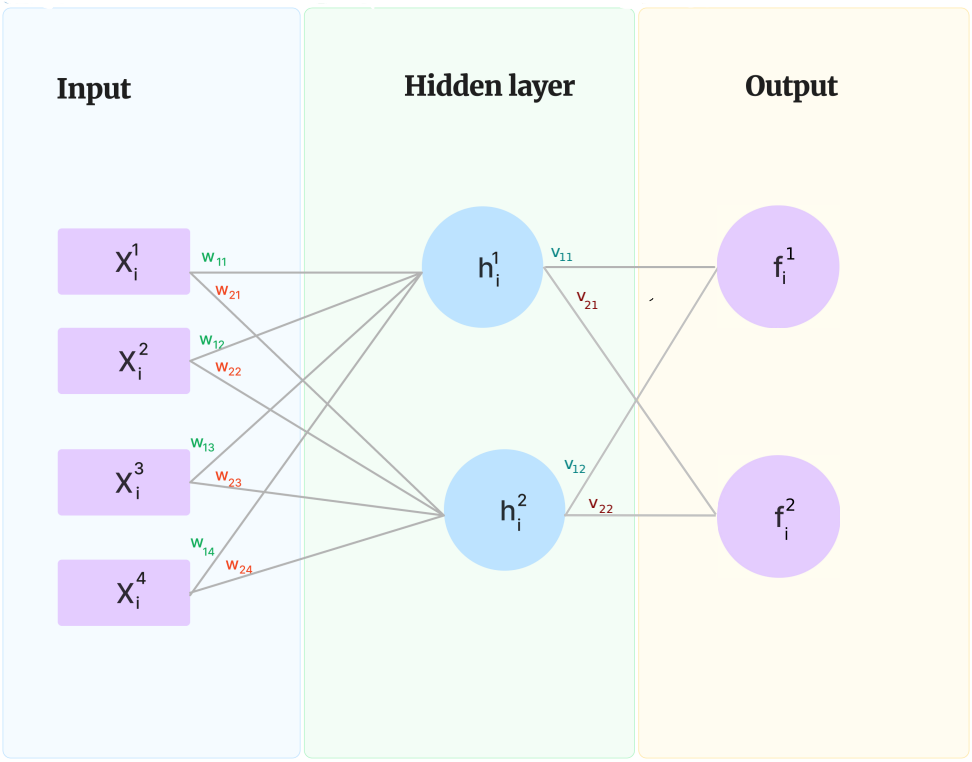
Aplikáciu vzťahov (6) a (23) – (26) môžeme vidieť znázornenú na Obr.3.



Obr. 3: Príklad backward passu.

1.3.3 Príklad backpropagation algoritmu pre jednoduchú neurónovú sieť

Backpropagation algoritmus si ilustrujeme aj na príklade neurónovej siete s jednou skrytou fully connected vrstvou.



Obr. 4: Príklad jednoduchej neurónovej siete s jednou skrytou vrstvou.

1. Forward propagation

V našom príklade to bude vyzerať následovne. Nech

$$v_{11} = 0,4, v_{12} = 0,25, v_{21} = 0,3, v_{22} = 0,35. \tag{39}$$

Najskôr si pomocou počiatočných váh (15) vypočítame vstup $\vec{h_i}$ našej jedinej skrytej vrstvy pre $i-tý$ obrázkový vstup

$$h_i^1 = w_{11}x_i^1 + w_{12}x_i^2 + w_{13}x_i^3 + w_{14}x_i^4, \tag{40}$$

$$h_i^2 = w_{21}x_i^1 + w_{22}x_i^2 + w_{23}x_i^3 + w_{24}x_i^4. \tag{41}$$

Nakoľko naša skrytá vrstva bude rozmerov 2×1, budeme počítať jednotlivé zložky h_i^1 a h_i^2. Pre konkrétne hodnoty prvého objektu (13) a počiatočných váh (15) prvej vrstvy takto získame

$$h_1^1 = 0,15 \times 0,99 + 0,1 \times 0,05 + 0,25 \times 0,6 + 0,1 \times 1,4 = 0,4435, \tag{42}$$

$$h_2^1 = 0,15 \times 0,15 + 0,1 \times 1,15 + 0,25 \times 0,15 + 0,1 \times 1,15 = 0,29. \tag{43}$$

Podobne aj pre druhý zo vstupných objektov (13)

$$h_1^2 = 0,3 \times 0,99 + 0,15 \times 0,05 + 0,2 \times 0,6 + 0,35 \times 1,4 = 0,9145, \tag{44}$$

$$h_2^2 = 0,3 \times 0,15 + 0,15 \times 1,15 + 0,2 \times 0,15 + 0,35 \times 1,15 = 0,65. \tag{45}$$

Keďže v skrytej vrstve v tomto príklade nevykonávame žiadne ďalšie operácie, budú vypočítané hodnoty $\vec{h_i}$ tvoriť zároveň aj výstup pre skrytú vrstvu a takisto vstup pre výstupnú vrstvu. Výstup výstupnej vrstvy, ktorá bude takisto ako skrytá vrstva dvojrozmerná, ďalej vypočítame následovne

$$f_i^1 = v_{11}h_i^1 + v_{12}h_i^2, \tag{46}$$

$$f_i^2 = v_{21}h_i^1 + v_{22}h_i^2. \tag{47}$$

Čo pre konkrétne hodnoty h_i dané (42) - (45) a počiatočné hodnoty váh výstupnej vrstvy (39) bude

$$f_1^1 = 0,4 \times 0,4435 + 0,25 \times 0,9145 = 0,406025, \tag{48}$$

$$f_2^1 = 0,4 \times 0,29 + 0,25 \times 0,65 = 0,2785, \tag{49}$$

$$f_1^2 = 0,3 \times 0,4435 + 0,35 \times 0,9145 = 0,453125, \tag{50}$$

$$f_2^2 = 0,3 \times 0,29 + 0,35 \times 0,65 = 0,3145. \tag{51}$$

Takto získame výstup našej jednoduchej neurónovej siete. Tento výstup následne využijeme pre vypočítanie hodnoty loss funkcie pre jednotlivé objekty (13) pomocou vzťahu (5) a skutočného priradenia objektov (14)

$$L_1 = -log\left(\frac{e^{f_1^2}}{e^{f_1^1} + e^{f_1^2}}\right) = -log\left(\frac{e^{0,453125}}{e^{0,406025} + e^{0,453125}}\right) = 0,290923, \tag{52}$$

$$L_2 = -log\left(\frac{e^{f_1^2}}{e^{f_1^1} + e^{f_1^2}}\right) = -log\left(\frac{e^{0,2785}}{e^{0,2785} + e^{0,3145}}\right) = 0,308918. \tag{53}$$

Výslednú hodnotu loss funkcie pre dané hodnoty parametrov (váh) našej jednoduchej neurónovej siete získame podľa vzťahu (6) pre $\lambda = 0$

$$L = \frac{1}{2}\left(-log\left(\frac{e^{0,453125}}{e^{0,406025} + e^{0,453125}}\right) - log\left(\frac{e^{0,2785}}{e^{0,2785} + e^{0,3145}}\right)\right) = \frac{0,290923 + 0,308918}{2} = 0,6. \tag{54}$$

2. Backward propagation

Spätná propagácia chyby v neurónovej sieti bude v našom príklade vyzerať nasledovne. Derivácia loss funkcie pre jednotlivé parametre bude zloženou deriváciou. Pre parametre prepájajúce skrytú vrstvu a výstup neurónovej siete najprv derivujeme loss funkciu podľa výstupu neurónovej siete a následne daným parametrom. Získame

$$\frac{\partial L}{\partial v_{pr}} = \sum_{i=1}^{2} \frac{\partial L}{\partial f_i^p} \frac{\partial f_i^p}{\partial v_{pr}}, \tag{55}$$

čo môžeme upravene zapísať aj ako

$$\frac{\partial L}{\partial v_{1r}} = \sum_{i=1}^{2} \frac{\partial L}{\partial f_i^1} \frac{\partial f_i^1}{\partial v_{1r}}, \tag{56}$$

$$\frac{\partial L}{\partial v_{2r}} = \sum_{i=1}^{2} \frac{\partial L}{\partial f_i^2} \frac{\partial f_i^2}{\partial v_{2r}}. \tag{57}$$

Postupujeme ďalej smerom k vstupu neurónovej siete. Môžeme vidieť, že čím je vrstva od výstupu neurónovej siete ďalej, tým narastá počet členov zloženej derivácie, ktoré určujú parametre prislúchajúce k tejto vrstve. Derivácia podľa parametrov prepájajúcich vstup a skrytú vrstvu teda bude

$$\frac{\partial L}{\partial w_{kl}} = \sum_{p=1}^{2} \sum_{i=1}^{2} \frac{\partial L}{\partial f_i^p} \frac{\partial f_i^p}{\partial h_i^k} \frac{\partial h_i^k}{\partial w_{kl}}, \tag{58}$$

resp. v upravenej podobe

$$\frac{\partial L}{\partial w_{1l}} = \sum_{i=1}^{2} \left(\frac{\partial L}{\partial f_i^1} \frac{\partial f_i^1}{\partial h_i^1} \frac{\partial h_i^1}{\partial w_{1l}} + \frac{\partial L}{\partial f_i^2} \frac{\partial f_i^2}{\partial h_i^1} \frac{\partial h_i^1}{\partial w_{1l}} \right), \tag{59}$$

$$\frac{\partial L}{\partial w_{2l}} = \sum_{i=1}^{2} \left(\frac{\partial L}{\partial f_i^1} \frac{\partial f_i^1}{\partial h_i^2} \frac{\partial h_i^2}{\partial w_{2l}} + \frac{\partial L}{\partial f_i^2} \frac{\partial f_i^2}{\partial h_i^2} \frac{\partial h_i^2}{\partial w_{2l}} \right). \tag{60}$$

Pre learning rate α tak aktualizácia váh bude vyzerať nasledovne

$$w'_{kl} = w_{kl} - \alpha \frac{\partial L}{\partial w_{kl}}, \quad k = 1, 2 \quad \text{a} \quad l = 1, 2, 3, 4, \tag{61}$$

$$v'_{pr} = v_{pr} - \alpha \frac{\partial L}{\partial v_{pr}}, \quad p = 1, 2 \quad \text{a} \quad r = 1, 2. \tag{62}$$

Pre hodnoty $\frac{\partial L}{\partial f_i^p}$ bude platiť rovnaký vzťah ako v (27) v predchádzajúcej časti 1.3.2. Pre $\frac{\partial f_i^p}{\partial v_{pr}}$ zas bude podľa (46)–(47) platiť

$$\frac{\partial f_i^p}{\partial v_{pr}} = \frac{\partial (v_{p1} h_i^1 + v_{p2} h_i^2)}{\partial v_{pr}} = h_i^r \quad \text{pre} \quad r = 1, 2. \tag{63}$$

Po vyčíslení hodnôt $\frac{\partial L}{\partial f_i^p}$ a ich dosadení do (55) a pre parametre v_{pr} dané (39) získavame

$$\frac{\partial L}{\partial v_{11}} = \frac{1}{2}\left(\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} - \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times h_1^1 + \frac{1}{2}\left(-\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} + \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times h_2^1 \tag{64}$$
$$= \frac{0,4882 \times 0,4435 - 0,5090 \times 0.29}{2} = 3.4460 \times 10^2,$$
$$\frac{\partial L}{\partial v_{12}} = \frac{1}{2}\left(\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} - \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times h_1^2 + \frac{1}{2}\left(-\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} + \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times h_2^2 \tag{65}$$
$$= \frac{0,4882 \times 0,9145 - 0,5090 \times 0,65}{2} = 0.5782 \times 10^2,$$
$$\frac{\partial L}{\partial v_{21}} = \frac{1}{2}\left(-\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} + \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times h_1^1 + \frac{1}{2}\left(-\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} + \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times h_2^1 \tag{66}$$
$$= \frac{-0,4882 \times 0,4435 + 0,5090 \times 0,29}{2} = -3.4460 \times 10^2,$$
$$\frac{\partial L}{\partial v_{22}} = \frac{1}{2}\left(-\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} + \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times h_1^2 + \frac{1}{2}\left(-\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} + \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times h_2^2 \tag{67}$$
$$= \frac{-0,4882 \times 0,9145 + 0,5090 \times 0,65}{2} = -0.5782 \times 10^2.$$

Pre $\frac{\partial L}{\partial w_{kl}}$ ďalej vypočítame jednotlivé činiteľe. Prvý činiteľ dvojitej sumy v (58) bude opäť daný vzťahom (27) z časti 1.3.2. Druhý činiteľ $\frac{\partial f_i^p}{\partial h_i^k}$ získame pomocou (46)–(47) ako

$$\frac{\partial f_i^p}{\partial h_i^k} = \frac{\partial (v_{p1}h_i^1 + v_{p2}h_i^2)}{\partial h_i^k} = v_{pk} \quad \text{pre} \quad k = 1, 2. \tag{68}$$

A nakoniec pre $\frac{\partial h_i^k}{\partial w_{kl}}$ bude podľa definície h_i^k v (42) až (45) platiť

$$\frac{\partial h_i^k}{\partial w_{kl}} = \frac{\partial w_{k1}x_i^1 + w_{k2}x_i^2 + w_{k3}x_i^3 + w_{k4}x_i^4}{\partial w_{kl}} = x_i^l \quad \text{pre} \quad k = 1, 2 \quad \text{a} \quad l = 1, 2, 3, 4. \tag{69}$$

Pre $\frac{\partial h_i^k}{\partial w_{kl}}$ teda nakoniec získame vzťahy

$$\frac{\partial L}{\partial w_{1l}} = \sum_{p=1}^{2}\left(\frac{1}{2}\left(\frac{e^{f_1^1}}{\sum_j e^{f_1^j}} - \frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times v_{p1} \times x_i^l + \frac{1}{2}\left(-\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} + \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times v_{p1} \times x_i^l\right)$$
$$= 0,2441 \times v_{11} \times x_1^l - 0,2545 \times v_{11} \times x_2^l + 0,2441 \times v_{21} \times x_1^l - 0,2545 \times v_{21} \times x_2^l$$
$$\text{pre} \quad l = 1, 2, 3, 4, \tag{70}$$
$$\frac{\partial L}{\partial w_{2l}} = \sum_{p=1}^{2}\left(\frac{1}{2}\left(-\frac{e^{f_1^1}}{\sum_j e^{f_1^j}}\frac{e^{f_1^2}}{\sum_j e^{f_1^j}}\right) \times v_{p2} \times x_i^l + \frac{1}{2}\left(\frac{e^{f_2^1}}{\sum_j e^{f_2^j}} - \frac{e^{f_2^2}}{\sum_j e^{f_2^j}}\right) \times v_{p2} \times x_i^l\right)$$
$$= -0,2441 \times v_{12} \times x_1^l + 0,2545 \times v_{12} \times x_2^l - 0,2441 \times v_{22} \times x_1^l + 0,2545 \times v_{22} \times x_2^l$$
$$\text{pre} \quad l = 1, 2, 3, 4, \tag{71}$$

čo bude po vyčíslení pomocou hodnôt váh (39) a objektov (13)

$$\frac{\partial L}{\partial w_{11}} = 0,1425, \frac{\partial L}{\partial w_{12}} = -0,1964, \frac{\partial L}{\partial w_{13}} = 0,0758, \frac{\partial L}{\partial w_{14}} = 0,0344,$$
$$\frac{\partial L}{\partial w_{21}} = -0,1221, \frac{\partial L}{\partial w_{22}} = 0,1683, \frac{\partial L}{\partial w_{23}} = -0,065, \frac{\partial L}{\partial w_{24}} = -0,0295.$$

Následne keď dosadíme tieto výsledky derivácie loss funkcie do vzťahov (61)–(62) pre aktualizáciu váh, získame pre learning rate $\alpha = 0,05$

$$w'_{11} = 0,15 - 0,05 \times 0,1425 = 0,1429, w'_{12} = 0,1 + 0,05 \times 0,1964 = 0,1098,$$
$$w'_{13} = 0,25 - 0,05 \times 0,0758 = 0,2462, w'_{14} = 0,1 - 0,05 \times 0,0344 = 0,0983,$$
$$w'_{21} = 0,3 + 0,05 \times 0,1221 = 0,3061, w'_{22} = 0,15 - 0,05 \times 0,1683 = 0,1416,$$
$$w'_{23} = 0,2 + 0,05 \times 0,065 = 0,2033, w'_{24} = 0,35 + 0,05 \times 0,0295 = 0,3515$$
$$v'_{11} = 0,4 - 0,05 \times 0,0345 = 0,3983, v'_{12} = 0,25 - 0,05 \times 0,0058 = 0,2497,$$
$$v'_{21} = 0,3 + 0,05 \times 0,0345 = 0,3017, v'_{22} = 0,35 + 0,05 \times 0,0058 = 0,3503.$$

Aplikáciu vzťahov (6) a (59) – (61) môžeme vidieť vizualizovanú na Obr. 5.
Pri vzťahu (58) si môžeme všimnúť, že ak si jednotlivé členy predstavíme ako prvky matice, vzťah ako celok je možné zapísať vo forme maticového násobenia. Ak si označíme

$$\frac{\partial L}{\partial f_i^p} = \begin{cases} \frac{e^{f_i^p}}{\sum_j e^{f_i^j}} - 1, & \text{ak} \quad p = \arg\max(\vec{y_i}), \\ \frac{e^{f_i^p}}{\sum_j e^{f_i^j}}, & \text{ak} \quad p \neq \arg\max(\vec{y_i}). \end{cases} = F_{ip}, \tag{72}$$

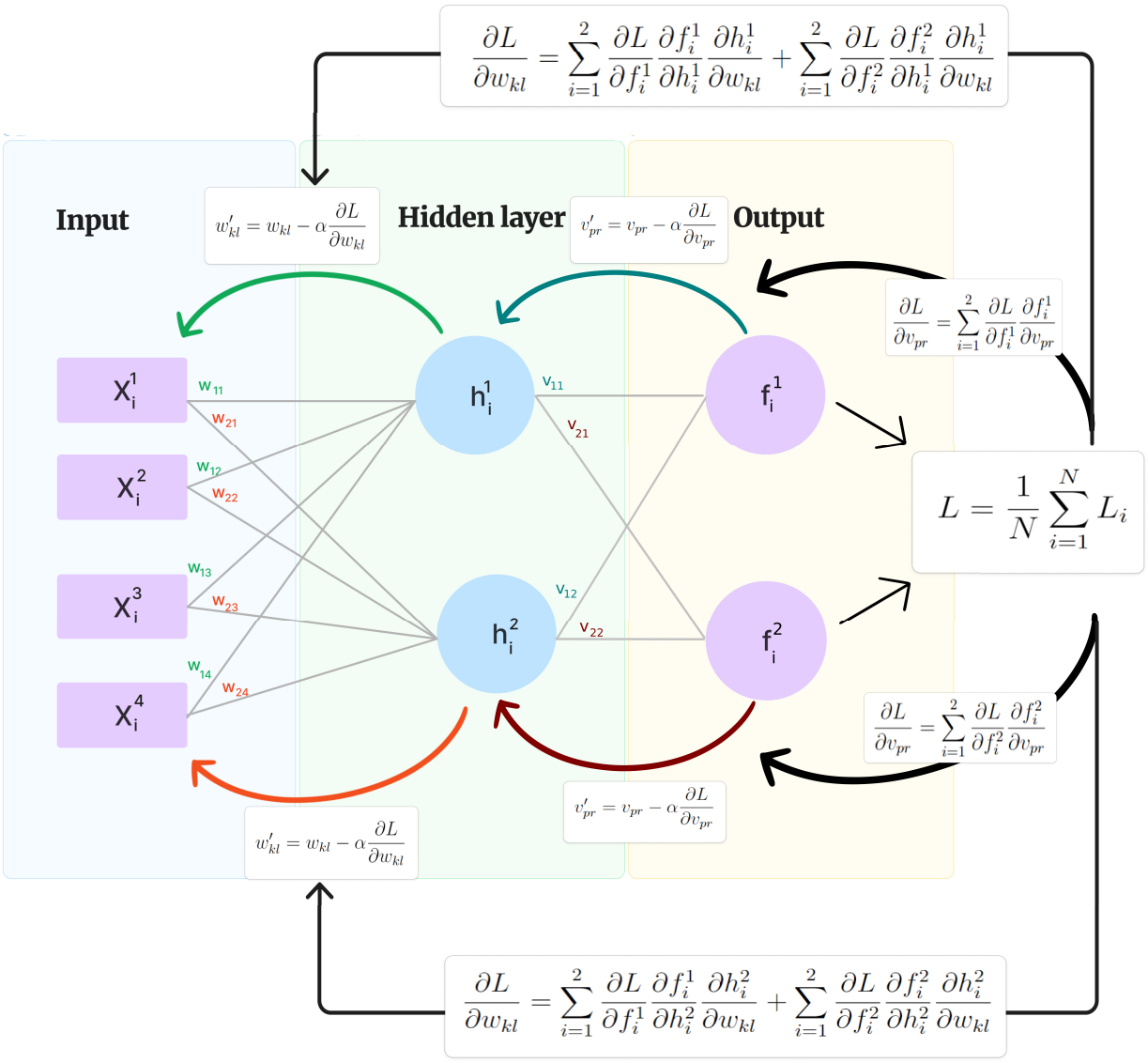
$$\frac{\partial f_i^p}{\partial h_i^k} = v_{pk} = V_{pk}, \tag{73}$$

$$\frac{\partial h_i^k}{\partial w_{kl}} = x_i^l = X_{il}, \tag{74}$$

potom získame vzťah (58) v podobe

$$\frac{\partial L}{\partial w_{kl}} = X^T F V. \tag{75}$$

Práve táto skutočnosť podtrhuje silu a efektívnosť backpropagation algoritmu. Pre pravidlo aktualizácie váh je kľúčový člen $\frac{\partial L}{\partial w_{kl}}$ a zo (72) vidíme, že tento člen vieme získať aj ako výsledok násobenia matíc, čo nám hovorí, že pre jeho výpočet vieme využiť vektorové operácie a výpočet tak značne zefektívniť.



Obr. 5: Príklad backward passu v jednoduchej neurónovej sieti s jednou skrytou vrstvou.

2 Konvolučné neurónové siete

V druhej kapitole si názorne vysvetlíme operácie konvolúcie a korelácie a ich význam v spracovaní obrazu na základe informácií obsiahnutých v [7], [38], [37], [27], [22], [20], [30] a [25]. Ďalej pomocou [22] a [29] popíšeme konvolučnú vrstvu a jej unikátne vlastnosti, ktoré ju robia vhodným kandidátom pre prácu s obrazovými vstupmi. Predstavíme aj ostatné vrstvy konvolučnej neurónovej siete a ich funkcie na základe [29], [22], [21] a [4].

2.1 Konvolúcia

2.1.1 Operácia konvolúcie

Konvolúcia je matematické zobrazenie z priestoru dvoch funkcií do priestoru jednej funkcie, ktoré má široké využitie – mimo iné v pravdepodobnosti, štatistike, v oblastiach diferenciálnych rovníc, spracovania signálov a samozrejme pri spracovaní obrazu. Označujeme ju symbolom *. V štatistike sa stretávame s konvolúciou definovanou pre funkcie reálnej spojitej premennej. Pre jednu spojitú premennú je konvolúcia definovaná ako

$$(o * g)(x) = \int_R o(\alpha) g(x - \alpha) d\alpha. \tag{76}$$

V problematike spracovania obrazu má však využitie predovšetkým diskrétna konvolúcia, ktorú pre konečné funkcie jednej diskrétnej premennej vieme zapísať ako

$$(o * g)(k) = \sum_{i=0}^{n-1} o(i) g(k - i) \quad \text{pre} \quad k = 0, 1, ..., i + n - 2. \tag{77}$$

Pre obrázky, ktoré majú dve priestorové rozmery - šírku a výšku, využívame podobu dvojrozmernej diskrétnej konvolúcie

$$(o * g)(k, l) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} o(i, j) g(k - i, l - j) \tag{78}$$

$$\text{pre} \quad k = 0, 1, ..., i + n - 2 \quad \text{a} \quad l = 0, 1, ..., i + n - 2.$$

V tomto kontexte vystupuje $o(k, l)$ ako funkcia obrazového vstupu a $g(k, l)$ ako filter (niekedy nazývaný aj ako konvolučná maska, prípadne filtračné alebo konvolučné jadro) [7], [38]. Úlohu konvolučného jadra plnia v konvolučných neurónových sietiach nízkorozmerné matice.

V oblasti spracovania obrazu často dochádza k zamieňaniu konvolúcie s operáciou vzájomnej korelácie. Korelácia sa v oblasti spracovania obrazu resp. spracovania signálov zvykne označovať symbolom $\star$.

Korelácia a konvolúcia sú si ako zobrazenia veľmi podobné, rozdiel v ich použití na obrazové vstupy vieme popísať za pomoci filtračného jadra – aplikovaniu konvolúcie prostredníctvom nejakého filtračného jadra zodpovedá aplikovanie korelácie po tom, ako dané filtračné jadro dvakrát priestorovo prevrátime – raz horizontálne a raz vertikálne [22].

Rovnako ako pri konvolúcii, v problematike spracovania obrazu pracujeme s diskrétnou koreláciou. Pre obrázky, ktoré majú dve priestorové rozmery – šírku a výšku, využívame podobu dvojrozmernej diskrétnej korelácie [3]

$$(o \star g)(k,l) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} o(i,j) g(k+i, l+j) \tag{79}$$

$$\text{pre} \quad k = 0, 1, ..., i+n-2 \quad \text{a} \qquad k = 0, 1, ..., i+n-2.$$

Vzájomný vzťah medzi týmito zobrazeniami vieme matematicky vyjadriť ako

$$(o \star g)(k,l) = (o(-k,-l) * g(k,l))(k,l). \tag{80}$$

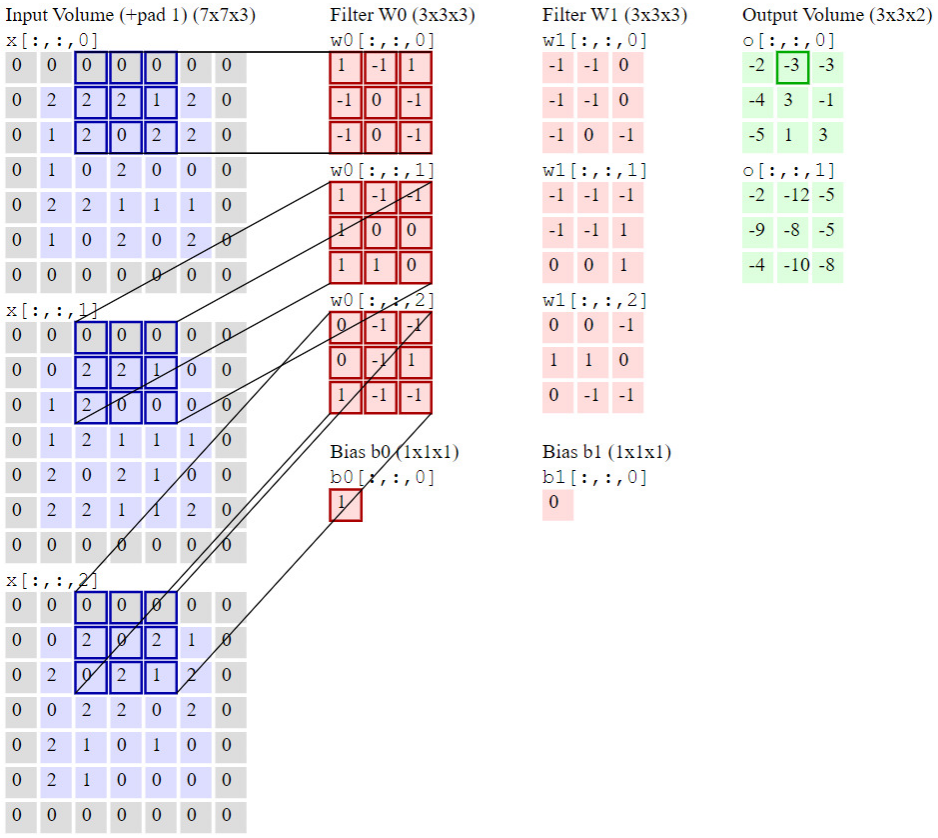
V praxi sa v konvolučných neurónových sieťach používa tak konvolúcia ako aj korelácia, v rámci názvov sa však zaužívala iba konvolúcia. Preto všetky dôležité pojmy – konvolučná vrstva, konvolučné neurónové siete, konvolučné masky a pod. – môžu paradoxne referovať aj na masky, vrstvy či neurónové siete, v ktorých sa využíva korelácia [22].

2.1.2 Konvolúcia v spracovaní obrazu

V súlade s [22], [29] a [25] jednoducho ilustrujeme fungovanie konvolúcie pri spracovaní obrazu. V spracovaní obrazu nachádza svoje využitie diskrétna konvolúcia. Tradične sa aplikácia konvolúcie prezentuje pomocou "sliding window"(Obr.6). Znamená to, že na vstupný obrázok postupne prikladáme konvolučnú masku. Začína sa (podľa konvencie) v ľavom hornom rohu vstupného obrázku a maska sa postupne posúva doprava o daný počet prvkov. V každej pozícii, ktorú maska priložená na vstupnom obrázku nadobudne,

[3] Vzorec pre koreláciu je uvedený v podobe, v akej vystupuje v kontexte konvolučných neurónových sietí, t.j. nepoužívame značenie pre komplexne združenú funkciu ako je to vo všeobecnej definícii korelácie, keďže v našom prípade sa komplexné združenie prejaví iba v zmene znamienka.

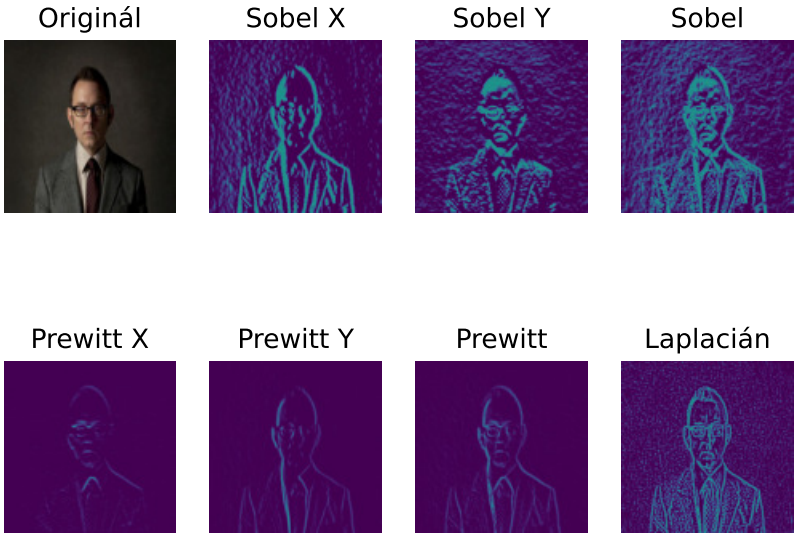
sa vykoná súčet prvkov Hadamardovho súčinu masky a časti obrázkovej matice, na ktorej sa maska práve nachádza.



Obr. 6: Príklad použitia konvolučného filtra. Na trojrozmernú obrázkovú maticu s rozmermi $7 \times 7 \times 3$ sú v slede za sebou použité dva konvolučné filtre $w0$ a $w1$ s rozmermi $3 \times 3 \times 3$ spolu s jednorozmernými vektormi odchýliek $b0$ a $b1$. Na obrázku je znázornené použitie prvého filtra $w0$ a odchýlky $b0$. Na každú dvojrozmernú maticu pozdĺž tretieho rozmeru matice (t.j. x[:,:,0], x[:,:,1], x[:,:,2]) je aplikovaná príslušná matica konvolučného filtra $w0$ (t.j. w0[:,:,0], w0[:,:,1], w0[:,:,2]). Začína sa (podľa konvencie) v ľavom hornom rohu vstupného obrázka a filter sa postupne posúva doprava o zvolený počet prvkov. V každej pozícii, ktorú filter priložený na vstupnom obrázku nadobudne, sa vykoná súčet prvkov Hadamardovho súčinu konvolučného filtra a časti obrázkovej matice, na ktorej sa filter práve nachádza. Jednotlivé súčty Hadamardovho súčinu pre dvojrozmerné matice prislúchajúce obrázku sa následne ešte sčítajú medzi sebou. Nakoniec je k nim pripočítaná odchýlka $b0$ a získame tak výsledok jedného kroku konvolúcie pre konvolučný filter $w0$ a odchýlku $b0$. Zdroj: [29].

Najdôležitejšie filtre využívané pri práci s obrazom a často aj vrámci konvolučných neurónových sietí sú napríklad vyhladzovací filter, filter na rozostrenie či filter na detekciu

hrán [38]. Slúžia na prvotnú úpravu obrazového vstupu podľa potreby použitých modelov resp. vrstiev neurónovej siete. Filter na rozostrenie slúži napríklad na odstránenie vysokofrekvenčných súčastí obrázka, ktoré sa môžu javiť ako hrany, a používa sa preto často pred použitím filtra na detekciu hrán. Na tento účel sa používa napríklad veľmi známy Gaussov filter. Detekcia hrán je zasa zásadná pre identifikáciu vzorov na obrázku. Existuje viacero populárnych filtrov s touto funkciu, napríklad Prewittow filter (zložený z vertikálneho a horizontálneho filtra), Sobelov filter (zložený z vertikálneho a horizontálneho filtra) alebo tzv. Laplacián. V hlbších konvolučných neurónových sieťach vystupujú ako parametre jednotlivých vrstiev konvolučné filtre, ktoré sú trénované na identifikáciu znakov špecifických pre danú úlohu neurónovej siete.



Obr. 7: Príklad použitia filtra na detekciu hrán. Pôvodný obrázok je pred použitím filtra transformovaný do odtieňov sivej. V prípade Sobelovho a Prewittovho filtra sú použité filtre na detekciu hrán najskôr osobitne v horizontálnom (X) a vertikálnom (Y) smere a až následne sú spojené do jedného filtra pre oba smery. Laplacián je už primárne aplikovaný v oboch smeroch súčasne. Zdroj pôvodného obrázku: [26].

2.2 Konvolučná vrstva neurónovej siete

2.2.1 Charakteristika konvolučnej vrstvy

Konvolučná vrstva je základným stavebným blokom konvolučnej neurónovej siete. Jej špecifiká jej dovoľujú predpokladať vstup vo forme obrázku a zabezpečujú afínnu invariantnosť pri rozpoznávaní vzorcov v obrázku – teda rozpoznanie vzorcov v obrázku bez ohľadu na ich situovanie v rámci obrázka. Konvolučná vrstva sa vyznačuje nasledujúcimi troma vlastnosťami.

1. **Lokálna/riedka konektivita** - ako je ilustrované aj na Obr.6, v prípade konvolučnej neurónovej vrstvy sú prepojenia konvolučnej vrstvy sústredené do konvolučného filtra.

2. **Priestorové usporiadanie** - zároveň usporiadanie trénovaných parametrov do podoby konvolučného filtra aplikovaného na obrázok bez zmeny jeho priestorovej štruktúry zabezpečuje, že sa berie do úvahy priestorové usporiadanie jednotlivých pixelov obrázka.

3. **Zdieľanie parametrov** - zdieľanie parametrov medzi jednotlivými neurónmi konvolučnej vrstvy má za následok výraznú redukciu počtu parametrov, ktoré potrebujeme natrénovať. Ďalšou veľkou výhodou je skutočnosť, že vo väčšine prípadov chceme, aby schopnosť identifikovať daný objekt na obrázku nebola citlivá na situovanie objektu na obrázku, v čom je práve nápomocná táto štruktúra [29].

2.2.2 Hyperparametre konvolučnej vrstvy

- **Depth - hĺbka** - korešponduje s počtom filtrov, ktoré chceme v danej vrstve použiť.

- **Stride** - vyjadruje veľkosť kroku, ktorým budeme náš filter prechádzať po vstupnom obrázku.

- **Padding resp. zero - padding** - vstup „obložíme“ nulami po jeho obvode, hodnota parametra padding hovorí o dimenzii („hrúbke“) obloženia [29], [22].

2.2.3 Príklad jednoduchej konvolučnej neurónovej siete

Vezmeme si príklad jednoduchej konvolučnej siete, ktorá bude mať dve konvolučné vrstvy, každú s jedným filtrom (maticou) o rozmeroch 3×3, kde jednotlivé filtre budú nasledovné

$$Filter1 = g_1(x) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}, Filter2 = g_2(x) = \begin{bmatrix} -1 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{81}$$

Budeme používať krok $S = 1$ a padding $P = 1$. Ako vstup si zoberieme obrázok v odtieňoch sivej rozmerov 5×5:

$$o(x) = \begin{bmatrix} 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 & 1 \\ 1 & 1 & 0 & 5 & 0 \\ 2 & 4 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{82}$$

Premenná x je dvojrozmernou priestorovou premennou $x = (x_1, x_2)^T$. Každá vrstva obsahuje práve jeden filter, preto aj hĺbka každej vrstvy bude $D = 1$. Rozmery výstupu 1.vrstvy vieme získať pomocou rozmerov vstupného obrázka, ktorými sú $Width_1 = 5, Height_1 = 5$, a hodnôt hyperparametrov S, P, D.

$$Width_2 = \frac{Width_1 - F + 2P}{S} + 1, \tag{83}$$

$$Height_2 = \frac{Height_1 - F + 2P}{S} + 1, \tag{84}$$

$$D_2 = 1. \tag{85}$$

Rozmery výstupu 1.vrstvy budú rovnaké ako rozmery vstupu, a síce 5×5. Analogicky výpočet pre 2.vrstvu vráti opäť rozmery tentokrát už finálneho výstupu ako 5×5. Výstup 1. vrstvy (a vstup 2. vrstvy):

$$h_1(x) = (g_1 * o)(x) = \begin{bmatrix} 0 & -3 & 0 & 2 & 0 \\ -2 & 2 & -3 & 1 & 5 \\ -4 & -1 & 4 & 2 & 0 \\ -1 & 1 & -4 & 0 & 5 \\ -4 & 2 & 4 & 0 & 0 \end{bmatrix}. \tag{86}$$

Výstup 2. konvolučnej vrstvy, a teda aj celej tejto jednoduchej konvolučnej neurónovej siete bude

$$h_2(x) = (g_2 * h_1)(x) = (g_2 * g_1 * o)(x) = \begin{bmatrix} 2 & -8 & 3 & 4 & 1 \\ 0 & 2 & -1 & 5 & 5 \\ -5 & -1 & 2 & 1 & -1 \\ 2 & 1 & -3 & 0 & 3 \\ -5 & 7 & 3 & -1 & 0 \end{bmatrix}. \tag{87}$$

2.3 Ostatné vrstvy konvolučných neurónových sietí

2.3.1 Prehľad základnej architektúry konvolučnej neurónovej siete

2.3.2 Input/Vstup

Vstupom je obrázok v podobe 3D (resp. 2D v prípade čierno-bieleho obrázku) matice s rozmermi šírka × výška × počet farebnných kanálov (resp. v prípade tradičného obrázku z R,G,B farebnej schémy šírka × výška × 3). Je rozumné patrične upraviť rozmery vstupných obrázkov. V súčasnosti, keď obrázky dosahujú veľkého rozlíšenia, je nutné ošetriť ich rozmery s ohľadom na výpočtovú náročnosť pri trénovaní neurónovej siete. Preto sa pristupuje k redukcii rozlíšenia obrázkov. Aj v prípade moderných architektúr konvolučných neurónových sietí pred ich použitím musíme dbať na to, aké sú vyžadované rozmery pre ich vstupy. Konvolučná neurónová vrstva je schopná spracovať obrázok ľubovoľných rozmerov, ale FC vrstva má vstup fixných rozmerov. FC síce môže byť situovaná iba na konci architektúry, no problém nastane už ak nebude vstupný obrázok presne daných rozmerov. Je to spôsobené tým, že rozmery obrázku/objektu po prechode konvolučnou vrstvou sú dané pravidlami predstavenými v (83)–(85). Riešenie tohto problému predstavuj operácia s názvom pooling. Nie všetky architektúry však v sebe obsahujú patrične zabudovaný pooling.

2.3.3 Konvolučná vrstva

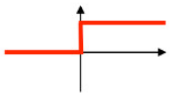
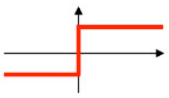
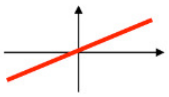
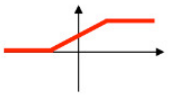
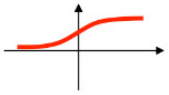
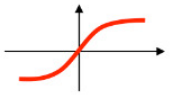
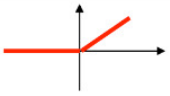
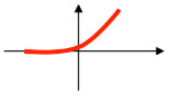
Konvolučná vrstva berie ako svoj vstup obrázok a aplikuje naň filter, pomocou ktorého sa snaží rozpoznať vzorce vo vstupe. Konvolučná vrstva je detailnejšie opísaná v častiach 2.1.2 a 2.2.

2.3.4 RELU vrstva

Vrstva, ktorá na svoj vstup aplikuje RELU aktivačnú funkciu, pričom nemení rozmery svojho vstupu. Aktivačná funkcia slúži na predstavenie nelinearít v neurónových sieťach. Predstavenie nelinearít prináša možnosť zachytiť komplikovanejšie vzťahy medzi vstupmi a výstupmi siete. Použitie aktivačnej funkcie hrá tiež kľúčovú úlohu pri trénovateľnosti neurónovej siete. Pri pohľade na náš príklad lineárneho klasifikátora (2) z časti 1.2.1, vidíme, že v každom neuróne prebieha len lineárna transformácia vstupu pomocou matice váh a vektora odchýliek. V prípade, že by sme vytvorili neurónovú sieť len pomocou neurónov, ktoré uskutočnujú lineárnu transformáciu, nezáležalo by na hĺbke siete. Zloženie lineárnych transformácií by bolo totiž tiež len lineárnou transformáciou. Všetky vrstvy siete by sa teda správali rovnako. A využitie gradientu a backpropagation algoritmu by sa pri tréningu neurónovej siete ukázalo ako neučinné.
V praxi sa používajú rôzne aktivačné funkcie, podľa druhu riešeného problému. V prípade konvolučných neurónových sietí je v súčasnosti často používanou práve RELU aktivačná funkcia.

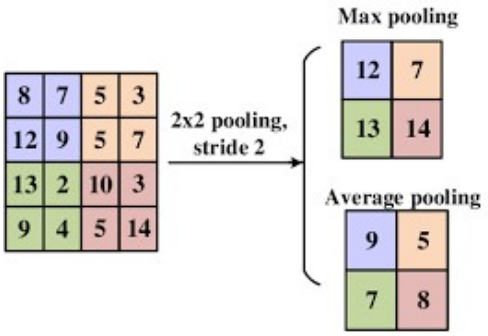
2.3.5 Pooling vrstva

Vrstva, ktorá na svoj vstup aplikuje funkciu, pomocou ktorej zredukuje veľkosti jeho dimenzií a vráti výstup fixných rozmerov. Ide teda o downsampling operáciu. Princíp jej použitia môžeme prirovnať k použitiu konvolučnej masky. Na vstupnú maticu je prikladané okno daného rozmeru, ktoré je posúvané daným krokom. Na oblasti vstupnej matice, na ktorej je priložené vykoná pooling - napríklad vyberie maximálny prvok z danej oblasti (*max pooling*) alebo vytvorí priemer prvkov danej oblasti (*average pooling*) [4].

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016 (http://sebastianraschka.com)

Obr. 8: Príklady najpoužívanejších aktivačných funkcií. Zdroj [32].



Obr. 9: Príklad max a average pooling operácie. Zdroj: [41].

2.3.6 Dropout vrstva

Dropout vrstva predstavuje formu regularizácie modelu, ktorej účelom je predísť jeho pretrénovaniu. Regularizácia pomocou dropout vrstvy spočíva v tom, že počas tréningu zmra-

zíme hodnoty niektorých parametrov a zabránime ich ďalšej aktualizácii počas tréningu. Počet a tiež voľba neaktívnych parametrov v danej vrstve sú dané pravdepodobnosťou zmrazenia/zneaktívnenia (inak nazývaný *dropout rate*) [21].

2.3.7 Fully connected (FC) vrstva

Tak ako pri klasických neurónových sieťach aj pri konvolučných neurónových sieťach je na konci (a niekedy nie len na konci) siete fully connected (plne prepojená) vrstva, ktorej každý neurón je prepojený so všetkými neurónmi z predchádzajúcej vrstvy, čo je znázornené na Obr. 4. Fully connected vrstva aplikuje na svoje vstupy lineárnu transformáciu, ako sme ilustrovali v časti 1.3.3. Z toho vyplýva, že každý jeden výstup tejto vrstvy je ovplyvnený každým jej vstupom.

3 Region based konvolučné neurónové siete

Osobitou triedou konvolučných sietí sú region-based konvolučné siete, ktoré sa používajú na segmentáciu obrazu. V tejto kapitole si predstavíme vývoj modelov tejto triedy, ktorý viedol k modelu Mask R-CNN – modelu určenému na úlohu segmentácie obrazu, s ktorým budeme pracovať v časti 4.4. Modely tejto triedy sú predstavené [9], [13]a [14] a takisto aj analyzované v [12], [36], [35], [31], [1] a [43].
Segmentačný model sa neskladá len z jednej neurónovej siete, ale z viacerých komponentov, ktoré dohromady zabezpečujú segmentáciu obrazového vstupu. Kľúčové komponenty modelu sú nasledovné [12].

1. **Návrh regiónov (region proposals)** - pomocou vhodného algoritmu sú vygenerované návrhy oblastí, v ktorých sa môže nachádzať objekt (väčšinou ich je aspoň niekoľko tisíc).

2. **Vektor čŕt (feature vector)** - pre každú navrhnutú oblasť (region proposal) sa určia jej časti – črty, ktoré sú dôležité na to, aby sme určili, či sa v nej nachádza objekt. V triede region-based konvolučných neurónových sietí sa na určenie týchto čŕt používa niektorá z hlbokých neurónových sietí, napríklad Resnet50, ktorý sme používali aj na klasifikáciu.

3. **Klasifikácia** - pomocou vektora čŕt (feature vector) je určené, či a aký objekt sa v danej navrhnutej oblasti (region proposal) nachádza. Oblasť môže byť buď označená ako pozadie alebo ako objekt z niektorej z daných tried. Populárnou metódou klasifikácie navrhnutých oblastí (region proposals) pred rozšírením neurónových sietí bývala metóda **support vector machine (SVM)**.

3.1 R-CNN

Prvým zo série modelov predstavených v tejto kapitole je R-CNN. R-CNN je skratka reprezentujúca region-based convolutional network. R-CNN je neurónová sieť, ktorá položila základ celej triedy region-based konvolučných neurónových sietí. Vyvinutá v roku 2014, bola prvým pokusom o vytvorenie modelu pre detekciu objektov, ktorý extrahuje črty pomocou predtrénovanej konvolučnej neurónovej siete [12], [9].

Treba podotknúť, že konvolučná neurónová sieť bola použitá len pri extrahovaní čŕt.
Pre potreby návrhu oblastí bol použitý *selective search for object recognition* a pre klasifikáciu zasa *support vector machine (SVM)*. Dôsledkom toho sú v R-CNN návrhy regiónov, extrakcia čŕt a klasifikácia nezávislými komponentmi viacstupňového modelu. Z toho vyplýva, že R-CNN sa nedá trénovať ako celok, ale je potrebné trénovať jednotlivé komponenty modelu osobitne. Taktiež každý návrh regiónu sa spracúva konvolučnou neurónovou sieťou nezávisle [12].

3.2 Fast R-CNN

Prvým rozšírením R-CNN publikovaným v roku 2015 bola Fast R-CNN. Fast R-CNN predstavila Region of interest (ROI) pooling vrstvu, ktorá zabezpečuje, že z každého navrhnutého regiónu (region proposal) sa extrahuje vektor čŕt (feature vector) rovnakej, fixnej dĺžky. Zároveň zabezpečila, aby návrh regiónov (region proposals), extrakcia vektora čŕt (feature vector extraction) aj klasifikácia boli súčasťou jednej konvolučnej neurónovej siete a bežali súbežne. Konvolučná neurónová sieť extrahujúca vektory čŕt (feature vector) prechádza výpočty cez všetky návrhy regiónov (region proposals).

ROI pooling vrstva

Každý navrhnutý región (region proposal) je rozdelený mriežkou nejakej danej veľkosti. Z každej bunky mriežky je vybraná maximálna hodnota z prvkov vstupnej feature mapy. Dostaneme vektor čŕt (feature vector) fixnej dĺžky závislej od veľkosti mriežky. Vektor čŕt (feature vector) je potom vstupom finálnej FC (fully connected) vrstvy, ktorá je rozdelená do dvoch vetiev:

- softmax vrstva predikujúca skóre pre jednotlivé triedy,
- FC vrstva predikujúca bounding boxy detekovaných objektov [13], [12].

3.3 Faster R-CNN

Faster R-CNN je rozšírením Fast R-CNN z roku 2016. Návrh regiónov (region proposal) je tvorený jednoduchou konvolučnou neurónovou sieťou (region proposal network - RPN). Výpočty sú zdieľané medzi RPN konvolučnou neurónovou sieťou a konvolučnou neurónovou sieťou z Fast R-CNN architektúry.

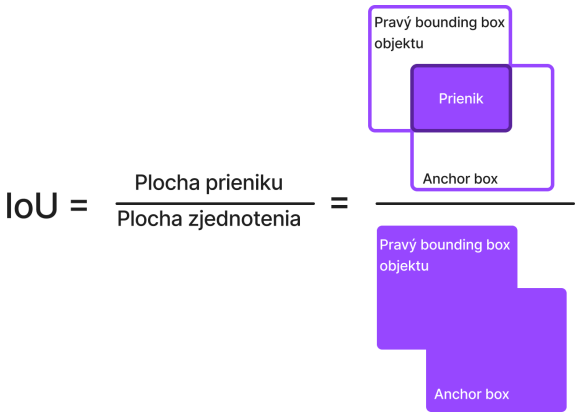
Region proposal network (RPN)

Region proposal network (RPN) funguje na základe využitia tzv. **anchor boxov**. **Anchor box** - bounding box (ohraničená obdĺžniková oblasť v rámci obrázku, ktorá by mala obsahovať objekt) s 2 parametrami - škála, pomer strán (scale, ratio), ktorý parametrizuje návrh regiónu (region proposal).

Zloženie region proposal network (RPN) tvoria štyri základné časti.

1. Vstupom sú čŕty extrahované z minulej konvolučnej vrstvy.
2. Vygenerovanie anchor boxov.
3. Pomocou anchor boxes je vygenerovaný vektor čŕt (feature vector).
4. Výstup tvorí intermediate vrstva tvorená dvoma vetvami
   - regulárna FC - 4D bounding box,
   - FC binárny klasifikátor - podľa hodnoty objectness skóre príradí návrhu regiónu jednu z dvoch tried – pozadie alebo objekt.

Objectness skóre - pri trénovaní RPN dostane každý anchor box kladné alebo záporné objectness skóre pomocou hodnôt ukazovateľa IoU (Intersection-over-Union), pre ktorého hodnoty platí $IoU \in [0, 1]$.



Obr. 10: Znázornenie výpočtu ukazovateľa Interesection over Union.

Hodnoty objectness skóre sú určované na základe nasledujúcich podmienok.

1. Ak má anchor box hodnotu $IoU > 0{,}7$, je mu priradené kladné objectness skóre.

2. Ak žiaden anchor box nemá $IoU > 0{,}7$, potom kladné objectness skóre dostane anchor box s najvyšou hodnotou IoU (ale chceli by sme, aby ten anchor box mal aspoň $IoU \geq 0{,}5$).

3. Ak má anchor box hodnotu $IoU < 0{,}3$ pre všetky ground-truth boxy, tak jeho objectness skóre je záporné.

4. Ak má anchor box $0{,}3 \leq IoU \leq 0{,}5$, tak nemá objectness skóre ani kladné ani záporné.

3.3.1 Trénovanie Faster R-CNN

Tréning architektúry Faster R-CNN je možný v dvoch formách.

1. **Alternating training** - najprv sa natrénuje RPN a až následne Fast R-CNN štruktúra, ako prebieha trénovanie Fast R-CNN zároveň sa aj upravujú váhy pre RPN prostredníctvom spoločných konvolučných vrstiev a takto sa alternuje medzi trénovaním RPN a Fast R-CNN. Váhy spoločných konvolučných vrstiev inicializujeme pomocou predtrénovaných váh natrénovaných na ImageNet datasete, ostatné RPN váhy nastavíme náhodne.

2. **Approximate joint training** - RPN a Fast R-CNN sa berú ako jedna sieť, ktorá sa natrénuje [12].

3.4 Mask R-CNN

Jedným z najnovších rozšírení triedy region-based konvolučných neurónových sietí je Mask R-CNN, ktorého účelom je segmentácia obrazu. Táto štruktúra bola vytvorená ako rozšírenie Faster R-CNN. **Segmentácia obrazu** je identifikácia kontúry/obrysu objektu na obrázku na úrovni jednotlivých pixelov. Pre rozlíšenie si uvedieme rozdiely medzi klasifikáciou, detekciou objektov a segmentáciou.

- **Klasifikácia** - určuje, či sa objekt na obrázku nachádza alebo nenachádza.

- **Sémantická segmentácia** - určí všetky pixely, ktoré prislúchajú danému objektu na obrázku.

- **Detekcia objektov** - určí, že na obrázku sa nachádza n prípadov výskytu daného objektu.

- **Inštačná segmentácia (instance segmentation)** - určí, ktoré pixely na obrázku prislúchajú jednotlivému z n prípadov výskytu daného objektu [36].

Dôležitou štruktúrou pre segmentáciu obrazu, ktorú budeme v našej práci a neskôr aj pri tvorbe segmentačného modelu v časti 4.4 využívať je **Mask regional convolutional neural network** alebo skrátene **Mask R-CNN**. Popis tejto štruktúry bude vychádzať z [10], [1]. Mask R-CNN pozostáva z dvoch fáz - najskôr preskúmanie obrázku a vytvorenie návrhov regiónov (region proposals), a následne klasifikácia navrhnutých regiónov (region proposals) a generácia bounding boxov a masiek.
Mask R-CNN je zložená zo štyroch modulov.

1. **Backbone/Kostra** - kostru tvorí regulárna konvolučná neurónová sieť (v našom prípade Resnet50), ktorá extrahuje čŕty (najskôr veľmi jednoduché, napríklad okraje, rohy a pod. a neskôr už komplikovanejšie ako napríklad auto, človek, obloha, atď.). Jej výstupom je feature mapa (mapa čŕt) o rozmeroch $32 \times 32 \times 2048$.

   Feature pyramid network - rozšírenie pre lepšiu reprezentáciu objektov v rôznych škálach, dovoľuje prístup čŕt (features) v každej vrstve siete prístup k črtám (features) z ostatných vrstiev.

2. **Region proposal network (RPN)** - jednoduchá neurónová sieť, ktorá prechádza obrázok pomocou sliding window, aby našla oblasti, ktorá môžu obsahovať hľadaný objekt. Oblasti, ktoré RPN prechádza (skenuje) sa nazývajú **anchor boxes** a majú rôzne veľkosti a aspect ratios. RPN neskenuje pôvodný obrázok, ale feature mapu, ktorá je výstupom kostry. Pre každý **anchor box** máme dve výstupy - **anchor trieda** – pozadie alebo nepozadie a **zjemnenie bounding boxov** – ktoré odhaduje deltu (percentuálnu zmenu v súradniciach pozorovanej oblasti), aby bounding box čo najpresnejšie fitoval objekt (toto RPN vykonáva len pre anchor boxy, ktorým je priradená trieda objekt).

3. **ROI klasifikátor and bounding box regresor** - berie na vstupe návrhy oblastí (označované aj *Regions of interest* – ROI) z Region proposal network a každému

ROI vráti dva výstupy – **triedu** (zaradí ROI buď do jednej zo špecifických tried klasifikátora alebo do triedy „pozadie“) a **zjemnenie bounding boxov** (podobne ako pri RPN, chceme ďalšie zjemnenie pozície a veľkosti bounding box pre čo najpresnejšie obsiahnutie objektu).

Keďže pri zjemňovaní bounding boxov pracujeme so vstupmi rôznych rozmerov potrebujeme použiť **ROI pooling** - upravenie rozmerov bounding boxov na fixné hodnoty pomocou využitia feature mapy.

4. **Segmentačná vetva** - segmentačná vetva je konvolučná neurónová sieť, ktorá berie na vstupe **pozitívne** regióny vybraté klasifikátorom *ROI* a generuje pre nich masky v nízkom rozlíšení (28x28 pixelov). Ide o generáciu masiek na úrovni pixelov, a teda výstupom sú takzvané **soft** masky.

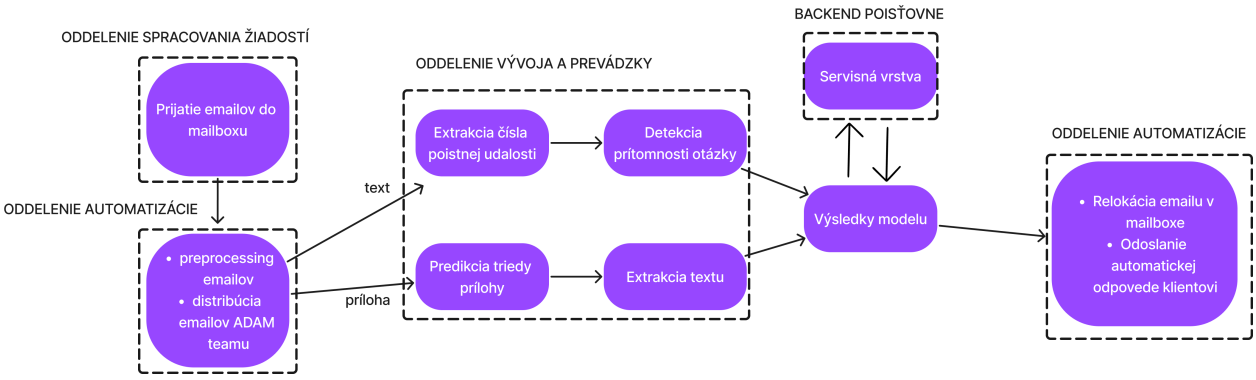
4 Praktické problémy - Likvidácia poistných udalostí

V nasledujúcej kapitole sa budeme venovať praktickému využitiu konvolučných neurónových sietí a to konkrétne v problematike automatizácie spracovania dokumentov pri likvidácií poistných udalostí.

Poistenie vo všeobecnosti podlieha viacerým možným deleniam. Jedným zo základných delení odvetví poistenia podľa slovenskej legislatívy je delenie na **životné** a **neživotné** [42]. **Životné poistenie** sa týka výhradne poistenia ľudského života, teda smrti alebo dožitia. **Neživotné poistenie** pokrýva poistenie všetkých ostatných rizík - napríklad zdravie človeka (úrazy, choroby a pod.), poškodenie hnuteľného a nehnuteľného majetku, zodpovednosť za škodu, záujmy poistníka (právna ochrana, pohľadávky, úvery a pod.) a pod. [28], [40].

V našej práci sa venujeme likvidácii poistných udalostí v rámci produktov neživotného poistenia. Spolupracujeme s oddelením spracovania žiadostí ČSOB, ktoré má na starosti komunikáciu s klientom počas doby likvidácie jeho poistnej udalosti. Každý deň oddelenie spracuje desiatky e-mailov s klientskými otázkami, informáciami a prílohami dokumentov dôležitými pre likvidáciu poistných udalostí. Mesačný priemer za rok 2023 je 5000 e-mailov, z toho 3200 e-mailov s prílohami. Predovšetkým manuálne spracovanie príloh - ich správne zatriedenie a následné nahrávanie do databázy - tvorí monotónnu a časovo náročnú činnosť. Likvidácia poistných udalostí je taktiež časovo obmedzená zákonnou lehotou. S využitím moderných štatistických nástrojov však vieme aspoň čiastočne zautomatizovať spracovanie e-mailov prijatých oddelením spracovania žiadostí, čo môže jednak urýchliť spracovanie e-mailov ako aj alokovať energiu a čas zamestnancov iným, produktívnejším úlohám.

Spolupracujeme aj s oddelením automatizácie, ktorý zabezpečuje vytiahnutie doručených e-mailov z mailboxu a ich následne poslanie do vstupu modelu oddelenia vývoja a prevádzky, ktorý vyvinieme na ich spracovanie. Po spracovaní e-mailov a ich príloh sú prílohy (a prípadné iné informácie zo spracovaných e-mailov a príloh) nahrané do servisnej vrstvy backendu poisťovne a oddelenie automatizácie sa postará o odoslanie automatickej odpovede klientovi a relokáciu spracovaných e-mailov.



Obr. 11: Znázornenie procesu automatizácie spracovania emailov helpdesku oddelením spracovania žiadostí. E-mailová komunikácia prijatá oddelením spracovania žiadostí je oddelením automatizácie presmerovaná do modelu oddelenia vývoja a prevádzky, ktorý e-maily a ich prílohy spracuje. Výsledky modelu sú následne spracované backendom poisťovne a klientovi je odoslaná automatická odpoveď o spracovaní jeho e-mailu.

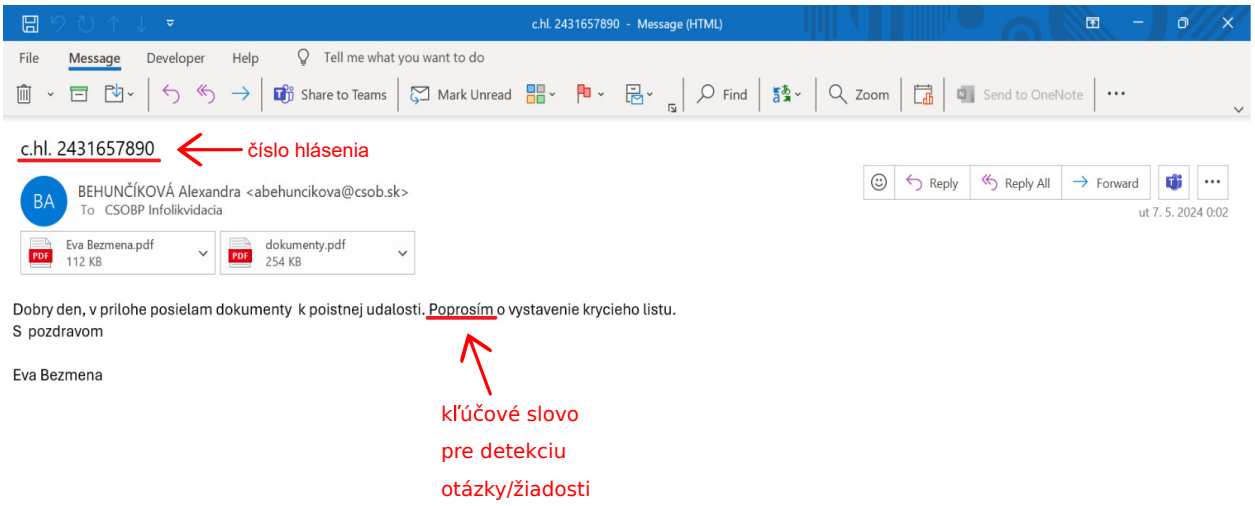
Najskôr si predstavíme niektoré dôležité praktické problémy, ktorým sme potrebovali čeliť pri spracovaní emailov a ich príloh. Ich riešenia nám ďalej poslúžia pri klasifikácií príloh ako aj pri celkovej automatizácii ich spracovania počas procesu likvidácie poistných udalostí.

4.1 Extrakcia eml súborov

Pri likvidácii poistných udalostí je od účastníkov poistných udalostí adresovaných poisťovni množstvo e-mailov obsahujúcich doplňujúce detaily alebo prílohy dôležité pre likvidáciu. Na komunikáciu s klientmi je využívaný Microsoft Outlook, kde každý prichádzajúci e-mail vieme spolu so všetkými jeho súčasťami uložiť vo formáte eml súboru. Mali sme k dispozícii 707 eml súborov, z ktorých sme sa snažili extrahovať časti dôležité pre oddelenie spracovania žiadostí, t.j.:

- predmet e-mailu,
- text e-mailu,
- odosielateľa e-mailu,
- príjemcu/adresáta e-mailu,
- dátum poslania,

- html e-mailu,
- prílohy e-mailu.



Obr. 12: Príklad e-mailu prijatého oddelením spracovania žiadostí. Číslo hlásenia nezodpovedá jeho reálnemu formátu.

Extrahované prílohy, texty a predmety emailov sme následne využili na ďalšiu analýzu a vyťažovanie podrobnejších informácií. Funkciu použitú na extrakciu uvádzame v prílohe B.

4.1.1 Vyťažovanie čísel poistných udalostí

Z textov a predmetov emailov sme sa snažili vyťažiť čísla poistných udalostí a poistných hlásení. V prípade čísla poistnej udalosti ako aj čísla poistného hlásenia išlo o desaťmiestne číslo. Bolo potrebné jednotlivé čísla od seba odlíšiť navzájom, ale predovšetkým obe odlíšiť od rodného čísla, na ktoré sa obe svojou štruktúrou podobali. Obe hľadané čísla sa vyznačovali špecifickou štruktúrou, ktorú bolo potrebné vziať do úvahy pri ich detekcii. Z dôvodu internej bezpečnosti neuvádzame presnú štruktúru vyťažovaných čísel. Správne vyťaženie čísla poistnej udalostí a čísla poistného hlásenia je dôležité pre priradenie mailovej komunikácie a jej príloh k správnej likvidovanej poistnej udalosti. Obe čísla figurujú

väčšinou v texte alebo predmete e-mailu, veľmi zriedkavo aj ako názov prílohy e-mailu. V prípade, že čísla poistných udalostí aj čísla poistných hlásení boli prítomné v e-mailovej komunikácie, sa nám podarilo obe vyťažovať so 100 % úspešnosťou.

4.1.2 Hľadanie otázok

V e-mailovej komunikácii s oddelením spracovania žiadostí sa prirodzene vyskytuje množstvo otázok a požiadaviek ohľadom likvidačného procesu. Je preto potrebné všetky tieto otázky a požiadavky zachytiť a zabezpečiť, že budú adresované. Preto všetky e-maily, v ktorých bude detekovaná otázka alebo požiadavka, budú okrem spracovania našim modelom ešte dodatočne špeciálne označené. K takto označeným e-mailom sa následne vráti zamestnanec poisťovne, ktorý adresuje otázky alebo požiadavky uvedené v texte e-mailu. Samotné vyhľadávanie otázok sa bude viazať na text a predmet e-mailu. Používame regulárne výrazy (alebo skrátené regexy) – postupnosti znakov, ktoré budú vyhľadávané v texte a predmete e-mailu. Na vyhľadávanie regulárnych výrazov v texte sme použili `python` knižnicu `re`, ktorá v sebe zahŕňa operácie pre regulárne výrazy. Ako kľúčové slová, ktorých prítomnosť budeme sledovať, sme zvolili množinu: „ako“, „kedy“, „preco“, „kde“, „prosim“, „prosime“, „ziadam“, „ziadame“. Osobitne sme ešte hľadali prítomnosť otáznika. Otázka/požiadavka bola detekovaná v 37,7% zo všetkých eml súborov. Otáznik bol identifikovaný v 5,7% súborov a nejaké z kľúčových slov v 35,8% súborov.
Vo všeobecnosti sme sa snažili, aby každá otázka či požiadavka bola detekovaná. Prípady, kedy by sme detekovali otázku či požiadavku v e-maile, v ktorom sa žiadna nenachádza, sú prípustnejšie ako e-maily obsahujúce otázku či požiadavku, v ktorých ale nebola detekovaná. V každom prípade bola cieľom čo najvyššia presnosť. Proces detekcie otázok a požiadaviek v e-mailovej komunikácii sa neustále vyvíja. Väčšinou, ak po určitom čase zaznamenáme frázu či kľúčové slovo, ktoré relatívne často vystupuje ako spoľahlivý ukazovateľ prítomnosti otázky alebo požiadavky, pridáme ich do existujúcej množiny kľúčových slov. Naopak texty e-mailov niekedy obsahujú automatické zakončenia či upozornenia, ktoré vedia spôsobiť falošne pozitívnu detekciu otázky. Výrazy z takýchto formulácií zasa zaradíme medzi zakázané výrazy a vyradíme tak e-maily, ktoré obsahujú iba takýto výraz, ale žiadne iné kľúčové slovo spomedzi e-mailov s detekovanou otázkou. Vo všeobecnosti sa ťažko určuje presnosť detekcie otázok/požiadaviek, keďže variabilita e-mailovej komuni-

kácie je veľká a výsledok tak závisí na zvolenej vzorke. Na vzorke 140 náhodne vybraných e-mailov však presnosť identifikácie prítomnosti otázky/požiadavky v e-maily bola 96,5 %, pričom v štyroch e-mailoch bola falošne identifikovaná otázka a jednom nebola zachytená v dôsledku veľmi špecifickej verbálnej formulácie.

4.2 Klasifikácia príloh pri likvidácii poistných udalostí pomocou konvolučnej neurónovej siete

4.2.1 Dáta

V rámci automatizácie nahrávania dokumentov z príloh e-mailov k likvidácii poistných udalostí by sme radi čo najpresnejšie klasifikovali jednotlivé prílohy do príslušných tried. Prílohy sú vo valnej väčšine prípadov tvorené obrázkami v bežných formátoch a pdf súbormi. Pdf súbory mohli byť tvorené buď jednou triedou prílohy (napríklad kalkulácie alebo niektoré faktúry sa vyznačovali tým, že mali viac strán) alebo v sebe obsahovať viacero tried príloh. V prípade pdf súborov sme ku každej stránke súboru pristupovali ako k samostatnému obrázku. Klasifikátor bol zostrojený pre klasifikáciu na úrovni obrázkov resp. stránok, pri čom každému obrázku prislúcha práve jedna vizuálna trieda.
V prípade viac-stranových príloh sa po klasifikácii jednotlivých strán pristupuje k finálnej agregácií výsledkov modelu na úrovni dokumentu. Pre začiatok sme uvažovali 3 skupiny sprievodných dokumentov rozdelené do 15 vizuálnych tried

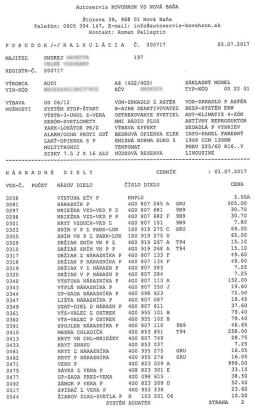
- dokumenty - triedy **bločky**, **faktúra**, **kalkulácia typu A**, **kalkulácia typu B**, **dokument**, **správa o nehode**,
- fotky - triedy **fotodetail**, **odometer**, **vin povrch auta**,
- doklady - vodičský preukaz (triedy **vodičák predná strana**, **vodičák zadná strana**), malý technický preukaz (triedy **mtp predná strana**, **mtp zadná strana**), veľký technický preukaz (triedy **vtp predná strana**, **vtp zadná strana**).



(a) blocky



(b)faktura



(c) kalkulacia typu A



(d) VIN povrch auta

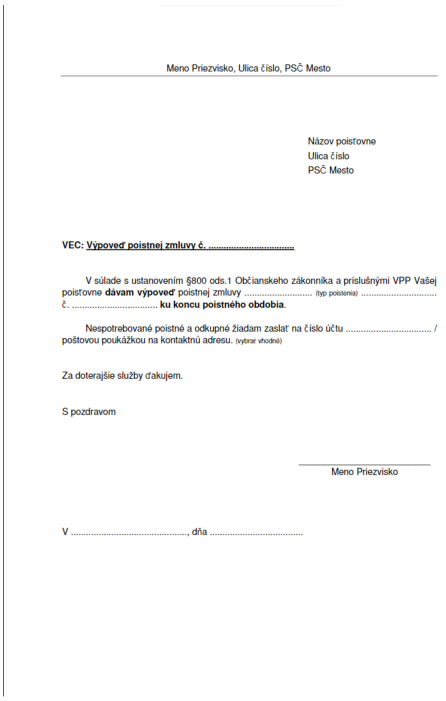


(e) mtp predna strana



(f) mtp zadna strana

Obr. 13: Vizuálne triedy klasifikovaných dokumentov. Všetky príklady jednotlivých tried sú získané z verejne voľne dostupných zdrojov a neprezentujú žiadne reálne osobné údaje.
Pre triedu kalkulácia typu B sa nám nepodarilo získať príklad, ktorý by spĺňal tieto požiadavky.
Zdroje: [23], [3], [18], [19].


Meno Priezvisko, Ulica číslo, PSČ Mesto

Názov poisťovne
Ulica číslo
PSČ Mesto

VEC: Výpoveď poistnej zmluvy č. .....

V súlade s ustanovením §800 ods.1 Občianskeho zákonníka a príslušnými VPP Vašej poisťovne **dávam výpoveď** poistnej zmluvy ..... (typ poistenia) ..... č. ..... **ku koncu poistného obdobia**.

Nespotrebované poistné a odkupné žiadam zaslať na číslo účtu ..... / poštovou poukážkou na kontaktnú adresu. (vybrať vhodné)

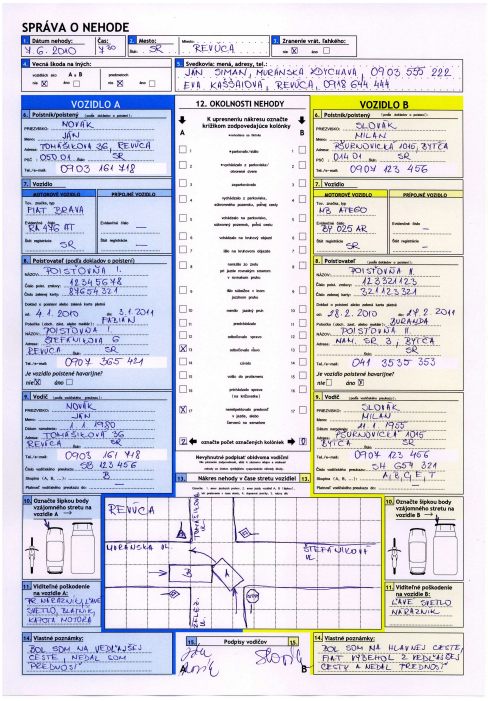
Za doterajšie služby ďakujem.

S pozdravom

Meno Priezvisko

V ....., dňa .....

(g) dokument



(h) sprava o nehode



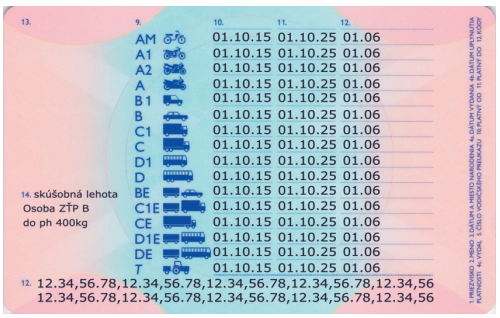
(i) odometer



(j) fotodetail



(k) vodicak predna strana

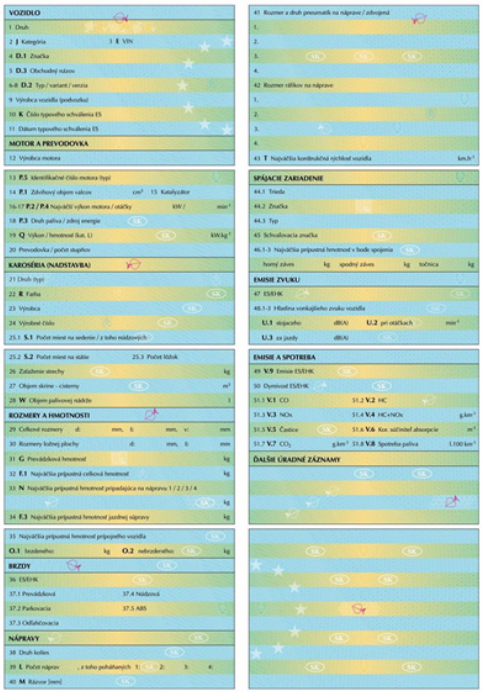


(l) vodicak zadna strana

Obr. 14: Vizuálne triedy klasifikovaných dokumentov. Všetky príklady jednotlivých tried sú získané z verejne voľne dostupných zdrojov a neprezentujú žiadne reálne osobné údaje. Zdroje: [23], [2], [24].



(m) vtp predna strana



(n) vtp zadna strana



(o) op predna strana



(p) op zadna strana

Obr. 15: Vizuálne triedy klasifikovaných dokumentov. Všetky príklady jednotlivých tried sú získané z verejne voľne dostupných zdrojov a neprezentujú žiadne reálne osobné údaje.
Zdroje: [23], [5].

Pre skupinu dokladov sme pre každý typ dokladu zaviedli dve vizuálne triedy – prednú a zadnú stranu. Urobili sme tak preto, že jednotlivé strany dokladov považujeme za dostatočne vizuálne odlišné a takéto rozdelenie za prospešné pre výkon klasifikátora. Vizuálne triedy pre prednú a zadnú stranu dokladu dodatočne agregujeme na jednu dokumentovú triedu, keďže pre nás je primárne dôležité, aký druh dokladu sa na obrázku nachádza a nie či ide o prednú alebo zadnú stranu.

Neskôr sme sa rozhodli pridať osobitnú triedu **vodičák predná zadná strana**, keďže sme pozorovali nezanedbateľný výskyt príloh, ktoré v rámci jednej stránky resp. jedného obrázku obsahovali obe strany vodičského preukazu. V závislosti od priestorového usporiadania fotiek prednej a zadnej strany vodičského preukazu na obrázku a farieb obrázku boli takéto prílohy buď klasifikované do triedy odlišnej od oboch tried pre vodičský preukaz (t.j. **vodičák zadná strana** a **vodičák predná strana**) alebo iba do jednej z týchto tried. Takto sme trénovali klasifikátor pre 16 tried.
Po získaní vhodného množstva dát boli pridané aj triedy pre občiansky preukaz – **op predná strana**, **op zadná strana**. A následne sme triedu **vodičák predná zadná strana** nahradili triedou **multicard**. Trieda **multicard** zahŕňala všetky obrázky, ktoré obsahovali v rámci jedného obrázka viac ako jednu fotku či sken kartičkovej prílohy (t.j. občiansky preukaz, vodičský preukaz, malý technický preukaz). Môže ísť o obrázok, na ktorom sú obe strany jedného druhu preukazu alebo aj rôzne druhy preukazov naraz. Finálny klasifikačný model tak pozostával z 18 vizuálnych tried.

4.2.2 Model

Ako klasifikačný model sme zvolili architektúru konvolučnej neurónovej siete - konvolučnú neurónovú sieť Resnet50. Ide o veľmi efektívnu architektúru zo skupiny residuálnych konvolučných sietí. Táto architektúra predstavuje dobrý kompromis medzi presnosťou predikcií a výpočtovou náročnosťou tréningu a predikcií. Vieme ju taktiež používať v kombinácií s region-based architektúrami predstavenými v kapitole 3, napríklad v našom prípade slúži Resnet50 ako kostra pre segmentačný model používaný neskôr v časti 4.4. Resnet50 bola pôvodne natrénovaná na známom datasete ImageNet. ImageNet tvoria rádovo stotisíce anotovaných obrázkov [17]. Pomocou princípu transfer learning sme schopní predtrénovaný model Resnet50 dotrénovať pre naše potreby klasifikácie len pomocou trénovacieho datasetu, ktorého veľkosť sa pohybuje v stovkách – tisíckach objektov. Pre implementáciu nášho klasifikačného modelu sme použili open-source `python` knižnicu `keras`. Knižnica `keras` slúži ako rozhranie pre prácu s neurónovými sieťami. Prostredníctvom `keras` rozhrania sme ďalej pristupovali k architektúre modelu ako aj ďalším stavebným blokom potrebným pre prácu s neurónovými sieťami.

4.2.3 Augmentácie

Jedným z veľmi účinných prostriedkov pre zvýšenie úspešnosti klasifikácie pomocou konvolučných neurónových sietí je pridanie augumentácií. Vo všeobecnosti ide o účinný postup, obzvlášť keď máme k dispozícii len malé trénovacie vzorky. K pôvodnej dátovej sade pridajú augmentácie vizuálnu variáciu. Pri prvom pohľade na počty dát v jednotlivých triedach sa zdá, že u väčšiny tried potenciálny nedostatok dát nehrozí. Problém však rozpoznáme, keď si uvedomíme, že niektoré dokumenty v jednotlivých triedach majú rôznu úroveň vizuálnej rozmanitosti. Ako príklad si môžeme zobrať napríklad triedu faktúry, kde napriek zdieľaným základným znakom sú od seba navzájom vizuálne jednotlivé faktúry značne odlišné. Ďalšou takouto triedou sú napríklad odometre alebo VIN čísla. Okrem vizuálnej variability samotných fyzických dokumentov je tu taktiež aj vizuálna variabilita nadobudnutá ich prenosom do digitálnej podoby (fotografia, fotokópia, sken, ...). Rôzne natočenie, posunutie, osvetlenie, pozadie a mnoho iných premenných, ktoré ovplyvnia ľudskú schopnosť klasifikácie veľakrát iba v extrémnych prípadoch, vedú často k rôznym klasifikáciam toho istého dokumentu konvolučnou neurónovou sieťou.
Existuje množstvo spôsobov augmentácie obrázka. My si predstavíme augmentácie, ktoré by mohli byť relevantné pri vizuálnom spracovaní dokumentov a ktoré sme aj v rôznej miere použili pri dokumentoch dôležitých pre likvidáciu poistných udalostí.



(a) **Otočenie**



(b) **Zoom**

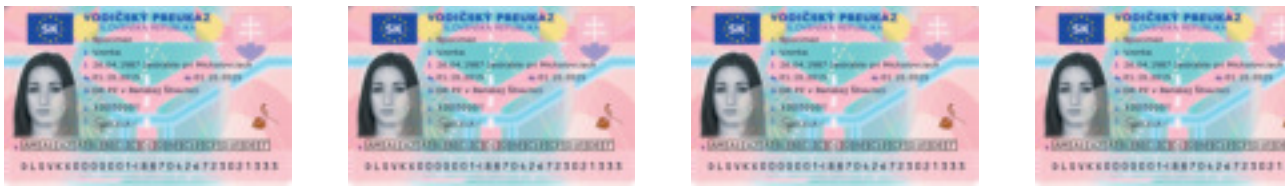
Obr. 16: Príklad augmentácií – časť 1.



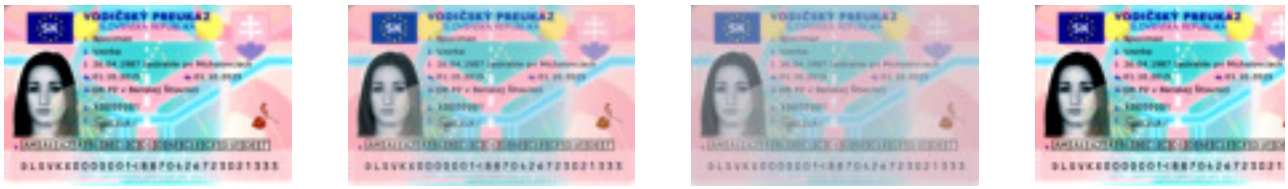
(c) **Shearing**



(d) **Posunutie**



(e) **Šum**



(f) **Zmena kontrastu**



(g) **Zmena farebných kanálov**

Obr. 17: Príklad augmentácií – časť 2.

Správnou kombináciou vstavaných augmentácií v knižniciach `keras` a `tensorflow` vieme dosiahnuť nielen replikáciu rôznych nedokonalostí vzniknutých pri fotení rôznych dokumentov ale aj napríklad efekt čierno-bieleho skenu pri kartičkových dokladoch.

4.2.4 Metrika úspešnosti

Presnosť klasifikácie nami natrénovaných modelov sme pozorovali pomocou confusion matrix na testovacom datasete. Confusion matrix predstavuje zobrazenie predikovaného priradenia jednotlivých obrázkov do tried klasifikátora oproti ich skutočnému priradeniu do tried. Nadobúda podobu matice, v ktorej každý riadok/stĺpec zodpovedá jednej z predikovaných/skutočných vizuálnych tried klasifikátora. Confusion matrix sme využívali pre vizualizáciu predikcií modelu, keď sme chceli identifikovať problematické vizuálne triedy.

Pre porovnávanie jednotlivých modelov medzi sebou sme používali ukazovateľ accuracy, ktorý udával, koľko percent objektov v testovacom datasete bolo modelom priradených správnej triede

$$accuracy = \frac{\text{počet obrázkov priradených správnej triede}}{\text{celkový počet objektov testovacieho datasetu}} \times 100\%. \tag{88}$$

4.2.5 Výsledky

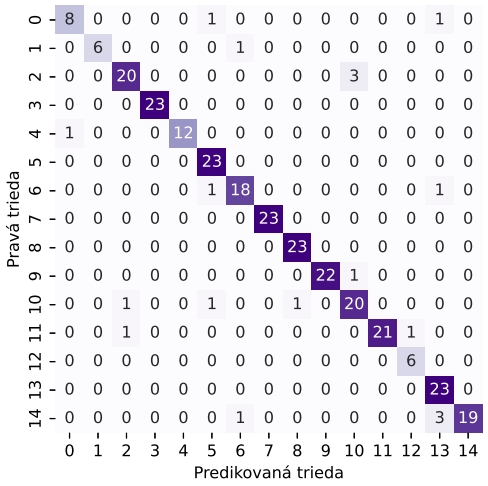
Model sme vo väčšine prípadov najprv trénovali v podobe **base freezed** – to znamená, že sa trénovali iba parametre z vrchnej časti modelu, ostatné parametre (v báze modelu) boli „zmrazené“ resp. pri tréningu neaktívne. Následne sme dotrénovali model pre všetky parametre – v podobe **base unfreezed**. Ako môžeme vidieť na Obr. 18 - 21, tento postup mal vo väčšine prípadov lepšie výsledky pre **base unfreezed** modely.

Počiatočné tréningy klasifikačného modelu pre 15 tried ukazovali sľubné výsledky, ako môžeme vidieť aj na Obr.18 – accuracy pre **base freezed** bola 93,4 % a pre **base unfreezed** 94,8 % . Nebrali však do úvahy prípady, keď bolo na jednom obrázku znázornených viac kartičiek. Tento problém sme sa najskôr pokúšali vyriešiť pridaním novej triedy **vodičák predná zadná strana**, keďže väčšinou sme najviac bádali tento jav pri vodičských preukazoch.

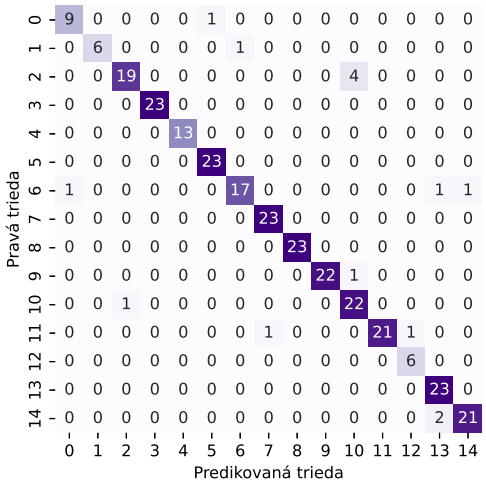
V modeloch so 16 triedami sme pozorovali vplyv využitia augmentácií v ich rôznom rozsahu avšak použité plošne na všetky vizuálne triedy naraz. Ako môžeme vidieť na Obr.19,

malé percento (10 %) augmentácií malo skôr negatívny efekt pre výkon klasifikátora – accuracy pre **base freezed** bola 70 %, pre **base unfreezed** 74,1 %. To mohlo byť spôsobené tým, že k už vizuálne rozmanitým dátam iba pridal ďalšiu variabilitu, ktorá však nebola v testovacom datasete zastúpená dostačne na to, aby sa ju neurónová sieť dokázala naučiť. Pre vyššie percento použitých augmentácií (40 %) už výsledky klasifikátora (Obr.21) na testovacom datasete boli o veľa lepšie – accuracy pre **base freezed** bola 85,5 %, pre **base unfreezed** 86,6 % – no stále porovnateľné s výsledkami klasifikátora bez použitia augmentácií (Obr.19) – kde accuracy pre **base freezed** bola 85,5 % a pre **base unfreezed** 87,9 %. Ukázalo sa, že najvhodnejšie je využívať augmentácie len pri určitých vizuálnych triedach, ak identifikujeme v dátach vzorec, ktorý nie je dostatočné zastúpený v trénovacej vzorke avšak zároveň nie je zanedbateľný pre úspešnosť predikcií. Toto zistenie sme aplikovali aj pri tréningu finálneho modelu.

Zároveň po otestovaní našich 16 triednych modelov na nových prichádzajúcich dátach, sa stala zrejmou potreba zaviesť nové triedy pre občiansky preukaz (**op predná strana** a **op zadná strana**). Tieto triedy neboli zaradené do predchádzajúcich modelov z dôvodu ich zanedbateľného množstva v pôvodných dátach využívaných pre tréning. Zároveň s touto zmenou počtu vizuálnych tried sme sa rozhodli nahradiť triedu **vodičák predná zadná strana** triedou **multicard**. Vizuálna trieda **multicard** brala do úvahy všetky prílohy obsahujúce viacero kartičiek v jednom obrázku namiesto toho, aby brala do úvahy iba prílohy obsahujúce výhradne len vodičské preukazy. Finálny počet vizuálnych tried, s ktorými sme pracovali bol 18. Trénovací dataset bol obohatený o nové dáta a augmentácie boli použité len pri triedach zo skupiny dokladov a pri triede **správa o nehode**. Finálny model (Obr.22) trénovaný pre 18 tried, na 6120 objektoch počas 3 epoch, mal accuracy vizuálnych tried 91 %. Pri sledovaní upravenej accuracy, pri ktorej sme nekládli dôraz na to, či šlo o prednú alebo zadnú stranu dokladu, bola presnosť finálneho modelu 96 %.

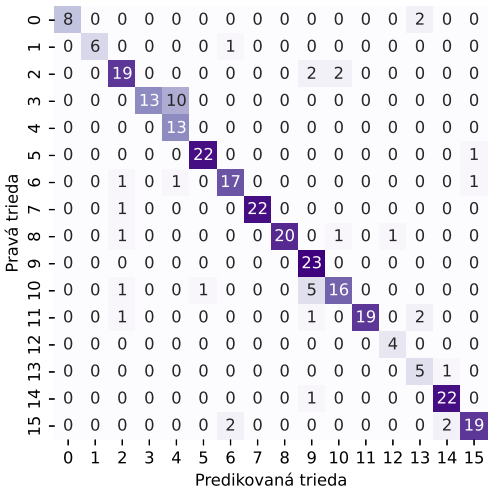


base freezed (8 epoch)

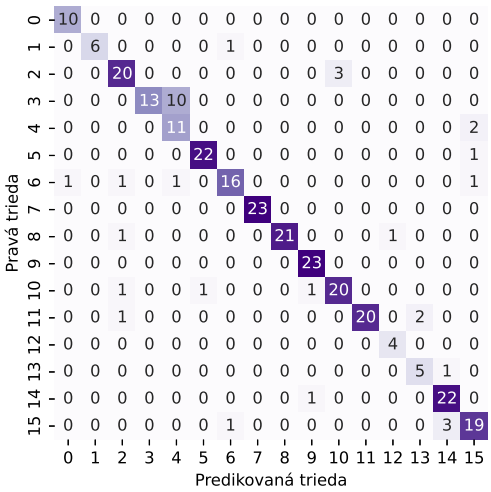


base unfreezed (5 epoch)

Obr. 18: Confusion matrix pre model trénovaný pre 15 tried na testovacom datasete o veľkosti 1905 objektov. Jednotlivým číslam tried zodpovedali vizuálne triedy - 0: blocky, 1: dokument, 2: fotodetail, 3: kalkulacia typu A, 4: kalkulacia typu B, 5: správa o nehode, 6: faktura, 7: mtp predna strana, 8: mtp zadna strana , 9: odometer, 10: vin povrch auta, 11: vodicak predna strana, 12: vodicak zadna strana, 13: vtp predna strana, 14: vtp zadna strana. Accuracy pre **base freezed** je 93,4 %, pre **base unfreezed** 94,8 %.

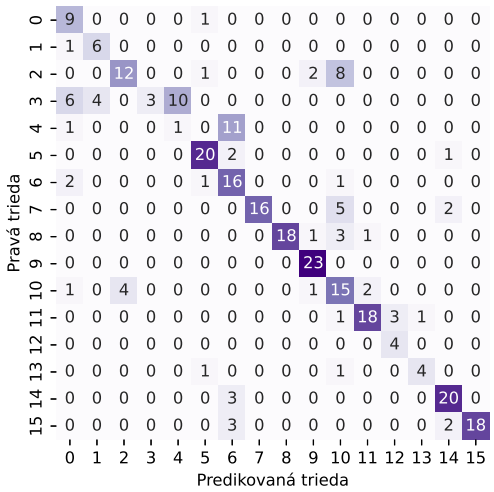


base freezed (8 epoch)

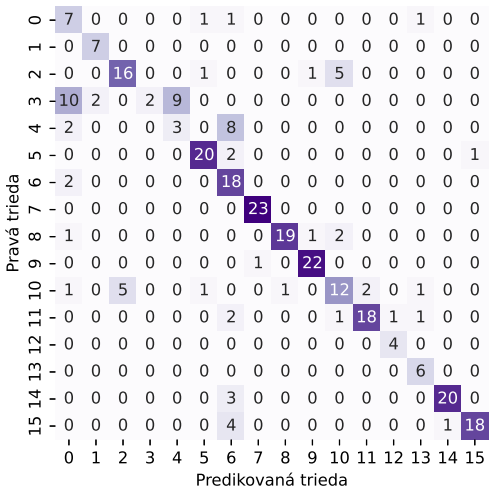


base unfreezed (5 epoch)

Obr. 19: Confusion matrix pre model trénovaný pre 16 tried na testovacom datasete o veľkosti 2032 objektov, bez augmentácií. Jednotlivým číslam tried zodpovedali vizuálne triedy - 0: blocky, 1: dokument, 2: fotodetail, 3: kalkulacia typu A, 4: kalkulacia typu B, 5: správa o nehode, 6: faktura, 7: mtp predna strana, 8: mtp zadna strana, 9: odometer, 10: vin povrch auta, 11: vodicak predna strana, 12: vodicak predna zadna strana, 13: vodicak zadna strana, 14: vtp predna strana, 15: vtp zadna strana. Accuracy pre **base freezed** je 85,5 %, pre **base unfreezed** 87,9 %.

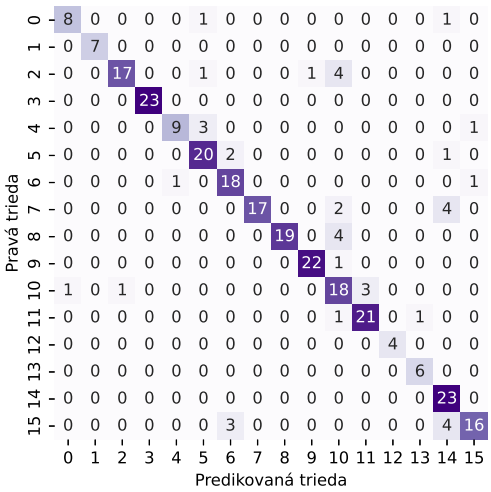


base freezed (8 epoch)

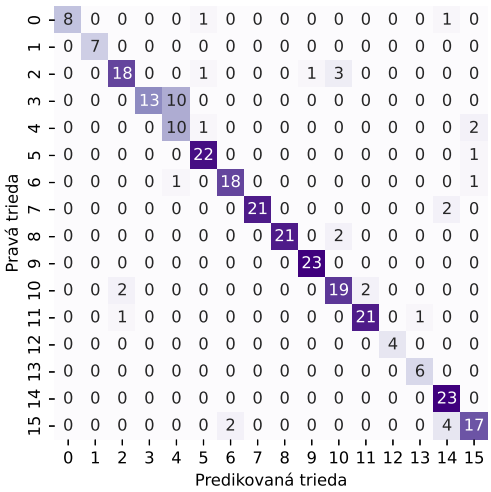


base unfreezed (5 epoch)

Obr. 20: Confusion matrix pre model trénovaný pre 16 tried na testovacom datasete o pôvodnej veľkosti 2032 objektov. Na testovací dataset bol použitý upsampling pomocou augmentácií použitých na 10% pôvodného testovacieho datasetu. Jednotlivým číslam tried zodpovedali vizuálne triedy - 0: blocky, 1: dokument, 2: fotodetail, 3: kalkulacia typu A, 4: kalkulacia typu B, 5: správa o nehode, 6: faktura, 7: mtp predna strana, 8: mtp zadna strana, 9: odometer, 10: vin povrch auta, 11: vodicak predna strana, 12: vodicak predna zadna strana, 13: vodicak zadna strana, 14: vtp predna strana, 15: vtp zadna strana. Accuracy pre **base freezed** je 70 %, pre **base unfreezed** 74,1 %.



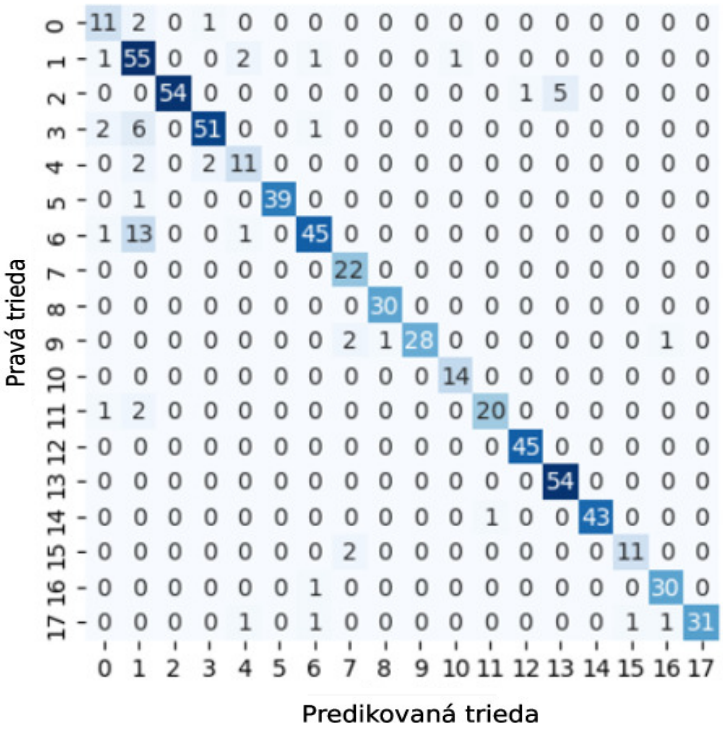
base freezed (8 epoch)



base unfreezed (5 epoch)

Obr. 21: Confusion matrix pre model trénovaný pre 16 tried na testovacom datasete o pôvodnej veľkosti 2032 objektov. Na testovací dataset bol použitý upsampling pomocou augmentácií použitých na 40% pôvodného testovacieho datasetu. Jednotlivým číslam tried zodpovedali vizuálne triedy - 0: blocky, 1: dokument, 2: fotodetail, 3: kalkulacia typu A, 4: kalkulacia typu B, 5: správa o nehode, 6: faktura, 7: mtp predna strana, 8: mtp zadna strana, 9: odometer, 10: vin povrch auta, 11: vodicak predna strana, 12: vodicak predna zadna strana, 13: vodicak zadna strana, 14: vtp predna strana, 15: vtp zadna strana. Accuracy pre **base freezed** je 85,5 %, pre **base unfreezed** 86,6 %

.



Obr. 22: Finálny model, 18 tried, 3 epochy, **base freezed**, veľkosť testovacieho datasetu 6120. Jednotlivým číslam tried zodpovedali vizuálne triedy 0: blocky, 1: dokument, 2: fotodetail, 3: kalkulacia typu A, 4: kalkulacia typu B, 5: správa o nehode, 6: faktura, 7: multicard, 8: mtp predna strana, 9: mtp zadna strana, 10: op predna strana, 11: op zadna strana, 12: odometer, 13: vin povrch auta, 14: vodicak predna strana, 15: vodicak zadna strana, 16: vtp predna strana, 17: vtp zadna strana. Najväčší problém pri predikcii tvorila nesprávna klasifikácia triedy **faktura** a triedy **kalkulacia typu A** ako triedu **dokument**. Tuto chybnú klasifikáciu ale vieme dodatočne vyriešiť hľadaním kľúčových slov v prílohách triedy **faktura** a triedy **kalkulacia typu A**.

4.3 Využitie OCR pri klasifikácii dokumentov a vyťažovaní konkrétnych informácií z niektorých druhov dokumentov

Optical character recognition (OCR) je proces extrakcie textu čitateľného počítačom z textu vo forme obrázku alebo fotografie. Zjednodušene premena vizuálnej informácie zo vstupného obrázku na počítačovom akceptovateľné znaky [33]. Pre využívanie OCR ňašom modeli pracujeme s `python` knižnicou `pytesseract`.

4.3.1 Využitie OCR a kľúčových slov pri triedení dokumentov

Niektoré triedy sprievodných dokumentov sú si vizuálne podobné svojou štruktúrou. V niektorých prípadoch dokumentové prílohy však vieme relatívne jednoducho rozlíšiť pomocou určitých kľúčových slov, ktoré sú význačné pre jednotlivé druhy príloh. Pomocou OCR vieme extrahovať text z príloh v obrázkovom formáte a následne využitím knižnice `re` vyhodnotiť, či daný dokument obsahuje kľúčové slová nejakej triedy, čo nám pomôže priradiť mu jeho potenciálne správnu triedu.

Jedna z pôvodných tried emailových príloh – trieda **dokument** – bola význačná tým, že v skutočnosti zastrešovala viacero druhov príloh. Identifikáciu založenú na hľadaní kľúčových slov sme využili pri dodatočnej klasifikácií príloh z triedy **dokument** podtried uvedených v tabuľke 1. Ako vzorec pre regulárny výraz zodpovedajúci IBAN-u sme použili štruktúry IBAN-ov Slovenskej republiky a jej susedných krajín.

Podtrieda dokumentu	Početnosť
splnomocnenie	13
čestné vyhlásenie:	3
žiadosť o krycí list	21
zápis o poškodení	13
platobný príkaz	3
uplatnenie poistného voči zodpovednému za škodu/ uplatnenie nároku poškodeného	7
odborný posudok	1
oznámenie o poukazaní poistného plnenie/informácia o čísle účtu	5
výpoveď/zmena poistnej zmluvy	57
vyzva na predloženie dokladov	2

Tabuľka 1: Súhrn podtried triedy **dokumenty**, početností ich výskytu vo vzorke 707 eml súborov, na základe ktorej sme tvorili náš model a volili kľúčové slová použité na identifikáciu jednotlivých podtried. Konkrétne kľúčové slová z bezpečnostných dôvodov neuvádzame.

4.3.2 Využitie OCR pri vyťažovaní konkrétnych informácií z dokumentov

Postup založený na OCR a následnom vyhľadávaní kľúčových slov sme využili aj pri získavaní celkových súm z kalkulácií. Identifikácia kľúčového slova pomôže určiť umiestnenie celkovej sumy kalkulácie v rámci spracovaného dokumentu, keďže kľúčové slova sú navrhnuté tak, aby boli zovšeobecnením pre frázy, ktoré vizuálne čo najbližšie obklopujú hodnotu celkovej sumy v prílohe. Mali sme dva hlavné druhy kalkulácií, každý so svojimi vizuálnymi špecifikami a kľúčovými slovami, ktoré sú znázornené v tabuľke 2.

Podtrieda dokumentu	Početnosť	Úspešnosť extrakcie sumy	Dĺžka OCR (v minútach)
Kalkulácie typu B	76	98%	13
Kalkulácie typu A	474	100%	40

Tabuľka 2: Vyťažovanie celkovej sumy z kalkulácie. Početnosti výskytu **kalkulácie typu A** a **kalkulácie typu B** vo vzorke použitej na tvorbu modelu, dĺžka trvania OCR pre jednotlivé typy kalkulácií. Konkrétne kľúčové slová z bezpečnostných dôvodov neuvádzame. Celková presnosť vyťažených celkových súm (t.j. vyťažené celkové sumy, ktoré zodpovedali skutočnej celkovej sume) pre obe triedy bola zhruba 95 %.

Pokúšali sme sa aj o vyťažovanie celkovej sumy z triedy faktúry. Problémom tejto triedy však bola veľká vizuálna a lexikálna variabilita jednotlivých faktúr (pre správne fungovanie OCR by boli vhodné rôzne nastavenia tesseractu pre jednotlivé dokumenty) a ťažké zovšeobecnenie kľúčových slov. Tieto skutočnosti spôsobili ťažkosti pri extrakcii kandidátov na celkovú sumu. Preto sme pristúpili k pridaniu dodatočných podmienok – požadovaná prítomnosť slova „faktúra“ a prítomnosť kandidáta na celkovú sumu – a pracovali na extrakcii celkovej sumy iba z dokumentov, ktoré tieto dve podmienky spĺňali.

Podtrieda dokumentu	Početnosť	Úspešnosť extrakcie sumy	Dĺžka OCR (v minútach)
Faktúra	102	89%	29

Tabuľka 3: Vyťažovanie celkovej sumy faktúry. Početnosť výskytu faktúr vo vzorke použitej na vývoj modelu, dĺžka trvania OCR pre danú početnosť a percentuálna úspešnosť vyťaženia celkovej sumy pre použité kľúčové slová. Konkrétne kľúčové slová z bezpečnostných dôvodov neuvádzame.

4.4 Segmentácia obrázkov

Pri klasifikácii dokumentov sa ukázalo, že množstvo chýb sa u všetkých iterácií nášho klasifikačného modelu vyskytlo pri dokumentoch typu kartičky (t.j. občianske preukazy, vodičské preukazy a malé technické preukazy). U týchto druhov dokumentov bol jednak značný výskyt rôznych natočení v kombinácii s veľkou variabilitou pozadí, ktoré pôsobili ako rušivý vplyv pri tréningu ako aj klasifikácií jednotlivých kartičiek a taktiež v rozsiahlom počte prípadov bolo viacero kartičiek v jednom obrázku. Rozhodli sme sa preto pre špecifický postup v prípade klasifikácie príloh obsahujúcich viacero kartičiek v jednom obrázku. Postup pozostával z nasledovných krokov.

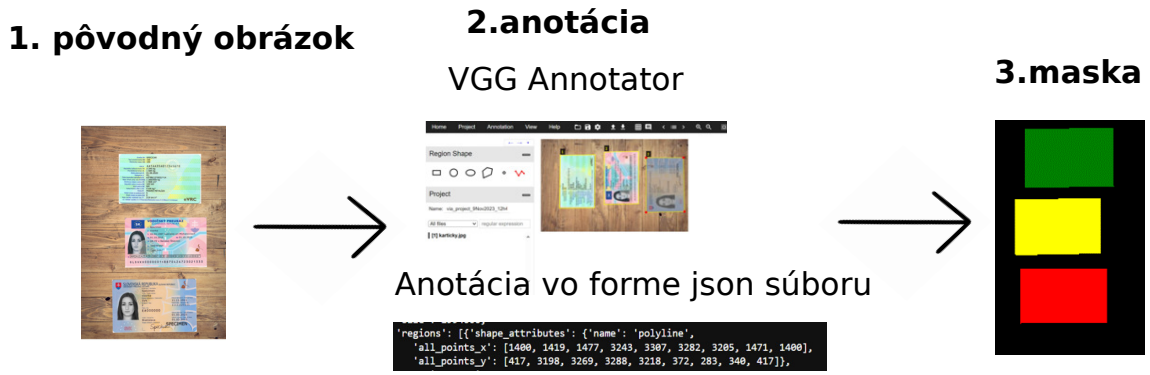
1. V pôvodnom klasifikačnom modeli zjednotiť všetky prílohy obsahujúce viacero kartičiek v jednom obrázku do triedy **multicard**.

2. Na všetky dokumenty klasifikované do triedy **multicard** použiť segmentáciu, ktorá na obrázku identifikuje daný preukaz a následné orezanie časti pôvodného obrázku obsahujúcej iba preukaz.

3. Orezané obrázky preukazových kartičiek následne klasifikovať ako patričné druhy preukazov pôvodným klasifikačným modelom.

4.4.1 Príprava dát na segmentáciu

Pred natrénovaním segmentačného modelu potrebujeme jeho vstupné dáta, ktorými budú prílohy obsahujúce viacero kartičiek v jednom obrázku, náležite pripraviť.
Prvým a najzákladnejším krokom je **výber vhodných trénovacích dát**. Musíme dbať na to, aby boli v datasete zahrnuté rôzne počty kartičiek na stránke ako aj rôzne verzie jednotlivých kartičkových typov dokumentov v rôznych vizuálnych podobách, ktoré sa v praxi vyskytujú (napríklad sken dokumentov alebo špecifické pozadie pri fotení dokumentov).
Pre účely vytvorenia masiek pre vstupné obrázky kartičiek potrebujeme jednotlivé obrázky anotovať - určiť, kde na vstupných obrázkoch sa nachádzajú kartičky. Na tento účel sme využili VGG - Annotator [11] - voľne prístupný softvér Oxfordskej univerzity, ktorý je možné spustiť lokálne a anotované dáta teda nie sú odosielané žiadnym tretím stranám. Výstupom VGG - anotátora boli polygónové anotácie vo formáte json. Z anotácií vo forme json súboru sme vytvorili masku vo vizuálnej podobe v bitmapovom formáte PNG.



Obr. 23: Vizuálne triedy klasifikovaných dokumentov.

Nakoľko aj keď majú obrázky dokumentov najrôznejšie rozmery, zväčša majú vysoké rozlíšenie. Pre zníženie výpočtovej náročnosti bolo potrebné zmenšiť rozmery vstupných obrázkov ako aj ich segmentačných masiek. Obrázkové prílohy sme bez značných problémov zmenšovali aj pri klasifikačnom modeli. Pri zmene rozmerov segmentačnej masky však došlo k zmene farby pixelov, ktoré sa nachádzali pri pôvodnej veľkosti na farebnom rozhraní. To spôsobilo, že ak maska v pôvodnej veľkosti obsahovala napríklad 3 farby (pozadie, kartička 1 a kartička 2), tak po zmene rozmerov, keď sa pixely na rozhraní farieb rozmazali, ich už obsahovala niekoľko desiatok.

Aby sme problém vyriešili, použili sme metódu zhlukovania pixelov segmentačnej masky v RGB farebnom priestore. Ako zhlukovaciu metódu sme použili *k-means*. Na každú masku sme po zmene rozmerov natrénovali zhlukovací *k-means* algoritmus pre $k =$ *počet kartičiek + 1* (jeden zhluk tvorilo pozadie) zhlukov. Farby na zmenšenej maske boli následne priradené k zhlukom zodpovedajúcim farbám na maske pôvodnej veľkosti.

4.4.2 Segmentačný model

Ako segmentačný model sme využili predtrénovaný model Mask R-CNN predstavený v časti 3.4. Model sme využili za pomoci `python` knižnice `torchvision.models`. Ako kostru (backbone) modelu sme použili Resnet50, ktorý sme využívali aj pri klasifikácii príloh. Na prvotný tréning sme využili trénovaciu vzorku veľkosti 130 stránok/obrázkov s kartičkami. Každý obrázok trénovacieho datasetu obsahoval od jednej do šiestich kartičiek. Výpočtová náročnosť tréningu pre 10 trénovacích epoch bola 4,5 hodiny a do 12 GB operačnej pamäte.

4.4.3 Výsledky modelu a úprava výstupov

Výstupy nami použitej Mask R-CNN architektúry mali špecifickú štruktúru, a síce, obsahovali väčšie množstvo návrhov masiek než ako bolo kartičiek na pôvodnom obrázku. Problémom bolo, že len niektoré z týchto návrhov aj reálne obsahovali kartičky. Bolo teda potrebné vhodnou úpravou výstupov modelu vyselektovať len návrhy masiek, ktoré obsahovali iba kartičku. Na túto selekciu sme použili nasledujúce podmienky:

- návrh masky musí mať vierohodnosť vyššiu ako 0,8,
- návrh masky musí mať plochu blízku ploche návrhu masky s najväčšou vierohodnosťou.

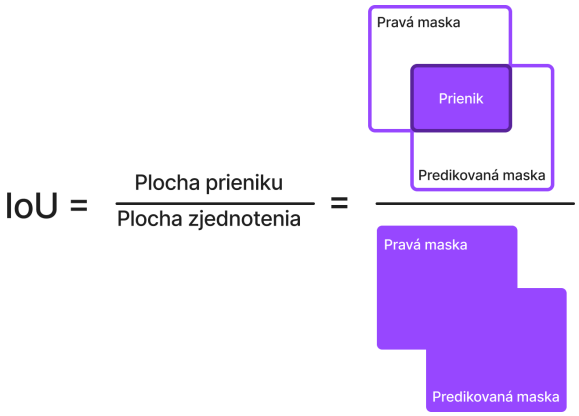
Následne sme zostrojili funkciu, ktorá po výbere relevantných predikcií segmentačných masiek, oreže kartičky z pôvodného obrázku a na tie potom opäť aplikuje klasifikačný model. Ako uvidíme aj v časti 4.4.5, v nami natrénovanom modeli sme pomocou týchto podmienok vo väčšine prípadov boli schopní získať taký počet masiek, aký bol aj skutočný počet kartičiek na obrázku.

4.4.4 Augmentácie pre segmentáciu

Pre zlepšenie predikcií segmentačného modelu sme navrhli augmentácie trénovacieho datasetu špecifické pre triedu **multicard**. Prvou augmentáciu bolo vytvorenie efektu naskenovaného kartičkového preukazu, ktorý aj z pôvodne nenaskenovaných obrázkov vytvoril obrázok vizuálne veľmi podobný skenu. Skenované kartičky predstavovali pre predikciu segmetnačnej masky výzvu v tom, že často splývali s pozadím. Z tohto dôvodu sme chceli mať v trénovacej vzorke dostatočné množstvo obrázkov zodpovedajúcim naskenovaným kartičkám. Ďalšou použitou augmentáciou bola augmentácia pozadia obrázku, ktorú sme spojili s augmentáciou natočenia kartičky na obrázku. Táto augmentácia mala pomôcť pri predikcii segmentačných masiek pre rôzne natočené kartičky.

4.4.5 Metriky úspešnosti

Ako metriku úspešnosti predikcií sme najskôr používali percentuálnu úspešnosť predikcie počtu kartičiek na obrázku. Na testovacej vzorke veľkosti 38 obrázkov obsahujúcich jednu až šesť kartičiek bol správny počet kartičiek identifikovaný v 90% obrázkov (34 z 38). Skutočný výsledok pri našom konkrétnom využití modelu vie byť však ešte optimistickejší. Podobne ako pri klasifikácii preukazov do tried s prednou a zadnou stranou, stačí nám, ak pomocou segmentácie na obrázku identifikujeme aspoň jednu stranu preukazu – už prítomnosť jednej strany preukazu nám podá informáciu o tom, že príloha má byť vložená do databázy ako daný typ dokladu.



Obr. 24: Znázornenie výpočtu ukazovateľa Interesection over Union.

Ďalšou použitou metrikou bola priemerná hodnota ukazovateľa IoU alebo Intersection over

Union, ktorý môžeme vidieť na Obr.24. Ide o ukazovateľ bežne používaný pri hodnotení segmentácie. Priemerná hodnota IoU bola 0,82.

4.5 Finálna agregácia dát

Konečné spracovanie emailov je nasledovné. Ak je email spracovaný automaticky, je klientovi odoslaná automatická odpoveď. Ak email obsahoval otázku, bude spracovaný a zároveň označený. Zamestnanec sa k nemu vráti, aby na prípadnú otázku alebo žiadosť reagoval. Niektoré emaily sú z rôznych dôvodov určené na ručné spracovanie - obsahujú neidentifikovateľný druh prílohy, prílohu s heslom, prílohu prevažne písanú rukou, neobsahujú číslo poistného hlásenia ani číslo poistnej udalosti a podobne. V súčasnosti je automaticky spracovaných približne 40 % e-mailov prijatých oddelením spracovania žiadostí.

Záver

V práci sme najskôr v prvej kapitole predstavili a ilustrovali základy počítačovej reprezentácie obrazu a princíp tvorby klasifikátora obrázkov dokumentových príloh. Pomocou konkrétnych príkladov backpropagation algoritmu bol tiež názorne demonštrovaný tréning klasifikátora. V druhej kapitole sme zostavili stručný prehľad rôznych vrstiev konvolučných neurónových sietí používaných aj v našich finálnych modeloch. Špeciálnu pozornosť sme venovali konvolučnej vrstve a aplikácii konvolúcie vo sfére neurónových sietí. V kapitole tretej sme popísali vývoj architektúry Mask R-CNN, ktorá bola v našej práci využívaná pre segmentáciu obrazu. Kapitoly 1-3 môžu slúžiť ako efektívny prehľad základných princípov pri štúdiu problematiky spracovania dokumentov pomocou konvolučných neurónových sietí.

Po položení teoretického základu sme v štvrtej kapitole nadviazali problematikou spracovania dokumentov a praktického využitia neurónových sietí pri automatizácii spracovania e-mailových správ oddelenia spracovania žiadostí. Vytvorili sme extraktor eml súborov, ktoré poskytujú informácie o e-mailovej komunikácií klientov poisťovne ohľadom likvidácie poistných udalostí, čo bolo základom pre ich ďalšie spracovanie. Následne sme navrhli algoritmus, ktorý z predmetu a textu e-mailov extrahoval čísla poistných hlásení a poistných udalostí, pomocou ktorých bolo možné prílohy e-mailu priradiť v databáze k príslušnej poistnej udalosti. Úspešnosť extrakcie čísel poistných hlásení a čísel poistných udalostí z e-mailov, v ktorých boli prítomné, bola 100 %. Texty a predmety e-mailových správ sme tiež prehľadávali za účelom detekcie otázky alebo požiadavky zo strany klienta. Nami navrhnutý systém detekcie otázok a požiadaviek založený na sledovaní prítomnosti kľúčových slov pre vzorku 140 náhodne vybraných e-mailov správne určil prítomnosť otázky v 96,5 % prípadov, pričom otázku nedetekoval iba v prípade jednej e-mailovej správy.

Centrálnou časťou našej práce bola tvorba klasifikátora príloh e-mailovej komunikácie oddelenia spracovania žiadostí. Ako klasifikátor bola použitá predtrénovaná štruktúra konvolučnej neurónovej siete Resnet50, ktorú sa nám podarilo natrénovať pre náš účel na relatívne malom trénovacom datasete (zhruba 6000 obrázkových príloh). Finálna presnosť klasifikácie bola 91 %. Pre potreby oddelenia spracovania žiadostí nie je dôležité, či ide o predné alebo zadné strany dokladu, ale len to, o aký typ dokladu ide. Podobne v prípade dvoch druhov kalkulácií, nie je dôležité o aký druh kalkulácie ide, len že je kla-

sifikátorom zaradená do triedy predstavujúcej kalkuláciu. Presnosť klasifikátora upravená na základe týchto skutočností bola 96 %.

Na klasifikačnú úlohu nadväzovalo využitie OCR pri hľadaní kľúčových slov, ktoré nám pomohli pri identifikácií niektorých podtried dokumentov ako aj pri vyťažovaní dôležitých informácií, potrebných pre likvidáciu poistnej udalosti z príloh e-mailov. Pri vyťažovaní celkovej sumy z kalkulácií sme dosiahli 95 % presnosť. Pri vyťažovaní celkovej sumy z faktúry sme dosiali 89 % presnosť.

Posledným cieľom tejto diplomovej práce bolo vytvoriť model pre identifikáciu dôležitých častí sprievodných dokumentov figurujúcich pri likvidácii poistných udalostí. Za týmto účelom sme úspešne natrénovali segmentačný model Mask R-CNN predstavený v 3.kapitole. Model sme využili pri dokladoch, ktoré mali formu kartičky. V prípade takýchto dokladov sme použili segmentáciu pre určenie presnej polohy kartičky v rámci obrázku. Na základe výsledku segmentačného modelu sme následne z pôvodného obrázku vyrezali iba časť obsahujúcu kartičku a tú pomocou natrénovaného klasifikátora priradili do správnej dokumentovej triedy. Výsledný segmentačný model dokázal detekovať správny počet kartičiek na obrázku v 90 % prípadov s priemernou hodnotou ukazovateľa IoU 0,82.

Neuronové siete netrénované pre klasifikáciu a segmentáciu sú súčasťou modelu nasadeného v poisťovni ČSOB. Prostredníctvom modelu sa v súčasnosti automaticky spracuje až 40 % e-mailovej komunikácie prijatej oddelením spracovania žiadostí. V budúcnosti je možné aplikovať nadobudnuté poznatky a súčasti modelu aj v rámci činnosti iných oddelení poisťovne.

Zoznam použitých zdrojov

[1] Abdulla, W.: *Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow*, Matterport Engineering Techblog, 20.3.2018, dostupné na internete (12.4.2024): https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b4

[2] Adobe stock images, dostupné na internete (8.12.2023): https://stock.adobe.com/search?k=odometer+miles

[3] Aider – Odškodňovanie po dopravných nehodách: *Plnenie rozpočtom: Ako získať viac o 100%-200%?*, dostupné na internete (8.12.2023): https://aider.sk/plnenie-rozpoctom-ako-ziskat-viac-o-100-200/

[4] Alruban, A., et al.: *A Comparison of Pooling Methods for Convolutional Neural Networks*, Journal of Applied Sciences, Volume 12, Issue 17, 2022, dostupné na internete (2.2.2024): https://www.mdpi.com/2076-3417/12/17/8643

[5] Bonnici, D.: *What is a vehicle identification number (VIN)?*, Which car? (14.9.2018), dostupné na internete (8.12.2023): https://www.whichcar.com.au/car-advice/what-is-a-vehicle-identification-number-vin

[6] Boyd, S.; Vandenberghe, L.: *Convex Optimization*, Cambridge University Press, 2004, kap.9.3

[7] Boyle R., Hlavac V.,Sonka M.: *Image processing, analysis, and machine vision*, Stanford: Cengage Learning, 2008

[8] Dabbudra, I.: *Gradient Descent Algorithm and Its Variants*, Medium, Towards data science, 31.12.2017, dostupné na internete (2.2.2024): https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

[9] Darrell, T.; Donahue, J.; Girshick, R.; Malik, J.: *Rich feature hierarchies for accurate object detection and semantic segmentation*, UC Berkley, 2014 dostupné na internete (2.2.2024): https://arxiv.org/pdf/1311.2524.pdf

[10] Dollár, P.; Girshick, R.; Gkioxari, G.; He, K.: *Mask R-CNN*, Facebook AI Research (FAIR), 2018, dostupné na internete (2.2.2024): https://arxiv.org/pdf/1703.06870.pdf

[11] Dutta, A.; Gupta, A.; Zisserman, A.: *VGG Image Annotator (VIA)*, dostupné na internete (2.5.2024): https://www.robots.ox.ac.uk/~vgg/software/via/

[12] Fawzy Gad, A.: *Faster R-CNN Explained for Object Detection Tasks*, Paperspace (2021), dostupné na internete (30.6.2023): https://blog.paperspace.com/faster-r-cnn-explained-object-detection/

[13] Girshick, R.: *Fast R-CNN*, Microsoft Research, 2015, dostupné na internete (2.2.2024): https://arxiv.org/pdf/1504.08083.pdf

[14] Girshick, R.; He, K.; Ren, S.; Sun, J.: *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, 2016, dostupné na internete (2.2.2024): https://arxiv.org/pdf/1506.01497.pdf

[15] Gonzales, R.C.; Woods, R.E.: *Digital image processing*, Pearson Education, 2007

[16] Harsh, A.: *Multiple Inputs Multiple Outputs in a Neural Network*, dostupné na internete (2.3.2024): https://neuralthreads.medium.com/multiple-inputs-multiple-outputs-in-a-neural-network-7860c5278a30

[17] ImageNet: *About ImageNet*, Stanford Vision Lab, Stanford University, Princeton University, 2020, dostupné na internete (2.4.2024): https://www.image-net.org/about.php

[18] Inv24 - fakturačný program: *Faktúry*, dostupné na internete (8.12.2023): https://www.inv24.sk/sk/faktura/

[19] Kasa FIK, Dokumentace: *Náležitosti pokladničného dokladu*, dostupné na internete (8.12.2023): https://manual.kasafik.cz/books/kasa-fik-ekasa-%28sk%29/page/n%C3%A1le%C5%BEitosti-pokladni%C4%8Dn%C3%A9ho-dokladu

[20] Kristeková, M.: Poznámky k predmetu *Spracovanie digitálnych signálov*, FMFI UK, 2024

[21] Krizhevsky, A. et al. : *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, Journal of Machine Learning Research 15, str. 1929-1958, 2014, dostupné na internete(21.3.2024) https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf?utm_content=buffer79b43&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer,

[22] Li, M.; Lipton, Z. C.; Smola, A. J; Zhang, A.: *Dive into deep learning*, Cambridge University Press, 2023, dostupné na internete(1.2.2024): https://d2l.ai/index.html

[23] Ministerstvo vnútra Slovenskej republiky: *Vzory dokladov a popis ochranných prvkov*, dostupné na internete (8.12.2023): https://www.minv.sk/?vzory-dokladov

[24] O'Donoghue, S.: *What to do if you hit a parked car? The dos and do nots, including insurance, fines and points.*, The Sunday Times Driving (12.11.2021), dostupné na internete (8.12.2023):https://www.driving.co.uk/car-clinic/hit-parked-car-dos-nots-including-insurance-fines-points/

[25] *Convolution*, 28.6.2018, songho.ca, dostupné na internete: https://www.songho.ca/dsp/convolution/convolution.html

[26] Person of interest, promotional photo, dostupné na internete (2.2.2024): https://www.fanpop.com/clubs/person-of-interest/images/31808811/title/harold-finch-season-1-promotional-photo-photo

[27] Petrou C., Petrou M.: *Image processing: The fundamentals*, John Wiley and Sons, 2010

[28] Portál finančnej osvety a a ochrany finančného spotrebiteľa MF SR: *ČO TO JE POISTENIE*, Ministerstvo financií Slovenskej republiky, 2020, dostupné na internete (2.2.2024): https://www.fininfo.sk/fininfo/financne-produkty/poistenie/co-je-poistenie/co-je-poistenie.html

[29] Poznámky k predmetu *CS231n: Convolutional Neural Networks for Visual Recognition*, Standfordská univerzita, dostupné na internete (9.1.2024): https://cs231n.github.io/

[30] Poznámky k predmetu *Signal Analysis*, FMFI UK

[31] Pujara, A.: *Object segmentation with Mask R-CNN*, Analytics Vidhya, 1.1.2024, dostupné na internete (12.4.2024): https://abhijeetpujara.medium.com/object-segmentation-with-mask-r-cnn-6e9c06cef81c

[32] Raschka, S.: *Activation Functions for Artificial Neural Networks* , Ahead of AI, 2016, dostupné na internete (2.2.2024): https://sebastianraschka.com/faq/docs/activation-functions.html

[33] Sable, A.; Zelic, F.: *Guide to Tesseract in Python with Pytesseract and OpenCV*, Nanonets (27.2.2023), dostupné na internete (8.12.2023): https://nanonets.com/blog/ocr-with-tesseract/

[34] Schmidhuber, J.: *Who Invented Backpropagation?*, Jürgen Schmidhuber's AI Blog, 2022, dostupné na internete (2.2.2024): https://people.idsia.ch/~juergen/who-invented-backpropagation.html

[35] Sharma, P.: *Computer Vision Tutorial: Implementing Mask R-CNN for Image Segmentation*, Analytics Vidhya, 27.2.2024, dostupné na internete (12.4.2024): https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/

[36] Slajdy k predmetu *Convolutional Neural Networks for Visual Recog- nition*, Stanfordská univerzita, 2017, dostupné na internete (2.2.2024): https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

[37] Szeliski, R.: *Computer Vision: Algorithms and Applications*, The University of Washington, 2021

[38] Šikudová, E. et al: *Počítačové videnie. Detekcia a rozpoznávanie objektov*, vydavateľstvo Wikina, Praha, 2014

[39] Trnovská, M.: Poznámky k predmetu *Metódy voľnej optimalizácie*, FMFI UK, 2022

[40] Veľas, A.: *Poisťovníctvo pre bezpečnostných manažérov*, Žilinská univerzita v Žiline, 2009

[41] Yingge, H. ; Ali, I.; Lee, Kang-Yoon: *Deep Neural Networks on Chip - A Survey*, IEEE International Conference on Big Data and Smart Computing (BigComp), 2020, str. 589 - 592.

[42] Zákon o poisťovníctve Úplné znenie zákona č. 8/2008 z 28. novembra 2007 o poisťovníctve a o zmene a doplnení niektorých zákonov ako vyplýva zo zmien a doplnení vykonaných zákonom č. 270/2008 Z. z., zákonom č. 552/2008 Z. z., , zákonom č. 186/2009 Z. z., zákonom č. 276/2009 Z. z., zákonom č. 129/2010 Z. z., zákonom č. 130/2011 Z. z., zákonom č. 332/2011 Z. z., zákonom č. 520/2011 Z. z. zákonom č. 547/2011 Z. z., zákonom č. 32/2013 Z. z., zákonom č. 132/2013 Z. z., zákonom č. 352/2013 Z. z. a zákonom č. 183/2014 Z. z. , Príloha 1, dostupné na internete (2.2.2024): https://www.nbs.sk/_img/documents/_legislativa/_uplnezneniazakonov/z0082008.pdf

[43] Zhang, X.: *Understanding Mask R-CNN Basic Architecture*, Shuffle, 14.11.2021, dostupné na internete (12.4.2024): https://www.shuffleai.blog/blog/Understanding_Mask_R-CNN_Basic_Architecture.html

Príloha A

Kód použitý pre ilustráciu príkladu v časti 1.2.4. Jednoduchý príklad backpropagation algoritmu v pythone.

```
import numpy as np

# rozmery vstupov a vystupov
input_nodes = 4
output_nodes = 2

# vstupy
X =  np.array([[0.99, 0.05],
               [0.6, 1.4]])

# inicializacia vah
W1 = np.array([0.15, 0.1, 0.25, 0.1])
b1 = 0.2
W2 = np.array([0.3, 0.15, 0.2, 0.35])
b2 = 0.2
b = np.array([b1, b2])
W = np.array([W1, W2])

# learning rate
lr = 0.05

# Vypocet loss funckie a jej derivacie

# aplikacia softmax normalizacie, ktora je sucastou cross vypoctu cross-
                                    entropy loss
def softmax(y):
    return np.exp(y) / np.sum(np.exp(y))

# derivacia softmax funkcie
def softmax_dash(y):
    I = np.eye(y.shape[0])
    return softmax(y) * (I - softmax(y).T)

# vypocet cross-entropy
```

```
def cross_E(y_true, y_pred):
    return -np.log(softmax(y_pred))
# derivacia cross-entropy
def cross_E_grad(y_true, y_pred):
    return softmax(y_pred) - y_true

# forward pass
def forward_propagation(X, W, b):
    y_pred = (np.dot(X, W1), np.dot(X, W2)) + b
    return y_pred

# backward pass
def backward_propagation(y_pred,W,b):
    error_upto_softmax = np.sum(cross_E(y, y_hat) * softmax_dash(
                                        in_output_layer), axis = 0).
                                        reshape((-1, 1))
    grad_w2 = error_upto_softmax .dot(out_hidden_1.T )

    error_grad_H1 = np.sum(np.sum(error_upto_softmax * w2, axis = 0).
                                        reshape((-1, 1)))
    grad_w1 = error_grad_H1*( x1.T )
```

Kód použitý pre ilustráciu príkladu v časti 2.4. Jednoduchý príklad implementácie backpropagation algoritmu v pythone.

```
import numpy as np

x1 =  np.array([[0.99, 0.05],
                [0.6, 1.4]])
x2 =  np.array([[0.99, 0.05],
                [0.6, 1.4]])
y = np.array([[1], [0]])

learning_rate = 0.05

def softmax(x):
    return np.exp(x) / np.sum(np.exp(x))

def softmax_dash(x):
```

```
    I = np.eye(x.shape[0])
    return softmax(x) * (I - softmax(x).T)

def cross_E(y_true, y_pred):
    return -np.log(softmax(y_pred))

def cross_E_grad(y_true, y_pred):
    return softmax(y_pred) - y_true

# random weight initialization
W = np.random.random(size = (hidden_nodes, input_nodes))
# b1 = np.zeros(shape = (hidden_nodes, 1))
V = np.random.random(size = (output_nodes, hidden_nodes))
# b2 = np.zeros(shape = (output_nodes, 1))

# forward pass
in_hidden_1 = W.dot(x1)
out_hidden_1 = in_hidden_1

in_output_layer = V.dot(out_hidden_1)
y_hat = in_output_layer

# loss

# backward pass
error_upto_softmax = np.sum(cross_E(y, y_hat) * softmax_dash(
                                    in_output_layer), axis = 0).reshape
                                    ((-1, 1))
grad_V = error_upto_softmax .dot(out_hidden_1.T )
#-----------------------------------------
error_grad_H1 = np.sum(np.sum(error_upto_softmax * V, axis = 0).reshape
                                    ((-1, 1)))
grad_W = error_grad_H1*( x1.T )

# Updating weights and biases with SGD
update_W = - learning_rate * grad_W
W += update_W
update_V = - learning_rate * grad_V
```

```
V += update_V
```

Kód bol inšpirovaný [16].

Príloha B

Kód použitý pre extrakciu eml súborov v časti 4.1.

```
# Based on script from https://stackoverflow.com/questions/31392361/how-
                                    to-read-eml-file-in-python
import email
from email import policy
from email.parser import BytesParser

import io
import base64
import unicodedata


def extract(eml_bytestring, key):
    """
    Extracts all data from e-mail, including subject, text, from, to,
                                        files etc., and returns them as
                                        a dictionary.

    Inputs
    - eml_bytestring: bytes. Bytestring encoded eml file [if input_type
                                        == 'bytes']
    - key: str for that particular Message. Can be a file name or
                                        anything.

    Output
    - msg_dict: dict, keys: from, to, subject, date, text, html, [,
                                        files]
            key files will be present only when message contained binary
                                            files.
    """
    msg_input = email.message_from_bytes(base64.b64decode(eml_bytestring
                                        ))
    fb = io.BytesIO(base64.b64decode(eml_bytestring.encode('utf-8')))
    msg = BytesParser(policy=policy.default).parse(fb)

    date = msg_input['date'] if msg_input.get("date") else ''
```

```
    # sender_str = get_element_from_header(msg_input, 'from')
    # recipient_str = get_element_from_header(msg_input, 'to')
    subject = get_element_from_header(msg_input, 'subject')

    sender_str = msg['from']
    recipient_str = msg['to']

    body_text, html, files = extract_content(msg_input, key)
    files = flatten_files_list(files)
    body_text = body_text.strip()
    html = html.strip()
    msg_dict = {"subject": subject, "from": sender_str, "to":
                                        recipient_str,
                "date": date, "text": body_text, "html": html, 'files':
                                                files}
    return msg_dict


def extract_content(m, key):
    """
    Extracts body_text, html, attachments from an e-mail message.

    Inputs
    - m: email.Message()
    - key: str, Initial message ID

    Outputs
    - Text: str, All text from all parts.
    - Html: str, All HTMLs from all parts
    - Files: list of dictionaries [{'filename': file_name, '
                                        bytestring_file': bytestring (
                                        bytes)}, ...]
    """
    html, text, files, file_dict = "", "", [], {}

    if not m.is_multipart():
        content_type = m.get_content_type()
        # Attachments
```

```
        if content_type in ['application/pdf', 'image/jpeg', 'image/jpg'
                                            , 'image/png', 'application/
                                            zip']:
            file_name = get_element_from_header(m)
            file_dict = {'filename': text_normalize(file_name),
                         'bytestring_file': m.get_payload(decode=True)}
        # text
        elif content_type == "text/plain":
            text += get_text_from_msg(m)
        # html
        elif content_type == "text/html":
            html += get_text_from_msg(m)
        else:
            # Something else!
            # Extract a message ID and a file name if there is one:
            # This is some packed file and name is contained in content-
                                                type header
            # instead of content-disposition header explicitly
            name_index_start = content_type.find("name=")
            name_index_end = content_type.find(";", name_index_start)
            if name_index_start > -1:
                file_name = content_type[name_index_start:
                                                    name_index_end]
                file_name = file_name.strip("'\"")
                file_dict = {'filename': file_name, 'bytestring_file': m
                                                    .get_payload(decode=
                                                    True)}
        files = [file_dict] if file_dict else []

    if m.is_multipart():
        for pl in m.get_payload():
            t, h, f = extract_content(pl, key)
            text += t
            html += h
            files += f
    return text, html, files
```

```
def get_element_from_header(msg_input, label=''):
    """
    Extracts to, from, subject and date from email.Message()

    Input
    - origin: email.Message() object

    Output
    - out_element: str
        If message doesn't contain element corresponding to the given
                                            label, empty strings will be
                                             returned
    """
    out_element = ''
    content = msg_input.get(label) or msg_input.get_filename()
    if content:
        byte_list = email.header.decode_header(content)
        if isinstance(byte_list[0][0], bytes) and byte_list[0][1]:
            out_element = byte_list[0][0].decode(byte_list[0][1])
        elif isinstance(byte_list[0][0], bytes) and not byte_list[0][1]:
            out_element = byte_list[0][0].decode('utf-8')
        else:
            out_element = byte_list[0][0]
    return out_element.strip()


def get_text_from_msg(msg):
    charset = msg.get_content_charset()
    if charset is None:
        text = msg.get_payload(decode=False)
    else:
        try:
            text = msg.get_payload(decode=True).decode(charset)
        except UnicodeError:
            text = msg.get_payload(decode=False)
    return text
```

```
def flatten_files_list(files):
    """TODO ADD DOCSTRING"""
    temp_list = []
    if isinstance(files, list) and files:
        for list_element in files:
            temp_list += flatten_files_list(list_element)
    elif isinstance(files, dict) and files.get('filename'):
        temp_list = [files]
    return temp_list


def text_normalize(text):
    text = ''.join(c for c in unicodedata.normalize('NFD', text) if
                                          unicodedata.category(c) != 'Mn')
    return(text)
```