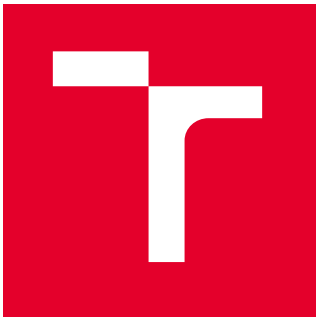


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

KLASIFIKACE KOLEJOVÝCH VOZIDEL

RAILWAY WAGONS CLASSIFICATION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Michal Kotrlý**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Peter Honec, Ph.D.**

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Michal Kotrlý ***ID:*** 200547

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Klasifikace kolejových vozidel

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit klasifikátor kolejových vozidel na základě obrazu / bočního pohledu. K dispozici je obrazová databáze průjezdů vlaků.

1. Seznamte se se základy zpracování obrazu, statistického a strukturního rozpoznávání a strojového učení.
2. Proveďte rešerši metod vhodných pro klasifikaci obrazu vzhledem k charakteru pořízených dat z reálné lokality.
3. Vytvořte třídy jednotlivých projetých vozů a lokomotiv.
4. Vytvořte vhodné rozhraní pro knihovnu (DLL) klasifikátoru.
5. Otestujte metody pro klasifikaci vozidel a vyberte nejvhodnější.
6. Zhodnoťte spolehlivost.

DOPORUČENÁ LITERATURA:

HLAVAC V., SONKA M., BOYLE R.: Image Processing, Analysis, and Machine Vision, ISBN 978-0495082521

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 17.5.2023

Vedoucí práce: Ing. Peter Honec, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Diplomová práce se věnuje klasifikaci kolejových vozidel na základě obrazové informace. V práci jsou teoreticky popsány a následně realizovány dva přístupy ke klasifikaci kolejových vozidel. Prvním přístupem je transformace snímků na histogramy vizuálních slov ze slovníku podle metody Bag of Visual Words a následná aplikace klasických klasifikátorů typu k-NN, SVM, Multinomial Naive Bayes, neuronová síť a Ensemble metoda typu voting classifiers. Druhým přístupem je klasifikace snímků pomocí ověřených architektur konvolučních neuronových sítí využitím metody transfer learning. Sítě AlexNet, VGG16 a ResNet50 byly předtrénovány na obsáhlém datasetu ImageNet a horní vrstvy byly dotrénovány na vlastním datasetu kolejových vozidel.

Oba přístupy byly vyladěny pro nejlepší možné výsledky klasifikace. Pro jejich srovnání byl sestaven trénovací dataset s 1773 snímky ve 27 třídách a testovací dataset obsahující 444 snímků. Na testovacím datasetu dosáhl nejúspěšnější klasifikátor s transformací snímků BoVW metodou správnosti 89%. Konvoluční neuronové sítě dosáhly správnosti 95-97%, což je výrazně lepší výsledek. V práci jsou také zohledněny doby predikce nových snímků pro oba přístupy.

Nad rámec práce byl implementován algoritmus pro dělení snímků vlakové soupravy na jednotlivé snímky vozů. V závěru jsou uvedeny limitace a popsány důvody omezené robustnosti algoritmu.

KLÍČOVÁ SLOVA

Klasifikace, Zpracování obrazu, Strojové učení, Konvoluční neuronové sítě, Bag of Visual Words, Rozpoznávání, Odhad přesnosti modelu

ABSTRACT

This Master's thesis deals with classification of railway wagons based on visual information. A theoretical background of two different approaches for a classification system is provided and both approaches are subsequently implemented. First approach includes transforming images of wagons to histograms of visual words, according to the Bag of Visual Words method. Afterwards, classifiers such as k-NN, SVM, Multinomial Naive Bayes, neural network and Ensemble method, specifically Voting classifiers, are applied. Second approach is classifying images using well known architectures of Convolutional Neural Networks and transfer learning. AlexNet, VGG16 and ResNet50 were pre-trained on a large ImageNet dataset and the upper layers were trained on the dataset of railway wagons.

Both approaches were fine-tuned for the best possible performance. For comparison of both approaches a training dataset with 1773 images in 27 classes and testing dataset with 444 images were compiled. On testing dataset the best classifier using BoVW method reached accuracy of 89%. Convolutional neural nets performed with 95-97% accuracy, which is an improvement. Prediction times of images to be classified are also considered.

Beyond the scope of the assignment of this thesis, an algorithm for splitting train images into images of individual wagons was developed. In the conclusion, limitations and reasons for limited robustness of this algorithm are presented.

KEYWORDS

Classification, Image Processing, Machine Learning, Convolutional Neural Networks, Bag of Visual Words, Recognition, Model Accuracy Estimation

KOTRLÝ, Michal. *Klasifikace kolejových vozidel*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2023, 88 s. Diplomová práce. Vedoucí práce: Ing. Peter Honec, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Michal Kotrlý
VUT ID autora:	200547
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Klasifikace kolejových vozidel

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Peterovi Honecovi, Ph.D. za odborné vedení, konzultace, zpětnou vazbu a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Diplomová práce se zabývá klasifikací snímků kolejových vozidel na základě obrazové informace. Úloha klasifikace spočívá v třídění (klasifikování) objektů v obraze do tříd. Třída je kolekce podobajících se, ale ne nutně stejných objektů. Objekty jedné třídy jsou pak odlišné od objektů tříd jiných. Pokud obraz obsahuje více objektů, jsou jednotlivé objekty od sebe navzájem a od pozadí izolovány a přiřazeny k odpovídající třídě.

Motivací této práce je průzkum možnosti nahrazení části stávající funkcionality systému CAMEA UniRAIL pro klasifikaci jednotlivých kolejových vozidel a zároveň zvýšení přesnosti této klasifikace. Současný systém, který je nasazen na reálné lokalitě, využívá sadu indukčních snímačů, které detekují průjezd železničních kol. Z časové značky průjezdu kola kolem dvou snímačů umístěných v definované vzdálenosti od sebe je vypočítána rychlost soupravy a z rychlosti a časových značek průjezdů náprav jsou měřeny rozvory dvojkolí jednotlivých vozů. Z rozvoru lze do určité míry detekovat typ vozu. Sada snímačů je však finančně nákladná a navíc pro umístění systému přímo v kolejišti musí splňovat příslušné drážní normy nebo zkušební napětí 4 kV, což dále zvyšuje cenu systému. Pro rozšíření systému do dalších lokalit železniční sítě pak cena hraje velkou roli. Klasifikace na základě rozvoru dvojkolí má také své limity – existují některé odlišné třídy kolejových vozidel, které mají rozvor stejný nebo velmi podobný a takové třídy pak nelze pouze na základě rozvoru spolehlivě odlišit. Diplomová práce se zabývá možností klasifikace kolejových vozidel ze snímků průjezdů souprav z řádkové kamery. Výhodou je, že kamera není umístěna přímo v kolejišti, a tak nepodléhá příslušné normě a odpadá nutnost instalace drahých snímačů.

První kapitola popisuje teoretické základy zpracování obrazu a popisuje metodu *Bag of Visual Words* (BoVW), která je kompaktní reprezentací obrazu pro použití standardních klasifikátorů. Druhá kapitola uvádí klasické metody strojového učení za účelem klasifikace, jako je k-NN, Support Vector Machines (SVM), neuronové sítě a další algoritmy strojového učení, které byly využity v praktické části. Rozebrány jsou i konvoluční neuronové sítě (CNN), jakožto moderní konkurent klasických přístupů ke klasifikaci obrazu. Dále kapitola popisuje metodiku odhadu přesnosti natrénovaných modelů. Třetí kapitola se zabývá analýzou datasetu snímků kolejových vozidel a praktickou realizací klasifikačního systému. Praktická realizace čerpá z teorie nabyté v předchozích dvou kapitolách. V rámci práce byly implementovány dva přístupy a jejich dosažené výsledky jsou v závěru kapitoly srovnány. Prvním přístupem je řetězec předzpracování obrazu, zpracování metodou BoVW a klasifikace klasickými klasifikátory. Druhým přístupem je klasifikace pomocí vybraných architektur konvolučních neuronových sítí. Pro oba přístupy byly vytvořeny moduly v

jazyce Python, které je možné použít v automatickém skriptu. Funkčnost modulů pro klasifikaci demonstruje jednoduchá uživatelská aplikace.

Nad rámec zadání byl vypracován algoritmus pro automatické dělení snímků osobních souprav na snímky jednotlivých vozů. Tento krok logicky zapadá do celého systému před samotnou klasifikací vozů.

1 Zpracování obrazu

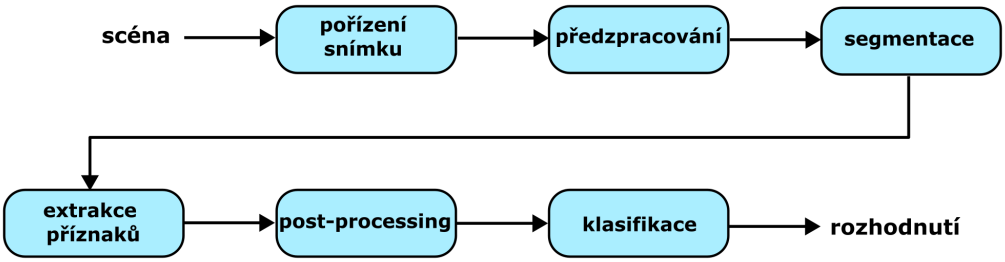
Obraz určený pro zpracování počítačem musí být reprezentován vhodnou diskrétní datovou strukturou. Obraz je spojitý dvourozměrný signál, digitalizací se pak rozumí navzorkování tohoto signálu do matice s M řádky a N sloupci, jejíž prvky jsou jasové hodnoty získané kvantizací signálu v příslušném obrazovém bodě (pixel). Čím je vzorkování a kvantizace jemnější, tím lépe je aproximován původní spojitý signál [1].

1.1 Interpretace obrazu

Interpretace obrazu počítačem je ve srovnání s pochopením obsahu člověkem i přes obrovský pokrok v oblasti počítačového vidění a zpracování obrazu stále omezená. Zatímco člověk zvládá určité typické úlohy počítačového vidění bez námahy, počítač je náchylný k chybám. Přes tyto omezení je v dnešní době počítačové vidění použito v široké škále reálných aplikací, mezi které patří rozpoznávání znaků a čtení kódů či SPZ, kontrola kvality v průmyslu, měření v rovině, defektoskopie, 3D digitální modely, zobrazovací techniky v oblasti medicíny, dohled (surveillance), rozpoznávání otisků prstů a měření biometrických údajů, klasifikace, navigace v prostoru, detekce pohybu, dopravní úlohy, bezkontaktní měření teploty, spojování obrazů a mnoho dalších [3]. Proces porozumění obsahu obrazu je obvykle dělen do několika úrovní. Od spodní úrovně obsahující pořízené obrazy, až po vysoce abstraktní popis potřebný k pochopení obsahu. Tuto hierarchii lze kategorizovat jako low-level processing a high-level processing. Low-level processing metody mají minimální znalost o obsahu obrazu, tyto metody zahrnují kompresi dat, pre-processing metody pro filtraci šumu, detekci hran, ostření obrazu atd. High-level processing je založen na znalosti a strategii, jak dosáhnout daného cíle využitím metod umělé inteligence. Na této úrovni se algoritmy snaží napodobit lidské kognitivní schopnosti a dojít k rozhodnutí na základě informací v obraze [1].

Obecný systém pro klasifikaci objektů v obrazu je složen z několika fází, jak je zobrazeno na obr.1.1. Fáze snímání využívá kameru vhodného typu a parametrů pro danou aplikaci. Mezi parametry kamerového systému patří rozlišení, dynamický rozsah, citlivost, zkreslení, úroveň, zaostření atd. [2]. Další fází je pre-processing obrazu, který je souborem operací na nejnižší úrovni abstrakce. Následuje segmentace a extrakce lokálních příznaků (features). Tento krok redukuje objem dat z obrazové informace pouze na charakteristiku těchto příznaků. Popis těchto příznaků je poté předán klasifikátoru, který vyhodnotí předložené indicie a zařadí objekt do určité třídy, pokud vlastnosti příznaků objektu spadají do tolerance této třídy [2]. V závislosti na povaze problému mohou či nemusí být použity všechny kroky tohoto

řetězce. Dále budou jednotlivé kroky klasifikačního systému, které jsou významné pro zadaný problém, popsány detailněji.



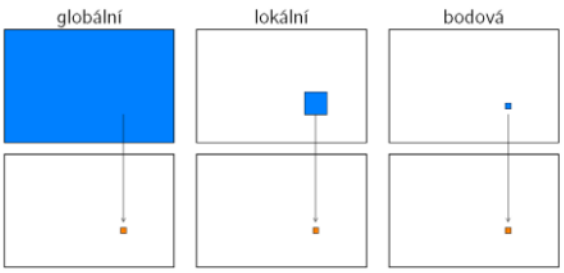
Obr. 1.1: Obecný systém pro klasifikaci [2], upraveno

1.2 Pre-processing

Vstupem i výstupem je matice jasových úrovní. Je důležité si uvědomit,že pre-processing nezvyšuje obsah obrazové informace, ale typicky ji snižuje. Z hlediska teorie informace je tedy nejvhodnější nepoužít žádný pre-processing a zaměřit se na pořízení kvalitních obrazů. Pre-processing je však vhodný ve většině aplikací, protože potlačuje irelevantní informaci pro danou úlohu zpracování obrazu. Cílem pre-processingu je tedy vylepšení obrazové informace odstraněním zkreslení a šumu a zvýrazněním některých vlastností obrazu pro další zpracování. Metody pre-processingu lze rozdělit do čtyř kategorií: jasové transformace, geometrické transformace, lokální předzpracování a restaurace obrazu [1]. V dalším budou nejdůležitější metody předzpracování shrnuty, lokální předpracování bude zahrnuto do kapitoly jasových transformací.

1.2.1 Jasové transformace

Při jasové transformaci dochází ke změně hodnoty obrazové funkce podle daného pravidla, ale vstupem i výstupem je obraz stejných parametrů (rozlišení, bitová hloubka). Tyto transformace lze dělit dle velikosti okolí vyšetřovaného bodu na globální, lokální a bodové.[přednáška jas.trans]. Mezi globální transformace patří např. Fourierova transformace, která umožňuje filtraci šumu a detekci hran ve frekvenční doméně. Nová hodnota pixelu je vypočítána z jasových hodnot celého obrazu. Hojně využívané jsou lokální jasové trasformace pro filtraci šumu a detekci hran v prostorové doméně. Nová jasová hodnota pixelu je dána váženým součtem jasových hodnot v jeho okolí. Tato operace se provede pro každý pixel se stejnou sadou vah, které se říká lineární filtr (kernel). Matematicky lze aplikování filtru popsat jako diskrétní



Obr. 1.2: Typy jasových transformací [23]

konvoluci vhodného kernelu a vstupního obrazu [4]. Existuje mnoho druhů kernelů pro průměrování (filtrace šumu), či detekci hran. Bodové transformace zahrnují jasové korekce a převodní charakteristiky. Nová hodnota jasu je určena z hodnoty téhož pixelu. Jasové korekce se využívají při korekci nerovnoměrného osvětlení, převodní charakteristiky pak pro zvýšení konstrastu, úpravě jasu, či prahování. Tyto funkce lze výhodně implementovat jako look-up tabulky, podle kterých jsou jasové hodnoty upravovány. Další bodovou transformací vhodnou pro zlepšení kontrastu je ekvalizace histogramu obrazu, která cílí na rovnoměrné rozložení jasových hodnot na celém přípustném rozsahu [1].

Úlohou ekvalizace je najít takovou převodní funkci intenzity obrazu $f(I)$, aby byl histogram obrazu po transformaci „rovnoměrný“. Nejdříve sestrojíme kumulativní histogram c(I) z histogramu h(I)

$$c(I) = \frac{1}{N} \sum_{i=0}^{I} h(i), \tag{1.1}$$

kde I je jasová hodnota (pro šedotónový obraz v rozsahu [0,255]) a N je počet pixelů. Kumulativní histogram je normováním na rozsah jasových hodnot odpovídajících výstupnímu obrazu. Kumulativní histogram se použije jako převodní charakteristika jasových úrovní [3]. Pro některé obrazy je vhodnější rozdělit obraz na M bloků a provést ekvalizaci histogramu pro každý blok zvlášť. Tyto metody adaptivní ekvalizace histogramu (AHE) zahrnují výpočet převodní funkce pro každý blok a bilineární interpolaci. Kromě středového pixelu každého bloku, který je přímo transformován převodní charakteristikou daného bloku, jsou jasové hodnoty všech ostatních pixelů interpolovány z převodních charakteristik čtyř nejbližších bloků [3]. Jednou z metod těchto adaptivních ekvalizací je CLAHE (*contrast limited adaptive histogram equalization*), která omezuje zesílení šumu v uniformních oblastech, což je vedlejší účinek AHE metod. Saturací histogramu bloku a redistribucí pixelů nad danou mez do celého rozsahu tak, aby byla jeho plocha zachována, způsobíme menší strmost kumulovaného histogramu. Strmost kumulovaného histogramu přímo odpovídá zvýšení

kontrastu. Menší strmost v uniformních oblastech tak zmenšuje kontrast a nezesiluje šum [18].

1.2.2 Geometrické transformace

Základem jsou parametrické 2D transformace jako je translace, rotace, měřítko a zkosení. Parametry všech transformací lze zapsat do matice rozměru 3x3 a s výhodou využít při transformaci v homogenních souřadnicích. Při dopředném mapování pixelu z původního obrazu do nových souřadnic vznikají prázdné pixely (díry), proto je preferováno zpětné mapování, kde je každý pixel v cílových souřadnicích vzorkován z původního souřadnicového systému [3]. Zatímco body cílového obrazu nabývají celočíselných hodnot, jsou hodnoty po mapování do původního obrazu neceločíselné. Jasové hodnoty jsou však známy jen v bodech o celočíselných souřadnicích. Pro získání hodnot v mezilehlých bodech se v praxi nejčastěji používá bilineární interpolace nebo interpolace vyššího řádu [6]. Geometrické transformace jsou využívány ke korekci zkreslení, nejčastěji radiálního zkreslení způsobeného čočkou při projekci scény na plochu obrazu. Tento efekt je patrnější, čím kratší je ohnisková vzdálenost kamery [5]. Radiální zkreslení se projevuje projekcí sítě čtverců do tzv. barelu nebo podušky. Korekce zkreslení je nutná pro měření vzdáleností v obraze [1]. Dalším typem je tangenciální zkreslení, které vzniká, pokud není soustava čoček rovnoběžná s rovinou obrazu.

1.3 Segmentace

Na této úrovni je obraz dělen na významné oblasti pro danou aplikaci. Zejména pak oddělení objektů zájmu od pozadí. Výsledkem kompletní segmentace je množina oblastí korespondujících s jednotlivými objekty v obraze, u částečné segmentace neodpovídají oblasti přímo objektům. Kompletní segmentace obrazu často vyžaduje výpomoc z vyšších vrstev zpracování, které mají určitou znalost úlohy. Avšak velkou skupinu segmentačních úloh lze řešit i nízkoúrovňovým zpracováním. V těchto případech jde většinou o kontrastní objekty na uniformním pozadí. Takové zpracování je nezávislé na kontextu, nebyl použit žádný model objektu. Cílem částečné segmentace je rozdělit obraz na homogenní oblasti vzhledem k jisté vlastnosti (jas,barva,textura atd.) [1].

Existují dva hlavní přístupy - region-based metody, kde jsou detekovány oblasti mající stejnou vlastnost a boundary-based metody, kde jsou detekovány hrany oblastí [2]. Tyto přístupy jsou k sobě duální, oblast tvoří uzavřenou hranici a každá uzavřená hranice definuje oblast. Protože povaha algoritmů obou přístupů je odlišná a mohou dávat různé výsledky segmentace, lze tyto přístupy kombinovat [1].

1.4 Detektory příznaků

Příznaky, v anglické literatuře často označované jako *features* nebo *keypoint features* jsou oblasti obrazu, jejichž charakteristika je unikátní. Jsou to body, jejichž okolí vykazuje vysoký gradient jasové funkce ve více než jednom směru. Podle [7] mají vhodné příznaky tyto vlastnosti:

- opakovatelnost - příznak je možné detekovat v jiném obrazu i přes geometrické a jasové transformace
- jednoznačnost - každý příznak je odlišitelný
- lokální - příznak popisuje malou oblast obrazu tak, aby byl robustní proti okluzi a překrytí (occlusion)
- kompaktnost a efektivita - méně příznaků než pixelů v obraze

Tyto příznaky se využívají pro hledání shodných lokací v odlišných obrazech, často za účelem výpočtu polohy kamery (*camera pose estimation* nebo také *extrinsic calibration*), spojování obrazů nebo v úloze rozpoznávání. Výhodou příznaků je možnost párování odpovídajících lokací i při velké změně měřítka a orientace objektů v obrazu. Typicky lze detektory příznaků funkčně rozdělit na extraktor a deskriptor příznaků. Extraktor nalezne oblasti kolem významných bodů obrazu a deskriptor tyto oblasti kompaktně popíše tak, aby mohly být deskriptory příznaků mezi sebou porovnány [3]. Pravděpodobně nejznámějším detektorem je SIFT detektor.

1.4.1 SIFT detektor

V [8] je popsána metoda detekce příznaků, které jsou invariantní ke škálování a rotaci obrazu, a částečně invariantní ke změně osvětlení a úhlu pohledu. Princip algoritmu je následující: Obraz je několikrát filtrován Gaussovým filtrem s konstantně se zvětšujícím rozptylem. Tyto filtrované obrazy tvoří tzv. *scale space*, ve kterém jsou vzdáleny o konstantu k, kterou se vždy násobí rozptyl pro další kolo filtrace (např. $k = \sqrt{2}$). Filtrace je konvoluce Gaussova filtru $G(x, y, \sigma)$ s obrazem $I(x, y)$

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y). \tag{1.2}$$

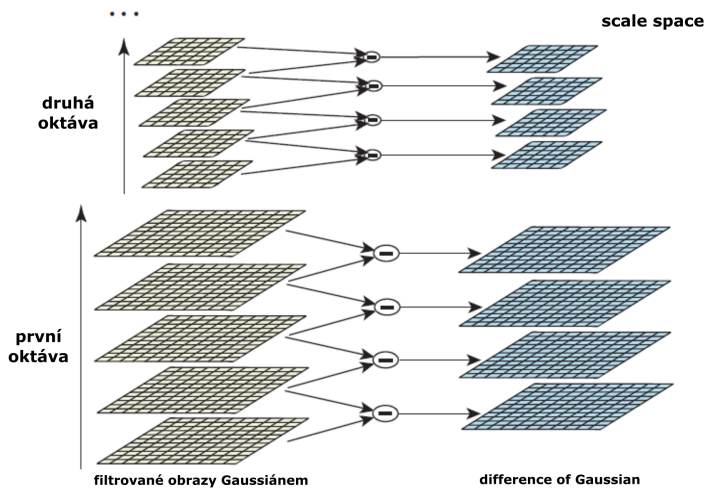
Gaussův filtr je dán výrazem

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}. \tag{1.3}$$

Sousední obrazy ve scale space, vzdáleny činitelem k, jsou od sebe odečteny a vznikne sada *difference-of-Gaussian* obrazů, které dobře aproximují LoG operátor (*Laplacian of Gaussian*).

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{1.4}$$

Poté podvzorkujeme obraz, který byl filtrovaný Gaussiánem s rozptylem 2σ (pro $k = \sqrt{2}$) tak, že každý druhý pixel v řádku a sloupci odstraníme. Celý proces filtrace opakujeme. Autor v [8] volí 4 úrovně vzorkování (oktávy) a $k = \sqrt{2}$, to pak znamená 5 různě filtrovaných obrazů pro každou oktávu.



Obr. 1.3: Ilustrace scale-space a vznik DOG obrazů [8], upraveno

Potencionální významné body jsou pak lokální extrémy difference-of-Gaussian obrazů hledané jak v prostoru, tak ve scale-space. Každý pixel je porovnán s osmi sousedními pixely v aktuálním obraze a devíti sousedy v přilehlých obrazech ve scale-space. Následuje přesná lokalizace významných bodů a zamítnutí lokalit s nízkým kontrastem a bodů podél hran. Přidělení orientace každému významnému bodu na základě lokálních vlastností obrazu umožní invariantnost k rotaci obrazu. V obraze $L(x, y)$, ve kterém byl významný bod nalezen, je v tomto bodě vypočtena velikost $m(x, y)$ a směr $\theta(x, y)$ gradientu obrazu. Gradient $\nabla L(x, y)$ je aproximován diferencemi

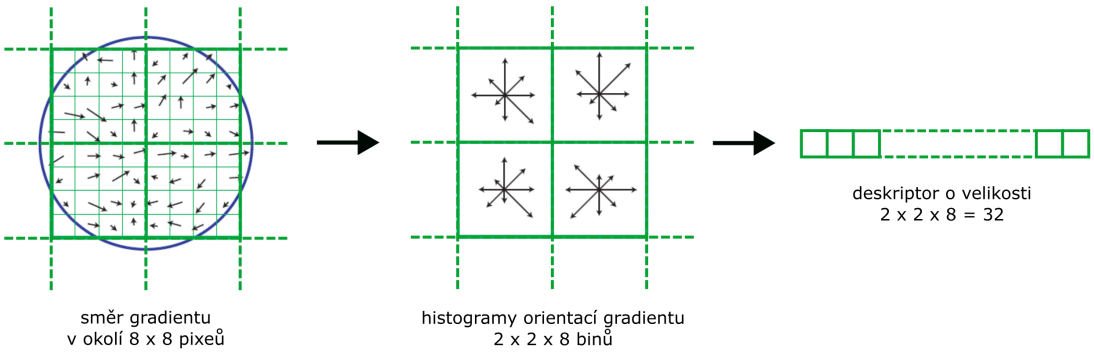
$$\nabla L(x,y) = \begin{bmatrix} \nabla_x L(x,y) \\ \nabla_y L(x,y) \end{bmatrix} \approx \begin{bmatrix} \Delta_x L(x,y) \\ \Delta_y L(x,y) \end{bmatrix} = \begin{bmatrix} (L(x+1,y) - L(x-1,y) \\ L(x,y+1) - L(x,y-1) \end{bmatrix}, \tag{1.5}$$

pak velikost a směr gradientu jsou dány rovnicemi

$$m(x,y) = \sqrt{\nabla_x^2 L(x,y) + \nabla_y^2 L(x,y)} \tag{1.6}$$

$$\theta(x,y) = \arctan\left(\frac{\nabla_y(x,y)}{\nabla_x(x,y)}\right). \tag{1.7}$$

Každý významný bod je popsán deskriptorem. Pro každý významný bod je deskriptor vytvořen nejdříve výpočtem velikosti a směru gradientu v každém bodě v okolí



Obr. 1.4: Ilustrace sestavení deskriptoru z okolí 8x8 pixelů. V prácí je použito okolí 16x16 pixelů, které vede na deskriptor délky 128 [8], upraveno

16x16 pixelů. Velikosti gradientu jsou následně ještě váhovány Gaussovým filtrem, aby body na okraji okolí měly menší vliv než ty blízko významného bodu. Tato informace je pak akumulována do histogramu orientací gradientu v každém subregionu o velikosti 4x4 pixelů. Vznikne tak matice histogramů velikosti 4x4, každý histogram obsahuje 8 „přihrádek“ (binů) pro směr gradientu. Každá přihrádka udává součet velikostí gradientů v subregionu, jejichž směr spadá do této přihrádky. Následně je sestaven vektor příznaku, neboli deskriptor, o velikosti 4x4x8=128 z číselných hodnot v osmi přihrádkách pro každý subregion. Normalizací vektoru je deskriptor částečně invariantní k osvětlení. Pokud histogram orientace gradientu vykazuje dva viditelné vrcholy, je významný bod považován za dva příznaky s různou orientací [8]. Obraz velikosti 500x500 pixelů typicky generuje asi 1000 významných bodů (features) [1].

1.5 Rozpoznávání

Obecné rozpoznávání objektů spadá do dvou obsáhlých kategorií, rozpoznávání instancí objektů (*instance recognition*) a rozpoznávání třídy objektů (*class recognition*).

První kategorie řeší rozpoznání známého 2D nebo 3D objektu, který může být ve scéně zachycen z jiného úhlu pohledu, na jiném pozadí nebo je částečně překryt. Druhá kategorie je mnohem náročnější problém rozpoznávání instancí objektů patřících do obecné třídy, jako je „kočka“, „auto“ nebo „květina“. Mnoho přístupů rozpoznávání instancí objektů zahrnuje extrakci příznaků z nového obrazu a použití některého z dostupných algoritmů k porovnání s příznaky extrahovanými z databáze obrázků. V anglické literatuře se tento proces nazývá *feature matching*. Pokud je nalezen dostatečný počet shodných významných bodů mezi novým obrázkem a obrázkem z databáze, provede se krok verifikace odhadem geometrické transformace

mezi obrázky. Nalezené domnělé shody hlasují o hypotéze pozice, orientace a měřítku objektu v obrazu, tedy o správné geometrické transformaci. Pokud má některá hypotéza geometrické transformace počet shodných bodů (inliers) nad prahem T, je instance objektu v novém obrazu rozpoznána.[3]

Klasifikace obrazů do tříd klasickými metodami se opírá o extrakci vhodných příznaků a jejich statistice, často za využití strojového učení k provedení finální klasifikace. Moderní přístupy jsou založeny na hlubokých neuronových sítích. Hluboké učení trénuje celý rozpoznávací řetězec, od pixelu až ke klasifikaci. Zástupcem klasických metod je *Bag of Visual Words*, metoda představená v článku [9]. Také se jí říká *Bag of Features* nebo *Bag of Keypoints*.

1.6 Bag of visual words

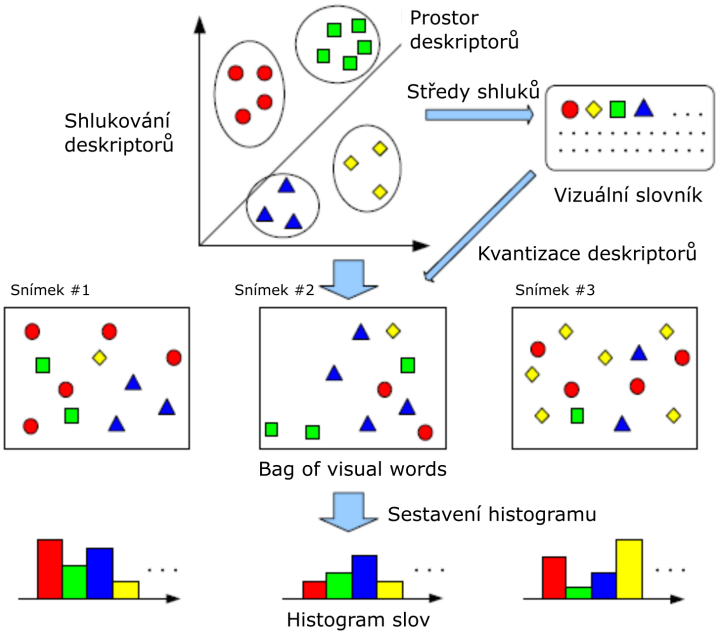
Bag of visual words reprezentuje obrazy nebo objekty jako neseřazenou kolekci deskriptorů příznaků. Každý obraz je popsán charakteristickým „pytlem“ příznaků ze slovníku, formálně histogramem četností každého příznaku v obrazu. Metoda vychází z reprezentace textů pomocí *bag of words*, tedy doslova „pytel slov“. Tento postup byl adoptován pro účely obrazové informace a slova byla nahrazena tzv. vizuálními slovy, vznikajícími kvantizací deskriptorů. Pro klasifikaci nového obrazu stačí pak vypočítat histogram vizuálních slov tohoto obrazu a srovnat s histogramy obrazů naučené databáze. Výhodou metody je její jednoduchost, výpočetní nenáročnost a robustnost [9].

Pro přehlednost popisu metody je nutné zavést určitou terminologii. Jako „deskriptor“ se bude označovat vektor popisující významný bod generovaný SIFT algoritmem z kapitoly 1.4, „vizuální slova“ nebo jen „slova“ vznikají kvantizací, tzn. shlukováním všech deskriptorů trénovací množiny a tvoří „vizuální slovník“ nebo jen „slovník“.

Trénování systému na trénovací množině zahrnuje tyto kroky:[9]

1. Detekce a popis významných bodů v testovací množině s anotovanými obrazy
2. Vytvoření vizuálního slovníku kvantizací prostoru deskriptorů metodou shlukování
3. Sestavení histogramu slov pro každý obraz z trénovací množiny
4. Naučení *multi-class* klasifikátoru na histogramech slov, které reprezentují každý obraz

Prvnímu kroku je věnována samostatná kapitola 1.4. Následný vznik vizuálního slovníku a sestavení histogramu slov je ilustrováno na obrázku 1.5 a popsáno v sekcích níže.



Obr. 1.5: Ilustrace metody Bag of visual words [10], upraveno

1.6.1 Vznik vizuálního slovníku

Všechny extrahované deskriptory z trénovací množiny jsou vstupem iterativního shlukovacího algoritmu, např. K-means, který je popsaný v sekci 2.3.1. Předem nastavený počet shluků udává velikost slovníku, tedy počet vizuálních slov. Po ukončení iterace shlukovacího algoritmu je nalezen střed (těžiště) každého shluku deskriptorů a je prohlášen za vizuální slovo. Slovník by měl být dostatečně obsáhlý, aby rozlišoval významné změny v obraze, ale ne příliš velký na to, aby postihnul nerelevantní variace v obraze jako je šum [9].

1.6.2 Sestavení histogramu

Pro popis obrazu je následně využito vytvořeného slovníku. Každému deskriptoru, extrahovanému z obrazu, je metodou k-NN přiřazeno nejbližší slovo ze slovníku a daný deskriptor je v reprezentaci obrazu nahrazen tímto slovem. Dochází tedy ke kvantizaci prostoru deskriptorů. Následně stačí pouze spočítat četnosti výskytu každého slova v obraze a sestavit histogram, který reprezentuje vstupní obraz. Reprezentace obrazu Metodou bag of words je tak vlastně ztrátovou kompresí, která redukuje vstupní informaci klasifikátoru. Pro ilustraci bude uveden příklad.

Předpokládáme, že vstupní obraz má velikost 960x540 pixelů. Extrakcí příznaků metodou SIFT vygenerujeme 500 deskriptorů, tedy 128-dimenzionálních vektorů. Řekněme, že velikost slovníku je 200 slov. Pak každý z deskriptorů je namapován na

nejbližší slovo ze slovníku a četnosti výskytu každého slova jsou zaznamenány a je sestaven histogram slov. Dimenze histogramu je tak shodná s nastavenou velikostí slovníku, v tomto případě je to hodnota 200. Obraz je tedy reprezentován 200-dimenzionálním vektorem.

1.6.3 Klasifikace nových obrazů

Hlavní kroky pro klasifikaci nového obrazu kopírují postup reprezentace obrazu jako histogram slov a následuje klasifikace naučeným klasifikátorem [9]. Klasifikaci na bázi strojového učení se věnuje kapitola 2.

1. Detekce a popis významných bodů
2. Přiřazení nejpodobnějšího slova ze *slovníku* ke každému deskriptoru
3. Sestavení histogramu slov, který obsahuje počty přidělených deskriptorů ke každému vizuálnímu slovu ze slovníku
4. Aplikace *multi-class* klasifikátoru, jehož vstupem je histogram slov a výstupem predikovaná třída obrazu.

1.6.4 Předchozí výzkum v oblasti metody BoVW

V článku [9] je kromě popisu metody BoVW obsažena sada experimentů klasifikace na vlastním datasetu se sedmi třídami a volně dostupném datasetu s pěti třídami. Vyšší úspěšnost klasifikace autoři dosáhli u SVM klasifikátoru, který předčil naivní bayesovský klasifikátor. Práce [10] přebírá metody klasifikace textu jako je váhování, výběr slov, velikost slovníku atd. a aplikuje je na reprezentaci obrazu slovníkem vizuálních slov. V práci se autoři zabývají vlivem nastavení vizuálního slovníku na kvalitu klasifikace obrazu. Práce tak poskytuje chybějící část navazující na předchozí výzkum, který se zabýval převážně vlivem výběru detektorů významných bodů a klasifikátorů na kvalitu klasifikace.

2 Strojové učení

Od reprezentace obrazu histogramem slov se tato kapitola přesouvá k využití metod strojového učení ke zpracování těchto histogramů a predikci třídy obrazu. Kapitola se zabývá nástíněním teoretického základu metod, se kterými bylo experimentováno v praktické části práce.

Strojové učení je obor zabývající se algoritmy schopnými učit se z dat. Formálně lze říci, že program (algoritmus) se učí ze zkušenosti E vzhledem k úloze T a metrice výkonnosti P, pokud se výkonnost na úloze T měřená pomocí P zlepšuje využitím E [12]. Zkušenost E, na které se program učí, jsou tedy dostupná data, nazývaná trénovací množina nebo trénovací dataset obsahující tzv. vzorky. Systémy strojového učení je možné rozdělit do kategorií podle:

- toho, zda jsou učeny s učitelem či nikoliv
- toho, zda jsou pouze nové vzorky porovnávány se vzorky z trénovací množiny nebo je z trénovací množiny sestaven prediktivní model
- toho, zda se systém může učit inkrementálně na nových datech či nikoliv

Tyto kritéria nejsou exkluzivní, jednotlivé metody strojového učení mohou spadat do více kategorií.

2.1 Řetězec úlohy strojového učení

Obecně každou úlohu strojového učení lze popsat procesem s několika kroky:

1. Definuj problém. Jakou úlohu chceme program naučit řešit?
2. Nasbírej data. Získej data pro trénovací a testovací množinu. Čím je dataset obsáhlejší a pestřejší , tím lépe.
3. Navrhni příznaky, které popisují data.
4. Natrénuj model. Vylaď parametry modelu na trénovací množině optimalizací.
5. Otestuj model. Zhodnoť výkonnost modelu na testovací množině. Pokud jsou výsledky nedostačující, přehodnoť volbu modelu a příznaků, nasbírej více dat [11].

2.2 Učení s učitelem

V případě učení s učitelem (*supervised learning*) obsahuje trénovací množina, která je algoritmu předložena, také žádané řešení, tedy správnou výstupní veličinu [12]. Datasetem je tedy kolekce N vzorků $\{(\mathbf{x_i}, y_i)\}_{i=1}^{N}$. Každý vektor $\mathbf{x_i}$ se nazývá vektor příznaků (*feature vector*), jekož elementy (dimenze) $r = 1, ..., D$ obsahuje hodnoty, které vzorek nějak popisují. Této hodnotě se říká příznak a je značena jako $x^{(j)}$. Např. pokud každý vzorek reprezentuje osobu, pak $x^{(1)}$ může být výška v centimetrech,

$x^{(2)}$ váha v kilogramech atd. Výstupní veličina (*label*) může být prvkem konečné množiny tříd $\{1, 2, ..., C\}$ nebo reálné číslo [13].

Případu, kdy je výstupní veličinou reálné číslo, se říká *regrese*. Úloha regrese tedy spočívá v naučení modelu na datech tak, aby bylo možné predikovat spojitou veličinu (*target*). Příkladem může být odhad ceny domu na základě jeho vlastností, jako je obytná plocha, stáří, lokalita atd. *Klasifikace* místo spojité veličiny predikuje diskrétní veličinu, neboli třídu [11]. Příkladem může být třídění emailů do tříd {spam, not_spam}. Nejznámějšími algoritmy učení s učitelem jsou: [12]

- k-Nearest Neighbors (kNN)
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural Networks

2.2.1 k-Nearest Neighbors (kNN)

Algoritmus k-Nearest Neighbors patří do skupiny učení s učitelem, kde jsou nové vzorky porovnávány se vzorky trénovací množiny a nevytváří se parametrizovatelný model. Takové algoritmy se označují jako *učení založené na instancích* (*Instance-Based Learning*). Na rozdíl od jiných algoritmů, které umožňují odstranění trénovacích dat po sestavení modelu, kNN uchovává celou trénovací databázi v paměti [13]. Při velkých datasetech má tedy vysoké paměťové nároky. Při klasifikaci nebo regresi nového vzorku je nalezeno k nejbližších sousedů v příznakovém prostoru, přičemž vzdálenost příznakových vektorů v tomto prostoru se nejčastěji určuje Euclidovou vzdáleností danou známým vztahem:

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{\sum_{r=1}^{D} (x_i^{(r)} - x_j^{(r)})^2} \tag{2.1}$$

kde $x_i^{(r)}$ je r-tý prvek příznakového vektoru $\mathbf{x_i}$. Podle vztahu 2.1 se vypočte vzdálenost $d(\mathbf{x_q}, \mathbf{x_i})$ pro $i = 1, ..., N$ mezi novým vzorkem $\mathbf{x_q}$ a všemi vzorky $\mathbf{x_i}$. Ze vzdáleností získáme k nejbližších vzorků $\mathbf{x_i}$ patřících do tříd $f(\mathbf{x_i})$ k novému vzorku $\mathbf{x_q}$. U klasifikace je odhadnutá třída dána vztahem:

$$\hat{f}(\mathbf{x_q}) = \underset{c \in C}{\operatorname{argmax}} \sum_{i=1}^{k} \partial(c, f(\mathbf{x_i})) \tag{2.2}$$

kde $\partial(c, f(\mathbf{x_i})) = 1$ když $c = f(\mathbf{x_i})$, jinak $\partial(c, f(\mathbf{x_i})) = 0$.

Vztah 2.2 říká, že odhad třídy nového vzorku $\hat{f}(\mathbf{x_q})$ je třída $c \in C$, která je mezi k nejbližšími vzorky $\mathbf{x_i}$ patřící do tříd $f(\mathbf{x_i})$ tou nejzastoupenější [16]. Aby byla

třída jasně určená a vyhlo se neurčitosti při vícero stejně zastoupených tříd mezi k sousedy, typicky se volí lichý počet sousedů k [11]. V případě regrese je predikovanou veličinou průměr hodnot výstupních veličin k nejbližších sousedů dle:

$$\hat{f}(\mathbf{x_q}) = \frac{\sum_{i=1}^{k} f(\mathbf{x_i})}{k} \tag{2.3}$$

2.2.2 Support Vector Machine (SVM)

Nejdříve si zavedeme terminologii lineárních klasifikátorů za předpokladu, že je dataset $\{(\mathbf{x}_i, t_i)\}_{i=1}^{N}$, kde $t_i \in \{-1, 1\}$ *lineárně separabilní*. Definujme lineární klasifikátor s hranicí (*decision boundary*) $y(\mathbf{x}) = \mathbf{w^T x} + b = 0$. Parametry modelu jsou dány veličinami $\mathbf{w}$ a b. Klasifikace je pak dána znaménkem projekce $\hat{t_i} = \operatorname{sgn} y(\mathbf{x}_i) = \operatorname{sgn}(\mathbf{w^T x}_i + b)$, tzn. pro $y(\mathbf{x}_i) < 0$ je $\hat{t_i} = -1$ a pro $y(\mathbf{x}_i) > 0$ je $\hat{t_i} = 1$. Model můžeme zobecnit na lineární kombinaci nelineárních funkcí vektoru příznaků pomocí bázových funkcí $\boldsymbol{\phi(x)}$. Předpokládáme jednoduchý případ $\boldsymbol{\phi(x)} = \mathbf{x}$. Model má tvar:

$$y(\mathbf{x}) = \boldsymbol{w^T \phi}(\mathbf{x}) + b \tag{2.4}$$

Předpokladem lineární separability je zajištěno, že existuje řešení $\mathbf{w}$ a b tak, že pro všechny vzorky $\mathbf{x}_i$ platí $t_i y(\mathbf{x}_i) > 0$. Samozřejmě může existovat mnoho řešení ke správnému rozdělení tříd hranicí. V metodě SVM je rozhodovací hranice vybrána tak, že maximalizuje *margin*, což je nejmenší vzdálenost hranice od kteréhokoliv vzorku [14]. Maximalizace marginu je ilustrována na obr. 2.1. Vzdálenost vzorku od nadroviny dané rovnicí $y(\mathbf{x}) = 0$, kde $y(\mathbf{x})$ vychází z 2.4, je definována jako

$$r_i = \frac{y(\mathbf{x}_i)}{\|w\|} = \frac{\boldsymbol{w^T \phi}(\mathbf{x}_i) + b}{\|w\|} \tag{2.5}$$

Vzdálenost nejbližšího vzorku od hranice (margin) můžeme definovat jako:

$$\min_i [t_i r_i] \tag{2.6}$$

S výhodou jsme eliminovali znaménko ve vzdálenosti díky tomu, že t_i a r_i mají vždy stejné znaménko. Maximalizace marginu je pak optimalizační úlohou parametrů $\mathbf{w}$ a b daná vztahem:

$$\underset{\mathbf{w},b}{\operatorname{argmax}} \left\{ \frac{1}{\|w\|} \min_i [t_i(\boldsymbol{w^T \phi}(\mathbf{x}_i) + b)] \right\} \tag{2.7}$$

Následující vztah platí pro všechny vzorky $\mathbf{x}_i$:

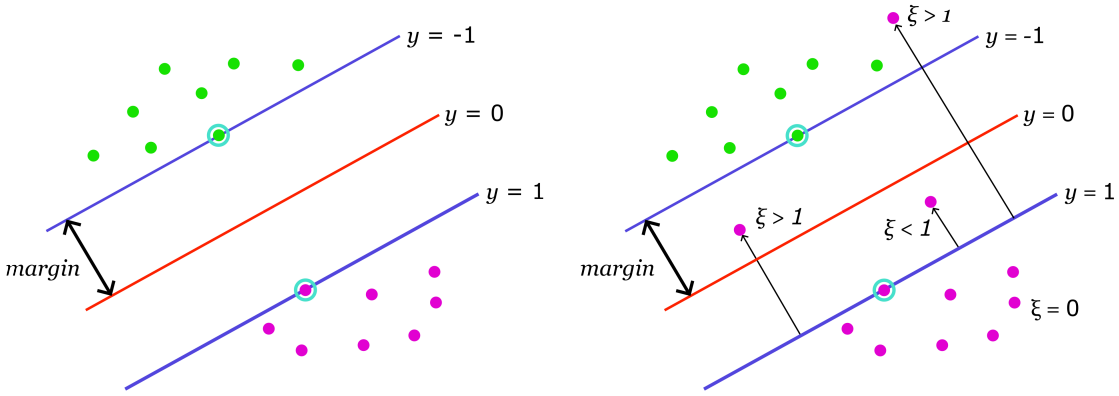
$$t_i(\boldsymbol{w^T \phi}(\mathbf{x}_i) + b \geq \min_i [t_i(\boldsymbol{w^T \phi}(\mathbf{x}_i) + b)] \tag{2.8}$$

Dále můžeme výraz $\min_i [t_i(\boldsymbol{w^T \phi}(\mathbf{x}_i) + b)]$ škálovat tak, aby se rovnal jedné, tedy, $\min_i [t_i(\boldsymbol{w^T \phi}(\mathbf{x}_i) + b)] \stackrel{!}{=} 1$. Nyní můžeme přepsat optimalizační úlohu do tvaru:

$$\underset{\mathbf{w},b}{\operatorname{argmax}} \frac{1}{\|w\|} \quad \text{za podmínky} \quad t_i(\boldsymbol{w^T \phi}(\mathbf{x}_i) + b) \geq 1 \ \forall i \tag{2.9}$$

Maximalizace výrazu $\frac{1}{\|w\|}$ je ekvivalentní k minimalizaci $\frac{1}{2}\|w\|^2$. Optimalizační úloha tedy nabývá tvaru:[14]

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \frac{1}{2}\|w\|^2 \quad \text{za podmínky} \quad t_i(\boldsymbol{w}^T\boldsymbol{\phi}(\mathbf{x}_i)+b)-1 \geq 0 \ \forall i \tag{2.10}$$



Obr. 2.1: Definice marginu (vlevo) a zavedení slack variables pro lineárně neseparabilní dataset (vpravo)

Tatu optimalizační úlohu s podmínkou lze řešit metodou *Lagrangeových multiplikátorů* [13][14]. Pro každý vzorek zavedeme Lagrangeův multiplikátor $a_i \geq 0$. Místo řešení původního problému, daného rovnicí 2.10 je vhodné převést na ekvivalentní problém reprezentovaný duální úlohou, která je dána rovnicí:

$$\tilde{L}(\mathbf{a}) = \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j t_i t_j k(\mathbf{x}_i\mathbf{x}_j) \tag{2.11}$$

s podmínkami

$$a_i \geq 0, \quad \sum_{i=1}^{N} = 0, \tag{2.12}$$

kde $\mathbf{a} = (a_1, ..., a_N)^T$. Duální problém je maximalizován vzhledem k $\mathbf{a}$:

$$\underset{\mathbf{a}}{\operatorname{argmax}} \tilde{L}(\mathbf{a}) \tag{2.13}$$

Jádrová funkce (*kernel function*) v rovnici 2.11 je dána vztahem $k(\mathbf{x}_i\mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. Problém popsaný vztahy 2.11,2.13 je opět optimalizační úloha kvadratického programování. Tyto problémy se efektivně řeší metodou *Sequential Minimal Optimization* (SMO).

Doposud jsme se zabývali striktně lineárně separablními datasety, kde se distribuční funkce příznaků jednotlivých tříd nepřekrývají. Zavedením tzv. *slack variables* $\xi_i \geq 0$ pro každý vzorek rozšíříme SVM na nelineárně separabilní datasety. Tyto proměnné umožňují určit parametry $\mathbf{w}$ a b tak, že některé vzorky (např. outliers)

mohou ležet na nesprávné straně rozhodovací nadroviny za cenu jejich penalizace v chybové funkci. Pro vzorky na správné straně rozhodovací nadroviny mimo margin je definováno $\xi_i = 0$, pro ostatní vzorky platí $\xi_i = |t_i - (\boldsymbol{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b)|$. Vzorky na správné straně marginu mají $0 < \xi_i \leq 1$ a vzorky na hranici $\xi_i = 1$. Pro všechny vzorky na nesprávné straně rozhodovací hranice tedy platí $\xi_i > 1$ a hodnota ξ roste se vzdáleností od hranice [14]. Hodnota ξ pro různé vzorky je ilustrována na obr. 2.1. Proměnná ξ_i je tedy měřítkem, jak moc i-tý vzorek porušuje hranici klasifikátoru [12]. Chybová funkce je pak kompromisem mezi maximalizací marginu a redukcí míry nesprávné klasifikace minimalizací ξ_i. Cílem je optimalizavat úlohu

$$\underset{\mathbf{w},b,\xi}{\operatorname{argmin}} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i \quad \text{za podmínek} \quad t_i(\boldsymbol{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \ \forall i. \tag{2.14}$$

Při optimalizaci této úlohy mluvíme o tzv. *soft margin* SVM [12]. Parametr C určuje nastavení kompromisu při minimalizaci chybové funkce a je hyperparametrem metody, jehož hodnota je určena experimentálně. Zavedením chybové funkce tzv. *hinge loss function* lze optimalizační úlohu soft margin SVM přepsat do tvaru

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \underbrace{\max\left(0, 1 - t_i(\boldsymbol{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b)\right)}_{\text{hinge loss}} \tag{2.15}$$

Datasety, které nejsou lineárně separabilní a zavedení soft margin SVM nemá kvůli povaze dat účinek, můžeme převést z původního prostoru do prostoru o vyšší dimenzi, kde jsou data lineárně separabilní. Právě k tomuto se s výhodou využívají jádrové funkce (kernely), díky kterým lze pracovat s daty v prostoru s vyšší dimenzí bez explicitní transformace dat do tohoto prostoru. Tomuto výpočetnímu usnadnění se říká *kernel trick* [13]. Běžnou jádrovou funkcí je polynomiální kernel M-tého řádu s konstantou c: $k(\mathbf{x}_i \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^M = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$. Transformace vektoru příznaků $\phi(\mathbf{x})$ do výšší dimenze a jejich skalární součin však není nutné počítat, stačí dosadit skalární součin $\mathbf{x}_i^T \mathbf{x}_j$ do požadovaného kernelu $(\mathbf{x}_i^T \mathbf{x}_j + c)^M$, protože významný je pouze skalární výsledek [13].

Pro klasifikaci nových vzorků vyhodnotíme znaménko rovnice 2.4. To lze vyjádřit Lagrangeovými multiplikátory a_i a jádrovou funkcí jako:

$$y(\mathbf{x}) = \sum_{i=1}^{N} a_i t_i k(\mathbf{x}, \mathbf{x}_i) + b \tag{2.16}$$

Lze si povšimnout, že vzorky, pro které platí $a_i = 0$, nemají vliv na sumu v rovnici 2.16, nehrají tedy roli při klasifikaci nových vzorků. Zbylým vzorkům, pro které platí $a_i \geq 0$, se říká *support vectors* a protože pro ně dále platí $t_i y(\mathbf{x_i}) = 1$, tyto body leží na nadrovinách maximálního marginu [14]. Support vectors jsou tedy jediné body, které určují rozhodovací hranici.

SVM je fundamentálně binární klasifikátor, je však několik přístupů použití SVM pro multi-class klasifikaci. Častou metodou je sestavení C SVM modelů, kde c-tý model provádí binární klasifikaci, tedy zda vzorek patří do třídy c, či nikoliv. Vzorky třídy c jsou pozitivní případy a vzorky zbylých $C-1$ tříd jsou brány jako negativní. Tomuto přístupu se říká *one-versus-all* [11][14]. Nyní máme C rozhodovacích hranic a známé vzorky třídy c splňují podmínky $y_c(\mathbf{x}) > 0$ a $y_j(\mathbf{x}) < 0$, kde $j = 1, ..., C, j \neq c$. a $y_c(\mathbf{x}), y_j(\mathbf{x})$ výchází z klasifikační rovnice 2.16 pro každou třídu. Všeobecně tímto způsobem však nelze definitivně určit třídu pro vzorky z celého příznakového prostoru, existují regiony bodů v prostoru, pro které nejsou podmínky současně splněny. Tyto body leží na „pozitnvní“ straně rozhodovací hranice více než jednoho klasifikátoru nebo na „negativní“ straně všech rozhodovacích hranic. Fúzí výsledků klasifikace všech SVM klasifikátorů obejdeme tento problém odhadnutím třídy dle: [11][14]

$$\hat{t}(\mathbf{x}) = \underset{c \in C}{\operatorname{argmax}}\, y_c(\mathbf{x}) \tag{2.17}$$

2.2.3 Naive Bayes

Dalším klasifikátorem, který se jeví jako vhodný, je tzv. Naive Bayes klasifikátor. Slovo „Naive“ poukazuje na fakt, že se předpokládá vzájemná nezávislost příznaků [17]. Klasifikátor využívá známé Bayesovy věty. Jeho varianta zvaná *Multinomial Naive Bayes* je vhodná pro multinomické rozdělení a běžně se používá pro klasifikaci textu, kde je text reprezentován vektorem četností slov ze slovníku. Klasifikátor je tedy analogicky použitelný pro klasifikaci vektorů četností (histogramů) vizuálních slov, vznikajících metodou BoVW (sekce 1.6). Autoři článku [9], který popisuje metodu BoVW, rovněž experimentovali s Naive Bayes klasifikátorem, proto je vhodné ověřit tuto možnost na vlastní implementaci metody BoVW.

Aposteriorní pravděpodobnost, že vzorek x_i, reprezentovaný vektorem četností vizuálních slov $\mathbf{x_i}$, patří do třídy c je úměrná výrazu

$$P(c|\mathbf{x_i}) \propto P(c) \prod_{1 \leq k \leq n} P(t_k|c) \tag{2.18}$$

kde $P(t_k|c)$ je podmíněná pravděpodobnost výskytu slova t_k ve vzorku třídy c [15]. Je to míra toho, jak moc přispívá přítomnost t_k k tomu, že c je správnou třídou. $(t_1, t_2, ..., t_n)$ je množina slov, vyskytujících se ve vzorku x_i, které jsou součástí slovníku. $P(c)$ je apriorní pravděpodobnost, že vzorek x_i patří k třídě c. n je celkový počet slov ve vzorku. Nejlepší predikcí třídy je *maximum a posteriori* (MAP) třída, daná

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, \hat{P}(c|\mathbf{x_i}) = \underset{c \in C}{\operatorname{argmax}}\, \hat{P}(c) \prod_{1 \leq k \leq n} \hat{P}(t_k|c) \tag{2.19}$$

kde $\hat{P}$ je odhad parametru P z trénovacího datasetu, jak bude ukázáno. Protože častěji se vyskytující třídy jsou s větší pravděpodobností třídy správné, je odhad apriorní pravděpodobnosti dán vztahem

$$\hat{P}(c) = \frac{N_c}{N}, \tag{2.20}$$

kde N_c je počet vzorků ve třídě c a N je celkový počet vzorků. Výpočet odhadu podmíněné pravděpodobnosti $\hat{P}(t|c)$ se liší v závislosti na typu Bayesovského klasifikátoru, pro Multinomial NB je dán jako relativní četnost slova t ve vzorcích třídy c [17]:

$$\hat{P}(t_k|c) = \frac{T_{ct}}{\sum_t T_{ct}}, \tag{2.21}$$

kde T_{ct} je počet výskytů slova t v trénovacích vzorcích třídy c a jmenovatel je celkový počet slov ve třídě c. Možnost $\hat{P}(t|c) = 0$ pro slovo t, které se v trénovacích vzorcích třídy c vůbec nevyskytuje, vyloučíme pomocí tzv. *Laplacian smoothing* přidáním jedničky.

$$\hat{P}(t_k|c) = \frac{T_{ct} + 1}{\sum_t (T_{ct} + 1)}, \tag{2.22}$$

Pokud tak neučiníme, i jediný výskyt slova v testovacím vzorku ze třídy c, které nebylo přítomno v žádném trénovacím vzorku třídy c by způsobil nulovou aposteriorní pravděpodobnost $\hat{P}(c|\mathbf{x_i})$ podle rovnice 2.19 a tedy nesprávnou klasifikaci [15]. Rovnice 2.19 násobí velké množství podmíněných pravděpodobností, z implementačního hlediska je vhodné zavést logaritmy, čímž se násobení převede na sčítání a nehrozí podtečení proměnných s plovoucí desetinnou čárkou. Třída s nejvyšším $\log \hat{P}(c|\mathbf{x_i})$ bude stále nejpravděpodobnější [15].

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} [\log \hat{P}(c) \sum_{1 \leq k \leq n} \log \hat{P}(t_k|c)] \tag{2.23}$$

Po určení $\hat{P}(c)$ pro všechny třídy a $\hat{P}(t_k|c)$ pro všechny třídy a vzory slov ze slovníku jsou známy parametry modelu pro predikci nových vzorků. Rovnice 2.19 může být upravena do tvaru podle BoVW článku [9] tak, aby se daly klasifikovat obrazy reprezentované vektorem četností slov (vzorů) $\mathbf{x_i}$ ze slovníku.

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, \hat{P}(c|\mathbf{x_i}) = \underset{c \in C}{\operatorname{argmax}}\, \hat{P}(c) \prod_{k=0}^{S-1} \hat{P}(t_k|c)^{x_i^{(k)}}, \tag{2.24}$$

kde $x_i^{(k)}$ je k-tý prvek vektoru $\mathbf{x_i}$ a odpovídá četnosti vzoru t_k v obraze a S je velikost slovníku.

2.2.4 Neuronové sítě (NN)

Neuronové sítě se skládají z *perceptronů*, každý z nich lze popsat matematicky jako lineární kombinace vstupů a transformace *aktivační funkcí*. Mějme tedy M lineárních

kombinací vstupní veličiny $x_1, ..., x_D$

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.25}$$

kde $j = 1, ..., M$ a index (1) značí, do jaké vrstvy sítě patří dané parametry. Parametrům $w_{ji}^{(1)}$ se říká váhy (*weights*) a parametry $w_{j0}^{(1)}$ jsou označovány jako předpětí (*biases*). Předpětí může být absorbováno do sady vah zadefinováním pomocného vstupu $x_0 = 1$ do tvaru:

$$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i \tag{2.26}$$

Lineární kombinace vstupů jsou tranformovány diferencovatelnou, nelineární funkcí $h(\cdot)$:

$$z_j = h(a_j) \tag{2.27}$$

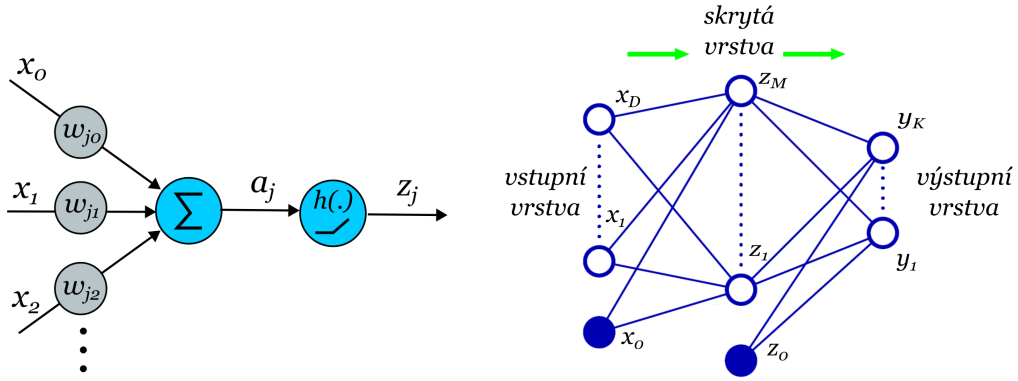
Výstupy z_j jsou vstupy do vyšší vrstvy sítě:

$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j \tag{2.28}$$

Předpokládejme, že tato vrstva je již výstupní, pak $k = 1, ..., K$ je počet výstupů a je použita vhodná aktivační funkce generující výstupy $y_k = \sigma(a_k)$. Celková funkce této jednoduché sítě je dána rovnicí

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=0}^{M} w_{kj}^{(2)} h \left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right) \tag{2.29}$$

a taková sít je ilustravána vpravo na obr. 2.2.



Obr. 2.2: Detail j-té jednotky (perceptronu) neuronové sítě (vlevo) a diagram neuronové sítě s jednou skrytou vrstvou (vpravo) [14], upraveno

Model neuronové sítě je tedy jednoduše nelineární transformace vstupů x_i na výstupy y_k, řízená nastavitelnými parametry $\mathbf{w}$. Vyhodnocení rovnice 2.29 je dopředná

propagace informace sítí (*forward propagation*). Trénování neuronové sítě zpočívá v ladění parametrů $\mathbf{w}$ tak, aby byla chybová funkce (*loss function*) $E(\mathbf{w})$ minimalizována. V závislosti na typu problému se volí vhodná chybová funkce a aktivační funkce výstupní vrstvy. Pro multiclass klasifikaci se volí tzv. *softmax* funkce a chybová funkce *multiclass cross-entropy*. Pro multiclass problémy je třída indikována jako kód 1 *z n* pomocí proměnné $t_k \in \{0, 1\}$ a výstup $y_k(\mathbf{x}, \mathbf{w})$ pro danou třídu je interpretován jako $y_k(\mathbf{x}, \mathbf{w}) = p(t_k = 1|\mathbf{x})$. Pak ztrátová funkce nabírá tvaru

$$E(\mathbf{w}) = -\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ki} \ln y_k(\mathbf{x}_i, \mathbf{w}) \tag{2.30}$$

a softmax funkce

$$y_k(\mathbf{x}, \mathbf{w}) = \frac{e^{a_k(\mathbf{x},\mathbf{w})}}{\sum_j e^{a_j(\mathbf{x},\mathbf{w})}} \tag{2.31}$$

zaručuje $0 \leq y_k \leq 1$ a $\sum_k y_k = 1$. Výstup $y_k(\mathbf{x}, \mathbf{w})$ opravdu tedy reprezentuje pravděpodobnost příslušnosti vzorku ke k-té třídě.

Úlohou optimalizace parametrů je najít takový vektor vah $\mathbf{w}$, aby byla funkce $E(\mathbf{w})$ minimalizována. Protože funkce $E(\mathbf{w})$ je hladká spojitá funkce $(\mathbf{w})$, nejmenší hodnotu nabývá v bodě, kdy gradient chybové funkce je roven nule, tedy:

$$\nabla E(\mathbf{w}) = 0 \tag{2.32}$$

Gradient $\nabla E(\mathbf{w})$ je vektor ve směru největšího růstu chybové funkce $E(\mathbf{w})$ a formálně parciální derivace chybové funkce. Nelze určit, zda bod, kdy $\nabla E(\mathbf{w}) = 0$, je globální nebo pouze lokální minimum, protože nelze najít analytické řešení této rovnice. Uchylujeme se tedy k iterativním numerickým metodám. Nejjednodušší je využít informaci o gradientu chybové funkce a upravit váhy o malý krok ve směru negativního gradientu

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)}) \tag{2.33}$$

kde $\eta > 0$ je tzv. *learning rate*. Výchozí vektor vah je zvolen náhodně. Po každém přenastavení vah je znovu vyhodnocen gradient a proces opakován. V každém kroku je vektor vah posouván ve směru největšího poklesu chybové funkce. Tomuto přístupu se říká *gradient descent* [14]. Parametr η určuje velikost kroku aktualizace vah. Pokud má příliš malou hodnotu, algoritmus projde mnoho iterací než dojde ke konvergenci. Naopak velká hodnota může způsobit přeskočení minima a divergenci algoritmu [12].

Pokud je využit celý dataset k výpočtu gradientu a až poté dojde k aktualizaci vah, mluvíme o tzv. *batch gradient descent*. Opačným extrémem je *stochastic gradient descent* (SGD), který vybere náhodný vzorek v každém kroku a aktualizuje váhy na základě gradientu vypočítaného pouze z tohoto vzorku. Algoritmus je pro velké datasety mnohem rychlejší, na druhou stranu vlivem náhodnosti chybová funkce

neklesá monotónně, ale fluktuuje a klesá jen v delším horizontu pozorování. To také umožňuje v některých případech opustit lokální minimum a najít optimálnější řešení. *Mini-batch gradient descent* je kombinací obou přístupů, gradient se počítá z náhodné skupiny vzorků (mini-batch) a následně se aktualizují váhy [12].

Trénovací algoritmy tedy iterativně minimalizují chybovou funkci a každá iterace probíhá ve dvou krocích. Prvně je nutno vypočítat gradient ztrátové funkce vzhledem ke každému parametru w_i. V druhém kroku jsou pak využity gradienty v metodách jako gradient descent k aktualizaci vah. Počet epoch udává, kolikrát byl celý dataset opakovaně použit v trénovacím algoritmu. Metoda *backpropagation* znamenala průlom v trénování neuronových sítí, protože představila efektivní způsob výpočtu gradientu. Nyní bude odvozen algoritmus backpropagation a jak jsou gradienty vypočítány. Chybová funkce $E(\mathbf{w})$ je dána součtem dílčích chybových funkcí, každá pro jeden vzorek:

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}) \tag{2.34}$$

Cílem je vypočítat gradient $\nabla E_n(\mathbf{w})$ pro každý z N vzorků a využít je pro metodu SGD nebo je akumulovat pro batch a mini-batch gradient descent metody. Každá jednotka (perceptron) obecné neuronové sítě provádí váhovaný součet vstupů

$$a_j = \sum_i w_{ji} z_i \tag{2.35}$$

a transformaci nelineární aktivační funkcí $z_j = h(a_j)$. Hledáme gradient funkce E_n, tedy parciální derivaci E_n podle w_{ji}. Řetězovým pravidlem dostáváme

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}. \tag{2.36}$$

Zavedeme notaci $\delta_j = \frac{\partial E_n}{\partial a_j}$ a dále z rovnice 2.35 můžeme napsat

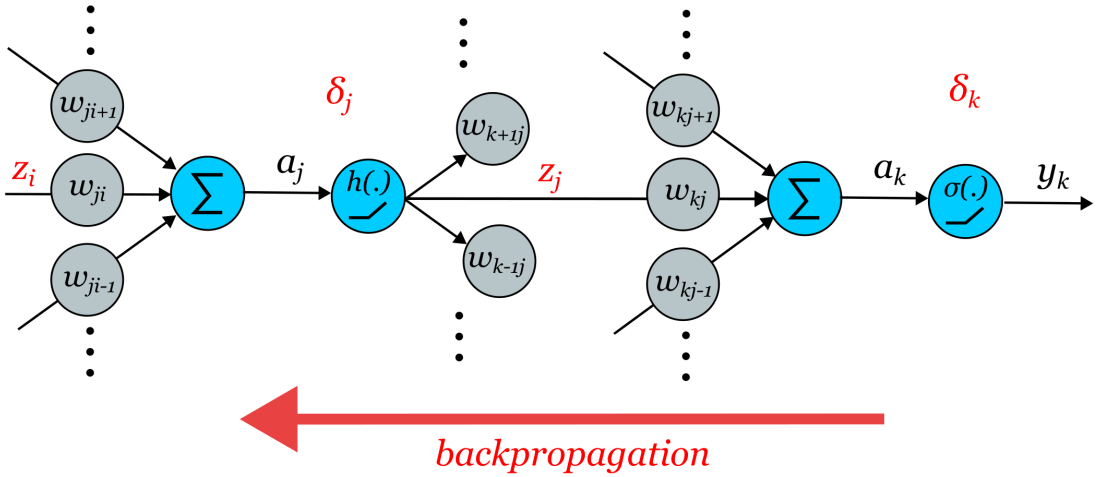
$$\frac{\partial a_j}{\partial w_{ji}} = z_i. \tag{2.37}$$

Substitucí δ_j a 2.37 do 2.36 obdržíme

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i. \tag{2.38}$$

Rovnice 2.38 nám říká, že potřebná parciální derivace je dána hodnotou δ dané jednotky a výstupu z předcházející jednotky procházející danou váhou. Takže pro výpočet všech parciálních derivací stačí určit δ_j pro každou jednotku a aplikovat rovnici 2.38. Výpočty jednotlivých δ začínáme od výstupní vrstvy podle

$$\delta_k = y_k - t_k \tag{2.39}$$



Obr. 2.3: Ilustrace výpočtu δ_j zpětnou propagací δ z vyšší vrstvy. Veličiny pro výpočet potřebných parciálních derivací chybové funkce jsou značeny červeně.

a postupujeme sítí pozpátku. Pro výpočet δ skrytých jednotek využijeme opět řetězové pravidlo:

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \tag{2.40}$$

kde suma je přes všechny jednotky k, do kterých vstupuje výstup jednotky j. Z rovnice 2.35 plyne $a_k = \sum_j w_{kj} z_j = \sum_j w_{kj} h(a_j)$ a pak platí

$$\frac{\partial a_k}{\partial a_j} = w_{kj} h'(a_j). \tag{2.41}$$

Substitucí 2.41 a $\delta_k = \frac{\partial E_n}{\partial a_k}$ do rovnice 2.40 dostáváme

$$\delta_j = \sum_k \delta_k w_{kj} h'(a_j) = h'(a_j) \sum_k w_{kj} \delta_k. \tag{2.42}$$

Tato rovnice je klíčová, protože nám říká, že lze získat δ pro kteroukoliv jednotku zpětnou propagací δ z jednotek ve vyšších vrstvách sítě (od výstupní vrstvy). Jakmile známe δ_j dané jednotky pak je jednoduché vypočítat žádanou parciální derivaci chybové funkce dle rovnice 2.38 [14].

Shrnutí metody backpropagation

1. Dopředná propagace vstupů x_n sítí a určení výstupů z všech jednotek užitím rovnic 2.35 a 2.27.
2. Vyhodnocení δ_k pro všechny jednotky ve výstupní vrstvě podle 2.39.
3. Zpětná propagace δ k výpočtu δ_j každé jednotky podle 2.42.
4. Výpočet parciálních derivací chybové funkce podle 2.38

Po získání parciálních derivací chybové funkce (gradient) je použit gradient descent pro aktualizaci vah. U batch metod se parciální derivace celkové chybové funkce určí jednoduchým součtem parciálních derivací chybové funkce dané jedním vzorkem přes všechny vzorky

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}} \tag{2.43}$$

2.2.5 Ensemble Learning

Ensemble learning nebo také meta učení je metoda využívající k regresi/klasifikaci kombinaci vícero modelů za předpokladu, že je možné kombinací modelů dosáhnout lepšího výsledku, než dosahuje nejlepší samostatný model [12]. Mezi nejzákladnější přístupy meta učení patří:

Voting classifiers je skupina klasifikátorů různých typů naučená na stejných datech. Jednoduchou možností, jak dosáhnout vyšší správnosti je shromáždit predikce jednotlivých klasifikátorů a vybrat tu s největším počtem výskytů. Tento přístup se nazývá *hard voting*. Pro regresní modely je predikcí aritmetický průměr. Pokud skupina klasifikátorů umí predikovat pravděpodobnosti jednotlivých tříd, vypočítá se aritmetický průměr pravděpodobnosti každé třídy přes všechny klasifikátory a vybere se třída s nejvyšší průměrnou pravděpodobností (*soft voting*) [12].

Bagging je metoda učení klasifikátorů stejného typu, ale každý model je učen na jiné podmnožině datasetu. Tyto podmnožiny jsou tvořeny výběrem prvků s opakováním, stejně jako pro metodu Bootstrap (sekce 2.4.3). Konečná predikce je určena opět hlasováním jednotlivých modelů [12].

Boosting Myšlenkou boostingu je řetězec modelů, tzv. *weak learners*, kde se další následující model více zaměřuje na vzorky, které byly špatně klasifikovány modelem předchozím. Nejznámnějším algoritmem této skupiny je AdaBoost. Základní model je natrénován na celém datasetu a následně se zvýší váhy nesprávně klasifikovaných vzorků. Druhý model je natrénován na datasetu s aktualizovanými váhami a proces se opakuje. Konečná predikce je vypočítána z váhovaných predikcí jednotlivých modelů. Váha modelu závisí na jeho správnosti klasifikace [12].

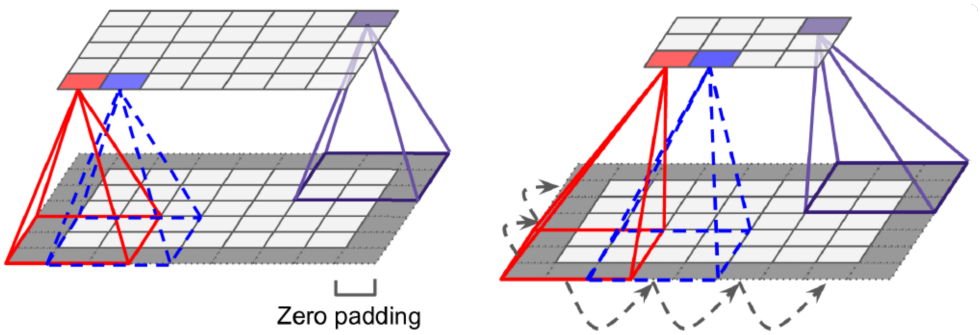
Stacking je podobným přístupem jako hlasování modelů různých typů, ale místo prostého hlasování (*hard voting*) jsou výstupy bázových modelů brány jako vektor příznaků pro natrénování jednoduchého modelu, který stojí o úroveň výš než bázové modely. Výstup nadřazeného modelu je konečnou predikcí [12].

2.2.6 Konvoluční neurové sítě (CNN)

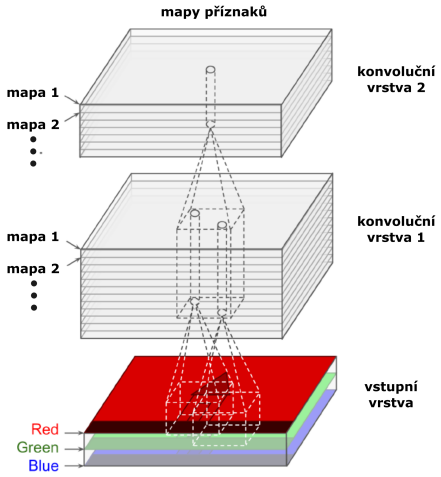
Speciálním typem vícevrstvých neuronových sítí jsou konvoluční neuronové sítě (CNN, ConvNet.), inspirované vnímáním vizuální informace žijících organismů. Hluboká CNN síť se skládá z konečného počtu vrstev, které se učí rozpoznávat charakteristiky vstupního obrazu s různou úrovní abstrakce [25]. Klíčovou vlastností CNN je schopnost naučit se hierarchii vzorů v obraze. První konvoluční vrstva se naučí rozpoznávat lokální vzory jako jsou hrany, druhá vrstva se naučí rozpoznávat složitější vzory z výstupu první vrstvy atd. [26]. Tradiční konvoluční síť je tvořena jedním, či více bloky konvolučních a tzv. pooling vrstev, následovaných několika plně propojenými vrstvami a výstupní vrstvou.

Konvoluční vrstva se skládá z několika konvolučních kernelů (filtrů), pomocí kterých se vypočítají tzv. mapy příznaků (*feature maps*) [25]. Každá z map příznaků vznikne konvolucí vstupního obrazu a jednoho filtru, podobně jak je popsáno v sekci 1.2.1 s tím rozdílem, že váhy filtrů nejsou určené inženýrem, ale jsou naučeny v rámci procesu tréninku konvoluční neuronové sítě. Následně je výstup transformován aktivační funkcí.

Pokud se na konvoluční vrstvu podíváme ze stejného pohledu, jako jsme popsali klasické neuronové sítě, pak do každého neuronu konvoluční vrstvy vstupují pouze hodnoty určitého okolí vstupního obrazu. Výstupem vrstvy neuronů se stejnými váhami (filtrem) je pak jedna mapa příznaků, která zvýrazňuje rysy v obraze, na které je daný filtr citlivý. V praxi využívané konvoluční vrstvy mají n filtrů a výstupem je n map příznaků. Výstup konvoluční vrstvy je reprezentován ve trojrozměrném prostoru „naskládáním“ map příznaků ve směru třetí dimenze. Konvoluční vrstva tak má jeden neuron pro každý pixel každé mapy příznaků. V rámci jedné mapy příznaků mají neurony identické váhy, váhy neuronů napříč mapami se liší. Stejné váhy v rámci neuronů v dané mapě příznaků dramaticky snižují celkový počet parametrů [12]. Konvoluce probíhá v zorném poli (*receptive field*) každého neuronu, daného velikostí filtru, a napříč všemi mapami příznaků z předcházející vrstvy. Často je žádoucí, aby výstupní mapy příznaků měly stejný rozměr jako vstupní obraz, toho se dosáhne přidáním “rámečku” nulových hodnot kolem vstupního obrazu (*zero padding*).Také je možné vytvořit mapy příznaků menších rozměrů zmenšením počtu neuronů a tedy zmenšením překryvu jejich zorného pole při zachování velikosti filtru. Tato vlastnost je řízena parametrem *stride*. Pro matematickou definici konvoluce by parameter stride odpovídal hodnotě 1. Obě situace jsou ilustrovány na obrázku 2.4. Každou konvoluční vrstvu následuje mapování výstupu aktivační funkcí [12].



Obr. 2.4: Ilustrace konvoluce se zero padding (vlevo) a zmenšení mapy příznaků parametrem *stride* = 2 (vpravo) [12]



Obr. 2.5: Ilustrace výpočtu map příznaků konvolucí [12], upraveno

Poolingová vrstva podvzorkuje vstupní mapy příznaků, čili zmenší jejich původní rozměr. Neurony této vrstvy nemají žádné váhy, provadí pouze agregační funkci typu max nebo mean na oknech s definovanou velikostí a parametrem stride. Nejčastější pooling vrstvou je max pooling vrstva [12]. Cílem pooling vrstvy je zmenšení dat, které je nutné ve vyšších vrstvách zpracovat, a také umožnit následujícím konvolučním vrstvám zpracovat větší oblasti obrazu (z hlediska původního vstupního obrazu) a rozpoznávat složitější vzory.

Horní vrstvy CNN jsou tvořeny klasickou *fully-connected* (FC) neuronovou sítí, kde je každý neuron vrstvy spojen se všemi neurony vrstvy předchozí. Mapy příznaků poslední konvoluční nebo pooling vrstvy jsou převedeny na jednorozměrný vektor, který je vstupem FC sítě [25]. FC síť pak vyprodukuje finální výstup CNN sítě, např. v případě klasifikace predikovanou třídu.

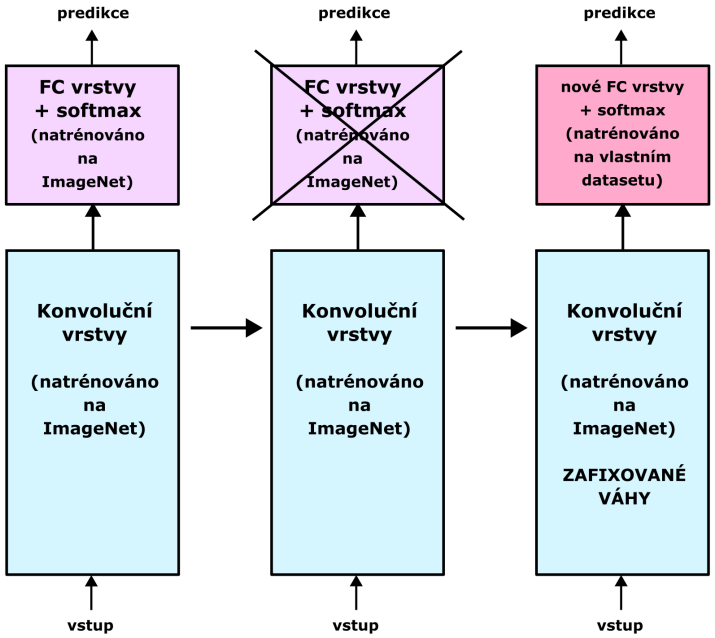
Architektura CNN Typickou CNN tvoří opakující se sekvence bloků tvořených několika konvolučními vrstvami s ReLU aktivační funkcí a pooling vrstvou. Rozměry map příznaků se průchodem sítí zmenšují, ale jejich počet se typicky zvětšuje. Nad těmito bloky konvolučních a poolingových vrstev je FC síť s několika skrytými vrstvami a výstupní vrstvou.

Transfer learning Intuitivním způsobem řešení dané úlohy strojového učení využitím CNN je navrhnout vlastní CNN a natrénovat ji na vlastním datasetu. Rychlejší cestou je použít některou z prověřených CNN architektur, která byla natrénována na velkém datasetu podobného charakteru, jako výchozí bod pro trénink sítě na vlastním datasetu [27]. Tomuto postupu se říká transfer learning. Výhodou je velké zrychlení trénovacího procesu a odpadá nutnost disponovat vlastním obsáhlým datasetem [12]. Typicky jsou použity váhy pouze části neuronové sítě s konvolučními vrstvami, než kompletně celé sítě. To je velmi efektivní, protože naučit spodní konvoluční vrstvy extrakci základních vzorů v obraze je lepší použitím obsáhlého datasetu [29].

Transfer learning tedy zahrnuje použití již ověřené architektury CNN. *Pretraining* je podobný koncept jako transfer learning, s tím rozdílem, že v rámci pretrainingu inženýr definuje vlastní architekturu CNN a tuto síť natrénuje na velkém datasetu jako je např. ImageNet. Pretraning umožňuje inicializaci vah využitím velkého datasetu a zároveň flexibilitu v návrhu architektury sítě [29].

Podle [27] lze transfer learning rozdělit do tří kategorií:

1. Využijeme celou síť s předtrénovanými váhami včetně FC vrstev- obvykle se používá, pokud je doména daného problému velmi podobné s velkým datasetem, na kterém byla síť natrénována. Není potřeba žádného tréninku na vlastním datasetu.
2. Využijeme část osvědčené sítě extrahující příznaky v obraze (konvoluční vrstvy) a optimalizované váhy „zmrazíme“. Navrhneme horní FC vrstvy s náhodnou inicializací vah, tyto váhy natrénujeme na vlastním datasetu. Zmrazit váhy znamená, že se jejich hodnota při tréninku nemění.
3. Tzv. fine-tuning, kdy jsou zmraženy váhy jen spodnějších konvolučních vrstev. Váhy horních konvolučních vrstev dotrénujeme na vlastním datasetu, stejně jako náhodně inicializované váhy FC vrstev. Kolik horních konvolučních vrstev si můžeme dovolit dotrénovat na vlastním datasetu závisí na velikosti a míře podobnosti vlastního datasetu s původním obsáhlým datasetem. Aktualizovat váhy spodních konvolučních vrstev se nedoporučuje, protože tyto vrstvy jsou již naučené extrahovat obecné vzory, které jsou společné pro téměř všechny úkoly zpracování obrazu, na které má být systém použit.

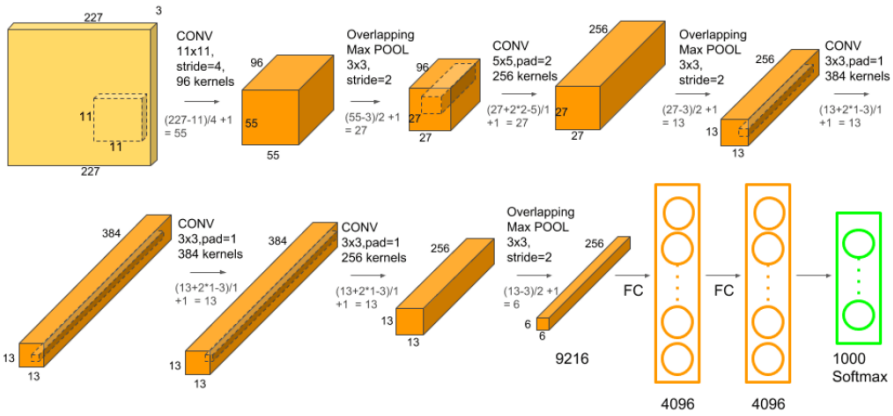


Obr. 2.6: Ilustrace transfer learningu pro obecnou CNN architekturu

2.2.7 Osvědčené architektury CNN

Tato sekce se zabývá popisem vybraných architektur konvolučních neuronových sítí pro klasifikaci, se kterými bylo experimentováno v praktické realizaci. Tyto architektury v průběhu let posunuly hranici možností v odvětví klasifikace obrazu. Měřítkem pokroku je soutěž ILSVRC ImageNet challenge. Dataset pro klasifikaci obrazu v rámci soutěže ILSVRC je podmnožinou obsáhlejšího datasetu ImageNet. Tato podmnožina obsahuje 1,2 milionu trénovacích, 50 tisíc validačních a 150 tisíc testovacích snímků, rozdělených do 1000 tříd (asi 1000 snímků v každé třídě) [30].

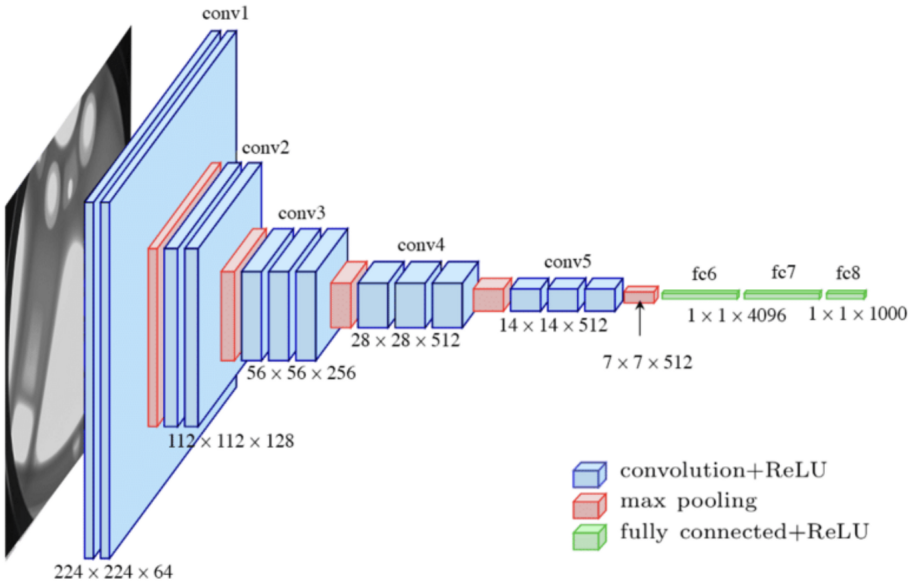
AlexNet AlexNet je jednou z nejznámějších klasických CNN architektur, která s velkým náskokem vyhrála soutěž ILSVRC-2012 v oblasti klasifikace. Publikace článku [30] a výsledek v soutěži vedly k velké vlně zájmu o oblast konvolučních neuronových sítí. Autoři uvádí, že ve své době to byla jedna z největších sítí natrénovaná na datasetu ImageNet. Síť se skládá z pěti konvolučních a tří FC vrstev. První konvoluční vrstva má filtr o velikosti 11x11 a po navazující poolingové vrstvě následuje druhá konvoluční vrstva s filtrem 5x5. Třetí, čtvrtá a pátá vrstva jsou spojeny bez poolingových vrstev mezi nimi a obsahují filtr 3x3. Architektura s rozměry vrstev je znázorněna na obr. 2.7.



Obr. 2.7: Ilustrace architektury sítě AlexNet [36]

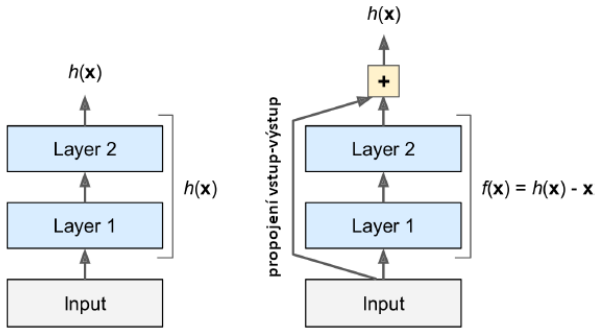
VGG16 je jednou z architektur týmu VGG, který v soutěži ILSVRC-2014 challenge v oblasti klasifikace obsadil druhé místo. VGG sítě se vyznačuji svou jednoduchostí a klasickým přístupem ke konvolučním sítím [12]. Číslovka v názvu odpovídá hloubce dané architektury (počtu vrstev). Vstupem jsou obrazy velikosti 224x224 pixelů s odečtenou střední hodnotou jasu trénovacího datasetu. Architekturu charakterizuje použití hlubší sítě, kde konvoluční vrstvy mají filtry s malými rozměry: 3x3. Autoři [31] architektury uvádí, že dvě konvoluční vrstvy s filtrem 3x3 za sebou mají zorné pole stejné jako konvoluční vrstva s filtrem 5x5; tři takové vrstvy mají zorné pole jako vrstva s filtrem 7x7. Výhodou stohu tří konvolučních vrstev s menším jádrem je použití tří ReLu aktivačních funkcí místo jedné, čímž je rozhodovací funkce více diskriminativní. Navíc je výrazně zredukován počet vah sítě. V praktické realizaci bylo experimentováno se 16vrstvou architekturou, která je v článku [31] označena jako konfigurace D. Schéma architektury VGG16 je na obr. 2.8.

ResNet je vítěz soutěže ILSVRC 2015. Vítězná variace architektury se 152 vrstvami potvrdila trend extrémně hlubokých modelů s nižším počtem parametrů [12]. Síť je tvořena stavebními bloky, tzv. *residuálními jednotkami*. Každá jednotka obsahuje vedle konvolučních vrstev i přímé propojení ze vstupu jednotky na výstup jednotky. Cílem je modelovat funkci $h(\mathbf{x})$. Pokud přidáme vstup $\mathbf{x}$ k výstupu konvolučních vrstev, konvoluční vrstvy budou nuceny modelovat funkci $f(\mathbf{x}) = h(\mathbf{x}) - \mathbf{x}$ místo $h(\mathbf{x})$. Tomuto se říká residuální učení, residuální blok je pak ilustrován na obrázku 2.9. Autoři v [32] uvádí, že residuální sítě se snadněji optimalizují a přesnost sítě se zlepšuje jejím prohlubováním. Varianty ResNet-50, ResNet-101 a ResNet-152 využívají residuální jednotky řazené za sebou s třemi konvolučními vrstvami a propojením vstup-výstup jednotky. Tyto sítě se pak liší pouze počtem takovýchto jednotek a číslo v názvu sítě odpovídá počtu vrstev. Síť je zakončena *global average*



Obr. 2.8: Ilustrace architektury sítě VGG16 [37]

pooling vrstvou a výstupní softmax vrstvou. Global average pooling vytvoří vektor hodnot pro softmax vrstvu výpočtem průměru každé ze vstupních feature map.



Obr. 2.9: Ilustrace residuální jednotky [12]

2.2.8 Augmentace obrazových dat

Augmentace dat je jednou z metod prevence přetrénování modelu (overfitting), tedy situace, kdy model špatně generalizuje při klasifikaci nových dat. Augmentované snímky uměle nafukují trénovací dataset. Předpokladem pak je, že skrze augmentaci je možné extrahovat z originálního datasetu více informací. Obecně se má za to, že větší dataset umožňuje lepší modely hlubokého učení [29].

Augmentace obrazovými transformacemi je základní augmentace obrazových dat vycházející z jednoduchých obrazových a jasových transformací, jako je např. zrcadlové otočení, rotace, jasové transformace, barevné transformace a náhodné oříznutí. Jasové transformace pomáhají s rozdílným osvětlením snímků v datasetu [29].

Augmentace dat na základě strojového učení se na rozdíl od augmentací výše, které jsou prováděny na vstupním obraze, provádí v prostoru příznaků na vektoru příznaků, který reprezentuje obraz v nižší dimenzi. SMOTE je oblíbená metoda augmentace méně zastoupených tříd v nevyváženém datasetu [29]. Metoda generuje syntetické vzorky následovně: Provede se náhodný výběr vzorku z dané třídy a jeho k nejbližších sousedů z této třídy. Vypočte se rozdíl vektoru příznaků náhodně vybraného vzorku a jednoho souseda. Nový syntetický příznakový vektor je určen vynásobením rozdílu náhodným číslem v intervalu $(0, 1)$ a přičten k vektoru příznaků onoho vybraného vzorku. Nový vzorek tak leží v prostoru příznaků na úsečce mezi náhodně zvoleným vzorkem a jeho sousedem.

Pokročilou metodou augmentace jsou Generative Adversarial Networks (GAN). GAN generuje nové instance, podobné původním, na základě interního soupeření dvou neuronových sítí. První síť (*discriminator*) rozhoduje o pravosti vzorku, zatímco druhá síť (*generator*) se snaží diskriminátor oklamat tak, aby vygenerovaný vzorek nerozpoznal od vzorků skutečných z datasetu. Po natrénování systému je generátor schopen generovat přesvědčivé syntetické vzorky [26].

2.2.9 Rychlejší optimalizace vah

Trénink hlubokých neuronových sítí může být časově náročný. Nahrazením základního optimalizačního algoritmu *gradient descent* (sekce 2.2.4) efektivnějším algoritmem se proces trénování urychlí. Pro připomenutí je uveden matematický předpis aktualizace vah pomocí algoritmu gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)}) \tag{2.44}$$

kde $\eta > 0$ je *learning rate* a $\nabla E(\mathbf{w})$ gradient chybové funkce. Gradient descent při aktualizaci vah nebere v potaz předchozí hodnoty gradientu, pouze aktuální hodnotu. Pokud je gradient malý, váhy se aktualizují pomalu [12].

Momentum optimization zohledňuje předchozí hodnoty gradientu, v každém kroku odečítá aktuální hodnoty gradientu od tzv. *vektoru setrvačností* $\mathbf{m}$ a aktualizuje váhy přičtením tohoto vektoru.

$$\mathbf{m}^{(\tau+1)} = \beta \mathbf{m} - \eta \nabla E(\mathbf{w}^{(\tau)}) \tag{2.45}$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \mathbf{m}^{(\tau)} \tag{2.46}$$

kde β je brzdící koeficient, který zabraňuje neomezenému růstu vektoru setrvačností. Nevýhodou algoritmu je zavedení druhého hyperparametru k vyladění, avšak typickou hodnotou je $\beta = 0.9$ [12].

Adagrad (*Adaptive Gradient Algorithm*) je algoritmus, který adaptivně škáluje learning rate η pro každou váhu v každé iteraci.

$$\mathbf{s}^{(\tau+1)} = \mathbf{s}^{(\tau)} + \nabla E(\mathbf{w}^{(\tau)}) \otimes \nabla E(\mathbf{w}^{(\tau)}) \tag{2.47}$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)}) \oslash \sqrt{\mathbf{s}^{(\tau)} + \epsilon} \tag{2.48}$$

kde $\otimes$ je násobení po prvcích, $\oslash$ dělení po prvcích a $\epsilon = 10^{-10}$ zabraňuje dělení nulou. V rovnici 2.47 vektor $\mathbf{s}$ akumuluje kvadrát gradientu a rovnice 2.48 je téměř shodná s gradient descent, akorát je gradient škálován výrazem $\sqrt{\mathbf{s} + \epsilon}$. AdaGrad pomáhá směrovat změnu vah k přímějšímu dosažení globálního optima. Nevýhodou je, že algoritmus často zastaví před dosažením optima, protože gradienty jsou škálovány na příliš malé hodnoty [12].

RMSProp opravuje tento nedostatek AdaGrad algoritmu. Místo akumulace všech gradientů do vektoru $\mathbf{s}$ od spuštění jsou gradienty z minulých iterací exponenciálně zapomínány (rovnice 2.49) [12].

$$\mathbf{s}^{(\tau+1)} = \beta \mathbf{s}^{(\tau)} + (1 - \beta) \nabla E(\mathbf{w}^{(\tau)}) \otimes \nabla E(\mathbf{w}^{(\tau)}) \tag{2.49}$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)}) \oslash \sqrt{\mathbf{s}^{(\tau)} + \epsilon} \tag{2.50}$$

Parametr zapomínání je typicky $\beta = 0.9$.

Adam algoritmus kombinuje princip algoritmů Momentum optimization a RMSProp, což je patrné v následujících rovnicích.

$$\mathbf{m}^{(\tau+1)} = \beta_1 \mathbf{m}^{(\tau)} - (1 - \beta_1) \nabla E(\mathbf{w}^{(\tau)}) \tag{2.51}$$

$$\mathbf{s}^{(\tau+1)} = \beta_2 \mathbf{s}^{(\tau)} + (1 - \beta_2) \nabla E(\mathbf{w}^{(\tau)}) \otimes \nabla E(\mathbf{w}^{(\tau)}) \tag{2.52}$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta \mathbf{m}^{(\tau)} \oslash \sqrt{\mathbf{s}^{(\tau)} + \epsilon} \tag{2.53}$$

Parametry zapomínání jsou typicky nastaveny na hodnoty $\beta_1 = 0.9$ a $\beta_2 = 0.999$. Adam je stejně jako AdaGrad a RMSProp adaptivní algoritmus a nevyžaduje přílišné ladění parametru η, což jeho použití činí velmi jednoduchým.

2.3 Učení bez učitele

V případě učení bez učitele (*unsupervised learning*) je datasetem kolekce vzorků $\{(\mathbf{x_i}, y_i)\}_{i=1}^{N}$ bez výstupní veličiny, kde $\mathbf{x_i}$ je opět vektor příznaků [13]. Např. shlukování *clustering* cílí na nalezení skupin podobných vzorků v datasetu, výstupem je pak ID skupiny (*clusteru*) pro každý vektor příznaků. Další skupinou učení bez učitele jsou algoritmy redukující dimenzi, výstupem je transformovaný vektor příznaků, který má menší počet příznaků než ten vstupní. Výstupem detekce anomálií (*anomaly, outlier detection*) je číslo indikující, jak je vzorek $\mathbf{x_i}$ odlišný od „normálních" vzorků v datasetu [13]. *Density estimation* odhaduje rozdělení hustoty pravděpodobnosti vzorků v prostoru příznaků [14]. V [12] jsou uvedeny příklady algoritmů jednotlivých úloh učení bez učitele:

- Shlukování
  - K-Means
  - DBSCAN
- Detekce anomálií
  - One-class SVM
  - Isolation Forest
- Redukce dimenze
  - Principal Component Analysis (PCA)

2.3.1 K-means

Cílem shlukovacího algoritmu k-Means je rozdělit dataset v prostoru příznaků do K shluků. Algoritmus inicializujeme výběrem D-dimenzionálních vektorů $\boldsymbol{\mu}_k$, kde $k = 1, ..., K$, které reprezentují středy K shluků. Běžně se jako $\boldsymbol{\mu}_k$ volí K náhodně vybraných vzorků (příznakových vektorů) z datasetu. Cílem je najít takové přiřazení vzorků do skluků, aby byl součet kvadrátu vzdáleností každého vzorku od nejbližšího středu $\boldsymbol{\mu}_k$ minimální [14].

Pro matematický popis algoritmu [14] zavedeme pro každý vzorek $\mathbf{x}_i$ binární proměnnou $r_{ik} \in 0, 1$,kde $k = 1, ..., K$, která uvádí ke kterému z K shluků je vzorek $\mathbf{x}_i$ přiřazen. Pokud vzorek $\mathbf{x}_i$ patří do shluku k, pak $r_{ik} = 1$ a $r_{ij} = 0$ pro $j \neq k$. Potom můžeme definovat účelovou funkci

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \left\| \mathbf{x}_i - \boldsymbol{\mu}_k \right\|^2, \tag{2.54}$$

která vyjadřuje součet kvadrátu vzdálenosti každého vzorku k jeho přiřazenému středu $\boldsymbol{\mu}_k$. Cílem je najít hodnoty r_{ik} a $\boldsymbol{\mu}_k$ tak, aby byla hodnota funkce J minimalizována. To se provádí iterativně, kdy každá iterace zahrnuje dva kroky optimalizace vzhledem k r_{ik} a $\boldsymbol{\mu}_k$. Po inicializaci středů $\boldsymbol{\mu}_k$ se v prvním kroku minimalizuje J

vzhledem k r_{ik} s konstantním $\boldsymbol{\mu}_k$. Tento krok je proveden přiřazením i-tého vzorku k nejbližšímu středu $\boldsymbol{\mu}_k$. Formálně tento krok lze vyjádřit jako:

$$r_{ik} = \begin{cases} 1, & k = \text{argmin}_j \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2, \\ 0, & \text{jinak.} \end{cases} \tag{2.55}$$

Druhý krok je optimalizace $\boldsymbol{\mu}_k$ s konstantním r_{ik}, který je matematicky vyjádřen vztahem:

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} r_{ik} \mathbf{x}_i}{\sum_{i=1}^{N} r_{ik}} \tag{2.56}$$

Jmenovatel výrazu 2.56 je roven počtu prvků v shluku k, rovnice lze interpretovat tak, že střed $\boldsymbol{\mu}_k$ je dán aritmetickým průměrem všech vzorků $\mathbf{x}_i$, které jsou přiřazeny do shluku k.

Tyto dvě fáze, přerozdělení vzorků do shluků a přepočítání středů shluků, jsou opakovány, dokud dochází v první fázi ke změně v přiřazení vzorků do shluků nebo dokud nebyl dosažen maximální počet iterací. Protože každá iterace snižuje hodnotu funkce J (2.54), konvergence algoritmu je zajištěna. Fáze K-Means algoritmu odpovídají fázím E (*expectation*) a M (*maximization*) EM algoritmu, proto se také kroky K-Means označují jako E a M [14].

SGD k-means

Klasická metoda k-means EM algoritmem trpí výpočetní náročností a dobou konvergence u velkých datasetů. Výpočetně rychlejší alternativou k řešení k-means EM algoritmem je gradient descent. Stejně jako u neuronových sítí se setkáváme s variantou stochastic gradient descent (SGD), která počítá změnu středů shluků po zpracování každého vzorku [19]. Stochastic gradient descent k-means funguje následovně:

1. Je dán počet shluků K a T iterací.
2. Inicializuj středy $\boldsymbol{\mu}_k$ jako náhodné vzorky $\mathbf{x}_i$.
3. Vynuluj čítače n_k, které udávají počet vzorků patřících shluku k.
4. V cyklu $t = 1$ až T proveď:
   (a) Přiřaď prvku $\mathbf{x}_i$ nejbližší střed $\boldsymbol{\mu}_k$.
   (b) Inkrementuj čítač n_k pro tento střed $\boldsymbol{\mu}_k$.
   (c) Urči learning rate jako $\eta = \frac{1}{n_k}$.
   (d) Aktualizuj střed tohoto shluku podle $\boldsymbol{\mu}_k = (1 - \eta)\boldsymbol{\mu}_k + \eta \mathbf{x}_i$.

Mini-batch varianta využívá k aktualizaci středů dávku o M prvcích a poté rozřadí vzorky do nových shluků:

1. Je dán počet shluků K, velikost dávky M a T iterací.
2. Inicializuj středy $\boldsymbol{\mu}_k$ jako náhodné vzorky $\mathbf{x}_i$.
3. Vynuluj čítače n_k, které udávají počet vzorků patřících shluku k.
4. V cyklu $t = 1$ až T proveď:
   (a) Vyber M vzorků do dávky.
   (b) V cyklu $\mathbf{x}_i$ až $\mathbf{x}_M$ proveď:
      i. Přiřaď prvku $\mathbf{x}_i$ nejbližší střed $\boldsymbol{\mu}_k$.
   (c) V cyklu $\mathbf{x}_i$ až $\mathbf{x}_M$ proveď:
      i. Inkrementuj čítač n_k pro tento střed $\boldsymbol{\mu}_k$.
      ii. Urči learning rate jako $\eta = \frac{1}{n_k}$.
      iii. Aktualizuj střed tohoto shluku podle $\boldsymbol{\mu}_k = (1 - \eta)\boldsymbol{\mu}_k + \eta\mathbf{x}_i$.

Mini-batch metoda má menší stochastický šum než SGD z jednotlivých vzorků, což umožňuje konvergenci do lepšího řešení, ale nevede k zvýšení výpočetní náročnosti. Dle experimentů v [19] mini-batch k-means konvergovala řádově rychleji než klasická EM implementace k-means algoritmu.

2.3.2 DBSCAN

Shlukovací algoritmus DBSCAN byl použit v implementaci hledání mezer mezi vagóny (sekce 3.5.3) v rámci algoritmu dělení souprav na vagóny, proto bude krátce představen.

DBSCAN definuje shluky jako oblasti příznakového prostoru s vysokou hustotou vzorků. Pro každý vzorek je spočítán počet vzorků, které leží ve vzdálenosti menší než ϵ. Této oblasti se říká ϵ-okolí. Pokud má vzorek nejméně n vzorků ve svém ϵ-okolí, je považován za tzv. *jádrovou instanci*. Jinak řečeno, jádrové instance leží v oblastech s vysokou hustotou vzorků. Všechny vzorky v ϵ-okolí jádrové instance patří do stejného shluku, což může zahrnovat i jiné jádrové instance, takže sekvence sousedících jádrových instancí tvoří jeden shluk. Každý vzorek, který není jádrovou instancí a nemá ji ani ve svém okolí, je považován za anomálii [12].

2.4 Odhad přesnosti modelu

No Free Lunch Theorem říká, že neexistuje ideální řešení pro klasifikační problém. Žádný klasifikátor negarantuje, že bude dosahovat nejlepších výsledků pro každý problém, na který tento model nasadíme [2]. Proto je nutné dodržet metodologii odhadu přesnosti modelu a na jejím základě vybrat nejvhodnější model a ověřit úspěšnost na nových datech.

Jak zjistit, zda naučený model dobře generalizuje na nových vzorcích, které budou získány v budoucnosti? Jediný způsob je nové vzorky modelu opravdu předložit. Samozřejmě není možné získat vzorky z budoucnosti, tuto situaci lze však simulovat rozdělením množiny vzorků na trénovací set a testovací set, který představuje vzorky z budoucnosti. Tomuto způsobu se říká metoda *hold out* [11].

2.4.1 Metoda hold out

Tato procedura rozdělí dataset na trénovací množinu, která se použije pro učení modelu, a testovací množinu, na které určíme úspěšnost modelu. Takto můžeme natrénovat několik modelů na trénovací množině a vybrat ten, který má nejvyšší úspěšnost. Obecně čím obsáhlejší je původní dataset, tím větší část může být prohlášena za testovací set a opačně. V praxi obsahuje testovací dataset 1/10 až 1/3 původního datasetu [11]. Pokud je chyba modelu na trénovací množině malá, ale na testovacích vzorcích velká, model správně negeneralizuje a došlo k tzv. *over-fittingu*. [12].

Nyní si představme, že se rozhodujeme mezi několika různými modely nebo nastavením hyperparametrů určitého modelu (např. počet vrstev a aktivační funkce neuronové sítě). Vybereme nastavení hyperparametru s nejmenší chybou na testovací množině. Navržený model zavedeme do aplikace a zjistíme, že je chyba na nových datech neočekávaně větší, než evaluace modelu na testovací množině ukázala. Problémem je, že byl hyperparametr nastaven tak, aby se model co nejlépe adaptoval na testovací dataset, což může způsobit nevalnou úspěšnost modelu na nových datech [12][17]. Častým řešením je tak další oddělení tzv. validačního datasetu z trénovací množiny. Máme tedy dataset trénovací, validační a testovací, v praxi často v poměru $0,6/0,2/0,2$. Několik modelů (různé druhy, odlišné nastavení hyperparametrů) je pak naučeno na trénovací množině a je vybrán model s nejlepším výsledkem na validačním datasetu. Vybraný model je pak natrénován na množině trénovací + validační a podroben testovacímu datasetu. Chyba modelu na testovacím datasetu je pak nezaujatým (unbiased) odhadem chyby na reálných nových datech [12]. Pozor, po evaluaci modelu na testovacím datasetu již nesmíme dále upravovat parametry modelu pro lepší výsledek vzhledem k testovací množině! [2] Došlo by k over-fittingu modelu na testovacích datech a na nových vzorcích by pak model vykazoval větší chybu, než bylo předpokládáno.

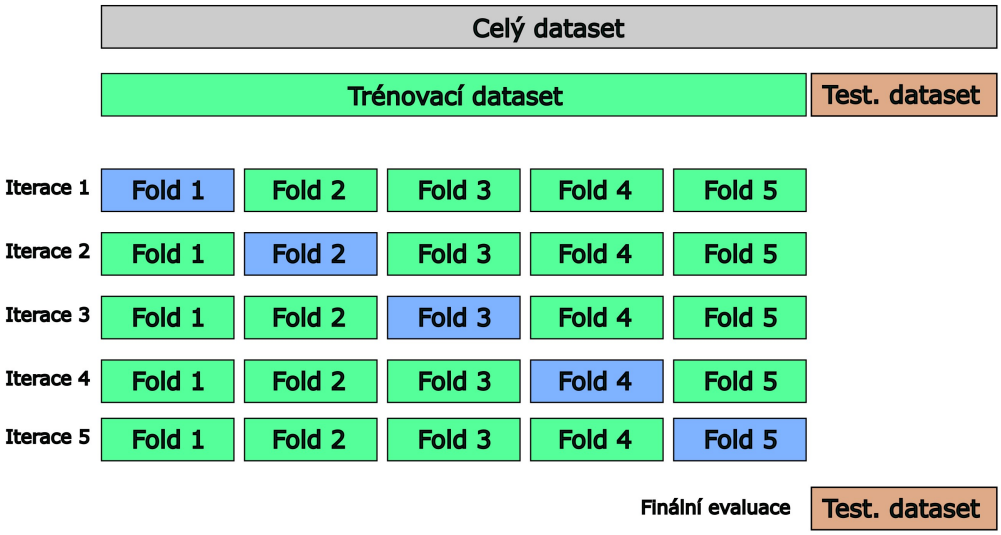
Celý postup je shrnut do následujících bodů:

(1) Rozdělení datasetu na trénovací, validační a testovací množinu
(2) Vyběr architektury modelu a jeho parametrů
(3) Naučení modelu na trénovací množině

(4) Evaluace modelu na validační množině
(5) Opakování kroků (2)-(4) s různými modely a nastavením parametrů
(6) Výběr nejlepšího modelu a naučení na datech z trénovací a validační množiny
(7) Verifikace kvality modelu na testovací množině

(Pozn. Tento postup předpokládá použití metody hold out. Je-li je použita metoda cross-validace nebo bootstrap, jsou kroky (3) a (4) opakovány pro každý z k foldů. Tyto metody jsou vysvětleny v následující sekci 2.4.2, resp. 2.4.3.) Pokud chyba na testovací množině nikterak neodpovídá chybě validační, je nutné se vrátit na začátek a navrhnout jiný model. Jelikož validační set vznikl oddělením části trénovacího datasetu, může se stát, že je validační dataset příliš malý a odhad chyby bude nepřesný, nebo naopak příliš velký, pak bude trénovací sada nereprezentativní. Řešením je metoda *cross-validace* [12]. I při cross-validaci by měla být z původního datasetu oddělena testovací množina pro finální evaluaci modelu. [17]

2.4.2 k-fold cross-validace



Obr. 2.10: Schéma metody 5-fold cross-validace [17], upraveno

Metodou k-fold cross-validace rozdělíme trénovací množinu na k disjunktních množin. Sjednocením $k-1$ množin je vytvořen trénovací dataset a zbylá množina (fold), na obr. 2.10 znázorněna modře, slouží jako validační. k-fold cross-validace opakuje tento krok k-krát, v každé instanci je použita odlišná množina jako validační a $k-1$ množin jako trénovací. Odhadem přesnosti nebo chyby modelu je

pak průměrná hodnota přesnosti, respektive chyby modelu, vypočítaná z k naučených instancí modelu. Odhad přesnosti/chyby modelu je tak robustnější než u metody hold out. Tato výhoda je však za cenu zhruba k-krát větší výpočetní náročnosti. Běžně se hodnoty k pohybují mezi $k = 3...10$ [11]. Speciálním případem je *leave-one-out* cross-validace, kde k je rovno počtu vzorků. V každé instanci je pak validační množinou pouze jeden vzorek. Tato variace ma význam u velmi malých datasetů. Pro nevyvážené datasety, kde se rozdělení četnosti tříd v datasetu neblíží rovnoměrnému,tj. četnosti vzorků jednotlivých tříd nejsou zhruba stejné, je vhodné provést sestavení k foldů pomocí tzv. *stratifikovaného výběru*. Ten zajistí poměrově stejné zastoupení všech tříd v každém foldu jako u původního datasetu [17].

2.4.3 Bootstrap

Alternativou k metodě cross-validace je tzv. *Bootstrap*, vhodný pro datasety s nedostatkem vzorků. V této metodě je vybráno N vzorků s opakováním z datasetu o velikosti N. Výběr s opakováním znamená, že vzorek již vybraný se vrací do původní množiny a může být vybrát opakovaně. Nevybrané vzorky tvoří validační množinu. Takto se postup opakuje, čímž vznikne k trénovacích souborů (foldů). Původní dataset je použit jako testovací množina. Pravděpodobnost výběru vzorku je $1/N$, tedy pravděpodobnost, že prvek nebude vybrán je $1 - 1/N$. Pokud jsou generovány soubory o N vzorcích výběrem s opakováním, pak pravděpodobnost, že určitý vzorek nebude vybrán po N výběrech je

$$(1 - 1/N)^N \approx e^{-1} = 0.368 \tag{2.57}$$

To znamená, že trénovací data obsahují $\sim 63.2\%$ vzorků [2].

2.4.4 Metriky pro evaluaci klasifikátoru

Existuje řada metrik pro kvantifikaci klasifikace, především pro binární problémy. Čtyřpolní tabulka nebo matice záměn, v zahraniční literatuře *confusion metrix*, je tabulka ilustrující jak dobře klasifikátor predikuje klasifikaci do tříd. Většinou je tabulka organizována tak, že řádky udávají skutečnou klasifikaci, tzv. *ground truth*, a sloupce predikované třídy. Pro binární problém mějme třídy pozitivní a negativní. Pak budou pole tabulky označeny jako *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) a *False Negative* (FN). Z hodnot v tabulce lze odvodit řadu metrik, které mohou být užitečné dle charakteru problému.

Precision udává, jaký poměr z predikovaných pozitivních případů je skutečně pozitivních. *Recall* pak říká, jaký poměr skutečně pozitivních případů bylo predikováno jako pozitivní. Mezi metrikami precision a recall vzniká kompromis, pokud se

		Predikce		
		Positive	Negative	celkem
Skutečnost	Positive	TP	FN	P
	Negative	FP	TN	N
	celkem	P'	N'	M

Tab. 2.1: Čtyřpolní tabulka pro binární klasifikaci

přenastavením prahu klasifikátoru zvětší hodnota precision, klesne hodnota recall. Správná volba kompromisu závisí na povaze problému. Např. pokud chceme navrhnout systém, který zve pacienty na preventivní onkologické vyšetření v závislosti na predispozici pacienta k rakovině, označíme pozitivní případ jako "pacient má podezření na rakovinu". Pak nejpíše zvolíme vysokou hodnotu recall, aby byli všichni pacienti s podezřením na rakovinu pozváni i za cenu toho, že mnoho z nich rakovinu skutečně nemá. Naopak chceme-li predikovat, které filmy z databáze jsou vhodné pro děti, zvolíme vysokou hodnotu precision. Do kategorie pro děti se tak zařadí pouze filmy skutečně vhodné. Kompromisem je, že se některé filmy skutečně vhodné pro děti klasifikují jako nevhodné. Čtyřpolní tabulka pro multi-class úlohy je matice typu $k \times k$, kde k je počet tříd. Řádky a sloupce pak ukazují četnosti skutečné, respektive predikované klasifikace všech vzorků. Čtyřpolní tabulka pro multi-class úlohy již nemá pouze čtyři pole, lépe ji tak vystihuje termín matice záměn nebo confusion matrix v anglicky psané literatuře. V ideálním případě, při bezchybné klasifikaci, by byly všechny prvky této matice mimo hlavní diagonálu nulové.

Metrika	Vztah
Správnost (Accuracy)	(TP+TN)/M
Chyba (Error)	(FP+FN)/M [=1-Správnost]
Precision	TP/P'
Recall	TP/P

Tab. 2.2: Metriky pro evaluaci binárního klasifikátoru

Pro multi-class klasifikaci nemá mnoho metrik přejatých z binárního případu pro většinu aplikací takový význam. Celkovou správnost klasifikace lze spočítat poměrem všech správně klasifikovaných vzorků k počtu všech vzorků. Správnost je užitečná metrika, pokud je chyba v predikci pro všechny třídy stejně významná [13]. Článek [20] uvádí metriky celkové klasifikace odvozené od binárních metrik pro každou třídu C_i dvěma způsoby: Metrika je průměr stejné metriky pro kaž-

dou třídu $C_1, ..., C_l$ (*macro-averaging*) nebo je vypočítána z kumulativních počtů TP,FN,TN,FP (*micro-averaging*). Macro-averaging dává stejnou váhu všem třídám, kdežto micro-averaging je ovlivněn více většími třídami [20]. Metriky pro multi-class klasifikaci jsou uvedeny v přehledové tabulce 2.3, micro-averaging je značen indexem μ, macro-averaging indexem M. V [21] je zavedena jednodušší notace: Matice C^k je

		Predikce			
		Třída 1	Třída 2	Třída 3	celkem
Skutečnost	Třída 1	c_{11}	c_{12}	c_{13}	$c_{1.}$
	Třída 2	c_{21}	c_{22}	c_{23}	$c_{2.}$
	Třída 2	c_{31}	c_{32}	c_{33}	$c_{1.}$
	celkem	$c_{.1}$	$c_{.2}$	$c_{.3}$	$c_{..}$

Tab. 2.3: Obecná matice záměn C^3 pro multiclass klasifikaci [22]

$k \times k$ maticí záměn a c_{ij} jsou prvky matice, kde $i, j = 1, 2, ..., k$. Řádky a sloupce pak ukazují četnosti skutečné, respektive predikované klasifikace. Počet vzorků skutečně patřících do třídy i je dán součtem hodnot v příslušném řádku:

$$c_{i.} = \sum_{j=1}^{k} c_{ij} \tag{2.58}$$

Podobně součet ve sloupci udává počet vzorků predikovaných jako třída j:

$$c_{.j} = \sum_{i=1}^{k} c_{ij} \tag{2.59}$$

Celkový počet vzorků je pak součet všech prvků v matici:

$$c_{..} = \sum_{i=1}^{k} \sum_{j=1}^{k} c_{ij} \tag{2.60}$$

Pak můžeme odvodit alternativní vztahy pro metriky v tab. 2.4. Správnost udává poměr správně klasifikovaných vzorků, F-score je harmonický průměr metrik precision a recall. CBA (*Class balance accuracy*) udává správnost s přihlédnutím k nevyváženosti datasetu. Autor v [22] uvádí další metriky jako *Matthew's Correlation Coeficient*, *Relative Classifier Information RCI*, *Confusion Entropy* a *G-Mean*.

Metrika	Vztah
Správnost (Accuracy)	$\frac{\sum_{i=1}^{k} \frac{TP_i+TN_i}{TP_i+FN_i+FP_i+TN_i}}{k} = \frac{\sum_i c_{ii}}{c_{..}}$
Chyba (Error)	$\frac{\sum_{i=1}^{k} \frac{FP_i+FN_i}{TP_i+FN_i+FP_i+TN_i}}{k}$
Precision_M	$\frac{\sum_{i=1}^{k} \frac{TP_i}{TP_i+FP_i}}{k} = \frac{1}{k}\sum_{i=1}^{k} \frac{c_{ii}}{c_{.i}}$
Recall_M	$\frac{\sum_{i=1}^{k} \frac{TP_i}{TP_i+FN_i}}{k} = \frac{1}{k}\sum_{i=1}^{k} \frac{c_{ii}}{c_{i.}}$
F-score_M	$\frac{1}{k}\sum_{i=1}^{k} \frac{2\frac{c_{ii}}{c_{.i}}\frac{c_{ii}}{c_{i.}}}{\frac{c_{ii}}{c_{.i}}+\frac{c_{ii}}{c_{i.}}}$
CBA_M	$\frac{1}{k}\sum_{i=1}^{k} \frac{c_{ii}}{\max(c_{.i}c_{i.})}$
Precision_μ	$\frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} TP_i+FP_i}$
Recall_μ	$\frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} TP_i+FN_i}$

Tab. 2.4: Metriky pro evaluaci multi-class klasifikátoru [20][21]

3 Praktická realizace

Kapitola se věnuje analýze poskytnuté databáze a popisuje praktickou realizaci systému klasifikace kolejových vozidel. Návrh systému čerpá z teoretické rešerše shrnuté do kapitol 1 a 2.

3.1 Dataset kolejových vozidel

3.1.1 Akvizice snímků

Obrazová databáze průjezdů vlaků obsahuje kolejová vozidla zachycena z bočního pohledu. Kolejová vozidla jsou snímána řádkovou kamerou s přísvitem v reálné lokalitě na železničním koridoru. Řádková kamera má pouze 1 řádek světlo-citlivých pixelů, který dokáže snímat s vysokou frekvencí. Ze znalosti rychlosti vlaku a nasnímaných řádků lze sestavit obraz projíždějícího vozidla. V nynější podobě systému, která byla naznačena v úvodu práce, je kamera spouštěna (trigger) ze signálů sady indukčních snímačů, které detekovaly přítomnost soupravy. Z časových značek signálů snímačů je rovněž vypočítána rychlost soupravy. V hypotetickém systému, jehož možností realizace se tato práce zabývá, by byla rychlost vlaku pro spouštění kamery měřena radarem, který rovněž nemusí být umístěný přímo v kolejišti. Výsledkem akvizice snímků je šedotónový obraz projíždějícího kolejového vozidla ve velmi vysokém rozlišení, typicky 6000×1283 až 9900×1283 pixelů na jedno kolejové vozidlo. Celá kolejová souprava tak může mít rozlišení až 1000 MPix. Dataset obsahuje snímky velkého množství typů kolejových vozidel za různých světelných podmínek (denní i noční scény), jak je patrné na obr. 3.1 a 3.2. Tato databáze byla pro práci poskytnuta a hlubšímu porozumění procesu akvizice těchto obrazových dat nebyla věnována pozornost.

3.1.2 Analýza pořízených snímků

Po nahlédnutí do databáze je patrné, že některé třídy vozidel trpí nízkou mírou podobnosti v rámci dané třídy. Např. novější typy lokomotiv jsou často vybaveny velkoplošným polepem reklamního charakteru nebo logem a firemními barvami provozovatele (obr. 3.1). Starší typy lokomotiv se mohou vizuálně velmi lišit, pokud došlo k jejich renovaci. Některé vozy jsou z velké části posprejovány nápisy vandalů. Naopak určité typy vozidel, především osobní vozy, jsou pro lajka pouhým pohledem téměř k nerozeznání (vysoká míra podobnosti mezi některými třídami). Oba fakty jsou komplikací pro klasifikaci, v ideálním případě si přejeme vysokou míru podobnosti v rámci třídy a co nejmenší míru podobnosti mezi třídami.

Některé snímky, pořízené zejména za tmy, jsou příliš podexponované, aby v sobě obsahovaly dostatečné množství informace. V daleko menší míře se také objevují přeexponované snímky za denního osvětlení. Všechny tyto negativní vlastnosti datasetu mohou ovlivnit úspěšnost klasifikace. Dalším problémem je nejasná definice jednotlivých tříd osobních vozů, v řadě případů se písmenné označení vozů liší, ale po vizuální stránce jsou vozy téměř nerozeznatelné. Např. třídy „Ampz“ a „Bmpz“ označují vozy 1., respektive 2. třídy nebo třída „Bpee“ má na rozdíl od třídy „Bee“ sedadla se střední uličkou, což ze snímku nelze rozeznat. Třídy byly tedy definovány dle úsudku a v případě potřeby je možné třídy předefinovat. U snímků lokomotiv je rozřazení triviální. Následující sekce popisují navržený systém klasifikace a tento systém je laděn na dvou datasetech, které byly sestaveny z dodaných snímků vedoucím práce. Redukovaný dataset obsahuje 1733 snímků rozdělených do 15 tříd s rovnoměrnějším rozložením četností. Plný dataset obsahuje 2217 snímků rozdělených do 27 tříd, kde některé třídy jsou zastoupeny v menších počtech jednoduše proto, že se takových exemplářů mezi dodanými snímky vyskytovalo málo. Oba datasety jsou detailněji popsány v tabulce 3.1, respektive 3.2.

Obr. 3.1: Ukázka snímků lokomotiv dvou typů

třída	230	242	362	380	386	Bdmtee	Bdpee	Bmz	ES64U4
četnost	124	129	111	81	104	141	67	92	87
rel. četnost [%]	7,15	7,44	6,40	4,67	6,00	8,14	3,87	5,31	5,02

třída	Panter_f	Panter_m	Railjet	Regiojet1	Regiojet2	Vectron
četnost	141	135	149	145	65	162
rel. četnost [%]	8,14	7,79	8,60	8,37	3,75	9,35

Tab. 3.1: Redukovaný dataset kolejových vozidel pro ladění hyperparametrů (15 tříd)

Obr. 3.2: Ukázka snímků osobních vozů typů Ampz a Bmpz

třída	230	242	362	380	386	Bdmtee	Bdpee	Bmz	ES64U4
četnost	124	129	111	81	104	141	67	92	87
rel. četnost [%]	5,59	5,82	5,01	3,65	4,69	6,36	3,02	4,15	3,92

třída	Panter_f	Panter_m	Railjet	Regiojet1	Regiojet2	Vectron
četnost	141	135	149	145	65	162
rel. četnost [%]	6,36	6,09	6,72	6,54	2,93	7,31

třída	753	842	Afmpz	Ampz	Ampz2	ARbmpz	Bee	Bmpz
četnost	14	15	73	42	19	69	36	56
rel. četnost [%]	0,63	0,68	3,29	1,89	0,86	3,11	1,62	2,53

třída	Bfhpvee	Bpmz	Regiojet_okna	350
četnost	36	39	63	22
rel. četnost [%]	1,62	1,76	2,84	0,99

Tab. 3.2: Kompletní dataset kolejových vozidel (27 tříd)

3.2 Návržený systém klasifikace metodou BoVW

3.2.1 Návrh frameworku pro evaluaci a testování

Po načtení snímků z jednotlivých složek odpovídajících třídám je oddělena testovací množina stratifikovaným výběrem. Zbytek dat je v rámci cross-validace, popsané v sekci 2.4.2, rozdělen na 5 foldů opět stratifikovaným výběrem. V každé iteraci cross-validace jsou čtyři foldy určeny jako trénovací a jeden jako validační. Trénovací snímky jsou před extrakcí významných bodů předzpracovány. SIFT detektor+descriptor extrahuje vektory příznaků, ty jsou shlukovány do vizuálních slov a je vytvořen slovník podle sekce 1.6, popisující metodu bag of visual words. Každý obraz z trénovacího datasetu je pak popsán jako histogram vizuálních slov. Tyto histogramy představují vektory příznaků a společně s hodnotou reprezentující třídu vzorku (snímku) jsou vstupem klasifikátorů. Práce se zabývá klasifikátory typu kNN, SVM, Multinomial Naive Bayes, neuronová síť a ensemble metoda typu voting classifiers. Všechny tyto klasifikátory jsou detailně popsány v sekci věnující se algoritmům typu učení s učitelem (2.2).

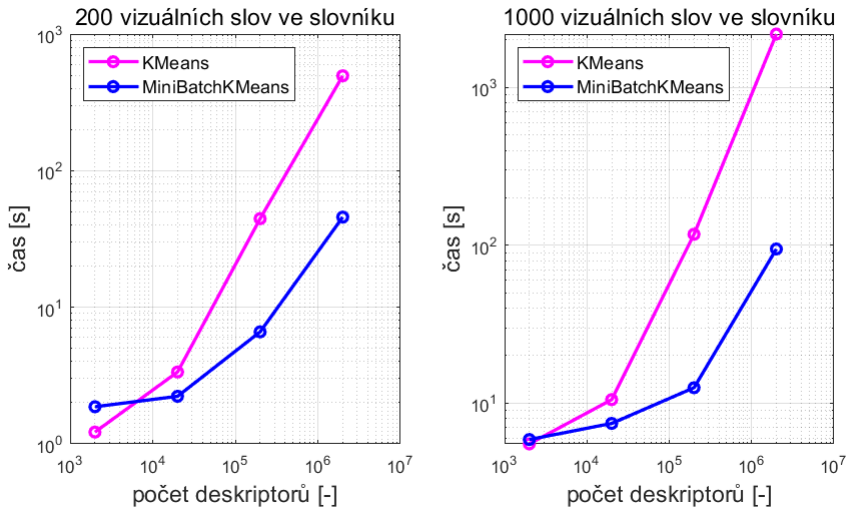
Druhým krokem je evaluace přesnosti klasifikace na validační množině. Z validačních snímků jsou extrahovány příznaky SIFT detektorem a každý je nahrazen vizuálním slovem ze slovníku. Následně je sestaven histogram slov pro každý snímek. Histogramy jsou předloženy natrénovaným klasifikátorům, které predikují třídu vzorku. Ze sestavené matice záměn (2.4.4) pro každý klasifikátor je vypočtena celková správnost na validační množině. Celý proces se opakuje v rámci cross-validace tak, aby byl každý z foldů použit jako validační dataset. Odhad správnosti jednotlivých klasifikátorů je pak dán jako aritmetický průměr správnostní všech pěti instancí cross-validace.

3.2.2 Předzpracování obrazu

Vzhledem k vysokému rozlišení pořízených snímků je extrakce významných bodů SIFT detektorem časově náročná operace. Vysoké rozlišení dovoluje škálovat obraz na menší velikost (downsampling) bez znatelné ztráty informace. Mapování pixelů probíhá z nových souřadnic do původního souřadnicového systému. Po mapování nemusí souřadnice být celočíselné, hodnota nového pixelu je tedy interpolována bilineární interpolací. Původní velké rozměry obrazu sice umožňují extrahovat větší počet významných bodů, většina jich je však pro BoVW model neužitečná. Dalším předzpracováním obrazu, jehož vliv je testován, je adaptivní ekvalizace histogramu metodou CLAHE (sekce 1.2.1). Vizuální kontrolou je patrné, že ekvalizace „vylepšuje“ snímky především v tmavých oblastech, jako jsou např. podvozky vozů, kde je možné extrahovat více detailů.

3.2.3 Rychlost vzniku vizuálního slovníku

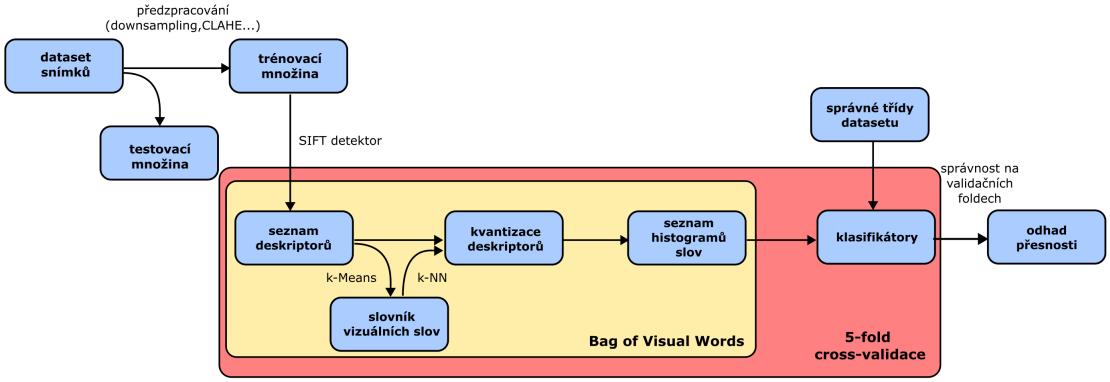
Při iterativním ladění hyperparametrů metody byla odhalena důležitost rychlosti běhu programu. Vlastní implementace metody BoVW ukázala, že největší překážkou, co se rychlosti běhu programu týče, je vznik vizuálního slovníku, popsaný v sekci 1.6.1. Tato část řetězce metody BoVW výrazně omezovala z časových důvodů možnosti testování, zejména kvůli cross-validaci, kdy slovník vzniká pro každý fold. Principem vzniku vizuálního slovníku je algoritmus K-means(2.3.1) Uspokojivým řešením je nahradit klasický algoritmus K-means za Mini-batch variantu (2.3.1) tohoto algoritmu. Mini-batch varianta dle [17] dosahuje lehce horší výsledky shlukování, pro účel vzniku vizuálního slovníku však dosahuje výsledků, které se negativně neprojevují na konečné kvalitě klasifikace snímků. Tento fakt byl ověřen na datasetu. Zrychlení vzniku vizuálního slovníku z deskriptorů je značné, jak ukazuje obr. 3.3. Při narůstající velikosti datasetu, a tedy zvyšujícím se počtu deskriptorů, prudce roste výpočetní náročnost vzniku slovníku klasickým algoritmem K-means. Pro 200 tisíc deskriptorů, což při určitém nastavení metody odpovídá extrakci příznaků z datasetu o velikosti 450 snímků, je doba konvergence mini-batch varianty o řád nižší. Obrázek 3.3 naznačuje dobu vzniku slovníku pro obě varianty K-means. Žádaný počet slov ve slovníku, který odpovídá počtu shluků metody k-means, dále ovlivňuje dobu sestavení slovníku.



Obr. 3.3: Doba sestavení vizuálního slovníku s 200 a 1000 slov

3.2.4 Ladění hyperparametrů klasifikátorů

Tato sekce se zabývá vlivem nastavení hyperparametrů jednotlivých klasifikátorů na kvalitu klasifikace. Zmenšený dataset podroben těmto testům obsahuje 1733 snímků



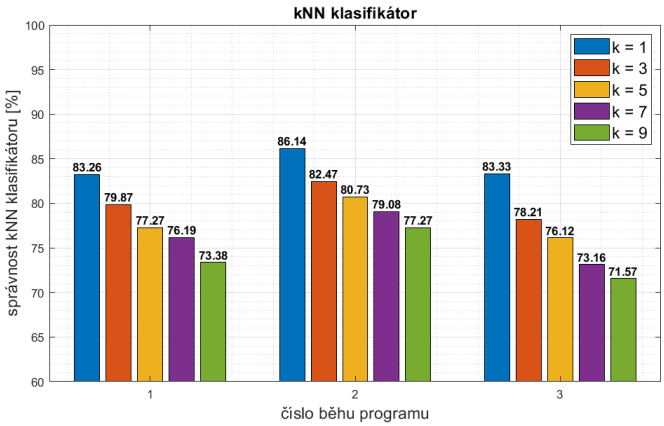
Obr. 3.4: Ilustrace pipeline pro ladění hyperparametrů klasifikátorů a metody BoVW

osobních vozů a lokomotiv rozdělených do 15 tříd. Detaily datasetu jsou uvedeny v tabulce 3.1. Z tohoto počtu je 20% (347 snímků) odděleno jako testovací dataset a zbylých 1386 snímků vstupuje do cross-validace. Jak je zmíněno v sekci 2.4.1, testovací dataset nebude v této fázi ladění hyperparametrů použit, slouží pro nezaujatý odhad chyby nejlépe vyladěných klasifikátorů.

Pro následující testy jsou některé parametry modelu zafixovány, aby bylo možné klasifikátory objektivně porovnávat. Velikost obrazu je zmenšena koeficientem 0,1 na šířku, respektive 0,33 na výšku. Takto škálované snímky mají velikost zhruba 750x430 pixelů, v závislosti na délce vozu. SIFT detektor extrahuje 500 nejvýznamnějších bodů a velikost vizuálního slovníku je nastavena na hodnotu 200. Všechny kvantifikované výsledky klasifikace jsou určeny 5-fold cross-validací, což je vhodný kompromis mezi kvalitou odhadu přesnosti modelů a časovou náročností výpočtu. Ilustrace pipeline pro ladění hyperparametrů klasifikátorů je na obr. 3.4.

kNN Pro kNN (sekce 2.2.1) je nejdůležitějším hyperparametrem počet nejbližších sousedů k. Obrázek 3.5 zobrazuje výsledky pro tři běhy programu. Je patrné, že klasifikace na základě jednoho nejbližšího souseda dosahuje v rámci cross-validace konstantně nejvyšší správnosti.

Linear SVM Dalším zkoumaným klasifikátorem je lineární SVM, implementovaný v knihovně *scikit-learn* buď pomocí třídy `LinearSVC` a chybovou funkcí *hinge* a *squared hinge* nebo `SVC` s lineárním kernelem. Histogramy vizuálních slov byly před vstupem do klasifikátorů normalizovány tak, že každá hodnota histogramu udává relativní četnost daného slova. Levý graf 3.6 ukazuje vliv hodnoty regularizačního parametru C a zvolené chybové funkce na kvalitu klasifikace. Význam parametru C je popsán v sekci 2.2.2.

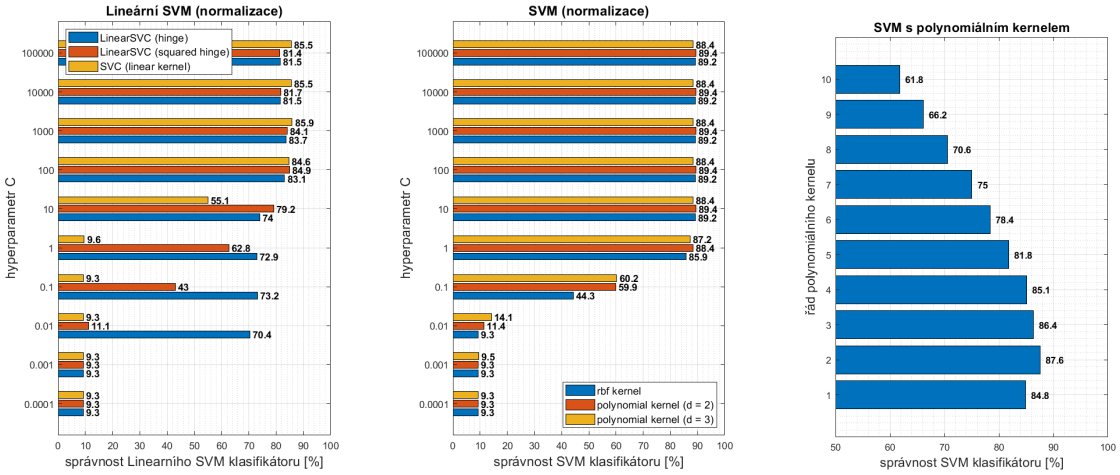


Obr. 3.5: Správnost klasifikace kNN klasifikátoru v závislosti na počtu sousedů k

SVM Dále je věnován prostor SVM klasifikátorům s polynomiálním a *Radial Basis Function* (RBF) kernelem. Na pravém grafu 3.6 je zobrazena závislost správnosti klasifikace na řádu polynomiálního kernelu. Z hodnot je patrné, že nejlepších výsledků dosahují kernely nižších řádů. Na prostředním grafu 3.6 je srovnán RBF kernel a polynomiální kernel druhého řádu pro různé hodnoty parametru C.

Neuronová síť Pro klasifikátor typu fully-connected neuronová síť byl převážně testován vliv počtu jednotek (neuronů) v jedné nebo dvou skrytých vrstvách. Autoři v [12] uvádí, že použití stejného počtu neuronů v každé vrstvě vede na stejně dobré výsledky jako klasické pyramidové uspořádání, kdy se počet neuronů ve vrstvách snižuje směrem od vstupu k výstupu. Navíc postačuje ladit jediný hyperparametr udávající počet neuronů ve všech vrstvách. Jako aktivační funkce všech jednotek byla zvolena ReLU (*Rectified Linear Unit*) s matematickým předpisem $h(a_j) = \max(0, a_j)$. Váhy byly optimalizovány algoritmem *Adam* - rychlejší alternativa k optimalizačnímu algoritmu stochastic gradient descent. (Popsáno v sekci 2.2.4.) Pro optimalizaci bylo zvoleno 20 epoch. Výstupní softmax aktivační funkce udává pravděpodobnost příslušnosti vzorku k jednotlivým třídám. Jako predikovaná třída je určena třída s nejvyšší pravděpodobností. Levá část obrázku 3.7 ukazuje závislost správnosti klasifikace na zvoleném počtu jednotek (perceptronů) v jediné skryté vrstvě a velikosti dávky. Pravá část porovnává neuronovou síť s dvěma vrstvami a neuronovou síť s jednou vrstvou. Všechny varianty sítě s jednou skrytou vrstvou dosahují lepších výsledků pro malé dávky. Z pravého grafu 3.7 je zřejmé, že přidaná skrytá vrstva nezlepšuje výsledek klasifikace.

Multinomial NB Dalším testovaným typem klasifikátoru je Multinomial Naive Bayes, teoreticky popsaný v sekci 2.2.3. Jediným parametrem je aditivní parametr α,



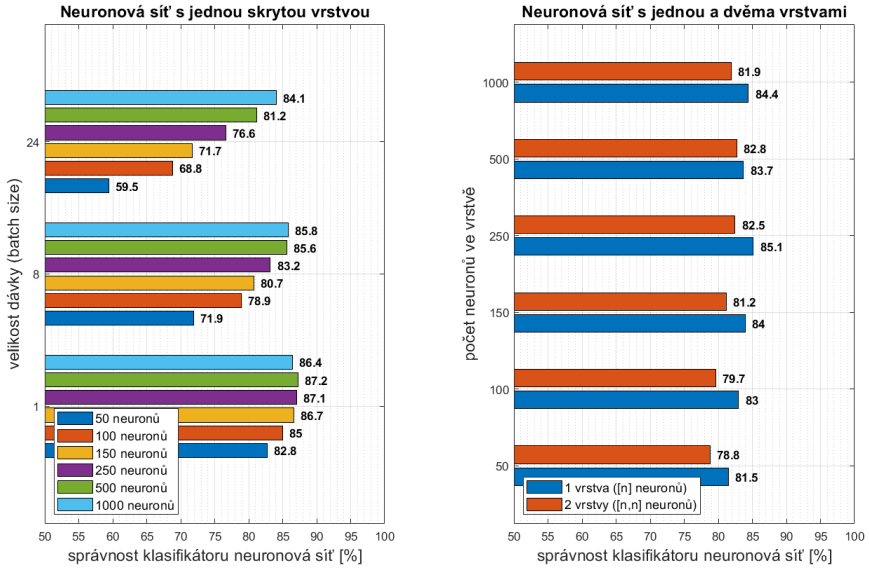
Obr. 3.6: Správnost klasifikace SVM klasifikátoru v závislosti na hyperparametru C a zvoleném kernelu (vlevo: lineární kernel, střed: polynomiální a rbf kernel, vpravo: polynomiální kernel)

který byl ponechán na hodnotě $\alpha = 1$, tak splňuje definici pro Laplacian smoothing z rovnice 2.22. NB klasifikátor tedy nebyl nijak laděn, odhad apriorní pravděpodobnosti třídy je dán relativní četností vzorků dané třídy v trénovacím datasetu (rovnice 2.20).

Ensemble learning V rámci testování využití metod Ensemble learningu (sekce 2.2.5) byla ověřena jednoduchá metoda voting classifiers. Pro klasifikátory typu kNN, lineární SVM, SVM s polynomiálním kernelem, Multinomial NB a neuronová síť byly agregovány predikce a zvolena ta s největším počtem výskytů. V dalších fázích byl klasifikátor Multinomial NB vyloučen z hlasování v metodě ensemble learning pro neuspokojivou úspěšnost klasifikace v porovnání s ostatními klasifikátory.

3.2.5 Ladění hyperparametrů BoVW metody

Pro ladění hyperparametrů BoVW metody byla použita pipeline z obr. 3.4. V cross-validaci je zahrnuta i metoda BoVW a v každém foldu cross-validace se tvoří nový slovník slov a sestavuje reprezentace obrazů v podobě histogramů vizuálních slov. Test je proveden na stejném datasetu jako pro ladění hyperparametrů klasifikátoru, je popsán na začátku sekce 3.2.4. V této sekci bude testováno nastavení velikosti vizuálního slovníku. Velikost slovníku mění velikost histogramu slov a tedy příznakového vektoru klasifikátorů. Pro testování vlivu velikosti slovníku jsou vybrány vyladěné klasifikátory různých typů, které dosahovaly nejvyšší správnosti na základě

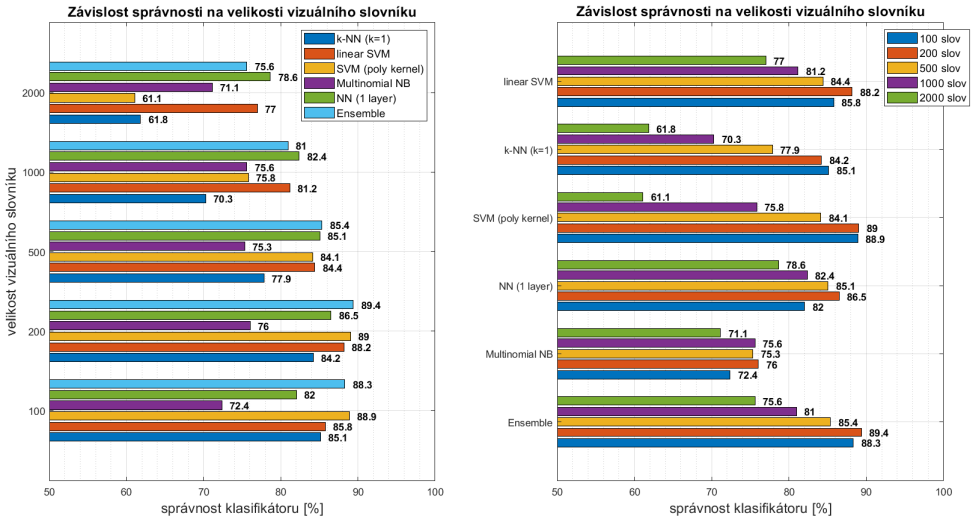


Obr. 3.7: Správnost klasifikace neuronové sítě v závislosti na počtu neuronů, vrstev a velikosti dávky

předchozích testů v sekci 3.2.4. Byly použity tyto klasifikátory;

1. k-NearestNeighbors (k=1)
2. LinearSVM (C=1000)
3. SVM (C=100, polynomiální kernel 2. řádu)
4. Multinomial Naive Bayes (Laplacian smoothing)
5. Neuronová síť (1 skrytá vrstva, 250 neuronů, ReLU aktivační funkce)
6. Ensemble learning - voting classifiers (kNN, LinearSVM, SVM, Neuronová síť)

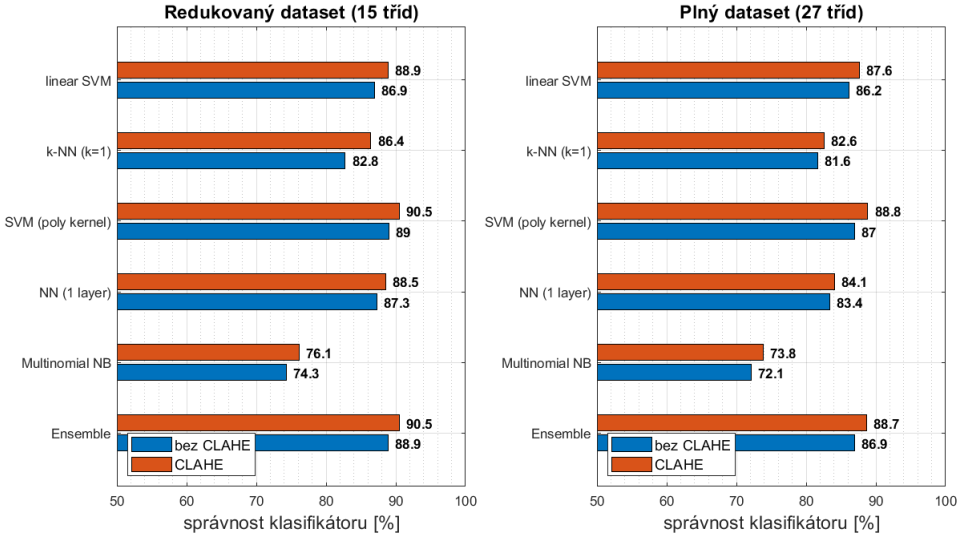
Velikost slovníku byla určena v rozmezí 100 až 2000. Na obou grafech 3.8 jsou zobrazena stejná data, pouze jinak interpretovány. Levý graf srovnává kvalitu klasifikace navržených klasifikátoru mezi sebou navzájem. Lze pozorovat, že Multinomial NB výrazněji zaostává. Pravý graf srovnává vliv velikosti vizuálního slovníku na výkon jednotlivých klasifikátorů. Je patrné, že klasifikátory jsou úspěšnější pro menší velikosti vizuálního slovníku a správnost se zvětšujícím se slovníkem klesá. Důvodem pravděpodobně je, že SIFT detektor extrahuje 500 významných bodů, které jsou mapovány na vizuální slova ve slovníku. Např. pro slovník o velikosti 2000 slov je extrahováno 500 významných bodů a jsou namapovány na slova ze slovníku. Pak je histogram výskytu vizuálních slov s 2000 příhrádkami, který popisuje obraz, řídkým vektorem (*sparse vector*) a není příliš vhodný jako vektor příznaků pro klasifikátory.



Obr. 3.8: Správnost klasifikátorů v závislosti na velikosti vizuálního slovníku

3.2.6 Vliv předzpracování snímků

Jak již bylo zmíněno, byl ověřen vliv předzpracování snímků, zejména pak ekvalizace histogramu algoritmem CLAHE. Motivací byla publikace [24], ve které je rozebrán pozitivní vliv předzpracování obrazu pomocí CLAHE pro extrakci významných bodů SIFT detektorem za různých světelných podmínek. Z grafu je patrné zlepšení přes-



Obr. 3.9: Vliv CLAHE na správnost klasifikace

nosti klasifikátorů, pro potvrzení hypotézy bude pro výsledky na plném datasetu

proveden párový t-test typu „$\mu_1 = \mu_2$“ [28]. Test nezahrnuje ensemble metodu, protože je pouhou kombinací ostatních klasifikátorů.

	acc_{i1} bez CLAHE	acc_{i2} s CLAHE	$acc_{i2} - acc_{i1}$
k-NN (k=1)	81,61	82,57	0,96
linear SVM	86,18	87,65	1,47
SVM (poly kernel)	86,97	88,83	1,86
Multinomial NB	72,14	73,83	1,69
Neural Net (1 layer)	83,36	84,09	0,73

Tab. 3.3: Vliv CLAHE na správnost klasifikátoru [%] (plný dataset)

- $H_0 : \mu_{acc_2} - \mu_{acc_1} = 0$; Rozdíl středních hodnot správnosti klasifikátorů bez CLAHE a s CLAHE je nulový a použití CLAHE nemá na klasifikaci vliv.
- $H_1 : \mu_{acc_2} - \mu_{acc_1} \neq 0$; Použití CLAHE má vliv na klasifikaci.

Z rozdílů $acc_{i2} - acc_{i1}$ vypočteme průměr $\overline{x} = 1,344$, odhad rozptylu pomocí výběrového rozptylu $\overline{s^2} = 0.231$ a $N = 5$. Testovací statistika T je rovna:

$$T = \frac{\overline{x} - 0}{\sqrt{\frac{\overline{s^2}}{N}}} = 6,249 \tag{3.1}$$

Kritická hodnota oboustraného t-testu pro $N - 1$ stupňů volnosti na hladině významnosti 0,05 je $t_{krit} = 2,571$. Protože je $T > t_{krit}$, zamítáme hypotézu H_0. Vliv CLAHE je potvrzen [28]. Stejného závěru bylo dosaženo i na datech z redukovaného datasetu. Test předpokládá normální rozdělení rozdílů $acc_{i2} - acc_{i1}$. Hypotéza na normální rozdělení rozdílů nebyla pomocí Shapiro-Wilkova testu zamítnuta, t-test je tak platný.

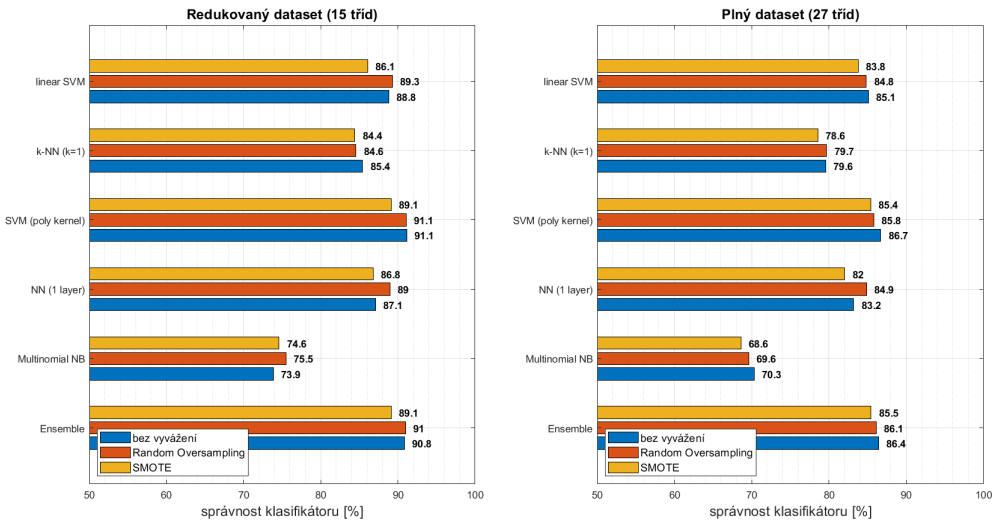
3.2.7 Metody pro vyvážení datasetu

Tato sekce se zabývá metodami, které umožňují vyvážení datasetu, tedy dosažení stejné četnosti vzorků v každé třídě. Přirozeně drtivá většina datasetů takovou vlastnost nemá, patří mezi ně i sestavený dataset kolejových vozidel. Přitom není jasně dána hranice nevyváženosti datasetu, která by jasně znamenala horší výkonnost klasifikátorů [22]. Problém nevyváženosti datasetu bude demostrován na příkladu. Mějme dataset o 100 vzorcích rozdělených do tříd následovně: 97 vzorků pozitivních, 2 vzorky negativní, 1 vzorek neutrální. Pak klasifikátor, který naivně klasifikuje všechny vzorky jako pozitivní, dosahuje správnosti 97%. Klasifikátor však nemá žádnou hodnotu pro zachycování vzorků neutrální a negativní třídy.

Jednoduchým řešením nevyvážených datasetů je vybalancování rozdělení četností ve třídách výběrem prvků. Tzv. *oversampling* zvýší zastoupení méně četné

třídy vytvořením kopie vybraných vzorků. *Undersampling* sníží zastoupení početné třídy vymazáním vybraných vzorků. Random undersampling může ztratit užitečnou informaci z vyloučených vzorků a random oversampling může způsobovat overfitting [22]. Metoda SMOTE, blíže popsána v sekci 2.2.8, generuje syntetické vzorky interpolací vzorků dané třídy v příznakovém prostoru, zde s výhodou využijeme vlastnosti použité metody BoVW, která transformuje snímky do příznakového prostoru v podobě histogramů četností vizuálních slov. Random Oversampling lze provádět před transformací snímků do příznakového prostoru nebo po transformaci, protože se jedná pouze o kopii vzorků a nezáleží na jejich reprezentaci.

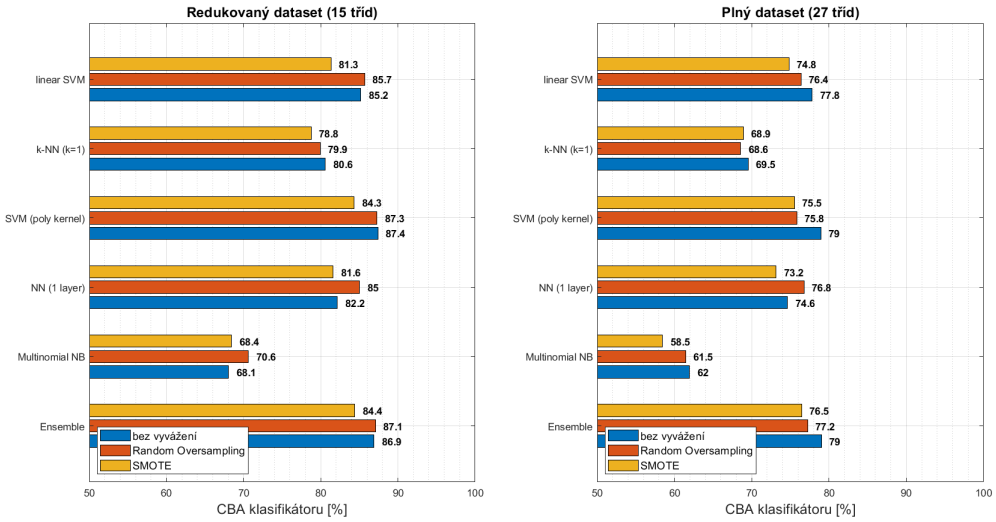
Cross-validací byly prověřeny metody vyvážení Random oversampling a SMOTE a porovnány s datasetem bez vyvážení. Obrázek 3.10 ukazuje vliv metod vyvážení datasetu pro redukovaný i plný dataset (tab. 3.1, respektive 3.2) na správnost klasifikace. Je patrné, že Random Oversampling má pozitivní účinky na správnost klasifikace u klasifikátoru typu neuronová síť, jinak metody vyvážení spíše zaostávají.



Obr. 3.10: Správnost klasifikace v závislosti na vyvážení datasetu

Jak bylo na začátku této sekce předvedeno na příkladu, nemá metrika „celková správnost“ vždy největší vypovídající hodnotu. Autor v [22] navrhuje metriku *Class Balance Accuracy* (CBA), jejíž vzorec výpočtu je uveden v tab. 2.4. CBA bere v potaz tři základní údaje z matice záměn pro každou třídu: počet správně klasifikovaných vzorků, počet vzorků predikovaných jako daná třída, a počet vzorků skutečně patřících do dané třídy. Matematický předpis ukazuje, že CBA pro každou třídu uvažuje metriky *Precision* a *Recall* (tab. 2.4) a zahrne do výpočtu tu, která udává nižší skóre. Tím pádem je CBA konzervativní reprezentací výkonnosti klasifikátorů a lépe

zohledňuje chyby klasifikace u málo zastoupených tříd. Z obrázku 3.11 je patrné, že metody Random Oversampling a SMOTE obecně nevedly ke zvýšení hodnoty metriky CBA kromě klasifikátoru typu neuronová síť s metodou Random Oversampling. Bylo však užitečné tento test provést, protože techniky vyvážení datasetu sice mohou snižovat celkovou správnost klasifikátorů, ale pokud by metrika CBA ukázala vyšší skóre v případě vyváženého datasetu, byla by to známka, že klasifikátory lépe klasifikují málo zastoupené třídy.



Obr. 3.11: Metrika CBA v závislosti na vyvážení datasetu

3.2.8 Trénování a inference

Po odladění všech hyperparametrů klasifikátorů a metody BoVW je možné „na ostro“ natrénovat klasifikační systém. Trénovací snímky jsou před extrakcí významných bodů podvzorkovány a předzpracovány algoritmem CLAHE. Z adresáře datasetu jsou extrahovány odpovídající *labels*, tedy správnou třídu pro každý snímek. SIFT detektor+descriptor extrahuje vektory příznaků, ty jsou shlukovány do vizuálních slov a je vytvořen slovník podle sekce 1.6. Každý obraz z trénovacího datasetu je pak popsán jako histogram vizuálních slov. Tyto histogramy představují vektory příznaků a společně s hodnotou reprezentující třídu snímku jsou vstupem klasifikátorů.

Při tzv. inferenci, tedy klasifikaci nových snímků, je použit vytvořený slovník a natrénované klasifikátory. Snímky určené ke klasifikaci jsou před extrakcí významných bodů předzpracovány. Extrahované vektory významných bodů jsou kvantovány

pomocí slovníku tzn. mapovány na nejbližší slovo. Každý snímek je pak reprezentován histogramem slov, který je klasifikován pomocí klasifikátorů. Výstupem je predikovaná třída pro každý snímek.

3.3 Navržený systém klasifikace metodou konvolučních sítí

Alternativním postupem klasifikace k popsané metodě „BoVW + klasifikátor“ je použití konvolučních neuronových sítí. Dosud popisovaná metoda představovala „klasický“ přístup - ruční návrh příznaků s následným využitím strojového učení. Konvoluční sítě se samy učí celý řetězec zpracování, od úrovně pixelů až po výstup.

3.3.1 Framework pro trénování CNN

V souladu s kapitolou 2.2.7 byly uvažovány pouze osvědčené architektury CNN za využití transfer learningu. Pro dotrénování konvolučních neuronových sítí byla zvolena cloudová platforma Google Colaboratory, která umožňuje využití běhu programu na GPU. Google Colaboratory je služba, která poskytuje možnost tvorby a běhu python kódu přímo v prohlížeči. Platforma je zejména vhodná pro strojové učení, analýzu dat a akademické účely. Kód je spouštěn na virtuálním počítači s přiřazenými zdroji včetně možnosti GPU. V bezplatné verzi jsou tyto zdroje omezené a fluktuují v čase. Výhodou jsou předinstalované knihovny pro strojové učení a analýzu dat jako je *TensorFlow, Keras, ScikitLearn, OpenCV, numpy*, atd. Všechny zdrojové kódy a data jsou dostupné z osobního Google Drive po připojení disku do prostředí Google Colaboratory.

Pro ověření kvality predikce pomocí konvolučních neuronových sítí byly zvoleny sítě AlexNet, VGG16 a ResNet50, popsané v sekci 2.2.7. Tyto sítě byly vybrány, protože mapují pokrok v oblasti klasifikace pomocí CNN (každá následující předčila výkonností tu poslední) a liší se základní filozofií návrhu. Při tréninku sítí se omezení zdrojů platformy Google Colaboratory projevilo v podobě nedostatečné operační paměti po načtení datasetu a zahájení trénovacího cyklu. Z tohoto důvodu bylo přistoupeno k dávkové koncepci, která je škálovatelná na jakkoliv obsáhlý dataset. Dataset je rozdělen na trénovací a validační+testovací množinu a načítán v dávkách definované velikosti. Šedotónové snímky jsou zmenšeny (podvzorkovány) na rozměry vhodné pro zpracování konvoluční neuronovou sítí (224x224) a jasové úrovně snímku jsou duplikovány do tří kanálů, protože CNN zpracovávají výhradně barevné obrazy s kanály RGB. Z jasových hodnot každého snímku je také odečtena střední hodnota jasu, předem vypočítána z celého datasetu. Toto předzpracování

je v souladu s publikovanými články všech tří sítí ([30],[31],[32]). Pro augmentaci a předzpracování dat je možné uplatnit *preprocessing layers* z knihovny TensorFlow, které se buď vloží přímo do modelu sítě, nebo se aplikují na dataset před vstupem do modelu sítě. Protože se implementované vrstvy pro augmentaci pro tuto aplikaci příliš nehodily, byly vytvořeny vlastní vrstvy, úpravou zdrojových kódů vrstev již implementovaných v TensorFlow.

Pro trénování sítí byl využit princip transfer learningu (2.2.6) a všechny sítě byly inicializovány s váhami, které byly předtrénovány na datasetu ImageNet s 1,2 milionem obrázků. Následně jsou konvoluční vrstvy zmrazeny, čímž jsou hodnoty vah (konvolučních filtrů) zafixovány. Horní FC vrstvy se optimalizují na zvoleném datasetu kolejových vozidel (obr. 2.6).

AlexNet není dostupná v knihovně Keras, proto byla použita předtrénovaná síť dostupná z [34] a dotrénována dle předchozího odstavce. Síť VGG16 je dostupná v knihovně Keras jako „black box“, ale pro lepší představu o architektuře byla implementována ručně vrstva po vrstvě. Předtrénované váhy na datasetu ImageNet pro konvoluční vrstvy byly získány z [33]. Síť ResNet50 byla použita z knihovny Keras jako „black box“ s předtrénováním na datasetu ImageNet.

3.3.2 Redukce FC vrstev

Protože jsou tyto architektury navržené pro klasifikaci datasetu ImageNet obsahující až 1000 tříd, nabízí se otázka, zda není možné zredukovat počty neuronů v FC vrstvách, když i výstupní softmax vrstva sítě pro klasifikaci kolejových vozidel bude mít daleko méně výstupů. Důsledkem by byl daleko menší počet vah (parametrů), které se budou optimalizovat na vlastním datasetu kolejových vozidel. Tabulky 3.4 ukazují počty trénovatelných vah v FC vrstvách a zafixovaných vah v konvolučních vrstvách. Horní tabulka odpovídá původním architekturám, kde FC vrstvy mají 4096 neuronů. Dolní tabulka zobrazuje počty vah v sítích při redukovaných FC vrstvách s 1024 neurony. Lze pozorovat výrazné zmenšení počtu parametrů při redukci počtu neuronů v FC vrstvách. To je i známka toho, že v CNN obsahují FC vrstvy většinu parametrů celé sítě. Počty zafixovaných vah se mezi tabulkami samozřejmě nemění, protože odpovídají konvoluční části sítí, která je stejná v obou případech. V síti ResNet50 se počty parametrů nemění, protože tato architektura neobsahuje horní FC vrstvy, ale pouze bezparametrovou global average pooling vrstvu a softmax vrstvu.

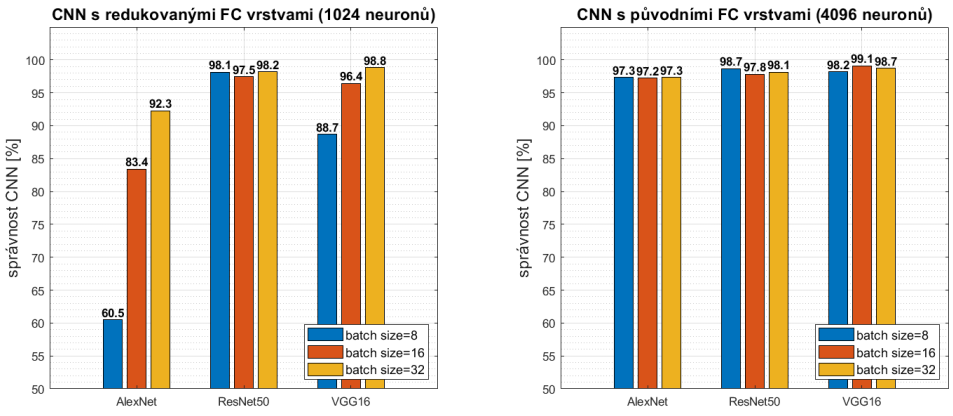
Pro architektury AlexNet a VGG16 byl ověřen vliv redukce počtu neuronů v FC vrstvách a velikost dávky pro optimální trénování. Architektura ResNet50, jak bylo zmíněno, není ovlivněna redukcí počtu neuronů, protože nemá FC vrstvy. Testy byly provedeny na plném datasetu (3.2) s 27 třídami a pro každé nastavení testu byly

CNN (FC s 4096 neurony)	AlexNet	VGG16	ResNet50
# trénovatelných vah	$54,6 \times 10^6$	$119,6 \times 10^6$	$55,3 \times 10^5$
# zafixovaných vah	$2,3 \times 10^6$	$14,7 \times 10^6$	$23,6 \times 10^6$

CNN (FC s 1024 neurony)	AlexNet	VGG16	ResNet50
# trénovatelných vah	$10,5 \times 10^6$	$41,5 \times 10^6$	$55,3 \times 10^5$
# zafixovaných vah	$2,3 \times 10^6$	$14,7 \times 10^6$	$23,6 \times 10^6$

Tab. 3.4: Počty vah (parametrů) pro testované CNN v původní (nahoře) a redukované (dole) velikosti FC vrstev

provedeny tři běhy tréninku a uveden aritmetický průměr. Pro trénink byl dataset rozdělen do trénovací (70%), validační (10%) a testovací množiny (20%). Bylo nastaveno 20 epoch tréninku a optimizér Adam (sekce 2.2.9). Grafy na obr. 3.12 zobrazují výkonnost konvolučních sítí pro architektury s původní velikostí FC vrstev (vpravo) a redukované FC vrstvy (vlevo). Pro verze s redukovanými FC vrstvami hraje velikost dávky podstatnou roli, naopak pro verze s FC vrstvami o 4096 neuronech jsou výsledky vyrovnané.



Obr. 3.12: Správnost konvolučních neuronových sítí v závislosti na velikosti dávky

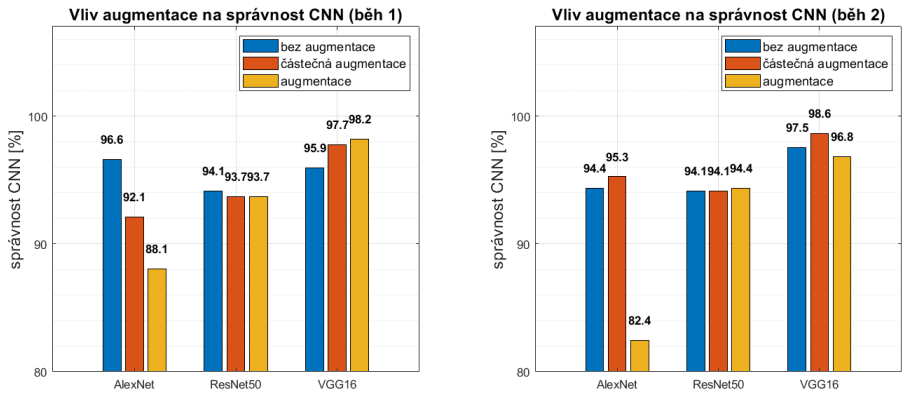
V dalších testech bude použita původní verze s FC vrstvami o 4096 neuronech, protože výsledky jsou stabilnější a síť zachovává původní ověřenou architekturu. Navíc tab. 3.5 ukazuje, že v prostředí Google Colab není doba tréninku AlexNet a VGG16 pro původní FC vrstvy výrazně delší, přestože obsahují daleko více parametrů, než varianty s redukovanými FC vrstvami.

doba trénování [s]	AlexNet	VGG16	ResNet50
FC s 4096 neurony	50,3	201,9	158,1
FC s 1024 neurony	37,2	190,2	158,1

Tab. 3.5: Doba trénování pro testované CNN v původní a redukované velikosti FC vrstev

3.3.3 Augmentace snímků

Augmentace snímků trénovací množiny je realizována pomocí již zmíněných, vlastních preprocessing vrstev. Tyto vrstvy, poskládané do řetězce, tvoří kopie snímků obrazovými a jasovými transformacemi. Zejména je využíváno zrcadlové otočení a úprava jasu přičtením a odečtením konstanty. Originální snímek je zrcadlově otočen a následně jsou oba snímky jasově upraveny. Z původního snímku tak vznikne 6 snímků, jeden tmavší, ten původní, jeden světlejší a další tři jako zrcadlové kopie.



Obr. 3.13: Vliv augmentace na správnost konvolučních neuronových sítí

Z grafu 3.13 je patrné, že augmentace má na správnost klasifikace sítě AlexNet spíše negativní vliv a pro ostatní architektury je přínos sporný a mění se mezi jednotlivými běhy tréninku. Dále byla ověřena možnost částečné augmentace, která nevyužívá zrcadlově otočené kopie snímků. Pro síť AlexNet se správnost oproti plné augmentaci zlepšila, což naznačuje, že zrcadlově otočené snímky nejsou pro zlepšení výkonnosti sítí příliš relevantní.

3.4 Srovnání metody BoVW a CNN

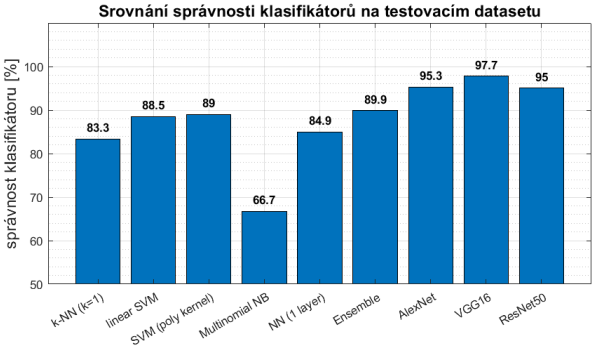
V této sekci budou porovnány výsledky klasifikace metodou BoVW s „klasickými“ klasifikátory a metody konvolučních neuronových sítí. Pro srovnání byly oba pří-

stupy podrobeny trénování a ověření správnosti na identickém trénovacím datasetu, respektive testovacím datasetu. Jedná se o plný dataset se 27 třídami (3.2) rozdělený na trénovací množinu s 1773 snímky a testovací množinu obsahující 444 snímků. Každý klasifikátor metody BoVW byl vyladěn podle nejlepších dasažených výsledků (sekce 3.2.4) a velikost vizuálního slovníku nastavena na základě grafu 3.8 na hodnotu 200. Bylo použito předzpracování ekvalizací histogramu snímků algoritmem CLAHE, jež zlepšuje správnost klasifikace (sekce 3.2.6). Metody pro vyvážení datasetu (sekce 3.2.7) nebyly využity, protože se neosvědčily. Tabulka 3.4 zobrazuje výsledky klasifikace pro 5-fold Cross-validace trénovacího datasetu a následně správnost klasifikace pro testovací dataset poté, co byly klasifikátory naučeny na celé trénovací množině. Správnost na testovací množině odpovídá výsledkům cross-validace, klasifikátory tak generalizují na nových vzorcích dle očekávání.

správnost[%]	k-NN	Linear SVM	SVM	Mul. NB	NN(1 layer)	Ensemble
5-fold CV	82,96	87,70	88,83	74,05	86,35	88,89
Test. dataset	83,33	88,51	88,96	66,67	84,91	89,86

Tab. 3.6: Správnost klasických klasifikátorů metodou BoVW

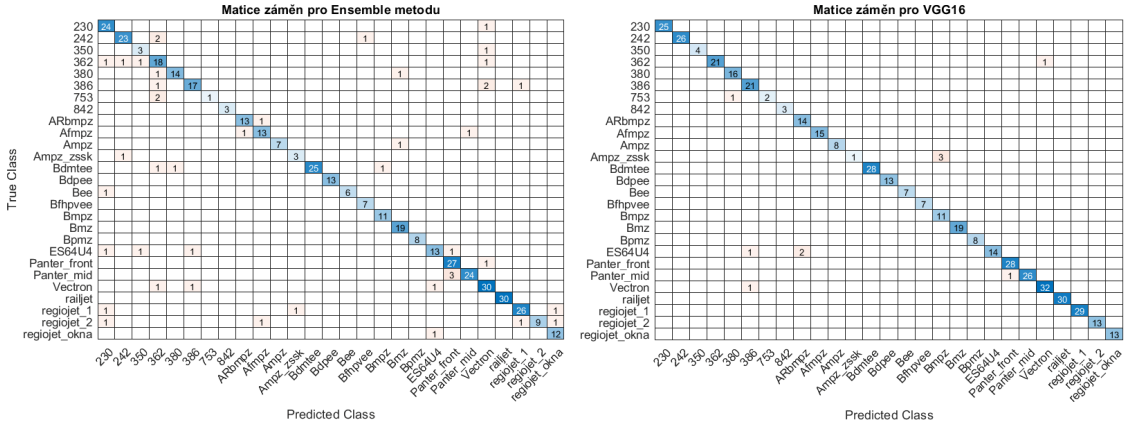
Pro CNN bylo nastaveno 20 epoch tréninku, optimizér Adam a velikost dávky 32 snímků. Dále byl zvolen trénink bez augmentace dat, protože pozitivní vliv augmentace nebyl jasně potvrzen.



Obr. 3.14: Porovnání výsledků klasifikace metodou BoVW a CNN

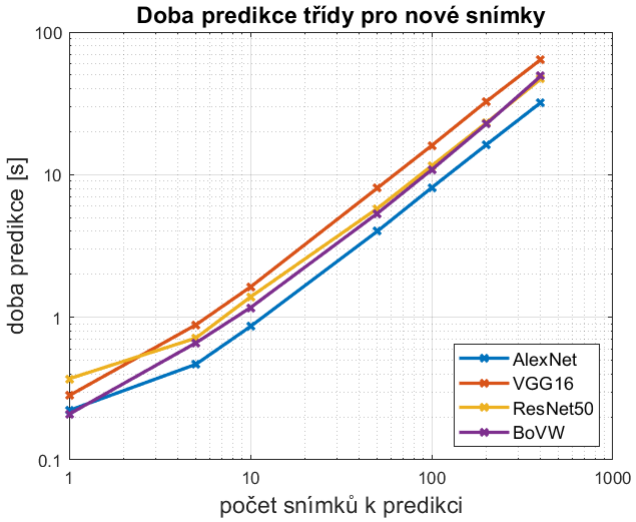
Správnost predikce V přímém srovnání výsledků klasifikace na obr. 3.14 obou přístupů je jasně vidět, že konvoluční neuronové sítě jsou úspěšnější. Pravděpodobně hraje roli předtrénování sítí na datasetu ImageNet s více než milionem snímků, které se sice nepodobají datasetu kolejových vozidel, ale umožňuje konvolučním

vrstvám naučit se extrahovat užitečnější obecné příznaky než jaké jsou extrahovány extraktorem SIFT v rámci metody BoVW.



Obr. 3.15: Matice záměn pro Ensemble metodu a síť VGG16 na testovací množině

Na obr. 3.15 jsou zobrazeny matice záměn pro nejlepší klasifikátor z obou přístupů. Pro klasické metody je zástupcem Ensemble metoda typu voting classifiers a CNN zastupuje síť VGG16. V případě Ensemble metody je nejvíce záměn mezi lokomotivami (levá horní část tabulky + třída Vectron), v případě VGG16 a ostatních dvou architektur je celková správnost uspokojivá až na jednu či dvě třídy, kde je nepoměrně více záměn.



Obr. 3.16: Doba predikce třídy pro nové snímky v logarimitckých souřadnicích

Doba predikce byla změřena pro jednotlivé konvoluční neuronové sítě a metodu BoVW na různě velkých množinách snímků. Měření bylo provedeno na kancelářském notebooku s procesorem AMD Ryzen 7 5700U (8 jader/16 vláken) 1,8/4,3 GHz bez dedikované GPU. Měření bylo opakováno pětkrát a doby predikce zprůměrovány. Výsledky jsou zobrazeny v loglog grafu 3.16. Je vidět, že s vyjímkou predikce jednoho snímku, kde časově převažuje režie implementované funkce nad samotnou dobou predikce klasifikátorů, roste doba predikce s počtem snímků lineárně. Mezi CNN je AlexNet asi dvakrát rychlejší, než síť VGG16, protože VGG16 má podle tab. 3.4 asi dvakrát více parametrů. Všechny klasické klasifikátory byly shrnuty do křivky „BoVW“, protože u této metody je doba predikce primárně dána dobou extrakce příznaků a transformace snímků na histogramy vizuálních slov. Samotná predikce klasickými klasifikátory z těchto histogramů trvá řádově kratší dobu. Doba predikce metodou BoVW je srovnatelná s dobou predikce sítí ResNet50.

3.5 Dělení vlakových souprav na vagóny

Jednou z úloh, která nebyla přímo zahrnuta mezi body zadání práce, ale byla vedoucím práce zmíněna, byla úloha automatického rozdělení snímku vlakové soupravy na jednotlivé vagóny.

Řádková kamera spustí snímání před průjezdem vlaku a do složky jsou uloženy snímky obsahující krátké fragmenty soupravy. Z těchto fragmentů je možné sestavit snímek celé vlakové soupravy ve vysokém rozlišení, příklad takového snímku je v horní části obrázku 3.17. Tyto snímky mají výšku pevně danou počtem pixelů řádkové kamery, ale šířka je proměnlivá, dle délky soupravy. Některé snímky přesahují šířku 100 tisíc pixelů. Snímky souprav tak mají tvar úzkých pruhů. Úlohou je „nasekat“ takovýto typ snímku na snímky zachycující jednotlivé vagóny. Tato úloha není zcela triviální, jelikož se scény snímků mohou mezi sebou výrazně lišit. Vlakové soupravy mají různý počet vagónů odlišných typů s různou délkou, navíc jsou soupravy zachyceny v různých denních dobách. Tato kapitola popisuje navržený algoritmus pro dělení vlakových souprav. Stěžejní částí navrženého řešení je princip metody *template matching*, která bude stručně popsána.

3.5.1 Template matching

Nejjednodušším způsobem nalezení známého objektu v obraze je hledání jeho pixelově přesné kopie, což implikuje nutnost nulové rotace a stejného měřítka objektu v obraze a vzoru objektu. V této zjednodušené formě tak hledáme shodu v obraze se vzorem (*template*), jehož podoba je známá [1].

Je dán vzor T o rozměrech $w_T \times h_T$ a obraz I s posunutím $\mathbf{x} = (x_a, x_b)$. Cílem je nalézt minimum součtu kvadrátů odchylek (*sum of squared differences*) jasové funkce [3].

$$\begin{aligned} E(x_a, x_b) &= \sum_{i=1}^{w_T} \sum_{j=1}^{h_T} (T_{i,j} - I_{x_a+i, x_b+j})^2 \\ &= \sum_{i=1}^{w_T} \sum_{j=1}^{h_T} (T_{i,j})^2 - 2 \sum_{i=1}^{w_T} \sum_{j=1}^{h_T} (T_{i,j} I_{x_a+i, x_b+j}) + \sum_{i=1}^{w_T} \sum_{j=1}^{h_T} (I_{x_a+i, x_b+j})^2 \end{aligned} \tag{3.2}$$

$E(x_a, x_b)$ měří míru podobnosti vzoru a obrazu v lokaci (x_a, x_b). V rovnici 3.2 je první člen rozepsané pravé strany konstantní a třetí člen se mění jen pomalu s x_a, x_b. Template matching může být tedy realizován maximalizací druhého členu, tj. korelace [1].

$$Corr_T(x_a, x_b) = \sum_{i=1}^{w_T} \sum_{j=1}^{h_T} (T_{i,j} I_{x_a+i, x_b+j}) \tag{3.3}$$

3.5.2 Předzpracování

Nejprve je snímek výrazně zmenšen pro urychlení zpracování, avšak stále v dostačujícím rozlišení. Následně jsou ze snímku odstraněny z obou stran okraje, na kterých není zachycena žádná část soupravy. To je provedeno metodou template matching, kdy se jako vzor (template) vezme vertikální pruh na levém konci snímku o šířce 10 pixelů, který zcela jistě neobsahuje žádnou část soupravy, pouze pozadí. Algoritmus template matching pak horizontálním posouváním vzoru po snímku soupravy vypočte hodnoty míry podobnosti v závislosti na posuvu. Empiricky nastaveným prahováním jsou pak pozice se součtem kvadrátů odchylek pod hodnotou prahu označeny za pozadí. Oříznutím lokalit označených jako pozadí z obou stran snímku je získán snímek obsahující pouze snímanou soupravu (viz. obr. 3.17).

Obr. 3.17: Separace vlakové soupravy (dole) od snímku soupravy s pozadím (nahoře)

3.5.3 Hledání mezer mezi vagóny

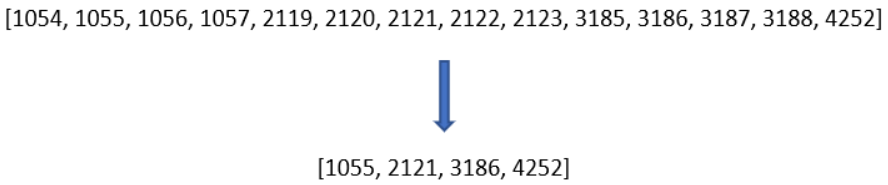
Dalším krokem je najít pozice mezer mezi vagóny. Analýza snímků souprav ukázala, že mezery jsou charakterizovány vertikálním pruhem pixelů s nízkou jasovou úrovní, způsobené stínem v mezeře. Tohoto faktu se s výhodou využije při opětovném použití metody template matching. Vzorem pro vyhledávání ve snímku soupravy bude

vertikální pruh o šířce 10 pixelů s jasovými hodnotami blízkými nule. Experiment ukázal, že je vhodné upravit šablonu (vzor) v závislosti na typu scény (noční/denní). Z vybrané oblasti pozadí je vypočítán aritmetický průměr a porovnáním s prahovou hodnotou je rozhodnuto, zda je snímek denní, či noční. Pro denní snímky je šablona s nízkými jasovými úrovněmi upravená tak, že je horní šestina pixelů nastavena na vysokou jasovou úroveň, aby odpovídala světlému pozadí denního světla.

Algoritmus template matching vrátí pozici všech shod, kde je součet kvadrátu odchylek pod empiricky daným prahem a tyto pozice jsou prohlášeny za kandidáty na pozice mezer mezi vagóny. Dodatečně se obdobně provede hledání mezery mezi lokomotivou a prvním vagónem, která je zpravidla u klasických souprav větší a nemá charakter tmavého pruhu. Pro tento účel je vhodná šablona pozadí a pozice s nejmenším součtem kvadrátu odchylek je určena jako mezera mezi lokomotivou a prvním vagónem. Dále je tedy pracováno s polem, které obsahuje pixelové pozice detekovaných mezer.

3.5.4 Filtrace kandidátů na správné pozice

Jelikož jsou mezery mezi vagóny často větší, než je šířka šablony, algoritmus template matching nalezne několik pozic mezery s malým posunem v rámci jednotek pixelů. Cílem je se zbavit takto duplikovaných pozic, které detekují stejnou mezeru. Na jednorozměrné pole pozic je aplikován shlukovací algoritmus DBSCAN, který na rozdíl od klasického K-means nemá pevně daný počet shluků, protože skutečný počet mezer není znám. DBSCAN sloučí navzájem blízké pozice do shluků a následně je vypočten aritmetický průměr pozic, který je označen za skutečnou pozici mezery. Dalším krokem filtrace je odstranění falešných kandidátů na okrajích snímku, které se občas objevovaly a skutečná přítomnost mezery v těchto oblastech je vyloučena.



Obr. 3.18: Pole obsahující pozice detekovaných mezer v pixelech a ilustrace filtrace duplikovaných mezer pomocí algoritmu DBSCAN

Posledním krokem je snaha o doplnění skutečně přítomných, ale nedetekovaných mezer vlivem osvětlení apod. Pokud je v poli rozdíl jakýchkoliv dvou pozic p_1, p_2 detekovaných mezer výrazně větší, než je průměrná délka vagónu, je tento fakt indikací,

že pravděpodobně nebyla detekována mezera mezi dvěma vagóny. Je pak předpokládáno, že rozdíl těchto dvou pozic $p_2 - p_1$ odpovídá délce dvou vagónů a nedetekovaná mezera leží mezi těmito pozicemi. Navržený naivní přístup pak předpokládá, že tyto dva vagóny mají stejnou či podobnou délku a pozice chybějící mezery je určena aritmetickým průměrem pozic $p_2 - p_1$. Při testování se ukázalo, že tento naivní přístup je dostatečnou aproximací odhadu polohy chybějící mezery.

Obr. 3.19: Ilustrace poloh detekovaných mezer (nahoře) a „nasekání“ soupravy na snímky jednotlivých vozů (dole)

Pomocí pole obsahujícího polohu mezer je jednoduché „nasekat“ snímek soupravy na snímky jednotlivých vagónů vhodné pro použití představených metod klasifikace vagónů. Ilustrace detekce mezer a rozdělení soupravy na vozy je na obrázku 3.19.

Postup dělení snímku vlakové soupravy na snímky vozů byl vyladěn na datasetu 128 snímků souprav různých délek, typu a světelných podmínek. Zejména hodnoty prahů pro algoritmus template matching byly laděny tak, aby na tomto datasetu bylo dosaženo co nejlepších výsledků. V tab. 3.7 je zaznamenána schopnost algoritmu pro dělení snímku soupravy. Ze 128 snímků bylo 9 snímků zařazeno do kategorie „nevhodný snímek“ z důvodu nedostačující kvality (přeexponováné snímky, v pozadí zachycena další souprava, atd.). Nekorektní dělení souprav z těchto snímků nebude bráno jako neschopnost dělícího algoritmu, ale jako vady akvizice snímků, které se tato práce nevěnuje. Mezi snímky zařazeny do kategorie „chybné dělení“ patří snímky, které mají vhodnou kvalitu, ale dělící algoritmus skutečně selhal. Do kategorie „správné dělení“ jsou zařazeny snímky, kde dělící algoritmus zafungoval bezchybně.

	správné dělení	chybné dělení	nevhodný snímek	celkem
počet snímků	112	7	9	128

Tab. 3.7: Robustnost algoritmu pro dělení snímku soupravy na snímky jednotlivých vagónů

3.5.5 Třída „Panter“

Třída vozů Panter, hojně zastoupená v testovacím datasetu, způsobovala kvůli svému barevnému vzhledu detekci velkého množství falešných mezer. Toto množství bylo tak velké, že bylo možné spolehlivě odfiltrovat soupravu třídy Panter od ostatních a použít specializovanou šablonu, která vede k detekci skutečných mezer bez množsví falešně pozitivních případů.

3.6 Struktura klasifikační knihovny a aplikace

Pro úlohu klasifikace snímků kolejových vozidel byly vytvořeny dvě třídy, v závislosti na použité metodě. Třída `BOVW_model` implementuje metodu transformace snímků do příznakového prostoru metodou BOVW a následně aplikuje vyladěné klasifikátory typu k-NN, Linear SVM, SVM s polynomiálním kernelem, neuronová síť, Multinomial Naive Bayes a Voting classifiers. Třída `Neural_model` implementuje klasifikaci metodou konvolučních neuronových sítí. Implementovanými architekturami jsou AlexNet, VGG16 a ResNet50.

3.6.1 Třída BOVW_model

Třída `BOVW_model` obsahuje metody pro natrénování, evaluaci úspěšnosti klasifikace, predikci nových snímků a ukládání a načítání modelů. Konstruktor instance třídy vytvoří objekt s nastavenými parametry, jako je počet extrahovaných významných bodů, velikost vizuálního slovníku a jména a počet klasifikovaných tříd. Metoda `train_classifiers` předzpracovává trénovací snímky, extrahuje významné body a vytvoří vizuální slovník podle sekce 1.6. Následně transformuje trénovací snímky do příznakového prostoru pomocí slovníku jako histogramy vizuálních slov. Na těchto histogramech jsou natrénovány vyladěné klasifikátory, které jsou následně uloženy do složky.

Metoda `evaluate_classifier` ověřuje správnost klasifikace na testovacím datasetu. Extrahuje významné body z testovacích snímků a pomocí již sestaveného slovníku je transformuje na histogramy vizuálních slov. Histogramy jsou klasifikovány natrénovanými klasifikátory a predikce porovnány se skutečnými třídami testovacích snímků. Z tohoto porovnání je vypočtena matice záměn, celková správnost a metrika CBA jednotlivých klasifikátorů. Tyto veličiny jsou výstupem metody.

Metoda `predict_image` načte snímky ze složky určené ke klasifikaci, extrahuje významné body a pomocí již sestaveného slovníku transformuje snímky na histogramy vizuálních slov. Na základě histogramů slov jsou sestaveny predikce natrénovaných klasifikátorů a ve složce se snímky je vytvořen CSV soubor „klasifikace.csv“,

kam je uloženo do řádku jméno snímku a predikce všech klasifikátorů. Každý řádek tak odpovídá jednomu snímku ze složky.

Interní metoda `_save_classifiers` slouží pro celkové uložení BoVW modelu. Ukládá natrénované klasifikátory, vizuální slovník a další parametry potřebné pro zpětné načtení modelu metodou `load_classifiers`.

Třída `BOVW_model` využívá modul `visual_words`, který implementuje dílčí funkce pro sestavení BoVW modelu, předzpracování dat, čtení a zápis snímků do složek a další užitečné funkce pro zpřehlednění implementace třídy.

3.6.2 Třída Neural_model

Třída `Neural_model` obsahuje metody pro trénování, evaluaci správnosti klasifikace, predikci nových snímků a ukládání a načítání uložených modelů. Konstruktor neinicializuje žádné hodnoty prostřednictvím parametrů, nastavení parametrů je prováděno v metodě pro trénink sítí.

Metoda `train_neural` trénuje konvoluční neuronové sítě a je možno ověřit jejich schopnosti na testovacím datasetu odděleném z trénovací množiny. Metoda načítá snímky z datasetu po dávkách definované velikosti a parametr `val_split` metody udává, jak velká část datasetu je oddělena jako validační+testovací množina. Je také možné natrénovat CNN na celém datasetu a ověřit schopnost na předem vytvořeném testovacím datasetu pomocí metody `evaluate_CNN`. Snímky jsou zmenšeny na požadovanou velikost a předzpracovány. Podle nastavení parametru augmentace jsou snímky z trénovací množiny příslušně augmentovány dle posledního odstavce v sekci 3.3. Následně jsou na trénovacím datasetu dotrénovány FC vrstvy předtrénovaných sítí typu AlexNet, VGG16 a ResNet50, v každé epoše je správnost klasifikace ověřena na validačním datasetu. Po natrénování je provedena predikce pro testovací množinu a predikce jsou porovnány se skutečnými třídami, z těchto dat je určena celková správnost pro každou ze tří zmíněných architektur sítě.

Interní metoda `_save_neural` ukládá do složky optimalizované váhy a jména tříd, pro které byly sítě natrénovány, tak, aby je bylo možné vyčíst metodou `load_neural` a váhami inicializovat nové instance modelů sítí. Metoda `load_neural` umožňuje načíst jen specifikovanou CNN nebo všechny dostupné typy.

Metoda `predict_image` načte snímky ze složky a předzpracuje je do podoby „stravitelné“ pro neuronové sítě a poté provede predikce pro všechny tři sítě nebo jednu zvolenou síť. Stejně jako ve třídě pro BoVW model je ve složce snímků určených k predikci vytvořen CSV soubor, kde jsou po řádcích uloženy jména snímků a predikce třídy.

Metoda `evaluate_CNN` ověřuje správnost klasifikace na testovacím datasetu. Využívá metody `predict_image` pro predikci a porovnává predikce tříd se skutečnými

třídami testovacích snímků. Z tohoto porovnání je vypočtena matice záměn, celková správnost a metrika CBA jednotlivých konvolučních sítí. Tyto veličiny jsou výstupem metody.

3.6.3 Třída Wagon_separator

Třída `Wagon_separator` obsahuje metody pro rozdělení snímku vlakové soupravy na snímky jednotlivých vozů.

Metoda `read_train_file` zpracuje RAW adresář průjezdu vlakové soupravy a sestaví ze zachycených fragmentů průjezdu dlouhý snímek celé soupravy.

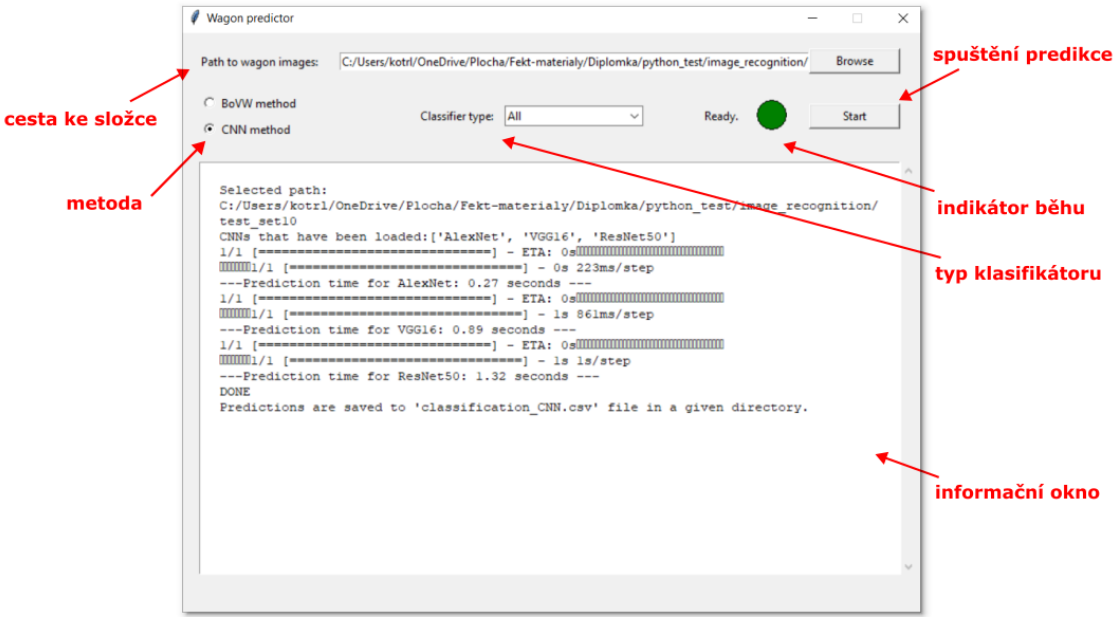
Metoda `separate_wagons` je hlavní metodou třídy a vykonává všechny kroky algoritmu pro rozdělení snímku soupravy na snímky vozů pomocí interních metod. Mezi tyto interní metody patří: `_preprocess_train` pro separaci soupravy od pozadí, `_find_gaps` pro hledání mezer mezi vozy, `_filter_gaps` pro filtraci falešně pozivních nálezů a `_cut_wagons` pro samotné rozdělení dlouhého snímku na snímky vozů. Snímky rozdělených souprav jsou uloženy do strukturovaného adresáře podle ID soupravy. Metoda `print_gaps` generuje snímek soupravy s ilustrací nalezených mezer mezi vozy.

3.6.4 Aplikace „Wagon predictor“

Pro demostraci klasifikační knihovny byla vytvořena jednoduchá aplikace pro klasifikaci snímků vozů využitím implementovaných knihovních funkcí. Aplikace má spíše demostrativní charakter, jelikož využití implementovaných funkcí je hlavně v rámci automatické klasifikace nových snímků pomocí skriptu. Nicméně aplikace umožňuje vybrat složku se snímky, vybrat metodu, uskutečnit predikci pro tyto snímky a výsledek uložit do csv souboru. Aplikace předpokládá dostupnost již natrénovaných modelů, které si podle potřeby načte.

Vzhled aplikace je na obr. 3.20. Uživatel vybere cestu ke složce, ve které se nachází snímky, pro které chce provést predikci. Následně vybere mezi BoVW a CNN metodou, zvolí vhodný dostupný klasifikátor pro danou metodu a spustí predikci. O běhu predikce je uživatel informován kontrolkou. O průběhu a dokončení predikce je uživatel informován prostřednictvím informačního okna. Po dokončení predikce je možné přenastavit požadovanou složku snímků, metodu, klasifikátory a predikci opětovně spustit.

Aplikace byla vytvořena pomocí knihovny `Tkinter`. Pro správnou funkci musí grafické rozhraní, složené z tzv. *widgetů*, nutně běžet v hlavním vlákně, aby „nezamrzlo“ po spuštění algoritmu predikce. Algoritmus predikce je spuštěn po stisknutí tlačítka „Start“ na pozadí v jiném vlákně. Pro predikci jsou využity implementované funkce klasifikační knihovny, jako je načtení natrénovaného modelu, predikce třídy snímků



Obr. 3.20: Vzhled a popis aplikace pro predikci snímků

a uložení výsledků do csv souboru. Standardní výstup je přesměrován z konzole do informačního okna pomocí Tkinter widgetu ze [35], který byl podle potřeby upraven.

Závěr

V diplomové práci byl prakticky ověřen potenciál systému klasifikace snímků kolejových vozidel na základě obrazové informace. Na základě provedených testů implementovaných metod lze zvážit možnost nahrazení části stávajícího systému pro klasifikaci kolejových vozidel na bázi indukčních snímačů kamerovým systémem.

Práce se věnuje klasifikaci snímků vozidel ve dvou rovinách, jde o dva odlišné přístupy. Za klasický přístup byla označena metoda Bag of Visual Words s následným využitím základních klasifikátorů. Druhým přístupem je použití moderních konvolučních neuronových sítí, které za poslední dekádu změnily pohled na zpracování obrazu. Oba přístupy byly detailně popsané v teoretické části, ve které byla rovněž provedena rešerše metod zpracování obrazu a metodiky vyhodnocení a odhadu přesnosti modelu. V části praktické byly implementovány patřičné algoritmy se snahou implementovat funkce analogicky pro oba přístupy. V závěru jsou oba přístupy srovnány.

První polovina praktické části se věnuje implementaci zpracování snímků a transformaci obrazové informace na histogram vizuálních slov v rámci metody BoVW. Následně byly tyto histogramy použity jako vektor příznaků pro vybrané klasifikátory, konkrétně kNN, Linear SVM, SVM s polynomiálním kernelem, Multinomial Naive Bayes, neuronová síť s jednou skrytou vrstvou a Ensemble metoda. Odhad přesnosti jednotlivých klasifikátorů byl proveden v souladu s nastudovanou metodikou a byla uskutečněna řada testů v rámci ladění hyperparametrů daných modelů na datasetu obsahujícím 1733 snímků osobních vozů a lokomotiv rozdělených do 15 tříd. Empiricky bylo dokázáno, že předzpracování algoritmem CLAHE zlepšuje konečnou správnost klasifikátorů. Naopak příznivý vliv vyvážení méně zastoupených tříd datasetu metodami Random Oversampling a SMOTE nebyl potvrzen. Rovněž byl prověřen vliv velikosti vizuálního slovníku na správnost klasifikace.

Druhá polovina praktické části se zabývá implementací a trénováním vybraných architektur konvolučních neuronových sítí (AlexNet, VGG16, ResNet50), jejichž konvoluční vrstvy byly předtrénovány na datasetu ImageNet. Pro CNN byly implementovány metody pro trénink, evaluaci a predikci, analogicky jako pro přístup BoVW. Dále byla testována augmentace snímků úpravou jasových hodnot a zrcadlovým otáčením snímku. Lépe se osvědčila jasová augmentace bez zrcadlově otočených kopií snímku.

V přímém srovnání obou přístupů byla ověřena schopnost predikce a doba predikce. Pro trénink byl použit dataset s 1773 snímky a 27 třídami. Testovací dataset obsahoval 444 snímků. Nejlepší klasifikátor z kategorie BoVW byl typ Ensemble, kde o konečné predikci rozhodovaly klasifikátory kNN, Linear SVM, SVM s polynomiálním kernelem a neuronová síť s jednou skrytou vrstvou. Klasifikátor dosahoval

správnosti cca. 89%. Správnost CNN se konstantně pohybovala mezi 95-98%, síť VGG16 navíc mírně předčila ostatní dvě architektury. Z časového hlediska je predikce sítě VGG16 asi 2x pomalejší, než u nejrychlejší sítě AlexNet. Doba predikce BoVW a ResNet50 je srovnatelná a časově leží mezi dobou predikce AlexNet a VGG16. Pokud není doba predikce důležitý faktor, je síť VGG16 nejlepší volbou.

Nad rámec zadání byl navržen algoritmus pro dělení snímků souprav na snímky jednotlivých vozů, které je pak možné klasifikovat výše zmíněnými metodami. Algoritmus pracuje na základě metody template matching pro hledání mezer mezi vozy. Objektivně lze říci, že navržený algoritmus není bezchybný a při tak různorodých snímcích souprav (denní a noční scény, různé délky a počet vozů v soupravě, různé typy souprav atp.) by byla vhodná větší robustnost systému. Navíc algoritmus nebyl vůbec navržen na nákladní soupravy, v některých případech by však pravděpodobně obstál. Tato oblast je místem pro budoucí zaměření pozornosti a zlepšení spolehlivosti.

Kromě zlepšení algoritmu pro dělení snímku soupravy je nutné lépe definovat, které třídy kolejových vozidel je třeba rozpoznávat. Následně je pro tyto třídy důležité nasbírat více vzorků s dostačujícím zastoupením ve všech definovaných třídách. Z provedených testů lze soudit, že potenciál metody BoVW byl pravděpodobně téměř naplněn a není možné dosáhnout takové úspěšnosti klasifikace, jako v případě konvolučních neuronových sítí.

Literatura

[1] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image processing, analysis, and machine vision*. Fourth edition. Stamford, CT: Cengage Learning, 2015. ISBN 978-1-133-59360-7.

[2] DOUGHERTY, Geoff. *Pattern Recognition and Classification*. New York, NY: Springer-Verlag, 2013. ISBN 978-1-4614-5322-2.

[3] SZELISKI, Richard. *Computer Vision: Algorithms and Applications*. 2nd ed. Cham: Springer International Publishing, 2022. ISBN 978-3-030-34371-2. Dostupné z: doi:10.1007/978-3-030-34372-9

[4] FORSYTH, David A. a Jean PONCE. *Computer Vision: A modern approach*. 2nd ed. New Jersey: Prentice-Hall, 2012, ISBN 978-0-13-608592-8.

[5] HARTLEY, Richard a Andrew ZISSERMAN. *Multiple view geometry in computer vision*. 2nd ed. Cambridge: Cambridge University Press, 2003, 655 s. ISBN 0-521-54051-8.

[6] SOJKA, Eduard, Jan GAURA a Michal KRUMNIKL. *Matematické základy digitálního zpracování obrazu*. VŠB-TU Ostrava a Západočeská univerzita v Plzni, 2011.

[7] LAZEBNIK, Svetlana. *Corner detection* [online]. University of Illinois, 2022 [cit. 2022-10-03]. Dostupné z: <`https://slazebni.cs.illinois.edu/fall22/`>

[8] LOWE, David G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* [online]. DORDRECHT: Springer Nature, 2004, **60**(2), 91-110 [cit. 2022-10-03]. ISSN 0920-5691. Dostupné z: <`https://doi.org/10.1023/B:VISI.0000029664.99615.94`>

[9] CSURKA, Gabriela, Christopher DANCE, Lixin FAN, Jutta WILLAMOWSKI a Cédric BRAY. Visual categorization with bags of keypoints. *ECCV International Workshop on Statistical Learning in Computer Vision* [online]. 2004 [cit. 2022-10-05]. Dostupné z: <`https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/csurka-eccv-04.pdf`>

[10] YANG, Jun, Yu-Gang JIANG, Alexander G. HAUPTMANN a Chong-Wah NGO. Evaluating Bag-of-Visual-Words Representations in Scene Classification.*Proceedings of the international workshop on Workshop on multimedia information retrieval*. 2007. Dostupné z: <`https://doi.org/10.1145/1290082.1290111`>

[11] WATT, Jeremy, Reza BORHANI a Aggelos K. KATSAGGELOS. *Machine learning refined: foundations, algorithms, and applications*. Cambridge: Cambridge University Press, 2016. ISBN 978-1-107-12352-6.

[12] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*. Second edition. Beijing: O'Reilly, 2019. ISBN 978-1-492-03264-9.

[13] BURKOV, Andriy. *The Hundred-Page Machine Learning Book*. 2019. ISBN 9781999579500.

[14] BISHOP, Christopher M. *Pattern recognition and machine learning*. [New York]: Springer, 2006. Information science and statistics. ISBN 0-387-31073-8.

[15] MANNING, Christopher D., Prabhakar RAGHAVAN a Hinrich SCHÜTZE, *Introduction to Information Retrieval*. [online]. 2008. Cambridge University Press. Dostupné z: <https://nlp.stanford.edu/IR-book/information-retrieval-book.html>

[16] HONZÍK, Petr, Karel HORÁK. Učení založené na instancích. *Strojové učení*. [online]. [cit. 2022-11-19]. Dostupné z: <http://vision.uamt.feec.vutbr.cz/?course=STU>

[17] PEDREGOSA, Fabian, Gaël VAROQUAUX, Alexandre GRAMFORT a Vincent MICHEL. Scikit-learn: Machine learning in Python. *Journal of machine learning research* [online]. 2011. [cit. 2022-11-01]. Dostupné z: <https://scikit-learn.org/stable/user_guide.html>

[18] PIZER, Stephen M., E. Philip AMBURN, John D. AUSTIN, Robert CROMARTIE, Ari GESELOWITZ, Trey GREER, Bart ter HAAR ROMENY, John B. ZIMMERMANN, Karel ZUIDERVELD. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*. 1987.

[19] SCULLEY, David. Web-scale k-means clustering. *Proceedings of the 19th international conference on World wide web*. 2010. Dostupné z: <https://doi.org/10.1145/1772690.1772862>

[20] SOKOLOVA, Marina, Guy LAPALME. A systematic analysis of performance measures for classification tasks. *Information Processing & Management* [online]. 2009 [cit. 2022-11-03]. Dostupné z: <https://doi.org/10.1016/j.ipm.2009.03.002>

[21] MORTAZ, Ebrahim. Imbalance accuracy metric for model selection in multi-class imbalance classification problems. *Knowledge-based systems* [online]. Amsterdam: Elsevier B.V, 2020, 210, 106490 [cit. 2022-11-08]. ISSN 0950-7051. Dostupné z: <https://doi.org/10.1016/j.knosys.2020.106490>

[22] MOSLEY, Lawrence. *A balanced approach to the multi-class imbalance problem.* Doctor of Philosophy Thesis, Iowa State University of Science and Technology, USA, 2013.

[23] HORÁK, Karel. Jasové transformace. *Zpracování vícerozměrných signálů.* [online]. [cit. 2022-11-17]. Dostupné z: <http://vision.uamt.feec.vutbr.cz/?course=ZVS>

[24] OLVERA, Raul David Palma, et al. A feature extraction using SIFT with a preprocessing by adding CLAHE algorithm to enhance image histograms. *2014 International Conference on Mechatronics, Electronics and Automotive Engineering.* 2014. Dostupné z: <https://doi.org/10.1109/ICMEAE.2014.41>

[25] GHOSH, Anirudha, A. SUFIAN, Farhana SULTANA, Amlan CHAKRABARTI and Debashis DE, year = 2020, *Fundamental Concepts of Convolutional Neural Network.* Dostupné z: <https://www.researchgate.net/publication/337401161_Fundamental_Concepts_of_Convolutional_Neural_Network>

[26] CHOLLET, François. *Deep learning with Python.* Shelter Island, NY: Manning, 2018, ISBN 978-1-61729-443-3.

[27] ELGENDY, Mohamed. *Deep learning for vision systems.* Simon and Schuster, 2020. ISBN 9781617296192.

[28] BAŠTINEC, Jaromír, Břetislav FAJMON, Jan KOLÁČEK. *Pravdìpodobnost, statistika a operaèní výzkum.* Ústav matematiky FEKT VUT v Brně. 2014.

[29] SHORTEN, Connor a Taghi M. KHOSHGOFTAAR. A survey on Image Data Augmentation for Deep Learning. *Journal of big data [online].* Springer International Publishing. 2019. [cit. 2023-02-08]. ISSN 2196-1115. Dostupné z: <https://doi.org/10.1186/s40537-019-0197-0>

[30] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM.* 2017. Dostupné z: <https://doi.org/10.1145/3065386>

[31] SIMONYAN, Karen a Andrew ZISSERMAN. *Very Deep Convolutional Networks for Large-Scale Image Recognition* [online]. 2014 [cit. 2023-02-08]. Dostupné z: <https://doi.org/10.48550/arxiv.1409.1556>

[32] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016 . ISBN 9781467388511. Dostupné z: <https://doi.org/10.1109/CVPR.2016.90>

[33] CHOLLET, François. *Trained image classification models for Keras.* [online]. [cit. 2023-03-04]. Dostupné z: <https://github.com/fchollet/deep-learning-models/releases>

[34] REYES, Armando a Eduardo RIVAS. *AlexNet TensorFlow with Weights.* [online]. [cit. 2023-03-01]. Dostupné z: <https://github.com/Eduardo-RP/Pretrained-AlexNet-TensorFlow-with-Weights>

[35] Tkinter, Console widget. [online]. [cit. 2023-05-09]. Dostupné z: <https://www.reddit.com/r/Tkinter/comments/nmx0ir/how_to_show_terminal_output_in_gui/>

[36] HASSAN, Muneeb ul. *AlexNet - ImageNet Classification with Deep Convolutional Neural Networks.* 2018. [online]. [cit. 2023-16-04]. Dostupné z: <https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/>

[37] FERGUSON, Max, Ronay AK, Yung-Tsun Tina LEE, a Kincho H. LAW. *Automatic localization of casting defects with convolutional neural networks.* 2017. Dostupné z: <https://doi.org/10.1109/BigData.2017.8258115>

Seznam symbolů a zkratek

SGD	Stochastic Gradient Descent
BoVW	Bag of Visual Words
CLAHE	Contrast Limited Adaptive Histogram Equalization
SMOTE	Synthetic Minority Oversampling Technique
CNN	Convolutional Neural Network
kNN	k-Nearest Neighbors
SVM	Support Vector Machines
NB	Naive Bayes
FC	Fully-connected
NN	Neural Network
GAN	Generative Adversarial Network
CBA	Class Balance Accuracy

A Obsah elektronické přílohy

Na přiloženém paměťovém médiu jsou soubory uložené následujícím způsobem:

- `/` ..... kořenový adresář přiloženého média
  - `datasets`
    - `raw_passing_trains` ...RAW složky vlakových souprav pro dělící algoritmus
    - `test_set` ..... plný testovací dataset jednotlivých snímků vozů
    - `test_set10` .....malý testovací dataset jednotlivých snímků vozů
    - `labels_test.pkl` .... seznam správných tříd pro snímky testovacího datasetu
    - `train_set` ..... adresář trénovacího datasetu s požadovanou strukturou
  - `libraries`
    - `bovw_lib`
      - `BoVW_Class.py` ..... zdrojový kód třídy `BoVW_model`
    - `neural_lib`
      - `alexnet.h5` ..... předtrénovaná síť AlexNet
      - `ConvNets.py` ..... zdrojový kód architektur CNN
      - `custom_layers.py` .....zdrojový kód vlastních augmentačních vrstev
      - `Neural_Class.py` .....zdrojový kód třídy `Neural_model`
      - `vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5`..předtrénované váhy pro VGG16
    - `separator_lib`
      - `panter_template.jpg` ..... speciální vzor pro soupravy panter
      - `Wagon_separator_Class.py` ..... zdrojový kód třídy `Wagon_separator`
    - `utility`
      - `visual_words.py` .modul funkcí pro BoVW model a další užitečné funkce
  - `models`
    - `trained_classifiers` ... adresář natrénovaného a uloženého BoVW modelu
    - `trained_convnets` .....adresář natrénovaného a uloženého modelu s CNN
  - `bovw_test.py` ..... ukázka použití třídy `BoVW_model`
  - `neural_test.py` .....ukázka použití třídy `Neural_model`
  - `separator_test.py` .....ukázka použití třídy `Wagon_separator`
  - `app_test.py` ..... zdrojový kód aplikace
  - `Wagon_predictor.exe` ..... spustitelná aplikace pro klasifikaci vozů
  - `README.md` ..... základní informace o kořenovém adresáři

B Seznam verzí použitých knihoven

Klasifikační knihovna a všechny její součásti byly implementovány v Pythonu verze 3.10.11 s následujícími verzemi využitých knihoven. V této konfiguraci jsou všechny součásti vlastního programu stabilní.

knihovna	verze
OpenCV	4.7.0
TensorFlow	2.12.0
keras	2.12.0
scikit-learn	1.2.2
scipy	1.10.1
imbalanced-learn	0.10.1

Tab. B.1: Seznam verzí použitých knihoven