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

ROZPOZNÁNÍ DISPLEJE EMBEDDED ZAŘÍZENÍ
SCREEN RECOGNITION OF EMBEDDED SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER THESIS

AUTOR PRÁCE — **Bc. Václav Novotný**
AUTHOR

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

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Václav Novotný ***ID:*** 168853
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Rozpoznání displeje embedded zařízení

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a zrealizovat systém na bázi strojového učení pro rozpoznání a analytický popis obsahu displeje embedded zařízení. Vstupem bude obraz obsahující relevantní část displeje embedded zařízení. Výstup tohoto systému bude využívat existující nadřazená aplikace poskytující zpětnou vazbu robotickému systému.

1. Seznamte se s problematikou strojového učení v počítačovém vidění a definujte požadavky na výstup systému.
2. Proveďte rešerši existujících typů strojového učení a diskutujte vhodnost a limitace jednotlivých řešení. Vyberte nejvhodnější řešení pro danou aplikaci.
3. Sestavte galerie učících a testovacích snímků pro jednoho výrobce zařízení a vyzkoušejte vybraný typ strojového učení.
4. Sestavte galerie pro vybranou skupinu výrobců embedded zařízení.
5. Implementujte možnost postupného doučování při přidání nového typu zařízení nebo obrazovky.
6. Integrujte vytvořený systém do existující aplikace poskytující zpětnou vazbu robotickému systému.
7. Otestujte a vyhodnoťte.

DOPORUČENÁ LITERATURA:

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

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 14.5.2018

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

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

Abstrakt

Tato diplomová práce se zabývá využitím metod strojového učení v oblasti počítačového vidění pro klasifikaci neznámých obrazů. V první části je provedena rešerše dostupných metod strojového učení, jejich limitace a vhodnost k řešené úloze. V další části jsou představeny přístupy k vytváření galerie. Následně je navrhnuto řešení klasifikátorů a architektury systému, které je v řešení realizováno a implementováno. Výsledný systém je nakonec otestován a vyhodnocen.

Klíčová slova

Strojové učení, umělá inteligence, počítačové vidění, neuronové sítě, deskriptor, metoda podpůrných vektorů, slovník vizuálních slov, SURF, SIFT, FREAK

Abstract

This master thesis deals with usage of machine learning methods in computer vision for classification of unknown images. The first part contains research of available machine learning methods, their limitations and also their suitability for this task. The second part describes the processes of creating training and testing gallery. In the practical part, the solution for the problem is proposed and later realised and implemented. Proper testing and evaluation of resulting system is conducted.

Keywords

Machine learning, artificial intelligence, computer vision, neural networks, descriptor, support vector machines, bag of visual words, SURF, SIFT, FREAK

Bibliografická citace

NOVOTNÝ, V. *Rozpoznání displeje embedded zařízení.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 76 s. Vedoucí diplomové práce Ing. Peter Honec, Ph.D..

Prohlášení

Prohlašuji, že jsem svou diplomovou práci na téma Rozpoznání displeje embedded zařízení vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové 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 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.

V Brně dne ....
(podpis autora)

Poděkování

Tímto děkuji vedoucímu mé diplomové práce Ing. Peteru Honcovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....
(podpis autora)

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Tato práce se zabývá problematikou zpracování obrazu a jeho klasifikace do konečného počtu tříd. Klasifikace je řešena použitím vybraných modelů strojového učení, které jsou součástí oblasti umělé inteligence.

Hlavním cílem této úlohy je vytvoření vizuální zpětné vazby nadřazenému systému, který ovládá embedded zařízení klikáním na akční elementy na displeji a sousledným cílevědomými kliky na něm provádí své testování. Pro správné vyhodnocení situace a pro zjišťování stavu testovaného objektu je nutná přesná a robustní zpětná vazba, která bude informovat testovací systém o aktuálním stavu testovaného zařízení, a která je tak náplní této práce. Aktuálním stavem je většinou obrazovka, kterou testované zařízení zobrazuje. Snímání displeje testovaného zařízení bude probíhat pomocí ethernetové kamery.

Testovaným zařízením může být obecně jakékoliv zařízení s dotykovým displejem nebo pasivním displejem s postranními ovládacími fyzickými tlačítky. Aktuálně jsou nejvíce testovány multifunkční tiskárny, konkrétně rozšíření třetí strany Y Soft SafeQ pro správu tisku, kopírování a skenování. Rovněž jsou podrobeny testování robotickým systémem i mobilní zařízení, IoT platformy a displeje $3D$ tiskáren. Největším problémem vydání nové aktualizace pro tyto zařízení je jejich včasné, důkladné a kompletní otestování. Každé zařízení, které rozšíření SafeQ podporuje, musí být před každým vydáním nové verze otestováno. Plnohodnotné otestování všech podporovaných zařízení trvá QA inženýrům bezmála čtrnáct dnů, což je neslučitelné s firemní strategií CI (Continuous Integration) a celkovým agilním přístupem k vývoji produktu. Rovněž v průběhu tohoto časového okna jsou nalezené chyby postupně opravovány, avšak testování po každé opravené chybě nezačíná od začátku, tudíž není možné zjistit, zda oprava nepoškodila jinou komponentu, která již dříve v průběhu vydávacího cyklu byla otestována. Robotický systém je schopen zkrátit dobu celého dosavadního testování do dvou dnů, a to převážně díky automatizované instalaci zařízení, nepřetržitému provozu, efektivnímu přístupu a paralelnímu testování všech zařízení souběžně použitím více robotických systémů. Občas se vyskytují dotazy, proč se zařízení netestují virtuálně a co vlastně robotický systém přináší navíc. Virtuální simulace zařízení a jeho testování existuje, nicméně ani zdaleka nepokrývá testy reálného zařízení, které se zásadně vždy chová jinak než ideálně. Koncový uživatel rovněž nebude využívat simulované zařízení ale reálné zařízení, proto je nutné otestovat funkčnost a vlastnosti, tedy nejen verifikaci ale rovněž i validaci jednotlivých komponent. Rovněž většina multifunkčních tiskáren nemá přístupné externí ovládání nebo hlášení stavu a bylo by tak nemožné ji programově ovládat pro účely testování. Další výhodou robotického systému je rychlá zpětná vazba pro vývojáře

o stavu aktuální větve vývoje. Vývojář, který udělá změnu v kódu, je schopen se promptně dozvědět hned následující den, zda jeho úprava kódu nerozbila funkčnost jiných komponent, a to díky automatickému nočnímu testování vždy aktuální master větvě.

Součástí testovacího systému je již přítomna vizuální zpětná vazba založena na extrakci deskriptorů z obrazu a hledání největší shody s deskriptory referenčních obrazů, toto hledání shody je realizováno algoritmem hrubé síly a přidruženými algoritmy. Druhým algoritmem pro rozpoznání obrazu je za použití optického rozpoznávání znaků (OCR), ten vyhledává texty v obraze a snaží se najít největší shodu s texty definovanými v databázi, které mají informace o tom, které obrazovce přísluší. Zmíněná metoda rozpoznávání obrazu pomocí OCR je v zásadě přesná, metoda používající deskriptory a hledání shody hrubou silou už tolik ne. Nicméně tyto algoritmy mají řadu nedostatků. Jeden z nich je výpočetní náročnost rozpoznávání využitím OCR, které trvá průměrně jednotky vteřin. Další z nevýhod aktuálního stavu je nízká úroveň generalizace, obtížná rozšiřitelnost a časově náročná definice nových obrazovek. Některé konkrétní úlohy aktuální zpětné vazby jsou například detekování aktivního stavu některých tlačítek, detekce počtu hvězdiček v poli pro zadávání pinu, nebo detekce vybraných tiskových úloh v seznamu úloh. Nevýhoda klasifikace obrazovek i těchto funkcionálních úloh spočívá v tom, že pro každé zařízení a změnu konfigurace, která vede k alespoň částečné změně rozložení nebo grafiky, je potřeba implementovat algoritmus, který bude přesně danou věc řešit, a to nejen stojí čas a peníze, ale hlavně to znemožňuje systém nasadit na úplně nové testovací zařízení bez úprav v kódu. Krom těchto konkrétních algoritmů ve vizuální zpětné vazbě je celý testovací systém ve vysoké několikavrstvé úrovni abstrakce, kde testy, které se na jednotlivých zařízeních provádí, jsou pouze jedny a nerozlišuje se v nich vůbec o jaké zařízení či o jakou konfiguraci jde, a proto je důležité zmíněné závislosti odstranit i z existující vizuální zpětné vazby, což je rovněž účelem této práce.

1 ROZBOR ZADÁNÍ

Prvním úkolem zadání je seznámení se s problematikou strojového učení a počítačového vidění a definice požadavků na výstup vytvořeného systému. Cíle výsledného systému jsou jasné, vytvořit takový systém, který z příchozího obrazu, zachycující displej libovolného dotykového zařízení, dokáže klasifikovat třídu, do které patří. Systém bude schopen se naučit na vytvořené trénovací galerii a rovněž bude schopen se doučit na nové třídy definované po hlavním učícím cyklu.

V druhé části zadání má být provedena rešerše dostupných metod strojového učení a následná diskuse o limitacích a vhodností jednotlivých přístupů a vybrání té, která teoreticky nejvíce vyhovuje řešené problematice. Pro tuto úlohu se výběr omezí pouze na metody typu učení s učitelem.

Ve třetí části zadání práce je za úkol sestavení učící galerie pro vybraného výrobce testovaných zařízení a vyzkoušení vybrané metody z kroku 2. Jak bylo zmíněno v úvodu, systém vytvořený v této úloze bude později součástí většího systému, který testuje zařízení. Tyto zařízení jsou různorodá a liší se velikostí i typem displeje, chováním integrovaného prostředí ale i omezeními, které na celý testovací systém přidává celou řadu požadavků. Všechny tyto proměnné nejčastěji závisí na výrobci zařízení, a právě pro jednoho tohoto výrobce bude sestavena trénovací galerie. Ve čtvrté části zadání budou pak sestaveny galerie i pro ostatní požadované výrobce zařízení.

Pátá část zadání definuje požadavek na výsledný systém, který má být schopen se doučit na novou klasifikační třídu nebo nově přidané zařízení. Schopnost doučení má být buď naprosto automatická po přidání záznamu do databáze, nebo při požadavku uživatele, v závislosti na zjištěné vhodnosti pro tuto úlohu.

V šesté části zadání je integrace řešení této práce do existující vizuální zpětné vazby robotického systému tak, aby byla schopná využívat naučené klasifikátory a klasifikovat příchozí snímky do jedné z klasifikačních tříd.

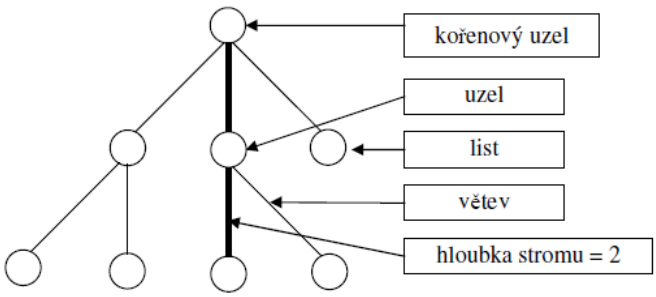
V rámci poslední části zadání má proběhnou testování výsledného řešení a vyhodnocení výsledků. Rovněž tak určení limitací systému a jeho možnosti rozšíření.

2 METODY STROJOVÉHO UČENÍ

Nejdříve je potřebné definovat požadované vlastnosti, které vybraná metoda musí splňovat. Všechny metody schopné řešit zadanou úlohu musí být typu učení s učitelem, jelikož při učení vždy známe výstupní třídu vzorku. Zvolená metoda musí být schopna vysoké úrovně generalizace a být schopna rozdělit nelineární multidimenzionální prostor, který bude určen vstupními daty, které se sestávají z množin atributů určeného typu deskriptoru. Rovněž vybraná metoda musí být schopna adaptace na nové trénovací vzory i třídy. Zvolená metoda musí mít co nejrychlejší proces vybavování, a to i na důraz vyšší časové i výkonové náročnosti procesu učení, jelikož systém, do kterého se tato metoda výsledně implementuje bude mít prostor pro delší přepočty v pozadí, avšak potřebuje velmi rychlé zjištění odezvy na neznámý obraz nejlépe blížící se zpracování v reálném čase.

2.1 Rozhodovací stromy

Rozhodovací stromy (RS) jsou hierarchické nelineární systémy umožňující ukládání znalostí, jejich následnou extrakci nebo použití k analýze nových dat. Jsou primárně používány pro klasifikaci kvalitativních závislých proměnných na základě vstupních proměnných. Vyznačují se zejména hierarchickou strukturou, nelinearitou, výbornou čitelností, snadnou interpretovatelností a flexibilitou. V závislosti na topologii a typu rozhodovacího stromu se dokáží natrénovat na kvalitativní i kvantitativní data, topologie může být binární nebo vícevrstvá a výstupem mohou být opět kvalitativní nebo kvantitativní data. Rozhodovací strom (Obrázek 2-1) se skládá z kořenového uzlu, který je počátečním uzlem RS, uzlů, které rozdělují vstupní data na větve, které jsou spojnicemi dvou uzlů dané úrovně. Pokud je uzel konečným bodem větvení, jedná se o list, který při jeho dosažení klasifikuje vstupní záznam na danou výstupní třídu. Hloubka stromu určuje úroveň (hloubku) vnoření do hierarchické struktury rozhodovacího stromu. [1]



Obrázek 2-1 Základní schéma rozhodovacího stromu [1]

Trénovací množinou je soubor záznamů, který obsahuje trénovací data (záznamy), které nesou informace o vstupních hodnotách atributů a také

o požadované výstupní třídě. Při vytváření rozhodovacího stromu se hledá způsob, jak stanovit co nejvhodnější podmínku, aby se po rozdělení v uzlu touto podmínkou dosáhlo co největšího informačního zisku. Podmínky mohou být navrženy na základě hodnoty jednoho nebo více atributů a mohou dělit strom do dvou nebo více větví v závislosti na binárním nebo vícerozměrném typu stromu. [1]

Existují algoritmy generování rozhodovacích stromů, které se zejména liší tím, jaké typy stromů z pohledu topologie, typů vstupních a výstupních proměnných generují. Mezi takovéto známé algoritmy patří ID3, CART nebo CHAID.

Rozhodovací stromy excelují v úlohách s malým počtem vstupních proměnných a s pestrým zastoupením kvalitativních nebo kvantitativních hodnot, avšak jsou nedostatečné pro úlohy s velkým počtem vstupních atributů, které jsou všechny navíc v podobném rozsahu kvantitativních hodnot.

2.2 Metody učení založené na instancích

Učení založené na instancích, Instance Based Learning (IBL), jsou metody schopné nalezení podobnosti neznámého vzoru se vzorem známým. Metody IBL se řadí mezi líné metody učení, které v procesu učení nevytváří žádný model a pouze ukládají trénovací data, která jsou složena z atributů a informace, do které klasifikační třídy patří. U základního algoritmu k-NN, metody *k* nejbližších sousedů (k nearest neighbor), jde tedy o učení s učitelem. Výpočet predikce se provádí až ve chvíli, kdy je předložen nový neznámý prvek ke klasifikaci. Klasifikace probíhá nalezením *k* nejbližších sousedů v *M* dimenzionálním prostoru a prostým průměrováním jejich klasifikačních tříd (rovnice 1). V případě kvantitativní veličiny se výstupní veličina vypočítá jako průměr *k* sousedů (rovnice 2). Důležitým parametrem je výběr metriky vzdálenosti mezi sousedy, kde u k-NN je často využívaná metrika Euklidovské vzdálenosti. Metoda nejbližších sousedů má inkrementální charakter a je tak jednoduchá adaptace na nový vzor nebo i novou třídu. Nevýhodou tohoto algoritmu je výpočetní časová náročnost při velkým počtu trénovacích prvků a velká citlivost na šum v trénovacích datech při nízkém k, kde s rostoucím k se zvyšuje náročnost na výpočet. [2]

$$f(x_q) = \underset{v \in V}{argmax} \sum_{i=1}^{k} \partial(v, f(x_i)), \tag{1}$$

kde v je vzdálenost náležící do maximální vzdálenosti V a k je počet sousedů.

$$f(x_q) = \frac{\sum_{i=1}^{k} f(x_i)}{k}, \tag{2}$$

kde k je počet sousedů a $f(xi)$ je kvantitativní hodnota příslušnosti.

Existují algoritmy pro výběr a uložení dat, které jednak zrychlují proces klasifikace, ale i zvyšují míru dosažené generalizace. Tyto metody se nazývají IB1

až IB4. Algoritmus učení IB1 pouze ukládá všechny trénovací instance, čímž vzniká vysoká náročnost na paměť a s rostoucím počtem trénovacích dat roste výpočetní náročnost predikce neznámého prvku, nijak tedy učení nemodifikuje. Algoritmus IB2 je modifikací IB1 a ukládá při procesu učení pouze ta trénovací data, která byla předchozími uloženými trénovacími daty špatně klasifikována, ostatní data ihned zahazuje. Výhodou IB2 je snížená náročnost na paměť a tím i snížený nárok na výpočetní výkon při klasifikaci, avšak je velice náchylná na šum v trénovacích datech, které jsou kvůli principu této metody vždy uloženy jako chybně klasifikované. Metoda učení IB3 modifikuje IB2 tím způsobem, že uložená data dělí do tří skupin: akceptované, zamítnuté a pozorované. Definuje pro ně dva prahy důvěryhodnosti α_1 a α_2, které tyto tři oblasti určují. Pokud je prvek mezi těmito dvěma prahy, také je ve stavu pozorován a je analyzováno, kolikrát predikoval dobře a kolikrát špatně, to se následně analyzuje statistickým významovým testem. Tímto algoritmem se zvyšuje robustnost vůči šumu, jelikož prvek není hned uložen při špatné klasifikaci, avšak IB3 je necitlivá na výjimky a pohlíží na ně jako na instance obsahující šum. Algoritmus IB4 rozšiřuje IB3 o relevanci jednotlivých atributů v učících datech a určuje tak jejich závažnost při klasifikaci. Tento algoritmus je nejúčinnější v úlohách s vysokým počtem atributů, které se následně často ukládají do *k-d* stromových struktur, které zrychlují proces nalezení nejbližších sousedů. Tato metoda může být však časově velmi náročná, jelikož se při klasifikaci vždy vytváří nový lokální model. [2]

Další metodou IBL je váhová metoda k nejbližších sousedů. Ta vychází z předpokladu, že vzdálenější sousedi od zkoumané instance mají mít menší vliv na klasifikaci než sousedi nejbližší. Váhová funkce (rovnice 3) je převrácená hodnota čtvereční vzdálenosti:

$$w_i = \frac{1}{d(x_q, x_i)^2}\ , \tag{3}$$

kde x_q je souřadnice nové instance v prostoru knn modelu a x_i je souřadnice i-tého souseda. V tomto modifikovaném algoritmu se následně upraví vztah pro výpočet klasifikace (rovnice 1) na:

$$f(x_q) = \underset{v \in V}{argmax} \sum_{i=1}^{k} w_i \cdot \partial(v, f(x_i))\,, \tag{4}$$

kde w_i je váha aktuální souseda vůči nové instanci. Vztah pro výpočet kvantitativního výstupu klasifikátoru se rovněž změní dle rovnice (5).

$$f(x_q) = \frac{\sum_{i=1}^{k} w_i \cdot f(x_i)}{k \sum_{i=1}^{k} w_i} \tag{5}$$

Nejčastějším problémem metod učení založeném na instancí je stanovení vhodného počtu sousedů. Jednou z možností je využití metody křížové validace, kde jsou data

rozdělena do N částí a vždy jedna je použitá jako klasifikační a zbytek jako trénovací. Vhodný počet k sousedů se získá prostým postupným zkoušením různých hodnot k a následnou analýzou výsledků. Tento přístup může být výpočetně i časově velice náročný v závislosti na počtu vzorů celé trénovací množiny. Optimální počet sousedů je zásadní části návrhu IBL klasifikátoru. Při nízkém k je metoda citlivá na šum a při vysokém k roste výpočetní náročnost. Speciálním případem volby parametru k je $k = 1$ a nazývá se Voroného graf, kde dochází k rozdělení plochy na polygony, které určují oblasti jednotlivých tříd, do kterých je neznámý prvek klasifikován. [2]

2.3 SVM – Metoda Podpůrných vektorů

Metoda podpůrných vektorů, nebo také support vector machines (SVM), je metoda nalezení optimální separující nadroviny dvou lineárně separabilních tříd. Optimální separace dvou tříd je dosáhnuto maximalizováním šířky prázdné oblasti (anglicky *margin*) mezi dvěma třídami v prostoru příznaků. Šířka této oblasti je definovaná jako vzdálenost mezi diskriminační nadrovinou v n dimenzionálním prostoru příznaků a nejbližším trénovacím vzorem, které se právě nazývají podpůrné vektory a určují samotnou diskriminační funkci. Silná stránka SVM spočívá ve schopnosti nalezení optimální separující nadroviny mezi dvěma třídami i v situaci, kdy těchto separujících nadrovin existuje velké množství, čímž i zamezuje přeučení modelu. K nalezení maximální šířky oblasti mezi dvěma třídami se musí minimalizovat proměnná $||w||$ z rovnice (6), pro tuto optimalizaci se používá kvadratické programování.

$$\omega_i(\mathrm{w} \cdot \mathrm{x_i} + \mathrm{b}) \geq 1, \tag{6}$$

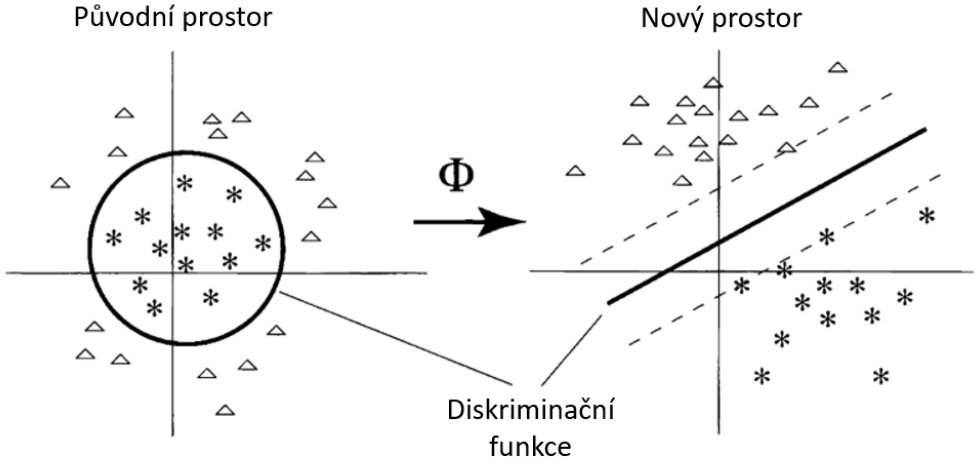
kde ω_i je klasifikátor třídy, w lineární parametr přímky, b je posun přímky a x_i je hodnota vstupního atributu. Výsledkem nalezení optimální separující nadroviny je vektor nenulových parametrů α, které slouží jako váhy a určují podpůrné vektory. Získáme je maximalizací účelové funkce L dle rovnice (7) jako

$$L(w, b, \alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \cdot \alpha_j \cdot \omega_i \cdot \omega_j \cdot (x_i \cdot x_j), \tag{7}$$

kde w je lineání parametr rovnice přímky, b je posuv rovnice přímky, ω je výstupní klasifikační třída a N je počet nenulových parametrů α. Rovnice (7) platí za předpokladu $\alpha \geq 0$ a $\sum_{i=1}^{N} \alpha_i \cdot \omega_i = 0$. [3]

Pro účely této úlohy je však důležitá separace nelineárně separabilních tříd. Aby SVM klasifikátor mohl být natrénován na data, která nejsou lineárně separující, využívá se tzv. *soft margin*, která povoluje i vzorky v opačné klasifikační oblasti. V lineární oblasti se jádrová funkce, která vyjadřuje podobnost dvou vzorů, určí skalárním součinem obou vzorů $k(x_i, x_j) = (x_i \cdot x_j)$. Lineární jádrová funkce je

pro nelineární úlohy nahrazena nelineární jádrovou funkcí $k(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$. Tímto se vytvoří nový nelineárně transformovaný prostor, ve kterém již lze zjistit lineární separující nadrovinu, která odpovídá nelineární dělící nadrovině v původním prostoru (Obrázek 2-2). [3]



Obrázek 2-2 Transformace původního prostoru do nového prostoru příznaků [3]

Nevýhoda SVM spočívá v jeho výhodě, kdy je schopen převést nelineárně separabilní problém na lineárně separabilní. K tomu je žádoucí vytvoření celé řady nových atributů a s tím související rozšíření dimenze nového prostoru, které je z tohoto důvodu časově velice náročné. Nejedná se totiž pouze o nalezení jedné transformace, ale o nalezení více transformací a vybrání té nejvhodnější. Řešení spočívá v tzv. jádrovém triku ve vzorci (8), kde se skalární součin $(x_i \cdot x)$ nahradí jádrovou funkcí v novém příznakovém prostoru $k \cdot (x_i, x)$, viz rovnice (9).

$$f(x) = \sum_{i \in SV} \alpha_i \cdot \omega_i (x_i \cdot x) + b, \tag{8}$$

kde α_i je váha pomocných vektorů a ω_i je výstupní třída vzorku.

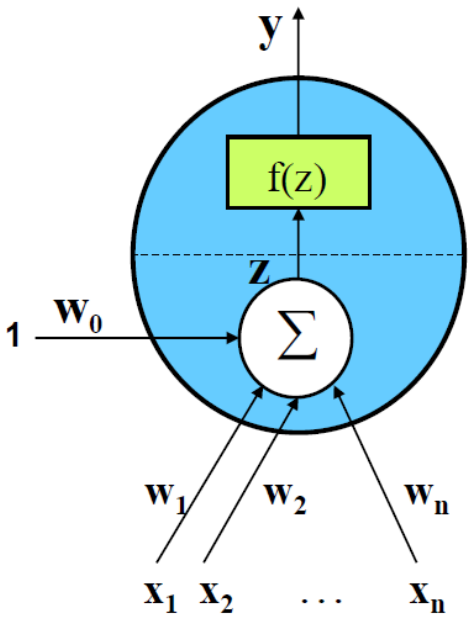
$$f(x) = \sum_{i \in SV} \alpha_i \cdot \omega_i \cdot k(x_i, x) + b, \tag{9}$$

kde $k(x_i, x)$ je jádrová funkce z nového příznakového prostoru. Existuje několik předdefinovaných jádrových funkcí a překvapivě její výběr z nich nepředstavuje problém, jako je tomu u jiných metod. Jakákoliv jádrové funkce z definovaných totiž vždy dává rozumné výsledky a volbou jiné lze docílit jen zpřesněných výsledků. [3]

Klasifikace více než dvou tříd je docíleno kombinováním N binárních klasifikátorů rozlišujících pouze dvě třídy, jednu specifickou třídu a druhou zastupující všechny ostatní třídy. [3]

2.4 Neuronové sítě

Neuronové sítě spadají do oblasti umělé inteligence a jsou obecně velmi silným nástrojem pro řešení klasifikačních úloh. Neuronové sítě se mezi sebou rozlišují převážně čtyřmi faktory, typem použitých neuronů, topologií, způsobem učení a vybavování. Neuron (Obrázek 2-3) se typicky skládá ze dvou částí, lineární a nelineární části. V lineární části se provádí nad vstupy a váhami jednotlivých vstupů matematická operace specifická pro konkrétní typ neuronové sítě. Vstupy neuronu jsou na obrázku označeny x_1 až x_n a nejčastěji bývají reálná čísla. Synaptické váhy jednotlivých vstupů zastupují znaky w_1 až w_n a práh neuronu w_0. Vnitřní potenciál neuronu je označován jako z a slouží zároveň jako vstup pro nelineární část neuronu. Ta je tvořena přenosovou funkcí neuronu, která v závislosti na typu neuronové sítě může mít diskrétní nebo spojitý výstup. Diskrétní výstupy zajišťují přenosové funkce unipolární a bipolární, spojité například sigmoida nebo hyperbolická tangenta. [5]

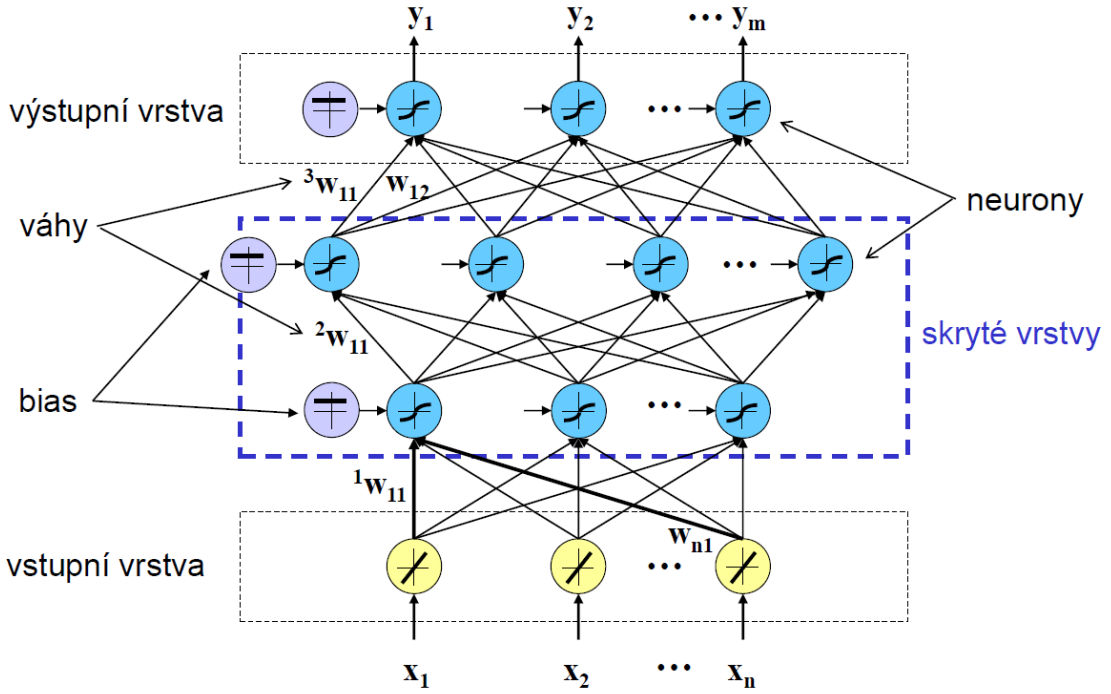


Obrázek 2-3 Schéma základního neuronu [5]

Základním typem neuronové sítě je perceptron. Perceptron je jednovrstvá přímovazební neuronová síť s nespojitou přenosovou funkcí, iterativním učením a jednorázovým vybavováním. Neuronová síť typu perceptron je schopna řešit pouze lineárně separabilní tréninkové množiny, a proto není vhodná pro tuto úlohu.

2.5 Vícevrstvá neuronová síť s algoritmem učení error back propagation

Nejužívanějším typem neuronové sítě je vícevrstvá neuronová síť s algoritmem učení se zpětným šířením chyby (error backpropagation algorithm, dále jen BP). Jedná se o vícevrstvou přímovazební síť neuronů se spojitou přenosovou funkcí. Učení je s učitelem a jedná se o iterační proces nastavování vah. Vybavování probíhá jednorázově. Topologie této sítě se sestává z minimálně tří vrstev neuronů – vstupní distribuční vrstvy, nejméně jedné skryté vrstvy a výstupní vrstvy (Obrázek 2-4).



Obrázek 2-4 Vícevrstvá neuronová síť [6]

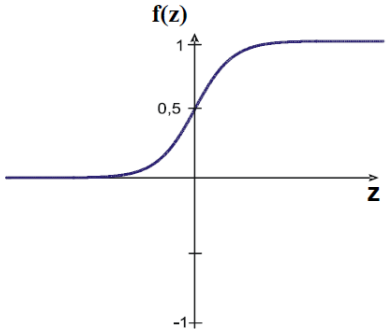
Každý neuron jedné vrstvy je spojen se všemi neurony vrstvy následující, tento typ zapojení neuronů se nazývá úplné propojení neuronů. U této sítě s rostoucím počtem neuronů ve skrytých vrstvách, popřípadě s počtem skrytých vrstev, se zvyšuje nelinearita chování sítě, což vede k řešení složitějších nelineárních závislostí vstupních atributů. Ovšem se zvyšujícím se počtem neuronů a vrstev se rovněž zvyšuje i výpočetní náročnost procesu učení, hlavně tedy z časového hlediska. [4]

Neurony ve vstupní vrstvě slouží pouze pro distribuci signálu, jejich přenosová funkce je tedy lineární a výstupní hodnota odpovídá hodnotě vstupní. V dalších vrstvách je struktura stejná jako na předešlém obrázku a mění se pouze způsob výpočtu vnitřního potenciálu z a typ přenosové funkce $f(z)$. Vnitřní potenciál se vypočítá jako suma násobků synaptických vah w_j a příslušných vstupů x a přičtením hodnoty prahu neuronu (rovnice 10)

$$y_j = f(w_{0j} + \sum_{i=1}^{N} x_{ij} \cdot w_{ij}), \tag{10}$$

kde j je index neuronu, i je index vstupu j-tého neuronu a N je počet vstupů do neuronu. Přenosová funkce u vícevrstvých neuronových sítí se často volí jako sigmoida nebo hyperbolický tangens. [4]

Sigmoida (Obrázek 2-5) je matematická nelineární funkce, která má výstupní hodnotu z intervalu 0 až 1 a má střed v hodnotě 0,5. Sigmoida je reálná funkce, která je definovaná pro všechny reálné vstupní hodnoty a v každém bodě na X ose je derivovatelná s kladným výsledkem. [4]



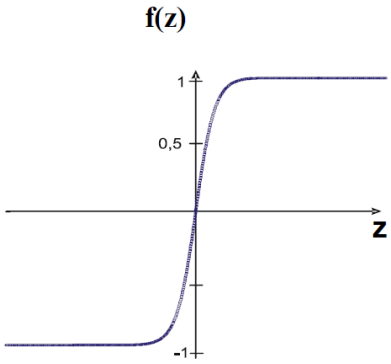
Obrázek 2-5 Přenosová funkce sigmoidy [6]

Výstupní hodnota sigmoidu se spočte podle rovnice (11) jako

$$f(z) = \frac{1}{1 + e^{-\sigma \cdot z}}, \tag{11}$$

kde σ určuje strmost funkce a z je vstup funkce.

Druhou používanou výstupní funkcí je nelineární funkce hyperbolická tangenta (Obrázek 2-6). Od sigmoidy se liší v intervalu výstupních hodnot, které nabývají od - 1 do 1, a v tom, že střed funkce hyperbolické tangenty je v nule. Tato funkce se u vícevrstvých neuronových sítí používá častěji, jelikož lépe rozděluje výstupní prostor.



Obrázek 2-6 Přenosová funkce hyperbolické tangenty [6]

Výstupní hodnota hyperbolické tangenty se spočte podle rovnice (12) jako

$$f(z) = \frac{1 - e^{-\sigma \cdot z}}{1 + e^{-\sigma \cdot z}}\ , \tag{12}$$

kde σ určuje strmost funkce a z je vstup funkce. [4]

Učení vícevrstvé neuronové sítě s algoritmem zpětného šíření chyby je definováno jako iterativní gradientní algoritmus učení, který minimalizuje čtverce chybové funkce. Pro učení musí být předloženy trénovací množiny podle tréninkového vzoru. Každý tréninkový vzor $[{}^{S}X\,{}^{S}D]$, kde S je počet vzorů, se sestává ze vstupního vektoru X a vektoru žádaných výstupů D. Velikost vektoru X *se* odvíjí od počtu neuronů vstupní vrstvy a velikost vektoru D od počtu neuronů ve výstupní vrstvě. Tyto tréninkové množiny tedy definují, jaké hodnoty mají být na výstupu při nastavení daných hodnot vstupů. To je vlastně princip neuronové sítě, která se snaží upravovat váhy jednotlivých neuronů ve skrytých vrstvách, aby co nejlépe klasifikovala všechny vstupní učící data dle jejich požadovaných výstupů. Existují doporučené požadavky na tréninkovou množinu, kde data pro jednotlivé výstupní třídy by měly být reprezentativní, jednotlivé třídy by měly být početně ekvivalentně zastoupeny (pokud to není dáno aplikačně jinak) a neměly by obsahovat pouze etalonové hodnoty, ale i zašuměné nebo nejlépe hodnoty přímo z prostředí, ze kterého se budou následná reálná data pro aktivaci získávat. V procesu učení se iterativně přes všechny vstupní množiny počítá výstup celé sítě a z něho chyba vzhledem k předloženému vzoru S podle rovnice (13) jako:

$${}^{S}E = \frac{1}{2}\sum_{j=1}^{m}({}^{S}y_j - {}^{S}d_j)^2, \tag{13}$$

kde y_j je výstup j-tého neuronu ve výstupní vrstvě, d_j je požadovaná hodnota j-tého výstupního neuronu, m je počet výstupních neuronů. Z těchto chyb se po předložením všech tréninkových vzorů počítá chyba sítě E_c, která udává míru naučenosti, tedy odchylky mezi skutečnými a požadovanými hodnotami výstupů neuronové sítě pro danou tréninkovou množinu. Celková chyba sítě je tedy definovaná podle vzorce (14) jako suma chyb sítě ${}^{S}E$ vzhledem k jednotlivým tréninkovým vzorům:

$$E_c = \sum_{s=1}^{p} {}^{S}E = \frac{1}{2}\sum_{s=1}^{p}\sum_{j=1}^{m}({}^{S}y_j - {}^{S}d_j)^2, \tag{14}$$

kde p je počet tréninkových vzorů. Hodnota jedné poloviny před sumami se uvádí z důvodu jednoduššího výpočtu derivace vnitřní funkce, po které se provede krácení a ruční výpočet je snazší.

V závislosti na celkové chybě sítě se adaptují váhy mezi výstupní a první skrytou vrstvou (z pohledu výstupu) dle rovnice (15)

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}\ , \tag{15}$$

kde t je aktuální krok, w_{jk} aktuální synaptické váhy a Δw_{jk}je adaptace váhy. Adaptace váhy se vypočte podle vztahu (16) jako

$$\Delta w_{jk} = -\propto \cdot \frac{\partial E_c}{\partial w_{jk}} = -\propto \cdot \sum_{s=1}^{p} \frac{\partial {}^{s}E}{\partial w_{jk}}\ , \tag{16}$$

kde α je koeficient učení, E_c je celková chyba sítě, ${}^{s}E$je chyba sítě vůči předloženému tréninkovému vzoru a w_{jk} je synaptická váha propojení *j*-tého neuronu v předešlé vrstvě a *k*-tého neuronu v aktuální vrstvě. Parciální derivace chyby sítě vůči předloženému vzoru ${}^{s}E$ podle synaptické váhy w_{jk} je definována dle vzorce (17) jako

$$\frac{\partial {}^{s}E}{\partial w_{jk}} = \frac{\partial {}^{s}E}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_{jk}}, \tag{17}$$

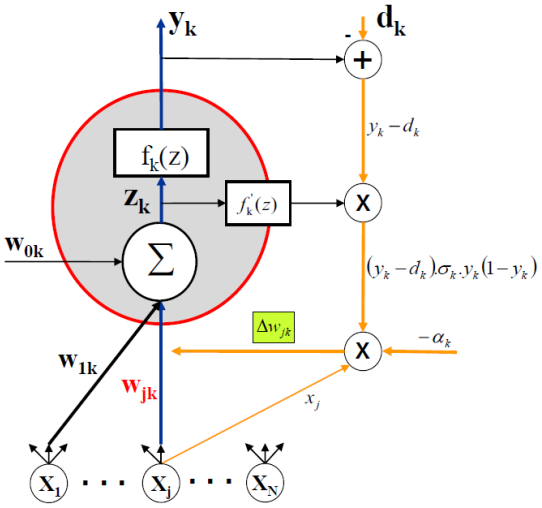
kde y_k je výstup *k*-tého neuronu a z_k je hodnota vnitřního potenciálu neuronu. První parciální derivace chyby sítě podle výstupu y_k se vypočítá jako rozdíl výstupu *k*-tého neuronu a požadovaná hodnota výstupu d_k. Druhá parciální derivace je parciální derivace přenosové funkce a spočte podle vzorce (18) jako

$$\frac{\partial y_k}{\partial z_k} = \sigma_k \cdot y_k \cdot (1 - y_k), \tag{18}$$

kde σ je strmost přenosové funkce a y_k je výstup *k*-tého neuronu. Poslední parciální derivace vnitřního potenciálu neuronu z_k podle synaptických vah w_{jk} je rovna x_j, které odpovídá hodnotě vstupu *j*-tého neuronu. Po dosazení všech parciálních derivací vyjde vzorec (19) pro výpočet chyby sítě pro adaptaci vah.

$$\frac{\partial {}^{s}E}{\partial w_{jk}} = (y_k - d_k) \cdot \sigma_k \cdot y_k \cdot (1 - y_k) \cdot x_j \tag{19}$$

Výpočet chyby mezi výstupní vrstvou a první skrytou vrstvou rovněž ilustruje Obrázek 2-7, kde je nejdříve proveden rozdíl výstupu y_k s požadovanou hodnotou výstupu d_k. Následně je výsledek vynásoben derivací přenosové funkce neuronu a v posledním kroku je vynásoben ještě vstupem x_j a koeficientem učení α.



Obrázek 2-7 Princip výpočtu algoritmu zpětného šíření chyby [6]

Po adaptaci synaptických vah mezi výstupní a první skrytou vrstvou je nutné provést přepočet vah i mezi skrytými vrstvami. Výpočet chyby mezi první a druhou skrytou vrstvou se provede podobně jako v rovnici (19), kde místo vstupu $x_j j$ se použije hodnota synaptické váhy w_{jk} na vstupu neuronu horní vrstvy. Adaptace vah se pak vypočítá podle vzorce (20) jako

$$\Delta w_{ij} = -\propto \cdot \sum_{l=1}^{p} \frac{\partial E_l}{\partial w_{ij}} \tag{20}$$

a následně se jednotlivé parciální derivace spočtou dle vzorce (21)

$$\frac{\partial E_l}{\partial w_{ij}} = \left(x_j - d_j\right) \cdot \sigma_j \cdot x_j \cdot \left(1 - x_j\right) \cdot s_i \cdot \sum_{k=1}^{m} (y_k - d_k) \cdot \sigma_k \cdot y_k \cdot (1 - y_k) \cdot w_{jk}, \tag{21}$$

kde proměnné s indexem j jsou vůči druhé skryté vrstvě a proměnné s indexem k jsou vzpjaty k první skryté vrstvě, s_i je i-tý vstup neuronu a m je počet neuronů v horní vrstvě. V případě dalších skrytých vrstev se provádí výpočet analogicky. Adaptace vah může probíhat dvěma způsoby, buď jako akumulované učení nebo adaptací po každém tréninkovém vzoru. Popsaný princip výše popisuje akumulované učení, kde se nejdříve předložení všechny tréninkové vzory, spočítá se celková chyba sítě E_c a následně se určí Δw_{jk} pro každý neuron. U učení s adaptací vah po každém tréninkovém vzoru se úprava synaptických vah provádí po každém

trénovacím vzoru, což zpomaluje konvergenci a také záleží na pořadí vzorů v trénovací množině. Učení u vícevrstvé neuronové sítě s algoritmem zpětného šíření chyb tedy probíhá iterativně, a to následným algoritmem:

1. Inicializace
2. Předložení tréninkového vzoru
3. Výpočet výstupu sítě
4. Výpočet chyby sítě E_S
5. Test na tréninkovou množinu
6. Výpočet chyby E_C
7. Test na ukončení
8. Adaptace vah
9. Iterace t+1

Při inicializaci se nastavují počáteční hodnoty synaptických vah, biasů, koeficientu učení, nastavení strmosti přenosových funkcí a nastavení maximální chyby jakožto podmínky pro ukončení. V dalších dvou krocích se předkládá aktuální tréninkový vzor a probíhá výpočet výstupu sítě. Po získání výstupu sítě, tedy vektoru výstupních hodnot, se vypočítá chyba sítě E_S. Testem na tréninkovou množinu je myšlena podmínka, která pokud nebyly předloženy ještě všechny trénovací vzory, tak skočí v algoritmu zpět a předloží následující trénovací vzor. Pokud byly však již všechny trénovací vzory předloženy přechází na další krok, a to výpočet celkové chyby sítě E_C podle vztahu (14). Algoritmus učení se zastaví, pokud vypočtená chyba E_C je menší než zvolená hodnota při inicializaci. Pokud je chyba větší, přistoupí se k adaptaci vah a započne nová iterace algoritmu. [4], [7], [8]

Učící algoritmus vždy konverguje, jelikož většinou obsahuje více ukončovacích podmínek, jako maximální počet iterací, nebo ukončení po N po sobě jdoucích správně klasifikovaných validací.

Vybavování je jednokrokový proces výpočtu výstupu sítě dle vzorce (10). Jedná se pouze o postupné násobení matic synaptických vah se vstupy a zjištění odezvy ve výstupní vrstvě.

2.6 Výběr vhodné metody

Pro souhrn základních vlastností probraných metod byla sestavena tabulka (Tabulka 1), která srovnává požadované vlastnosti na metodu strojového učení. Soubor probraných metod byl sestaven vzhledem k řešenému problému.

Tabulka 1 Srovnání metod strojového učení z pohledu zadané úlohy. RS – rozhodovací stromy, IBL – instrance based learning, SVM – support vector machines, NS s BP – neuronové sítě s algoritmem error back propagation

Metoda	Typ vstupu	Učení	Vybavování	Vhodné pro velký počet atributů (128+)
RS	Kvantitativní, kvalitativní	Rychlé, iterační	Rychlé, odvoditelné	Ne
IBL	Kvantitativní, kvalitativní	Rychlé, „líné“	Pomalé	Ano (IB4)
SVM	Kvantitativní	Velmi pomalé	Velmi rychlé	Ano
NS s BP	Kvantitativní	Pomalé, iterační	Rychlé, neodvoditelné	Ano

První z metod, a to rozhodovací stromy, ač jsou velmi rychlé jak v učení, tak v procesu vybavování, nejsou však příliš vhodné pro typ vstupních atributů, kde se bude jednat například o reálná čísla SURF deskriptoru, která jsou všechna v podobných intervalech. Rovněž je problémem dimenze vstupního vektoru, která je pro rozhodovací stromy problémová.

Druhá z metod, učení založené na instancích, by mohla být vhodným kandidátem, obzvláště s algoritmem učení IB4. Metoda IBL není limitována rozsahem hodnot vstupních atributů ani dimenzí vektoru atributů. Nevhodnost této metody však spočívá v časově náročném procesu vybavování. Pro tuto konkrétní úlohu není problémem dlouhý proces učení, ale důraz je kladen na rychlost procesu vybavování, které je u metody IBL pomalé.

Metoda podpůrných vektorů SVM je rovněž velice vhodná pro tuto úlohu. Dokáže precizně rozdělit příznakový prostor i pro velký počet výstupních tříd v mnohodimenzionálním prostoru. Výhodou této metody je v porovnání s ostatními zmíněnými metodami velmi rychlý proces vybavování.

Poslední ze zmíněných metod je neuronová síť s algoritmem učení BP. Tato konkrétní síť je schopna se naučit nelineární závislosti mezi vstupními atributy a lépe tak popsat nelineární příznakový prostor. Proces vybavování je dostatečně rychlý, avšak nelze zpětně určit z jakého důvodu se neuronová síť tak či onak rozhodla. Neuronová síť tohoto typu je vhodná pro zvolený typ vstupních dat i dimenzi vektoru atributů.

Metody SVM a NS s BP jsou si vhodností téměř ekvivalentní, nicméně proces učení u SVM je při zvolených vstupních datech podstatně časově náročnější. Co se přesnosti predikce týče, tak jsou si algoritmy téměř rovny. Nevýhodou SVM je složitější a časově náročnější adaptace na neznámý vzorek známé třídy i na neznámý vzorek ze třídy neznámé. Pro účely této úlohy jsou obě tyto metody vhodné a budou sloužit jako slabé klasifikátory, které v kombinaci utvoří silný klasifikátor.

3 ALGORITMY PRO EXTRAKCI VSTUPNÍCH DAT

Tato kapitola popisuje různé deskriptory významných bodů v obraze, jmenovitě SIFT, SURF a FREAK. Rovněž udává do kontextu princip slovníku vizuálních slov.

3.1 Scale Invariant Features Transform

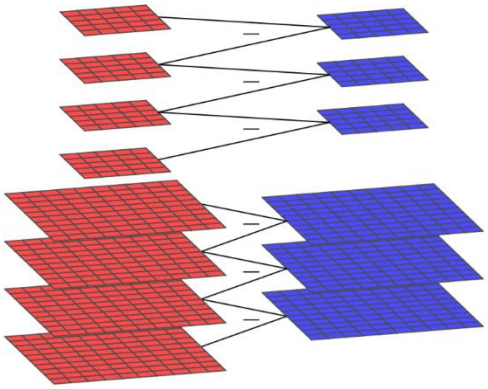
Scale Invariant Features Transform neboli SIFT je algoritmus pro detekci a popis významných bodů v obraze předložený Davidem Lowem, který je invariantní na rotaci a změnu měřítka. Pro nalezení stejných příznakových vektorů pro různé pohledy na stejný objekt slouží tzv. scale space (SP) s funkcí

$$\mathrm{L}(\mathrm{x},\mathrm{y},\sigma) = \mathrm{G}(\mathrm{x},\mathrm{y},\sigma) * \mathrm{I}(\mathrm{x},\mathrm{y}), \tag{22}$$

kde $G(x, y, \sigma)$ je Gaussova funkce a $I(x, y)$ je intenzita pixelu v bodě (x, y) obrazu I. Pro efektivní nalezení stabilního scale space extrému se poté používá konvoluce rozdílů Gaussianů

$$D(x, y, \sigma) = G(x, y, k\sigma) - G(x, y, \sigma), \tag{23}$$

kde k je faktor rozdílů měřítek mezi dvěma sousedními měřítky ve scale space. Jedná se o dostatečně blízkou aproximaci funkce Laplaceova transformace Gaussianu $\sigma^2 \cdot \nabla^2 \cdot G$, která oproti jiným funkcím představuje nejstabilnější obrazové příznaky. SP je následně rozdělen do několika oktáv. Při přechodu do vyšší oktávy se vybírá každý druhý pixel v řádku i sloupci, což ve výsledku redukuje velikost obrazu na čtvrtinu. Stabilní extrém SP se nalezne určením hodnoty funkce $L(x, y, \sigma)$ všech zkoumaných prostorů a odečtením sousedních obrazů ve scale space, jež utvoří skupinu možných kandidátů (Obrázek 3-1). Každý takovýto kandidát má devět sousedů následujícího a předchozího měřítka (úrovně scale space) a osm sousedů stejného měřítka. [9]



Obrázek 3-1 Výpočet rozdílových snímků ve všech oktávách utvářející difference of Gaussian (DoG). [9]

Kandidát je považován za extrém, pokud je menší nebo větší než hodnoty všech jeho 26 sousedů, nicméně pozice takto určeného extrému je pouhou aproximací. Taylorovým rozvojem funkce $D(x, y, \sigma)$ s počátkem v bodu nalezeného kandidáta a substitucí (x, y, σ) jako $\boldsymbol{x}$ dostaneme rovnici

$$D(\boldsymbol{x}) = D + \frac{\partial D^T}{\partial \boldsymbol{x}}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\frac{\partial^2 D}{\partial \boldsymbol{x}^2}\boldsymbol{x}, \tag{24}$$

kde $\boldsymbol{x} = (x, y, \sigma)$. Lepší odhad extrému $\hat{\boldsymbol{x}}$ se dosáhne diferenční rovnicí rovnice (24) a položením nule $\frac{\partial D}{\partial \boldsymbol{x}} + \frac{\partial^2 D^T}{\partial \boldsymbol{x}^2}\hat{\boldsymbol{x}} = 0$ a upravením do tvaru $-\frac{\partial D}{\partial \boldsymbol{x}}\frac{\partial^2 D}{\partial \boldsymbol{x}}^{-1} = \hat{\boldsymbol{x}}$ a dosazením do rovnice (24) dostaneme

$$D(\hat{\boldsymbol{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \boldsymbol{x}}\hat{\boldsymbol{x}}, \tag{25}$$

kde extrémy s malou hodnotou $D(\hat{\boldsymbol{x}})$ jsou považované za nestabilní a jsou odstraněny. Rovněž jsou odstraněny extrémy s hodnotou pod definovaným prahem $\tau, D(\hat{\boldsymbol{x}}) < \tau$. [9], [11], [12]

Poslední filtrací je odstranění hranových bodů, jelikož se za stabilní považují pouze body rohové. To se uskutečňuje výpočtem Hessovy matice z hodnot obrazu $DoG(x, y, \sigma)$ a pro $2D$ funkci platí

$$H\,f(x, y) = \begin{vmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x\,\partial y} \\ \frac{\partial^2 f}{\partial y\,\partial x} & \frac{\partial^2 f}{\partial y^2} \end{vmatrix} \tag{26}$$

a pro potlačení jiných, než rohových lokálních extrémů v obrazech DoG je stanoveno kritérium poměru vlastních čísel matice H

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}, \tag{27}$$

kde $Tr(H)$ je stopa Hessovy matice neboli součet prvků na hlavní diagonále a $Det(H)$ je determinant Hessovy matice. [9], [10]

Pro každého kandidáta v prostoru $L(x, y, \sigma)$, který splňuje výše popsané podmínky, se spočítá jeho reprezentace sestávající se z amplitudy gradientu $m(x, y)$ a směru tohoto gradientu $\theta(x, y)$

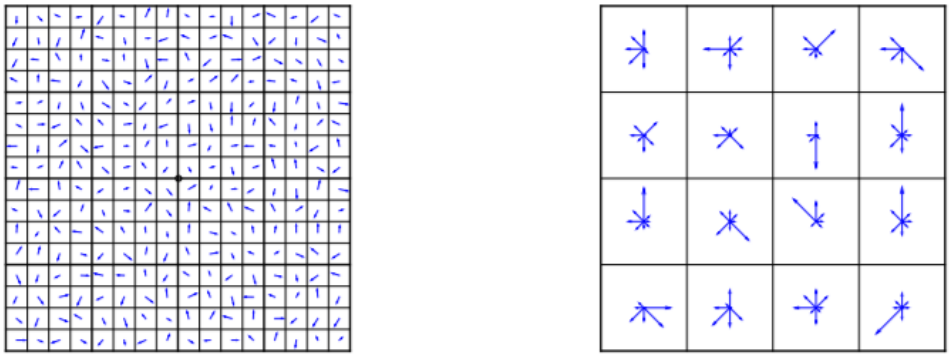
$$m(x, y) = \sqrt[2]{\big(L(x+1, y, \sigma) - L(x-1, y, \sigma)\big)^2 + \big(L(x, y+1, \sigma) - L(x, y-1, \sigma)\big)^2} \tag{28}$$

$$\theta(x, y) = arctan\left(\frac{L(x, y+1, \sigma) - L(x, y-1, \sigma)}{L(x+1, y, \sigma) - L(x-1, y, \sigma)}\right) \tag{29}$$

Následně se pro definované okolí extrému vypočítají směry gradientů θ_1 až θ_n a vytvoří se histogram s 36 kategoriemi po 10 stupních a každý jeho prvek je vážen příslušnou amplitudou gradientu a Gaussovým oknem umístěným na střed

vyšetřovaného bodu. Gaussovo okno slouží k preferenci bodů, které jsou blíže zkoumanému bodu. Kategorie histogramu s nejpočetnější reprezentací je vybrána jako výsledná orientace zkoumaného bodu. Pokud žádná kategorie histogramu gradientů není dominantní, tak je kandidát vyřazen. Všechny nevyfiltrované body jsou výslednými detekovanými body SIFT algoritmu a jsou popsány svou pozicí (x, y), velikostí měřítka σ a směrem gradientu $\theta(x, y)$. [9]

Detekované body SIFT algoritmu jsou následně reprezentovány deskriptorem, který se tvoří za pomoci 16x16 mřížky (Obrázek 3-2 a) umístěné ve středu detekovaného SIFT bodu v obrazovém prostoru $L(x, y, \sigma)$. Následně se vypočítá velikost amplitudy a směr gradientu. Poté se mřížka rozdělí do 16 regionu (Obrázek 3-2 b), kde každý region obsahuje 16 pixelů z předchozí mřížky a je popsána orientovaným histogramem s 8 kategoriemi. Výsledný deskriptor tvoří jednotlivé orientované histogramy váhované amplitudou gradientu příslušné oblasti a Gaussovým oknem, což ve výsledku utváří vektor o 128 prvcích. Následnou normalizací SIFT deskriptoru na jednotkovou délku se dosáhne invariance na afinní změnu intenzity bodů v obraze. [9], [10]



Obrázek 3-2 a) Princip SIFT deskriptoru, který pro významný bod (x,y) definuje pixelové okolí 16x16 orientovaných gradientů. b) Transformování 256 okolních pixelů do 16 regionů obsahující histogramy orientovaných gradientů s 8 kategoriemi. [9]

3.2 Speeded Up Robust Features

Speeded Up Robust Features neboli SURF je detektor i deskriptor významných bodů založený na algoritmu SIFT a snaží se o snížení výpočetní náročnosti a zároveň zachování robustnosti celého algoritmu. Od algoritmu SIFT se liší pouze ve způsobu tvoření scale space, které probíhá změnou velikosti masky, a nikoliv celého obrazu, a výpočtem lokálních extrémů. Namísto DoG pro detekci významných bodů využívá SURF aproximaci Haarových vlnek, které významně snižují výpočetní náročnost

celého algoritmu. Tato aproximace, tzv. Box functions, používají integrální obraz $I_{\sum}$,se kterým je následný výpočet konvoluce $I(x) \cdot g''(\sigma)$ pouze o třech operací $\pm$ namísto násobení matic. Matice H je pak ve tvaru

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{30}$$

a stopu $Tr(H)$ a determinant $Det(H)$ získáme jako

$$Tr(H) = D_{xx} + D_{yy} = \lambda_\alpha + \lambda_\beta \tag{31}$$

$$Det(H) = D_{xx} \cdot D_{yy} - {D_{xy}}^2 = \lambda_\alpha \cdot \lambda_\beta, \tag{32}$$

kde λ_α je největší vlastní číslo a λ_β je nejmenší vlastní číslo. Výsledný poměr $Tr(H)^2$ a $Det(H)$ se určí dle rovnice

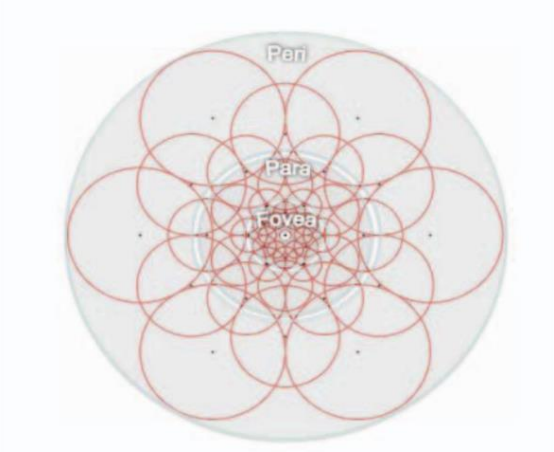
$$\frac{Tr(H)^2}{Det(H)} = \frac{(r+1)^2}{r}, \tag{33}$$

kde r je poměr λ_α a λ_β. Pro vyšetření kandidáta stačí tedy vyřešit $\frac{(r+1)^2}{r} < \tau$. [9]

Posledním důležitým rozdílem SURF oproti SIFT je ve fázi sestavování deskriptoru, kde SURF používá sumy a absolutní sumy derivací prvního řádu namísto histogramů orientovaných gradientů. [10]

3.3 Fast Retina Keypoint

Fast Retina Keypoint neboli FREAK je deskriptor významných bodů, který se snaží napodobit lidské vnímání detailů. V oblasti zájmu zkoumá větší množství detailů, tedy frekvence popisných bodů je vyšší a se zvětšující se vzdáleností od významného bodu se tato frekvence snižuje. Využívá k tomu topologii retina (Obrázek 3-3), která vytváří rovnoměrné rozložení bodů v soustředných kruzích se středem ve významném bodu. Na každý navzorkovaný bod je aplikovaná Gaussova maska, jehož velikost jádra je úměrná poloměru soustředných kružnic aktuálního navzorkovaného bodu. [13], [14]



Obrázek 3-3 Retina topologie rozložení bodů okolo významného bodu [13]

FREAK deskriptor tvoří binární vektor, jehož jeden prvek F se vypočítá jako

$$F = \sum_{0 \le a \le N} 2^a T(P_a) \tag{34}$$

$$T(P_a) = \begin{cases} 1, & I\left(P_a^{r_1}\right) - I\left(P_a^{r_2}\right) > 0 \\ 0, & jinak \end{cases}, \tag{35}$$

kde P_a je dvojice kružnic se středem v příslušném vzorkovacím bodě a o poloměru velikosti Gaussova jádra, N je požadovaná délka deskriptoru, $I\left(P_a^{r_1}\right)$ odpovídá intenzitě v první oblasti páru P_a. Jelikož je vzorkovací topologie FREAK deskriptoru kruhová a symetrická, tak se pozice i poloměr soustředné kružnice mění s měřítkem, a jelikož je výsledný binární vektor tvořen pokaždé jako rozdíl mezi páry jednotlivých kruhových oblastí s příslušnou velikosti Gaussova jádra, tak je FREAK deskriptor robustní vůči změně měřítka, rotace, intenzity osvětlení a šumu. [13], [14]

3.4 Bag of Visual Words

Bag of Visual Words neboli BOVW je slovník vizuálních slov, které frekvencí svého výskytu popisují daný obraz. Generování vizuálních slov probíhá po detekci významných bodů v obraze a extrahováním lokálních deskriptorů (Surf, Freak a jiné). Nalezené významné body, které mají podobné deskriptory, jsou shlukovány dostupnou metodou na bázi učení bez učitele. Nejčastější volbou je metoda K-Means, která příznakový prostor iterativně rozdělí do předem definovaného počtu shluků. Nejprve náhodně určí k bodů jakožto, následně se vypočítá vhodnou metodou vzdálenost každého bodu a určí se jeho nejbližší shluk, k němuž se přiřadí. Po přiřazení všech bodů se středy shluků přepočítají a všechny body opět hledají minimum funkce vzdálenosti ke všem shlukům. Algoritmus se zastaví, pokud žádný prvek nezmění za celou iteraci shluk. Středy konečných shluků pak tvoří vizuální slova. Velikost slovníku vizuálních slov se musí zvolit v závislosti na počtu obrazů a četnosti klasifikačních tříd. Poté, co je velikost slovníku určena, je každý vstupní obraz vyjádřen jako histogram naučených vizuálních slov celkově utvářející popisný vektor o pevné délce. Takový vektor je vhodný jako vstup pro metody strojového učení typu učení s učitelem. Nejčastější používanou metodou v korelaci s BOVW je metoda podpůrných vektorů SVM. [14], [16], [17]

4 DODATEČNÉ ALGORITMY POUŽITÉ PRO UČENÍ

Tato kapitola rozebírá dodatečné metody použité v rámci učení, a to metodu křížové validace, které měří výkonnost modelů, a genetické algoritmy, které jsou použité pro nalezení parametrů blízko optimálních hodnot.

4.1 Metoda Křížové validace

Křížová validace je statistická metoda vyhodnocující učící algoritmy rozdělováním dostupných dat na trénovací a validační a slouží k odhadu skutečné chyby modelu. Standartním typem metody křížové validace je k-fold, která rozdělí množinu dat na souměrně velkých $k-1$ trénovacích a 1 validační část. Pro každou iteraci je naučeno n modelů a následně každý z n modelů je požádán o predikci nad validačními daty, z čehož se získá statistická informace předem nadefinovanou metrikou, jako například přesnost nebo odchylka, pro právě naučený model. V další iteraci se vybere jiná část z dříve trénovacích částí jako validační a algoritmus se opakuje. Po dokončení všech iterací jsou dostupné hodnoty nastavené metriky pro každé k. Skutečná chyba modelu se dá určit jako součet všech vypočtených odchylek. Celková chyba modelu Err_{cv} je průměrem chyb všech dílčích modelů

$$Err_{CV} = \frac{1}{K}\sum_{i=1}^{K} Err\left(y_{\kappa(i)}, \tilde{f}^{-\kappa(i)}\left(x_{\kappa(i)}\right)\right), \tag{36}$$

kde K je počet částí vytvořených z dostupných dat, $\kappa(i)$ je i-tá část, $y_{\kappa(i)}$ a $x_{\kappa(i)}$ jsou výstupní hodnoty nastavených metrik a $\tilde{f}^{-\kappa(i)}$ je model naučený bez použití množiny $\kappa(i)$. [15], [20]

4.2 Genetické algoritmy

Genetické algoritmy jsou nástrojem pro modelování a simulaci složitých vícerozměrných nelineárních systémů. Snaží se o analogii s biologickým modelem evoluce. Na začátku je vytvořena populace tvořena jedinci, každý jedincem je charakterizován určitými atributy, které tvoří genom, či v případě jednodušších aplikací pouze chromozom, který určuje celkový charakter jedince. Z hlediska vývoje jedince dochází k adaptaci, která mění určité jeho vlastnosti pro zvýšení šance na přežití. Míra adaptace se v rámci genetických algoritmů vyjadřuje kvantitativně a nazývá se fitness funkce. Jedinci s vyšší hodnotou fitness funkce jsou kvalitnější a mají tak větší šanci na přežití. Jedinci s nižší hodnotu fitness funkce mohou být také vybrány do následující generace při určitých typech selekce. Selekce vybere z dané generace pouze část jedinců, kteří se následně stanou rodiči a mají

své vlastní potomky. Potomci vznikají jednak křížením vlastností svých rodičů, ale také náhodnými mutacemi. Soubor těchto rodičů a potomků se nazývá generace, která je následně opět podrobena selekčnímu tlaku utvářející cyklus vývoje. [18], [19]

Kódování jedinců může být binární nebo pomocí reálných čísel. Chromozom vytvořen binárním kódováním je složen z pouze jedniček a nul, které lze chápat tak, jestli chromozom danou binární informaci má nebo ne. Kódování pomocí reálných čísel vytváří chromozom jako řetězec reálných čísel. Oproti binárnímu kódování umožňuje tento přístup kratší a čitelnější zápis při dodržení stejné numerické přesnosti. Nevýhodou je však hledání vhodných typů genetických operátorů křížení a mutace, jejíchž správná volba je klíčová pro nalezení řešení. [18]

Křížení jedinců je jeden z operátorů genetických algoritmů a slouží k vytvoření nového jedince kombinací jeho dvou rodičů. Mechanismus závisí na typu kódování chromozom jedince, v případě binárního kódování se náhodně zvolí index rozdělení chromozomu obou rodičů a z prvního se vezmou hodnoty chromozomu do zvoleného indexu a doplní se hodnotami druhého rodiče počínající od indexu. Tím se vytvoří nový jedinec kombinací svých rodičů. V případě křížení jedinců kódovaných pomocí reálných čísel je rovněž možné použít stejnou metodu jako v případě binárního kódování, nicméně každá z reálných hodnot chromozomu může nabýt hodnot z jiných intervalů a vznikal by problém s překročením dovoleného rozsahu daného atributu. Z tohoto důvodu je výhodnější zavést jiné metody křížení. Jednou metodou může být poměrové zastoupení všech atributů obou rodičů vztahem

$$\boldsymbol{P} = R \cdot \boldsymbol{A} + (1 - R) \cdot \boldsymbol{B}, \tag{37}$$

kde $\boldsymbol{P}$ je nově vzniklý potomek, $\boldsymbol{A}$ *a* $\boldsymbol{B}$ jsou vybraní jedinci pro křížení a R je náhodně číslo z intervalu $(0,1)$. Vztah (37) lze rovněž rozšířit, aby se zvětšil obor hodnot a tím bylo křížení účinnější

$$\boldsymbol{P} = \left| \frac{|\boldsymbol{A} - \boldsymbol{B}|}{2} + \left| \left(\left(R \cdot K - \frac{K}{2} \right) \cdot |\boldsymbol{A} - \boldsymbol{B}| \right) \right| \right|, \tag{38}$$

kde K je parametr, jehož hodnota by měla být větší než 1. [17]

Mutace je dalším z operátorů genetických algoritmů a náhodně upravuje hodnoty atributů jedinců, což může vést k rychlejšímu nalezení řešení. Mutace, stejně tak jako křížení, závisí na typu kódování jedinců. U binárního kódování s definovanou pravděpodobností výskytu změní určitý počet hodnot atributů na opačné, tedy z nuly na jedničku a vice versa. U kódování s reálnými čísly nastávají v podstatě dvě možnosti, buď je číslo na vybrané pozici zvýšeno nebo sníženo o předem danou hodnotu, nebo je nahrazeno náhodně vygenerovaným číslem z definovaného intervalu. [18]

Pro vybraní nejsilnějších jedinců slouží operátor selekce. Jednou ze základních metod selekce je vážená ruleta. Každý jedinec má pravděpodobnost výběru úměrnou své kvalitě, tedy kvalitnější jedinci mají podstatně větší šanci na výběr než nejslabší jedinec. Pravděpodobnost výběru jedince se pak spočítá podle vzorce (39) jako

$$P_i = \frac{F_i}{\sum_{j=1}^{N} F_j}, \tag{39}$$

kde P_i je pravděpodobnost výběru i-tého jedince, F_i je kvalita i-tého jedince a $\sum_{j=1}^{N} F_j$ je suma kvalit všech jedinců v generaci. Selekce metodou vážené rulety však může uvíznout v lokálním extrému, pokud jeden jedinec má převážnou pravděpodobnost výběru. Tento problém je řešitelný pomocí metody poziční selekce, která podle hodnotící funkce seřadí jedince sestupně a následně jejich hodnoty nahradí indexem, kde nejslabší jedinec má index 1 a nejsilnější index N, kde N je počet jedinců v generaci. Dále se s jedinci pracuje jako u vážené rulety. Další metodou selekce je metoda Turnaj, která náhodně vybírá dvojici jedinců, ze kterých vítězí jedinec s vyšší kvalitou. Jako poslední často využívanou metodou selekce je Elitismus. Jedná se o jednoduchou techniku, která vybírá určitý počet jedinců s nejvyšší kvalitou, kteří se tak neúčastní selekčního tlaku a rovnou tak přechází do nové generace. [18]

Fitness funkce, nebo také hodnotící funkce, je taková funkce, která kvantifikuje kvalitu jedince. Obecně čím je výstupní hodnota fitness funkce větší, tím je jedinec kvalitnější. Její volba závisí na druhu optimalizačního problému. [18]

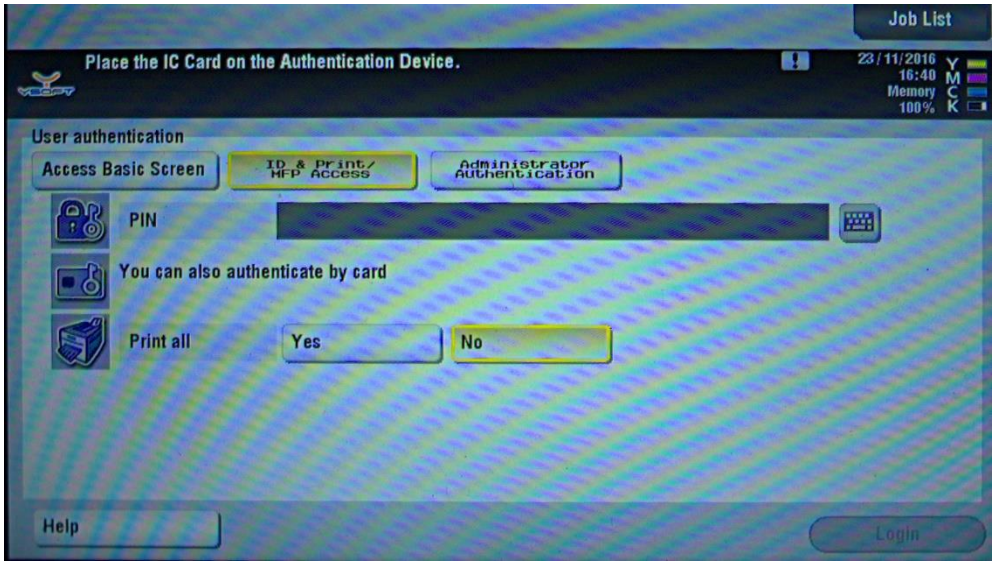
5 TRÉNOVACÍ A TESTOVACÍ GALERIE

V této kapitole jsou probrány dva vyzkoušené přístupy ke tvorbě galerie. První přístup se zakládá na objemné galerii, ve které jsou jednotlivé třídy zastoupeny desítkami snímků. Tento přístup byl funkční, avšak produkčně nerealizovatelný, jelikož by obsluha systému musela každou obrazovku (klasifikační třídu) snímat za různých podmínek, což by dramaticky snížilo rychlost přípravy nového zařízení do provozu. Z tohoto důvodu se od prvotní přístupu opustilo a přešlo se ke druhému přístupu, který využívá přístup k existující dostupné databázi referenční snímků robotického systému, ve které je 1 až *N* snímků ke každé klasifikační třídě. Pro každý referenční snímek se vytvoří syntetická data simulující změny podmínek. Tato simulace prodlužuje dobu učení, avšak řádově zjednodušuje práci obsluze.

5.1 Prvotní přístup tvoření galerie

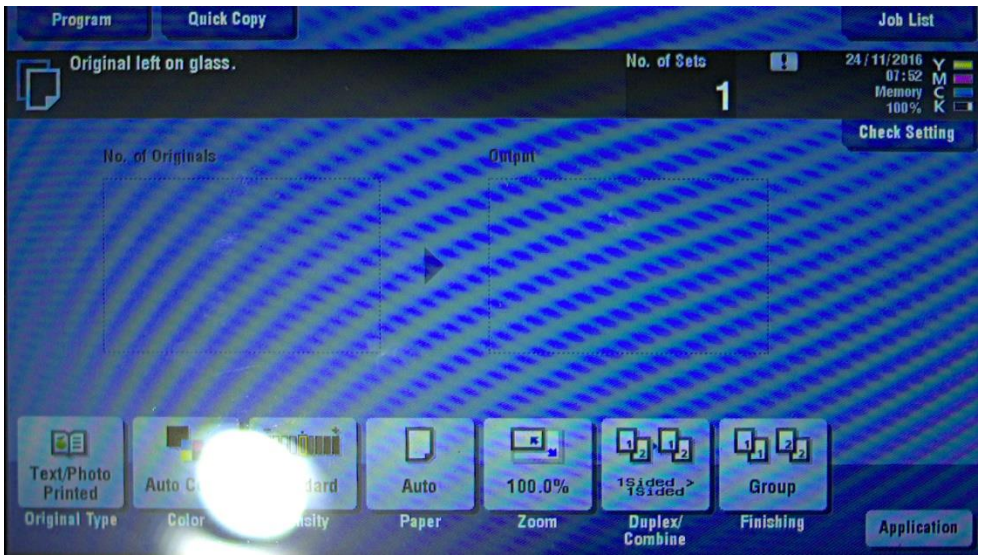
Prvotní galerie se sestávala z celkem 665 obrazů z toho 619 trénovacích a 46 testovacích obrazů zachycujících 9 různých obrazovek zařízení – přihlašovací obrazovka, klávesnice pro zadávání pinu, hlavní menu, SafeQ s tiskovými úlohami, SafeQ se skenovacími úlohami, obrazovka kopírování, obrazovka výběru účtovacího typu, obrazovka výběru typu papíru při kopírování a obrazovka výběru typu papíru při skenování. Všechny obrazu jsou patřičně perspektivně transformovány, oříznuty a velikostně normovány. Obrazy obou galerií mohou a často i obsahují šum (převážně typu moaré), odlesky, změny okolního osvětlení, mírné změny teploty barev, rozostření, prach z displeje zařízení, malé translační posuny i zachycenou část robotického manipulátoru. Při sestavování galerie byl kladen důraz na autentičnost dat a co největší přiblížení následným reálným datům, která budou zpracovávána v procesu vybavování v konečné aplikaci. Autentičnost byla zajištěna algoritmem implementovaným přímo do existující aplikace zpracování obrazu, který po úspěšné identifikaci automaticky uloží obraz s metadaty zakódovanými do názvu souboru. Všechny obrazy byly snímány skutečnou kamerou použitou v testovacím systému a zpracovány algoritmy, které i následně budou zpracovávat příchozí obrazy pro výsledný klasifikační systém této úlohy.

Na následujícím obrázku (Obrázek 5-1) je zobrazen příklad přihlašovací obrazovky se silným projevením šumu typu moaré. Šum tohoto typu má častý výskyt v reálných datech a je důležité na to data připravit a zajistit, aby se výsledný model nepřeučil a nesnažil se hledat závislosti v šumu. Kdyby byla trénovací data složena pouze z etalonů, vazba na šum by se mohla v procesu učení nepředvídatelně vytvořit a všechny obrazovky by se mohly z velké části podobat.



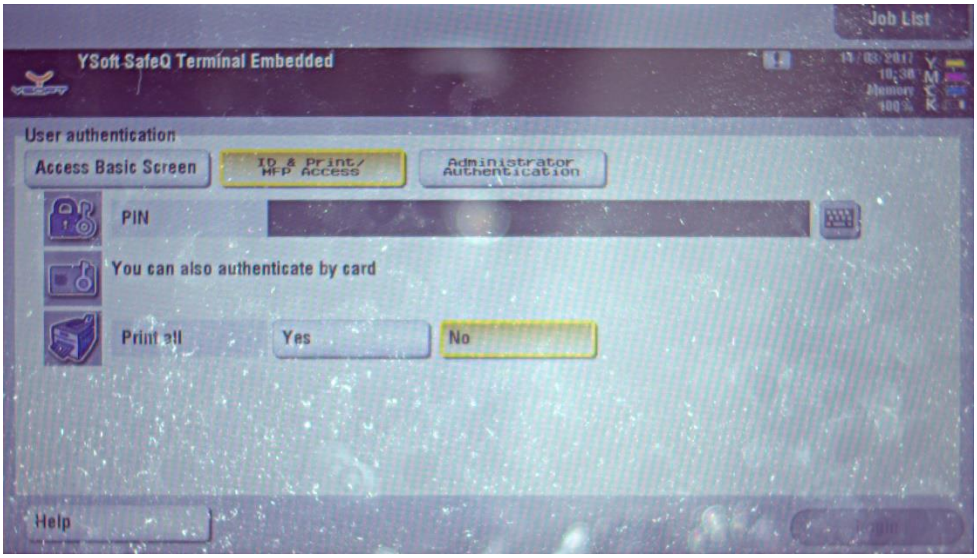
Obrázek 5-1 Testovací obrázek přihlašovací obrazovky se silným projevením šumu typu moaré

Pro testovací galerii byly vybrány obrazy, které obsahují značné deformace a poruchy, aby byla otestovaná robustnost vůči všem typům poruch, které mohou v reálné aplikaci nastat. Na obrázcích Obrázek 5-1 a Obrázek 5-2 jsou předvedeny dvě obrazovky z testovací galerie, které mají úroveň poruch velmi vysokou. Obrázek 5-2 zachycuje obrazovku kopírování a vystupuje v něm značný šum typu moaré, a navíc i silný odlesk od světla nešťastně umístěného nad displejem v úhlu pozorování displeje kamerou.



Obrázek 5-2 Testovací obraz s kopírovacím menu, ve kterém je silně zastoupen šum typu moaré a odlesk od světla umístěným nad zařízením

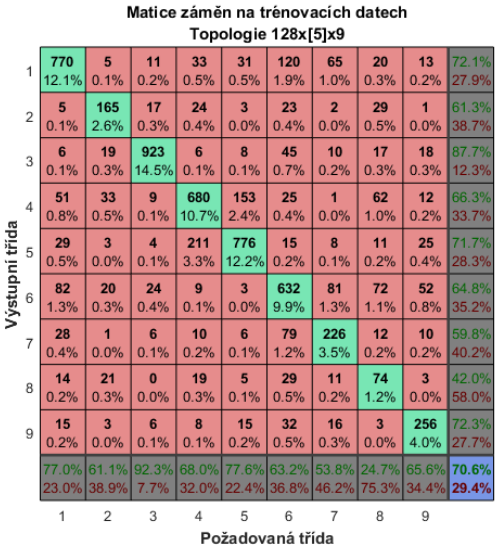
Na následujícím obrázku (Obrázek 5-3) je zachycen problém zaprášenosti displeje zařízení, na které působí přímé sluneční záření.



Obrázek 5-3 Testovací obrázek s přihlašovací obrazovkou s působením přímého slunečního záření na zaprášenou obrazovku

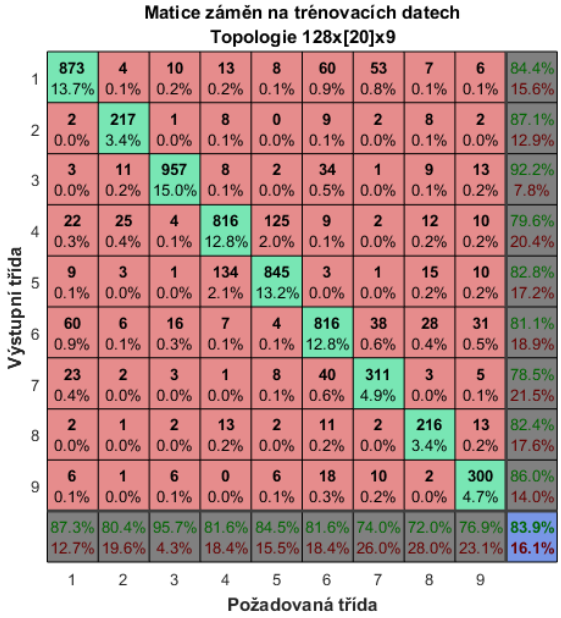
Pro ověření prvotního přístupu a pro vybraný model neuronovou síť s algoritmem učení se zpětným šířením chyby je ve vývojovém prostředí Matlab dostupný nástroj, který umožňuje rychlé vytvoření modelu neuronové sítě, nahrání trénovacích dat, spuštění učení a následné vyzkoušení naučeného modelu na testovacích datech. Nástroj však neumožňuje základní operace při návrhu neuronové sítě, a to konkrétní návrh topologie, nastavení počátečních vah ani nastavení učícího koeficientu a ostatních parametrů. Pro vytváření vlastních neuronových sítí slouží příkazy přímo do skriptu, které jsou rovněž využívány daným nástrojem na pozadí.

Prvním krokem při návrhu neuronové sítě je volba její topologie. Vstupní a výstupní vrstva je dána, 128 neuronů ve vstupní vrstvě odpovídá počtu atributů v příznakovém vektoru, který je vždy na vstupu sítě. Výstupní vrstva je v tomto případě složena z 9 neuronů, zastupujících 9 klasifikačních tříd, které trénovací data definují. Volba jedné nebo více skrytých vrstev a jejích neuronů je však větší problém, jelikož neexistuje obecný návod na volbu topologie. Některé heuristiky uvádějí volbu topologie skrytých vrstev jako 2x*N*, jakožto 2 skryté vrstvy každá s *N* neurony. Obecně také platí, že čím více skrytých vrstev a v nich neuronů, tím větší nelinearity je neuronová síť v trénovacích datech schopna pojmout. Pro začátek testování byly zvoleny topologie o jedné skryté vrstvě a několika málo neuronech (5 a 20) Síť se skrytou vrstvou o pěti neuronech nemá ani teoretické předpoklady, aby byla schopna se naučit závislosti mezi vstupními atributy, a tomu odpovídají i výsledky, které jsou zobrazené ve formátu matice záměn (Obrázek 5-4) pro trénovací data.



Obrázek 5-4 Matice záměn na trénovacích datech pro topologii s jednou skrytou vrstvou o pěti neuronech

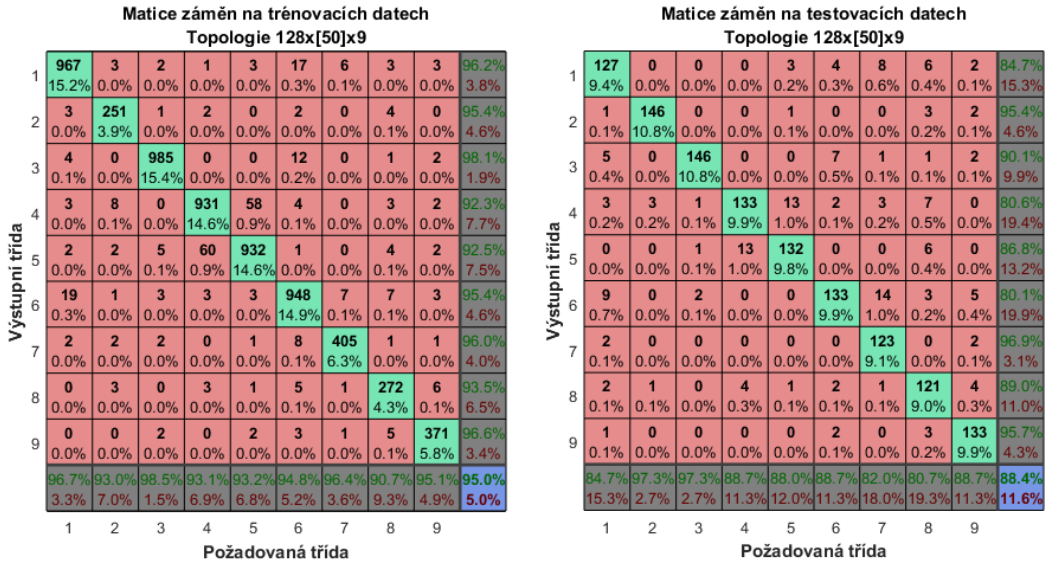
Síť s jednou skrytou vrstvou o dvaceti si vedla v rámci trénování lépe než předešlá síť. Z matice záměn (Obrázek 5-5) lze vypozorovat, že pro většinu tříd se daří správně klasifikovat vstupní deskriptory. K největším záměnám dochází mezi třídami 4 a 5, které jsou si i pro lidské oko značně podobné. Nicméně tato síť vykazuje stále velký počet chybně klasifikovaných vzorků už i pro trénovací data.



Obrázek 5-5 Matice záměn na trénovacích datech pro topologii s jednou skrytou vrstvou o dvaceti neuronech

V dalších pokusech byly otestovány topologie skrytých vrstev: jednovrstvá, dvouvrstvá a třívrstvá s postupně 50, 100 a 150 neurony v každé vrstvě. Již pro jednu skrytou vrstvu s padesáti neurony lze pozorovat v maticích záměn (Obrázek

5-6) zlepšení oproti předchozí síti. V matici záměn na trénovacích datech je stále řada nesprávně klasifikovaných vzorků, avšak v porovnání s celkovým počtem jde o přípustnou chybu.



Obrázek 5-6 Dvě matice záměn pro topologii s jednou skrytou vrstvou o padesáti neuronech, a) na trénovacích datech, b) na testovacích datech

Klasifikace na trénovacích datech není však směrodatná, pouze to vypovídá o faktu, že je daný model schopen se na trénovací data naučit, ale nezaručuje to kvalitu klasifikace budoucího předloženého neznámého vzoru. O kvalitě zařazení neznámého vzoru do známých tříd v procesu vybavování vypovídá matice záměn (Obrázek 5-6 b), která zobrazuje výsledky klasifikací po předložení testovacích vzorů. Všechny hodnoty predikcí jsou nad úrovní 80%, a už tato topologie by se mohla zkusit využít jako výsledný klasifikátor. Důležité také zmínit, že procentní hodnoty uvedené v maticích záměn jsou dobré k celkovému přehledu, ale nejsou tolik podstatné, jelikož vypovídají o celkové distribuci vzorků s danou třídou mezi všechny klasifikovatelné třídy. S rostoucím počtem výstupních tříd bude toto číslo zákonitě klesat a bude čím dále méně důležité. Podstatným zůstává relativní velikost hodnoty správné klasifikace oproti druhé největší hodnotě nesprávné klasifikace a rozdílu mezi nimi, to ve výsledku rozhodne o kvalitě klasifikace. Rovněž důležité bude odlišná výstupní funkce pro konečnou klasifikaci, kde se místo jednoho maxima budou brát všechny výstupní třídy nad určitým prahem. Některé deskriptory zákonitě můžou být velmi podobné až stejné s deskriptory jiných tříd, jelikož některé obrazovky mohou mít stejné části obrazu (například hlavička menu), a tak je přímo žádoucí tuto závislost v modelu umožnit, což zvedne robustnost systému opět o něco výš. Navzdory dobrým výsledkům topologie s jednou skrytou vrstvou o padesáti neuronech, zůstává otázkou, zda si takový počet neuronů poradí

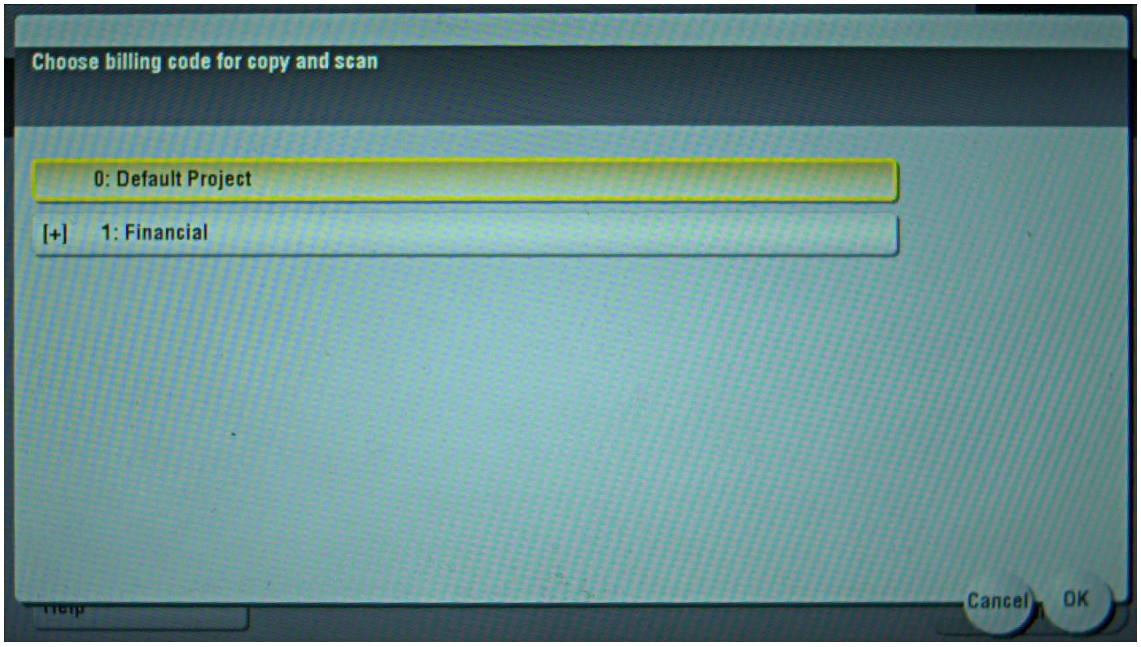
s více výstupními třídami, na které v době experimentu nebyly data, avšak v reálné aplikaci mohou jednoduše nastat.

Po přidání další skryté vrstvy o padesáti neuronech se výsledky obecně nezlepšily a zůstaly podobné. Tabulka 2 srovnává různé návrhy skrytých vrstev, vliv velikosti vrstev na dobu učení a počet iterací a zobrazuje nejmenší dosaženou kvalitu predikce neznámého vzoru s informací u jaké třídy nastala.

Tabulka 2 Porovnání vlastností navrhnutých topologií skryté vrstvy

Topologie	Doba učení [s]	Počet iterací	Nejmenší přesnost [%]	Nejhůře klasifikovaná třída
50	7	177	80,7	8
[50 50]	17	222	74,0	7
[50 50 50]	22	207	82,0	7
100	17	203	63,3	7
[100 100]	32	192	78,7	7
[100 100 100]	53	220	82,7	8
150	25	206	83,3	7
[150 150]	61	231	86,0	6
[150 150 150]	89	224	77,3	7

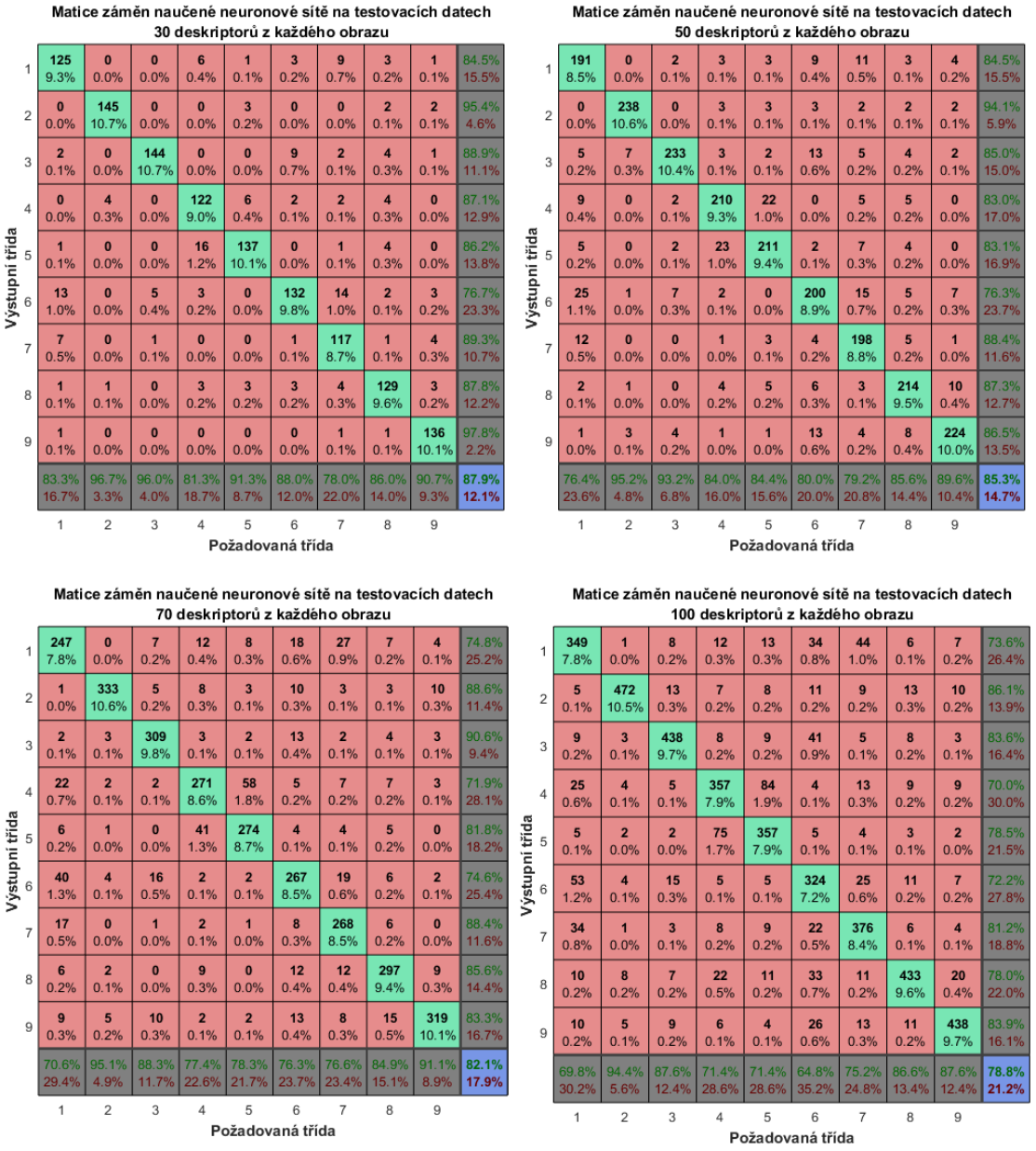
Podle dosažených výsledků pro různé topologie shrnutých v tabulce Tabulka 2 lze usoudit, že se zvětšujícím se počtem skrytých vrstev a počtem neuronů se zvyšuje časová náročnost na proces učení, rovněž lze z tabulky vyčíst, že největší obtíže má model s klasifikací deskriptorů patřící třídě s indexem 7, která obsahuje výběr účtovacího typu (Obrázek 5-7). Tato obrazovka není zrovna ideální pro deskriptory, jelikož v obraze jsou majoritou monotónní oblasti, nicméně i přesto je kvalita klasifikace dostatečná.



Obrázek 5-7 Testovací obrázek s nejproblémovější klasifikací ze všech testovacích dat

Vzhledem ke všem výsledkům topologií není jednoduché vybrat právě jednu topologii. Pro účely úlohy bude vybrána topologie s jednou skrytou vrstvou o padesáti neuronech s poznámkou, že by tato topologie nemusela stačit pro více výstupních tříd. Pro aktuálních devět klasifikačních tříd to však stačí, a tak bude použita jako výchozí topologie dalších testů.

Dalším testem byl počet deskriptorů extrahovaných z obrazů a jejich vliv na výslednou přesnost klasifikace. Pro trénovací i testovací data byly vytvořeny datasety, které popisovali každý obraz 30, 50, 70 a 100 deskriptory. Ze čtyř maticí záměn (Obrázek 5-8) lze pozorovat trend, kde s více deskriptory extrahovanými z obrazu se snižuje kvalita predikce relativně k celkovému počtu predikovaných vzorů. To je zapříčiněno snižující se kvalitou jednotlivých deskriptorů, které se postupně vybírají podle síly jejich odezvy. S rostoucím počtem deskriptorů klesá síla odezvy a tím i unikátnost deskriptoru vůči ostatním třídám. Nejlépe z tohoto testu vyšel počet třiceti deskriptorů na obraz. Tento počet nemá moc prostoru pro chyby, ale na druhou stranu využívá pouze nejvíce výrazných deskriptorů z obrazu a tím zanáší do klasifikace menší chybu.



Obrázek 5-8 Matice záměn klasifikovaných testovacích dat pro různý počet extrahovaných deskriptorů z obrazů

5.2 Druhý přístup tvoření galerie

Tento přístup využívá přístup ke komponentě robotického systému CRISA (Central Reference Image Storage Application), která zajišťuje pro všechny robotické systémy v lokální sítí skrze REST API přístup k nadefinovaným referenčním obrazům. Takto získaný obraz má informace o zařízení, ke kterému patří, identifikátor třídy a další atributy. Každá třída má typicky jednu referenční obrazovku, nicméně existují i takové, které mají nadefinovaných *N* variací. Variace

má za úkol definovat jiný vizuál nadefinované třídy, jenž se může měnit v závislosti na konfiguraci zařízení.

Generování syntetických učících a testovacích dat po stažení referenčních obrazů. Každý obraz je nejprve zbaven šumu, následně se vygeneruje daný počet předem definovaných simulací: Ztmavení obrazu o 15% a 30%, zesvětlení obrazu o 15% a 30%, translační posun obrazu o ± 10 pixelů v obou souřadnicích a ve všech kombinacích, rotační deformace obrazu o $\pm 1.5°$ a změna teplot barev na studenou, teplou a podzimní. K těmto simulačním obrazům se vygeneruje ještě počet obrazů s náhodnými deficity popsaným výše s rovněž náhodnými hodnotami v omezených oborech hodnot. Dodatečně simulační obrazy se generují, aby doplnily deficit vzniklý přítomností více variací různých tříd, tudíž aby některá ze tříd neměla početně majoritní zastoupení v rámci trénovací galerie. U testovací galerie není nevyváženost dat tak velký problém a bohatě dostačuje alespoň částečné zastoupení každé třídy. Proto se pro každý referenční obraz generuje 20 náhodných simulačních obrazů. Každý simulační jev má definovanou pravděpodobnost výskytu definované v následující tabulce (Tabulka 3) a tak jeden simulační obraz může mít minimálně jednu, ale při určité pravděpodobnosti i všechny simulační jevy. Pravděpodobnosti po sumaci nedávají 100%, jelikož se mezi jevy nerozhoduje, ale každý jev může nastat nezávisle na ostatních. V případě, že nenastal žádný ze simulačních jevů, je aplikována změna intenzity.

Tabulka 3 Definované simulační jevy a jejich pravděpodobnost výskytu a obor hodnot. Tyto hodnoty se aplikují pro trénovací data doplňující deficit a pro všechna testovací data. Nová hodnota simulačního jevu je vždy náhodně vygenerované číslo z příslušného oboru hodnot.

Simulační jev	Pravděpodobnost [%]	Obor hodnot	Jednotka
Intenzita	70	$\langle -40; 40 \rangle$	**[%]**
Posun	20	$\langle -20; 20 \rangle$	**[Px]**
Rotace	10	$\langle -3; 3 \rangle$	**[°]**
Teplota barev	40	$\langle 10; 20 \rangle$	**[%]**

Trénovací galerie pro M klasifikačních tříd má po generačním procesu počet snímků $N_{gallery}$ podle rovnice (40) jako

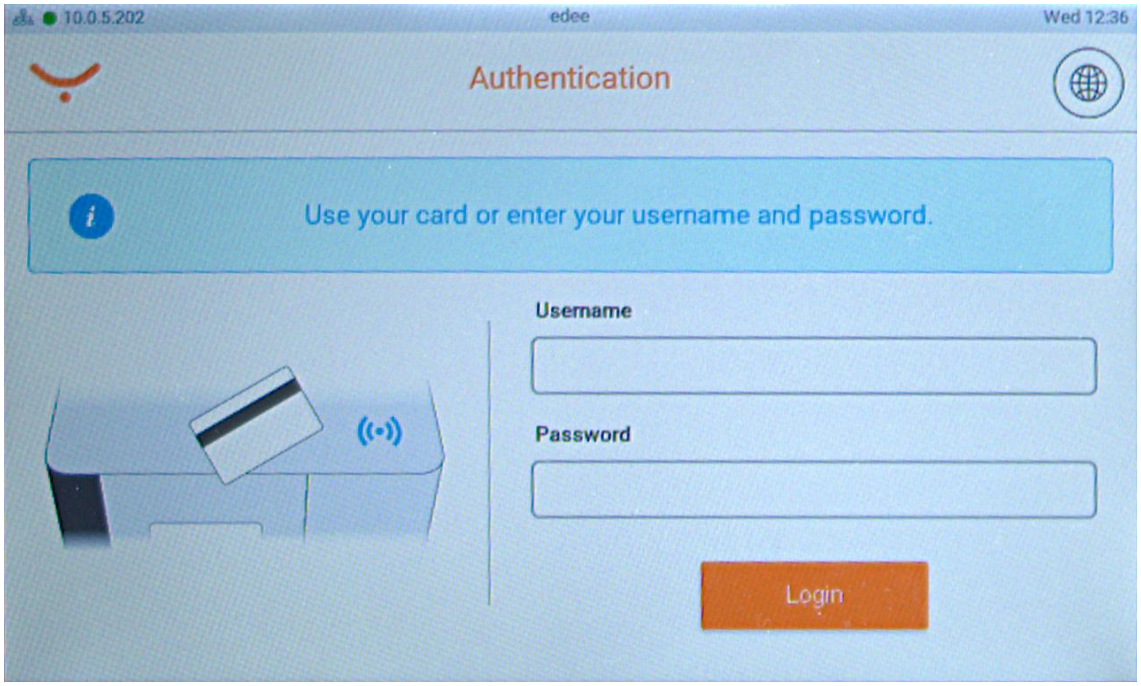
$$N_{gallery} = N \cdot \Big(\big(N_{si} + N_{sp} + N_{sr} + N_{st} \big) + N_{def} + 1 \Big), \qquad (40)$$

kde N je počet všech referenčních obrazů, N_{si} je počet simulací intenzity, N_{sp} je počet simulací s translačním posunem, N_{sr} je počet simulací s aplikovanou rotací, N_{st} je počet simulací se změnou teploty barev, N_{def} je počet simulací pokrývající deficit aktuální třídy. Jednička přičtená na konci rovnice je pro originální obraz, který se ke kolekci přidává. Deficit se vypočte podle rovnice (41)

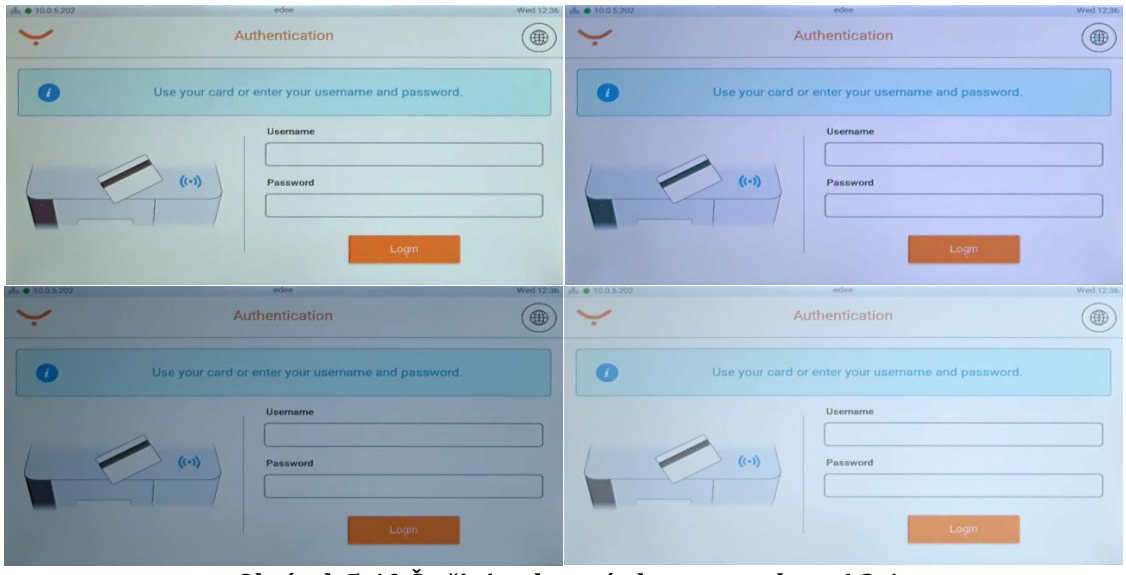
$$N_{def} = (N_{max} - N_{curr}) \cdot \big(N_{si} + N_{sp} + N_{sr} + N_{st} \big), \qquad (41)$$

kde N_{max} je nejvyšší počet variací jedné třídy a N_{curr} je počet variací aktuálně počítané třídy.

Pro uvedení příkladu je provedena simulace jevů na obrázku (Obrázek 5-9). Na dalším obrázku (Obrázek 5-10) jsou zobrazeny čtyři výsledky simulací, na kterých lze pozorovat změny intenzity jasu a změny teploty barev.



Obrázek 5-9 Originální obrázek použitý pro simulaci jevů trénovacích obrazů.



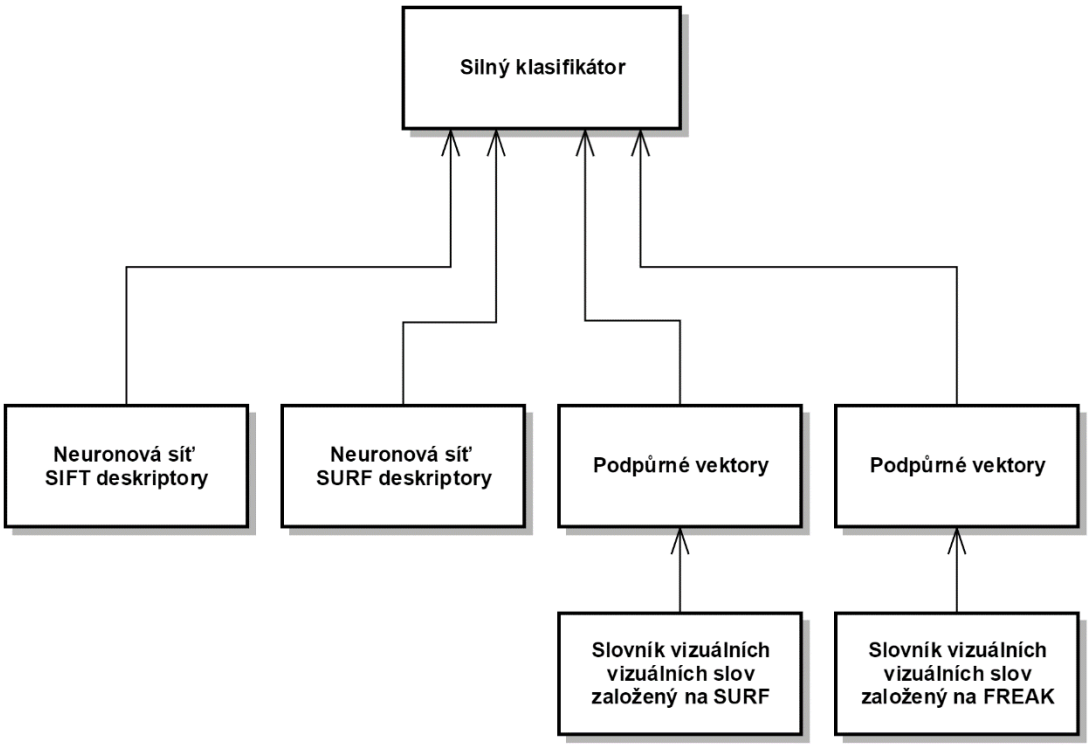
Obrázek 5-10 Čtyři simulované obrazy pro obraz 6.2-1.

6 NÁVRH ŘEŠENÍ

V této kapitole jsou předvedeny návrhy klasifikátoru a architektury výsledného systému.

6.1 Návrh klasifikátoru

Pro tuto úlohu je důležitá přesnost, se kterou bude klasifikátor pracovat, rovněž tak je důležitá stabilita a opakovatelnost při klasifikaci. Z tohoto důvodu se nemůže spoléhat pouze na jeden slabší klasifikátor a je vhodné využití více slabých klasifikátorů pro získání jednoho silného klasifikátoru. Obrázek 6-1 zobrazuje schéma navržených slabých klasifikátorů, které se sdružují v jeden silný klasifikátor. Dva slabé klasifikátory se učí na základě neuronových sítí a zbylé dva pracují se slovníkem vizuálních slov, které jsou vstupem do metody podpůrných vektorů.



Obrázek 6-1 Schéma slabých klasifikátorů, které vytváří silný klasifikátor.

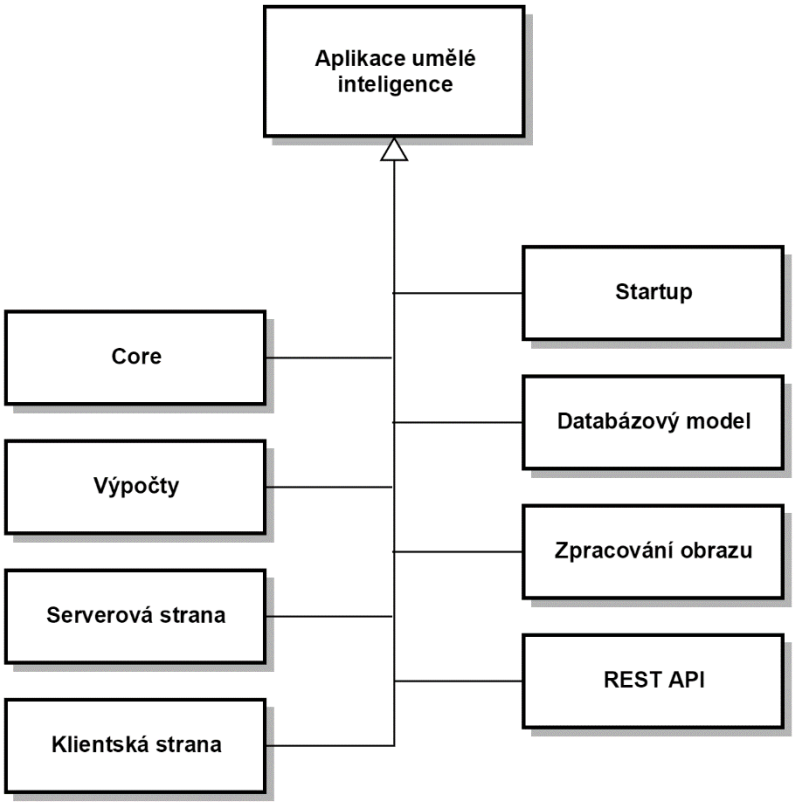
Vstupem do klasifikátorů s neuronovou sítí jsou SIFT a následně SURF deskriptory. U neuronové sítě se SURF deskriptory je vstupní vektor ještě doplněn o pozici, na které se významný bod nachází. Vstupní vektor pro neuronovou síť se SIFT deskriptory obsahuje pouze atributy deskriptoru. Přidáním pozice

do jednoho z deskriptorů zanese závislost nalezeného a klasifikovaného deskriptoru na souřadnicích, což může vést k lepším výsledkům.

Metody podpůrných vektorů mají vstup ve formě vizuálního slova z předem naučeného slovníku vizuálních slov, který je vytvořen buď na základě SURF nebo FREAK deskriptorů. Výhoda použití slovníku spočívá v pevné délce výstupního vektoru, který je přímo vhodný jako vstup do některého z učících algoritmů, nejčastěji tedy SVM. Nevýhoda však spočívá v přidané logistické náročnosti, jelikož se spolu s naučeným SVM modelem musí transportovat i naučený slovník a při přidání nové třídy se musí celý slovník znovu naučit a tím i SVM model.

6.2 Návrh architektury systému

Aplikace umělé inteligence, která je podstatou této práce, je samostatná aplikace schopná běžet v cloudu (Obrázek 6-2).

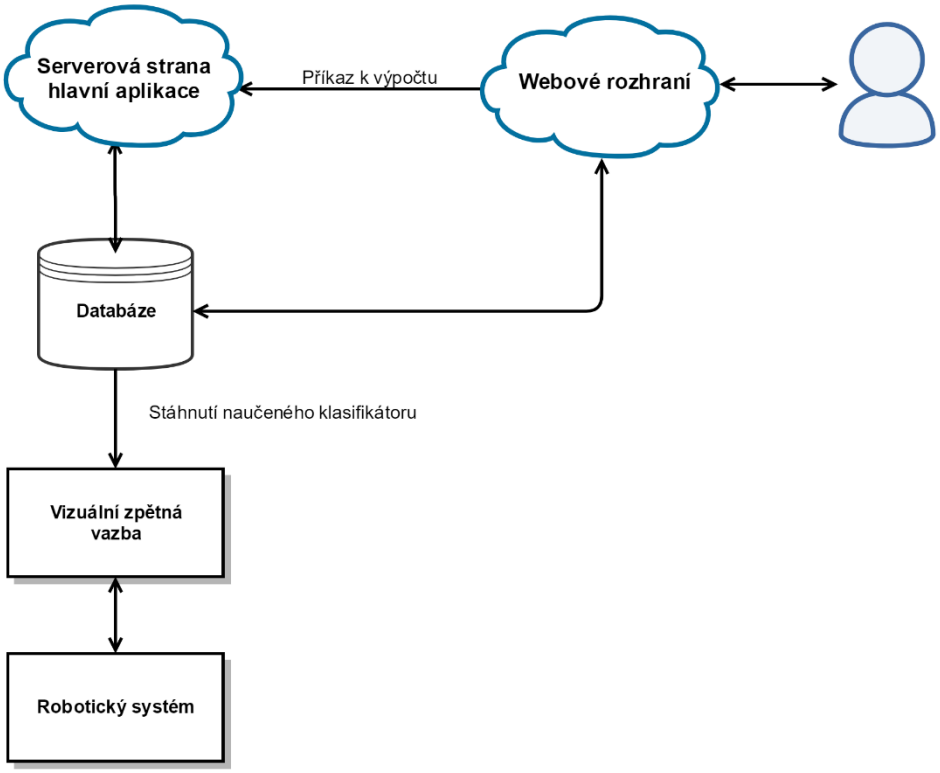


Obrázek 6-2 Schéma hlavní aplikace umělé inteligence

Řešení distribuuje klientskou knihovnu, skrze kterou je možné naučené klasifikátory využívat. Další součástí je databázový model obsahující datové typy pro práci s databází. Podstatnou částí je REST API, které slouží jako přístupový bod k serverové straně. Přijímá požadavky na výpočet nových klasifikátorů, které následně předá serverové straně. Část Core obsahuje všechny třídy, které jsou

používané mezi jednotlivými projekty celé aplikace. Zpracování obrazu poskytuje možnosti filtrování obrazu, generování simulačních obrazů a extrakci příznaků všemi zmíněnými deskriptory. V části Výpočty se provádí veškeré učící algoritmy, následné analýzy a testy.

Na následujícím obrázku (Obrázek 6-3) je zobrazené komunikační schéma mezi jednotlivými zúčastněnými komponentami. Uživatel ovládá hlavní aplikaci umělé inteligence skrze webové rozhraní, v němž dává příkazy k novým výpočtům, sleduje stavy zadaných příkazů, má možnost prohlížet již naučené klasifikátory, jejich průběhy a statistické informace. Rovněž má skrze webové rozhraní možnost definovat topologie a parametry neuronových sítí obecných, nebo vztažených ke konkrétnímu zařízení. Serverová strana komunikuje s databází, ve které jsou uložené referenční obrazovky všech zařízení, topologie a parametry neuronových sítí, informace o definovaných zařízeních a v neposlední řadě již naučené klasifikátory. Serverová strana přijme příkazy k výpočtu a následně naučený klasifikátor uloží do databáze. Klientská strana je nainstalována ve vizuální zpětné vazbě. Při požadavku od robotického systému na rozpoznání displeje zařízení, vizuální zpětná vazba načte naučený klasifikátor z databáze a provede klasifikaci nad vstupním obrazem a vypočtený výsledek vrátí robotickému systému.



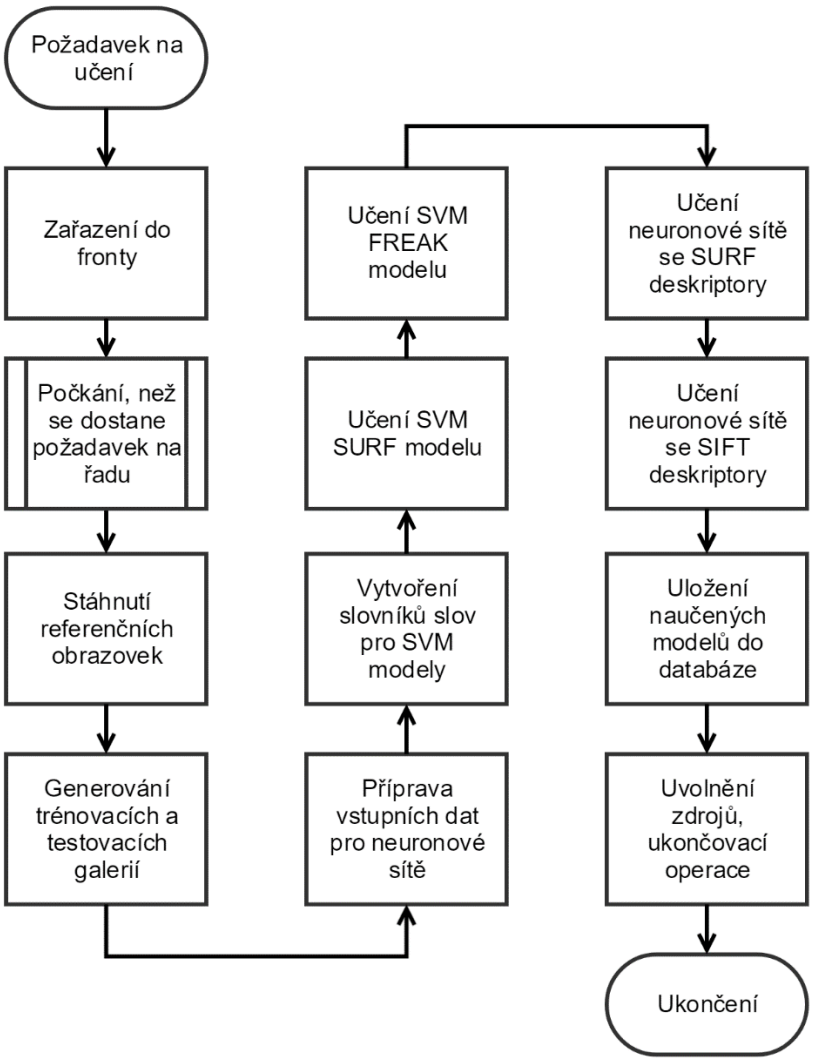
Obrázek 6-3 Schéma komunikace mezi komponentami.

7 ŘEŠENÍ

V této kapitole je rozebráno konkrétní řešení algoritmů, které obstarávají proces učení nového klasifikátoru a způsob agregace výsledků slabých klasifikátorů ve výsledek silného klasifikátoru. Rovněž jsou popsány rozlišovací mechanismy pro kompatibilní třídy.

7.1 Učení nového klasifikátoru

Jakmile dostane hlavní aplikace příkaz k učení nového klasifikátoru, přidá požadavek do výpočetní fronty a započne proces učení (Obrázek 7-1).



Obrázek 7-1 Proces učení

7.1.1 Příprava dat

Každý požadavek nejprve čeká ve frontě, než jsou předešlé požadavky odbaveny. Poté, co požadavek vystoupí ze fronty, získá referenční obrázky pro konkrétní zařízení z databáze, ty se následně velikostně normalizují, a začne se s generováním trénovacích a testovacích obrazů, které je popsané v kapitole Trénovací a testovací galerie.

Pro neuronové sítě je potřebné transformovat vhodně obrazy na data, která budou sloužit jako vstupy a cílené výstupy neuronových sítí. Obrazy se předloží příslušnému extraktoru deskriptorů významných bodů, který detekuje významné body v obraze a následně je i patřičným deskriptorem popíše. U SURF deskriptoru se používá implementace z knihovny OpenCV, avšak nepoužívá se přímo detektor významných bodů, který je v rámci implementace dodán. Místo toho jsou významné body nalezeny agregací výsledků z detektoru konvexních tvarů (Blob detektor), které jsou doplněny na požadovaný počet o body nalezené Harrisovým detektorem významných bodů seřazené sestupně dle jejich kvality. Pro nalezené body jsou pak extrahovány hodnoty atributů SURF deskriptorů. Výsledný deskriptor je o velikosti 130 atributů a sestává se ze dvou hodnot označující pozici nalezeného významného bodu X a Y, které následuje 128 atributů SURF deskriptoru. Hodnoty atributu pozic jsou navíc multiplikovány nastavitelným kvalifikátorem, který určuje váhu, jakou má chyba v atributech pozice vůči celému deskriptoru. U SIFT deskriptoru se významné body hledají stejným způsobem jako tomu je u SURF deskriptorů. Výsledný deskriptor není doplněn o informace o pozici významného bodu, a tak se spíše soustředí na lokální informaci o bodu a sestává se z pouze 128 hodnot. Oba deskriptory jsou sice postavené na podobných teoretických základech, nicméně díky rozdílnému počítání pyramidy oktáv dávají odlišené výsledky, tudíž jsou vhodné pro kombinaci slabých klasifikátorů.

7.1.2 Učení SVM modelů

Před učením SVM modelů je nutné vytvoření slovníků vizuálních slov z dostupných trénovacích obrazů. Způsob vytváření slovníku se odvíjí od typu použitých deskriptorů, které vlastní vizuální slova tvoří, a metody shlukování. Výchozím vytvářením slovníku je nastavení se SURF deskriptory a binárním dělením. Druhý slovník se tvoří za využití FREAK deskriptorů a shlukování metodou K-Means. Nejdůležitějším parametrem pro oba způsoby je počet konečných shluků odpovídající počtu vizuálních slov ve slovníku, a tedy i konečné pevné délce deskriptoru. Slovník vizuálních slov se dá teoreticky naučit na kterýkoliv lokální deskriptor významných bodů, jako jsou například HOG (Histogram of Oriented Gradients) či LBP (Local Binary Pattern), nebo jakoukoliv vlastní implementaci typově kompatibilní s dostupným algoritmem. Nicméně žádný z dalších již

implementovaných deskriptorů nevykazoval lepší výsledky, než tomu je právě u SURF a FREAK deskriptorů. S vytvořenými slovníky je provedena transformace trénovacích obrazů na zastoupení v podobě vektorů vizuálních slov, které jsou následně použity jako vstupy do SVM modelu. SVM modely se učí pomocí algoritmu SMO (Sequential minimal optimization), který řeší problém kvadratického programování. Po naučení SVM modelu se provede analýza nad trénovacími a testovacími daty a zjistí se počet nesprávně klasifikovaných, vypočítá se přesnost klasifikace a průměrné skóre klasifikace testovacích dat, které slouží v agregaci výsledků slabých klasifikátorů k normalizaci výsledku.

Učení SVM modelů je rovněž možné místo s pěvně stanovenými parametry také nalezením optimálních nebo alespoň optimu se blížících hodnot parametrů. Pro to byly vyzkoušeny dvě metody, metoda křížové validace a metoda genetických algoritmů. S použitím křížové validace byly nadefinovány množiny parametrů, které se postupně na vytvořeném modelu a galerii vyzkoušeli, zjistila se hodnota výkonnostní funkce a vybrala se ta nejlepší. S použitím genetických algoritmů se jako atributy chromozomů použily parametry SVM modelu a fitness funkce byl výsledek validačního testu na testovacích datech. Vytvářeli se tak nové generace, dokud alespoň jeden jedinec neměl 100% validaci nebo dokud se nedosáhlo maximálního času trénování. Obě metody nalézání parametrů pro učení SVM modelu byly funkční, avšak byly velice časově náročné.

7.1.3 Učení neuronových sítí

Dalším krokem algoritmu procesu učení je učení neuronových sítí. Nejprve jsou vykonstruovány objekty neuronových sítí v závislosti na datech v databázi, které pro konkrétní zařízení definují topologie sítí a nastavení parametrů jako je ku příkladu koeficient rychlosti učení (learning rate) a spád gradientu (momentum), ale i ukončovací podmínky. Pro každé definované zařízení je možné nadefinovat vícero těchto sítí, jejichž učení probíhá paralelně na principu *závodu*. V tomto principu mezi sebou soutěží jednotlivé algoritmy, v tomto případě nastavení sítí, a ta síť, která bude schopna se naučit jako první, je považována za nejlepší a konečnou. Tímto elegantním způsobem odpadá vysoká náročnost na obsluhu, která může udělat hrubý odhad topologie a parametrů více sítí a algoritmus sám najde tu nejlepší. Příliš jednoduchá síť se nedokáže konkrétní problém naučit a příliš složitá síť se jej bude učit příliš dlouho, a tak bude vybrána vždy optimální síť ze všech nadefinovaných. Tento systém má však své limitace, a to maximální počet paralelních sítí, který je výpočetní zařízení schopno zvládnout, a alespoň částečnou znalost obsluhy o problematice neuronových sítí a doporučených postupech při nastavování parametrů. Nicméně v nejhorším scénáři se žádná ze sítí nenaučí, nebo bude složitější, než by mohla být. Pokud se žádná ze sítí nenaučí, operátor

nastaví jiné hodnoty a opětovně spustí učení. Pokud se naučí složitější síť, bude učení trvat déle, rovněž hrozí riziko přeučení, obzvláště při nadměrném počtu a velikosti skrytých vrstev.

Po vytvoření objektů neuronových sítí se přechází k vlastnímu učení neuronových sítí. Učení probíhá s algoritmem se zpětným šířením chyby a má hned několik ukončovacích podmínek, mezi které patří podmínka validace, maximálního počtu dosažených epoch, maximální doby učení nebo při dosažení minimální chyby. Nejdůležitější ukončovací podmínkou je validace. Aby se proces učení pro neuronovou síť ukončil validační podmínkou, musí být všechny trénovací i testovací data správně klasifikována, navíc musí být skóre klasifikace každého obrazu větší než stanovené minimum a rovněž musí být hodnota správné klasifikace x krát větší, než je druhá největší hodnota ve výsledku, tedy maximum z ostatních tříd. Velikost multiplikátoru závisí na typu validačních dat, zda se aktuálně validuje trénovací nebo testovací obraz, v případě trénovacího je výchozím nastavením multiplikátor 3 a tedy podmínka je ve tvaru $X_{corr} > 3 \cdot \max(\boldsymbol{X} - \{X_{corr}\})$. Pro testovací data je pak multiplikátor roven hodnotě 2. Popsaným způsobem se nejen zaručí, že po validaci jsou všechna data správně klasifikována, ale že mají i dostatečnou robustnost. Nicméně tento přístup vyžaduje absolutní klasifikaci všech trénovacích a testovacích vzorů, která ve skutečnosti nemusí nikdy nastat. Z toho důvodu je přidán mód zmírněné validace, který automaticky nastává po dosažení předem definovaného počtu epoch učení sítě. Tento mód má za cíl zmírnit podmínky validace, umožnit menší množství špatných klasifikací testovacích vzorů a zabránit tak možnému přeučení. K určení přesného počtu povolených nesprávných klasifikací je použitý vzorec (42)

$$Q_{vr}(i) = (Q_{vr}(i-1) - 0.9) * k + 0.9, \qquad (42)$$

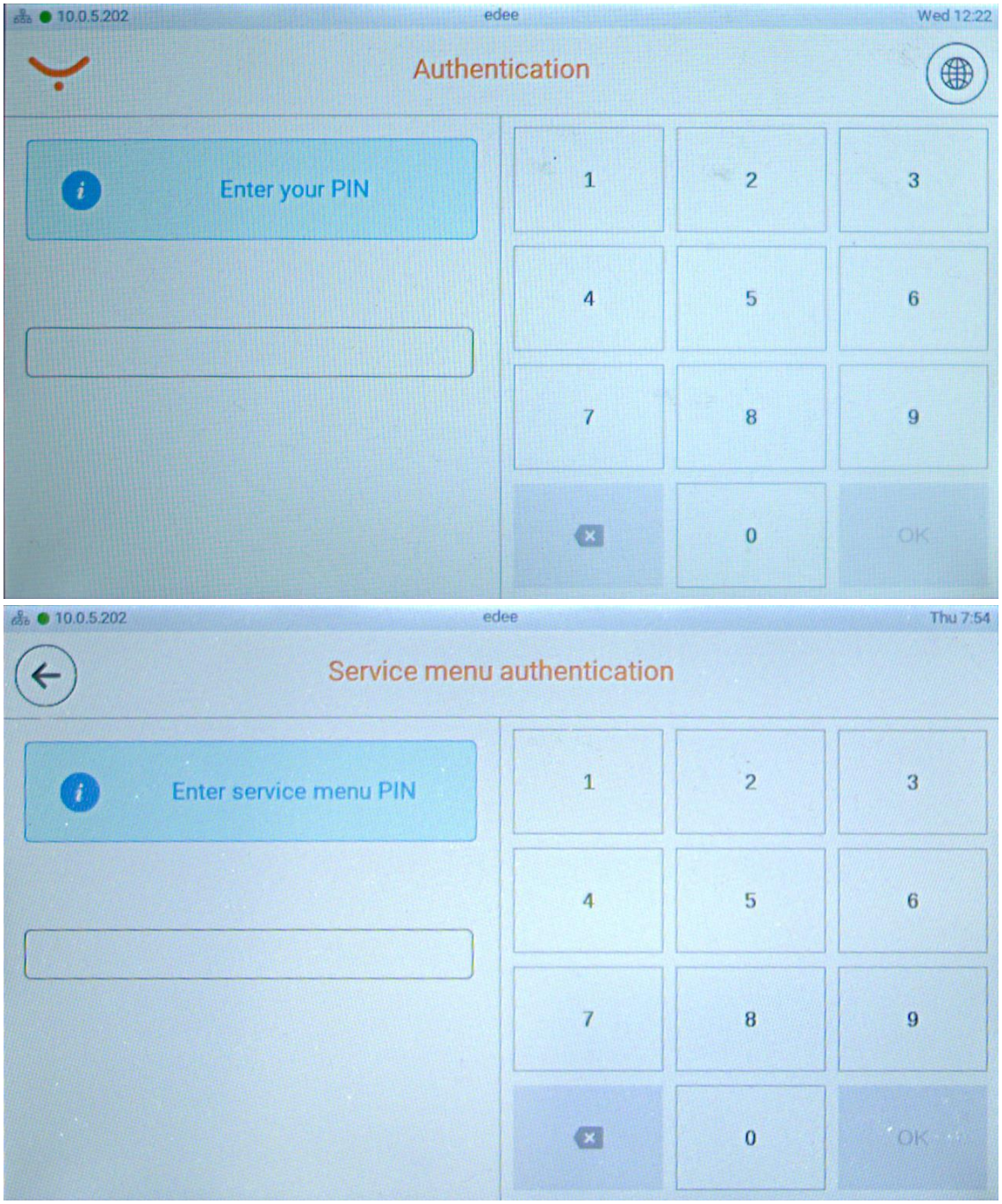
kde i je počet iterací po nastavení módu zmírněné validace, $Q_{vr}(i)$ je kvalifikátor validace v i-té iteraci a k je rychlost poklesu, které je nastavené na 0.975 ve výchozích podmínkách. Hodnota 0.9 definuje minimum funkce Q_{vr}, kterého může dosáhnout. Při dosažení minima je pro validační mechanismus možné opomenout až 10% nesprávně klasifikovaných vzorů. V rámci módu zmírněné validace se upravuje ještě hodnota multiplikátoru X_{corr} na hodnoty 1.25 pro trénovací a 1 pro testovací data.

Zmírněná validace rovněž zavádí kompatibilitu tříd, což jsou třídy, mezi kterými nemá ambice klasifikátor rozlišovat. Pokud by se klasifikátor mezi takovými třídami snažil rozlišovat, je pravděpodobné, že by se nikdy nenaučil nebo se přeučil. Pokud se v módu zmírněné validace stane, že se vzor klasifikuje jako třída, která sice není správná, ale nachází se v množině kompatibilních tříd s aktuálně žádanou třídou, tak je validace daného vzoru úspěšná. Příkladem mohou být následující dva obrázky (Obrázek 7-2), které jsou si svým obsahem téměř

totožné a liší se pouze v textu v horní části obrazu a pak textem v modré informační oblasti. Tyto rozdíly jsou tak malé, že jsou v jiných případech považovány v podstatě za šum, nicméně robotický systém je potřebuje rozeznávat. K naplnění požadavků robotického systému je dodán algoritmus, který mezi konkrétními třídami rozhoduje buď na základě textu nebo malé části obrazu. Rozlišovací algoritmus je blíže popsán v podkapitole 7.3 Rozlišování kompatibilních tříd.

Jakmile jsou neuronové sítě prohlášeny za naučené, provede se analýza nad všemi daty. Tato analýza zprvu zkoumá počet nesprávně klasifikovaných trénovacích i testovacích vzorů, následně spočítá statistické informace jakožto zprávu o učení pro uživatele, která obsahuje průměrné skóre, průměrnou rezervu, nejnižší dosažené skóre i rezervu, nejslabší a nejsilnější klasifikační třídu a percentil 95 reserv.

Po ukončení části neuronových sítí se všechny naučené modely uloží do příslušného databázového objektu, který se se všemi potřebnými informacemi uloží do databáze. Po uložení do databáze je již možné nový silný klasifikátor pro dané zařízení používat. Před ukončením učícího procesu se ještě uvolňují prostředky použité pro učení k uvolnění paměti.



Obrázek 7-2 Příklad dvou a) a b) nerozlišitelných obrazů zastupující odlišné klasifikační třídy

7.2 Evaluace naučeným klasifikátorem

Vizuální zpětná vazba robotického systému využívá naučeného silného klasifikátoru ke klasifikaci příchozích snímků obrazovek. Snímek je nejprve upraven do požadovaného tvaru pro každý slabý klasifikátor uvnitř silného klasifikátoru. Následně je po každém slabém klasifikátoru požadována predikce pro daný snímek včetně úrovně věrohodnosti výsledku. Zde nastává problém, jakým způsobem

výsledky agregovat, když každý model predikuje v jiném oboru hodnot. Bylo tak žádané najít takovou úpravu výstupních hodnot modelů, která by zaručila pro každý model férové posouzení kvality predikce v závislosti na uskutečněném učení. Nejvíce vyhovující se stala metoda normalizace pomocí průměrného skóre na testovacích datech po ukončení učení.

Jakmile jsou získány a normalizovány výsledky klasifikace všech slabých klasifikátorů, dochází k samotné agregaci výsledků. Ve výchozím případě jsou sečteny pravděpodobnosti jednotlivých predikovaných tříd, kde vítězí třída s největším zastoupením. Pokud však nastane případ, kdy se všechny modely neshodnou ani na jedné třídě, tedy kdy počet zastoupených tříd je 4, tak je prioritizovaný klasifikátor podroben testu, ve kterém je sledován poměr mezi skórem predikce a rezervy a poměr mezi dosaženým skórem predikce a průměrným skórem při učení. Pokud je skóre 3x větší než rezerva a zároveň je skóre větší než $\frac{1}{5}$ průměrného skóre, tak je za výsledek klasifikace silného klasifikátoru považována právě tato třída. V opačném případě je příchozí obraz prohlášen za nerozpoznaný a třída tak za neznámou. Pokud je klasifikovaná třída součástí některé z množin kompatibilních tříd, výsledek silného klasifikátoru je označen za nerozhodný a musí podstoupit rozhodovací mechanismus na vyšší úrovni.

7.3 Rozlišování kompatibilních tříd

Kompatibilní třídy jsou třídy, mezi kterými nemá klasifikátor ambice rozlišovat. Typicky se jedná o obrazovky, které jsou si svým obsahem natolik podobné, že rozdíly jsou pod úrovní šumu. Robotický systém nicméně potřebuje i tyto třídy nějakým způsobem rozlišit, aby mohl pokračovat ve své činnosti, kdy chyba klasifikace může mít fatální důsledek na celý běh testu. Rozlišování kompatibilních tříd tak probíhá s použitím OCR a rozeznáváním textu v definovaných oblastech, nebo vyříznutím části obrazu a nalezením největší shody.

7.3.1 Textové rozlišování kompatibilních tříd

Pro každou třídu dané množiny kompatibilních tříd je definovaná oblast v obraze a text, který se v oblasti nachází. Pro příchozí obraz se pomocí OCR přečtou texty ve všech definovaných oblastech, specifických pro každou z množiny kompatibilních tříd. Nalezený text v definované oblasti je porovnán s předlohou, která by se měla v oblasti nacházet. Porovnání spočívá ve výpočtu vzdálenosti metodou Levenshtein a nalezením nejnižší vzdálenosti napříč všemi definovanými slovy. Levenshtein metoda výpočtu vzdálenosti je metrika pro určení textového rozdílu mezi dvěma řetězci textu. Určuje tak počet minimálního množství úprav pomocí mazání, přidávání nebo substitucí jednotlivých znaků prvního řetězce

pro dosažení řetězce druhého. Hledání vzdálenosti probíhá oběma směry, hledání nejmenší vzdálenosti definovaných slov k nalezeným slovům pomocí OCR a vice versa, ze kterých se následně vybere maximum, které tak vypovídá o shodě sekvenci slov. Kompatibilní třída s nejmenší hodnotou Levenshtein vzdálenosti je prohlášena za rozpoznanou.

7.3.2 Vizuální rozlišování kompatibilních tříd

Podobně jako u textového rozlišování je definovaná oblast v obraze pro každou třídu, která ale definuje přímo obrazová data, která jsou následně porovnávána s příchozím obrazem. V procesu rozlišování jsou pak postupně použity všechny definované oblasti a je vyhledávána největší obrazová i poziční shoda. Metodou vyhledávání části obrazu v příchozím obraze je metoda vyhledávání šablony (template matching). Pro každou oblast jsou získány dvě hodnoty, míra shody x_s a pozice P_{loc}, která určuje levý horní roh místa největší shody. Následně se vypočítá euklidovská vzdálenost d_E jako $d_E = \sqrt{\left(P_{loc} - P_{def}\right)^2}$, kde P_{Def} je skutečná pozice definované oblasti. Pomocí rovnice $f = d_e + \frac{1}{x_s \cdot 10}$ se určí hodnoty funkce f jednotlivých oblastí kompatibilních tříd a vybere se třída s minimální hodnotou této funkce, která má tak nejlepší poměr vzdálenosti od definované lokace a míru podobnosti se vzorem.

8 IMPLEMENTACE

K implementaci celého řešení je použit vysokoúrovňový objektově orientovaný programovací jazyk C# platformy .Net framework, vývojové prostředí Visual Studio, framework pro vytváření webových rozhraní DotVVM a knihovny pro zpracování obrazu a umělou inteligenci OpenCV a AccordNet. Implementace se zaobírá třemi různými částmi, hlavní aplikací, která uskutečňuje samotné učení modelů a vytváří konečný klasifikátor, webovým rozhraním, skrze které má uživatel možnost zadávat příkazy na nové učení klasifikátorů, a klientskou část integrovanou do existujícího řešení vizuální zpětné vazby robotického systému, která využívá naučené klasifikátory uložené v databázi ke klasifikaci příchozích snímků.

Pro práci s obrazem je použita knihovna OpenCV, která nabízí konkrétní implementace známých metod, které se v oblasti počítačového vidění nacházejí. Knihovna OpenCV nemá oficiální podporu pro jazyk C#, a z toho důvodu je použita knihovna OpenCvSharp, která volá její nativní funkce. [21], [22]

Metody strojového učení použité v hlavní aplikaci využívají implementace z knihovny Accord.Net, jenž poskytuje implementace neuronových sítí, metod podpůrných vektorů i genetických algoritmů. [23]

DotVVM je front-end framework pro vytváření webových aplikací v programovacím jazyku C# a značkovacím jazyku HTML založený na návrhovém vzoru MVVM (model-view-viewmodel). Velká výhoda této knihovny spočívá v odstínění programátora od Javascriptu. [24]

8.1 Hlavní aplikace

Hlavní aplikace je implementována dle obrázku Obrázek 6-2. Řešení tedy hlavní aplikace se rozděluje do několika logických částí: Výpočetní část, ve které jsou veškeré výpočty související s učení a která slouží jako jádro klientské části, serverová část, která zařazuje nové požadavky do fronty, kterou postupně odbavuje, část pro přijímání webových dotazů, databázovou část, která obsahuje kontext a model potřebný k přístupu k datům v databázi a část pro práci s obrazem, ve které jsou metody na generování syntetických obrazů a metody na extrakci příznaků deskriptory.

Celá aplikace je postavena na návrhovém vzoru obráceného řízení (Inversion of Control, IoC) a DI (Dependency Injection). Jedná se o techniku, kde jeden objekt plní požadavky na reference ostatním objektům, k čemuž využívá zaregistrované typy objektů nazývané servisy. Požadavky na reference se tak nejčastěji uvádějí v konstruktoru dané třídy, která tak dává najevo komukoliv, jenž vytváří její instanci, že musí rovněž dodat instance požadovaných objektů. Rovněž je

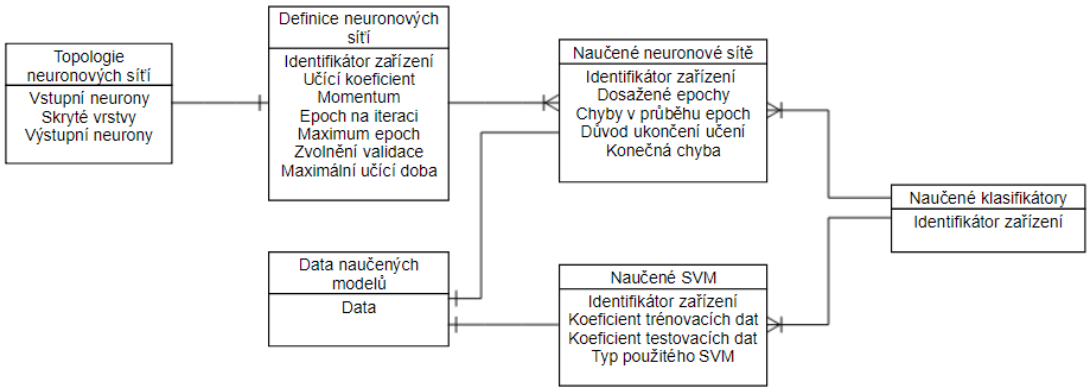
doporučeno požadované objekty mít ve formě rozhraní, které tak dovoluje třídě podstrčit libovolnou implementaci daného rozhraní. Tento způsob je velice rozšířený, neboť umožňuje třídě nezávislost na jedné konkrétní implementaci a rovněž umožňuje budoucí testování. Pro třídu, která využívá IoC a DI, je pak dle návrhového vzoru zakázáno vytvářet nové instance (využívat klíčové slovo *new*) jakékoliv třídy, která je součástí seznamu servis, a v ještě lepším případě nepoužívání klíčového slova *new* vůbec.

8.2 Databázový model

Databázový návrh a model je důležitou součástí celé implementace, jelikož k databázi přistupuje webové rozhraní i serverová a klientská část. Do databáze se ukládají natrénované klasifikátory včetně statistických údajů z trénování. Rovněž jsou v databázi uložené množiny kompatibilních tříd a jejich rozlišovací regiony. Dále se v databázi nachází definice topologií a parametrů neuronových sítí.

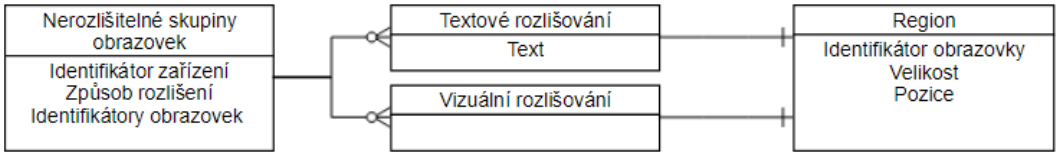
Databázový model je implementován za využití knihovny Entity Framework a přístupem Code first. V rámci tohoto přístup, místo návrhu databáze a následného tvoření objektů odpovídající databázového schématu, se vytváří nejprve třídy, podle kterých se následně vytvoří databázové schéma. Pro vytvoření databáze se využívá již dostupný PostgreSQL server, který je použit rovněž pro databázi robotického systému.

Modelové schéma klasifikátorů (Obrázek 8-1) obsahuje entity definic neuronových sítí, naučených neuronových sítí, naučených SVM modelů a naučených silných klasifikátorů. Každá definice neuronové sítě obsahuje referenci na tabulku topologií, ve které jsou definovány hlavně velikosti skrytých vrstev. Definice obsahuje vlastnosti identifikátor zařízení, nastavený učící koeficient, momentum, počet epoch na iteraci při učení, maximální počet počítaných epoch, počet epoch pro mód zmírněné validace a maximální učící dobu neuronové sítě. Entita naučené neuronové sítě obsahuje cizí klíč do tabulky definic neuronových sítí, počet dosažených epoch při učení, kolekci chyb, které při učení nastali v jednotlivých epochách, použitou ukončovací podmínku a výslednou chybu sítě. Entita také odkazuje na tabulku Data naučených modelů, ve které jsou jednotlivé naučené modely konvertované do řetězce ve formátu Base64, ve kterém jsou zakódované binární data modelů. Entita naučených SVM modelů má informace o zařízení, pro které se výpočet prováděl, koeficienty trénovacích a testovacích dat a typ použitého SVM, tedy jestli se jedná o SURF nebo FREAK deskriptory. Konečná entita naučených klasifikátorů obsahuje pouze vlastnost identifikátor zařízení a následně reference na 2 naučené neuronové sítě a 2 naučené SVM.



Obrázek 8-1 Databázové schéma klasifikátorů

Pro rozlišení kompatibilních tříd slouží regiony definující oblast v obraze, podle které je algoritmus schopný rozeznat, o kterou ze množiny definovaných tříd se jedná. Databázové schéma pro skupiny kompatibilních tříd a jejich regionů (Obrázek 8-2) definuje entitu nerozlišitelných skupin obrazovek, které obsahují informace o konkrétním zařízení, způsobu rozlišení, zda textově nebo vizuálně, a kolekci identifikátorů obrazovek. Tato entita pak obsahuje v závislosti na typu rozlišování *N* textových nebo vizuálních regionů, které obsahují informace o velikosti a pozici v rámci obrazu. Při rozlišování obrazovek se z databáze získá skupina obrazovek, která obsahuje daný identifikátor obrazovky, v rámci této skupiny se získají i regiony, které se následně použijí pro extrakci informace.



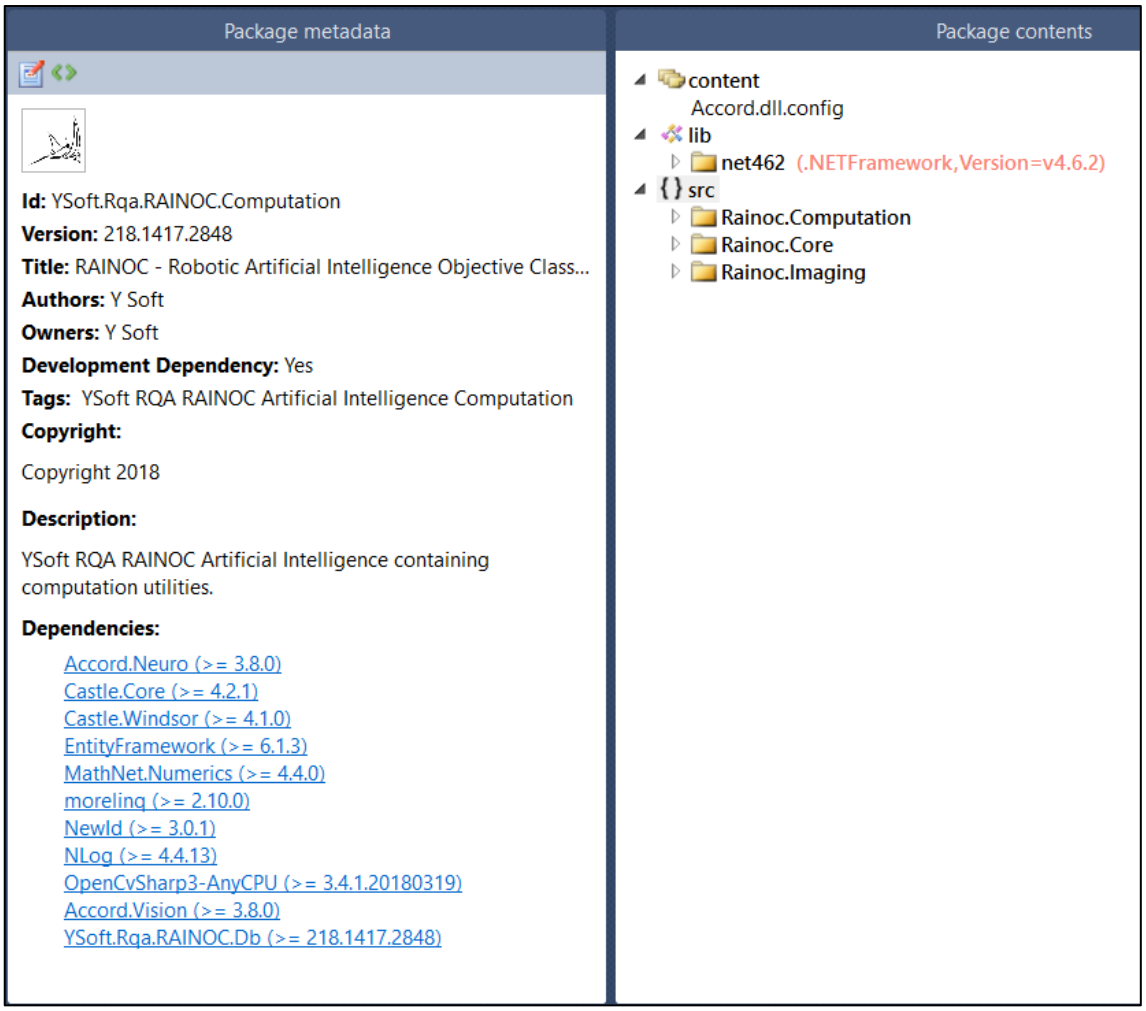
Obrázek 8-2 Databázové schéma oblastí pro rozlišování kompatibilních tříd

8.3 Implementace do existující aplikace pro zpětnou vazbu

Pro distribuci kódu klientům slouží balíčková služba NuGet. Je to nástroj, skrze který mohou vývojáři vytvářet, sdílet a používat svůj kód, který je zabalen do balíčků obsahující kompilované zdrojové kódy, například DLL soubory, které s potřebných dodatečným obsahem mohou být použity v cílených projektech. Tyto balíčky se dají sdílet skrze veřejné nebo privátní hostitele. Po nainstalování balíčku do tíženého projektu se dají funkcionality obsažené v DLL připojených k balíčku využívat přímo v kódu cílového projektu. [25]

Vygenerovaný NuGet balíček klientské strany pro tuto aplikaci (Obrázek 8-3) obsahuje identifikátor, verzi, název, autora, popis a seznam závislostí na jiné

knihovny. Balíčky pro tuto aplikaci se generují dva, balíček s kódem pro výpočty a další balíček s databázový modelem a přístupem k databázi. Pro vizuální zpětnou vazbu se využijí oba, nicméně pro webové rozhraní postačuje balíček s databázovým modelem.



Obrázek 8-3 NuGet balíček klientské strany

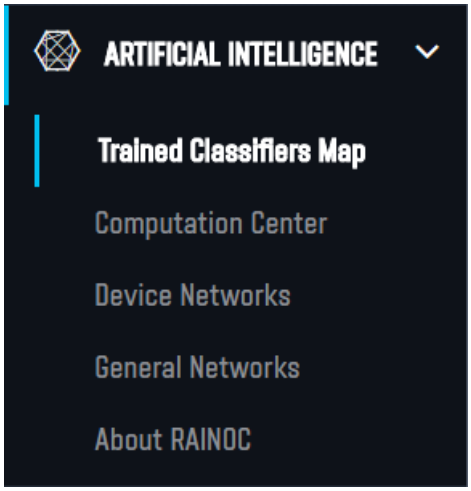
Balíček s výpočty, mimo jiné, zpřístupňuje třídu *ClassifiersEvaluator.cs*. V rámci této třídy má uživatel přístup ke dvěma metodám, metodě *Load* starající se o nahrání naučeného klasifikátoru konkrétního zařízení, a metodě *Evaluate*, která předaný obraz klasifikuje. Výsledkem evaluační metody může být identifikovaná třída, nerozpoznaná třída, nebo množina nerozlišitelných tříd.

8.4 Implementace do existujícího webového rozhraní

Existující webové rozhraní slouží robotickému systému jakožto hlavní administrátorské prostředí, skrze které má uživatel kontrolu nad celým systémem, má možnost upravovat data v databázích, nechávat si zobrazovat statistické informace o běhu robotů a mnohem více. V rámci této úlohy přibyly dvě nové

funkcionality, konfigurace a zobrazování klasifikátorů a možnost definování a správu regionů pro rozlišování kompatibilních tříd.

Menu správy klasifikátorů (Obrázek 8-4) obsahuje celkem 5 stránek. Bráno vzestupně, položku obecných informací, které pomohou novým uživatelům uvedení do kontextu, co je vlastně umělá inteligence, dále stránku konfigurací obecných neuronových sítí a stránku konfigurací neuronových sítí pro konkrétní zařízení. Další položkou je výpočetní centrum, ve kterém má uživatel možnost zažádat o výpočet nového klasifikátoru, sledovat aktuální stavy všech příkazů ve frontě, popřípadě rušit čekající výpočty. Poslední položkou menu je mapa všech natrénovaných klasifikátorů, ve které má uživatel možnost prohlížet klasifikátory, zobrazovat statistiky při trénování a popřípadě mazat nevydařené klasifikátory.

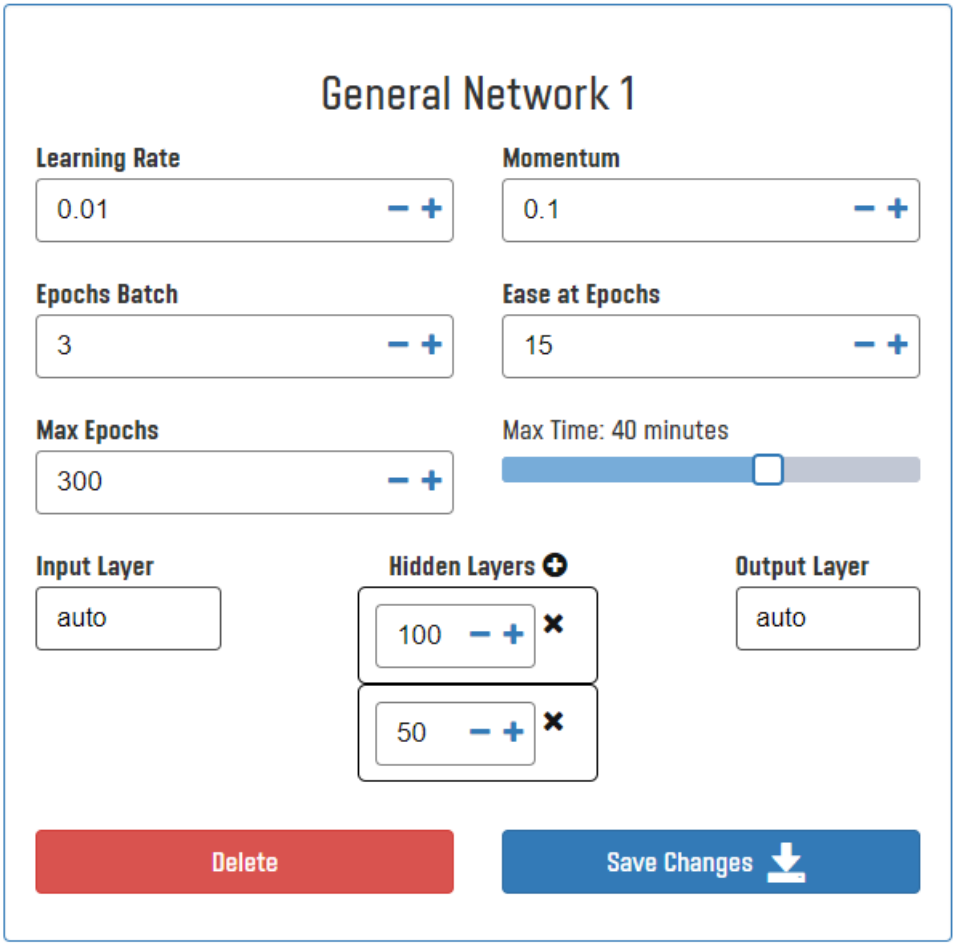


Obrázek 8-4 Menu umělé inteligence ve webovém rozhraní

Stránka obecných konfigurací neuronových sítí (Obrázek 8-5) obsahuje tabulku nadefinovaných konfigurací sítí. Uživatel má tak možnost prohlížet nadefinované topologie a parametry, které má možnost zároveň upravovat. Pod tabulkou definicí je funkcionalita na přidávání nových obecných konfigurací. Po kliknutí na ikonu editace se uživateli otevře detail požadované konfigurace (Obrázek 8-6). V této oblasti může uživatel upravovat parametry sítě a topologii, tedy pouze topologii skryté vrstvy, jelikož vstupní vrstva je odvozena od typu neuronové sítě a počet neuronů ve skryté vrstvě je ekvivalentní počtu klasifikačních tříd.

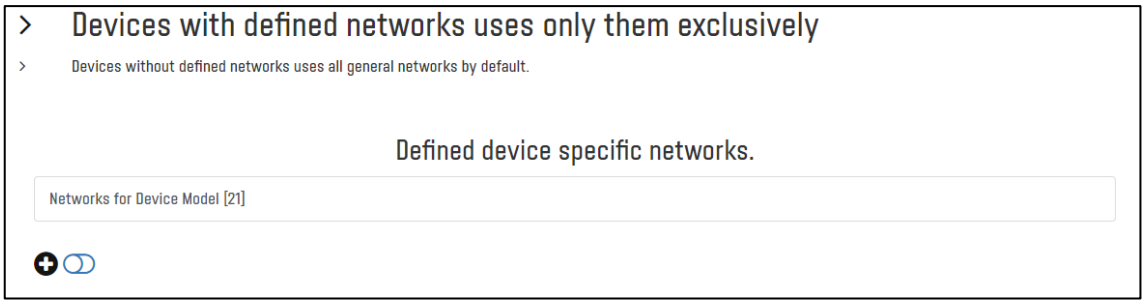
ID	Topology	Learning Rate	Momentum	Epochs Batch	Max Epochs	Max Learn Time	Epoch Ease	Edit
1	auto | 100 x 50 | auto	0.01	0.1	3	300	00:40:00	15	
3	auto | 100 x 100 | auto	0.02	0.1	3	300	00:41:00	15	
4	auto | 75 x 75 | auto	0.01	0.05	3	300	00:40:00	15	
5	auto | 100 | auto	0.01	0.05	3	300	00:40:00	15	
6	auto | 200 | auto	0.05	0.8	3	300	00:40:00	15	
2	auto | 150 x 75 | auto	0.03	0.4	3	300	00:40:00	15	

Obrázek 8-5 Seznam nadefinovaných obecných konfigurací neuronových sítí



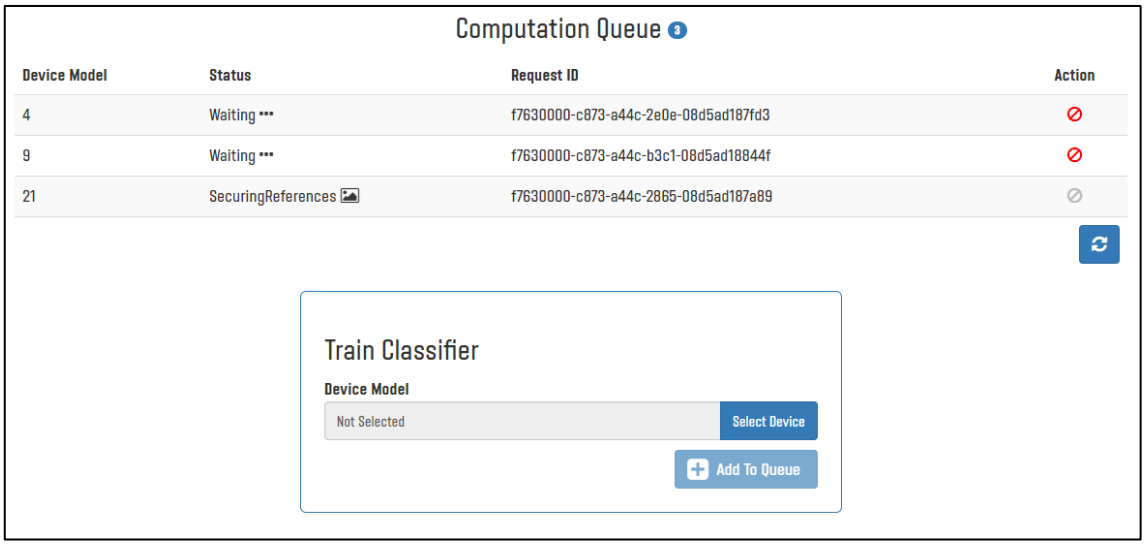
Obrázek 8-6 Detail konfigurace neuronové sítě

Stránka konfigurací pro konkrétní zařízení (Obrázek 8-7) obsahuje seznam konfigurací neuronových sítí, které mají explicitní nastavení. Zařízení, které má tento typ konfigurace definované, jej bude výhradně používat. V opačném případě využívá konfigurací obecných sítí. Po kliknutí na konkrétní nadefinované zařízení je pohled podobný obecným konfiguracím sítí (Obrázek 8-5).



Obrázek 8-7 Konfigurace konkrétních zařízení

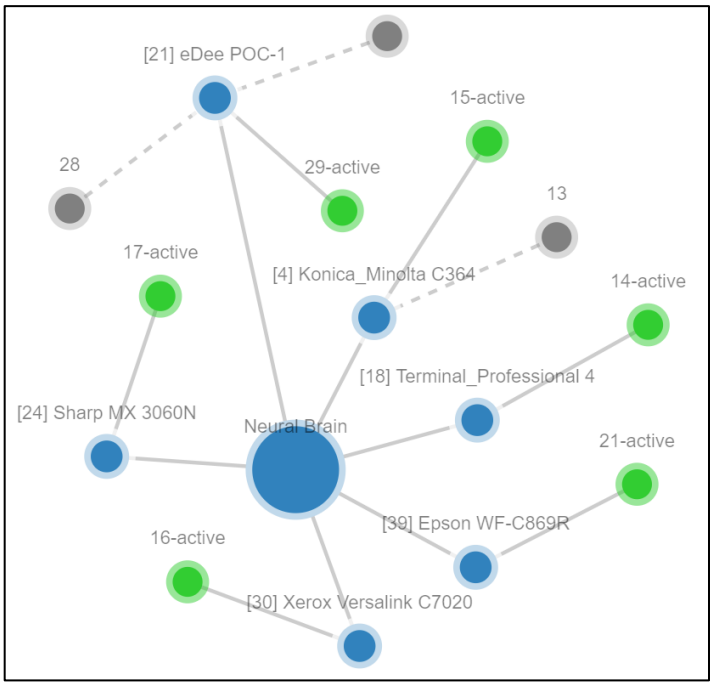
Stránka výpočetního centra (Obrázek 8-8) obsahuje seznam aktuálních požadavků na výpočet nového klasifikátoru. Tabulka se sestává z identifikátoru zařízení, statusu požadavku, identifikátoru požadavku a dostupné akce. Ve spodní části stránky je formulář pro zadání nového příkazu k výpočtu.



Obrázek 8-8 Výpočetní centrum webového rozhraní

Poslední částí této sekce je mapa natrénovaných klasifikátorů (Obrázek 8-9) zobrazující všechna naučená zařízení a jejich klasifikátory. Modře jsou označeny uzly zařízení, zeleně aktivní klasifikátory konkrétního zařízení a šedě neaktivní klasifikátory konkrétního zařízení, které slouží jako záloha, na kterou se dá, v případě špatného natrénování nového klasifikátoru, vrátit. Mapa byla vytvořena pomocí knihovny NetJsonGraph [26], která při poskytnutí dat v JSON formátu,

dokáže vygenerovat graf sítě. Obsah dat musí být podle definované struktury, která utváří hierarchii sítě, která může mít teoreticky nekonečnou hloubku. V rámci této struktury se definují body (nodes) a spojovací přímky (links). Každý definovaný bod v síti může být spojen jednou nebo více přímkami s dalšími body. Po kliknutí na některý z konečných bodů se zobrazí detail obsahující záložky jednotlivých slabých klasifikátorů s obsaženými informacemi o průběhu trénování, u neuronových sítí je navíc zobrazen graf vývoje chyby za uplynulé epochy.



Obrázek 8-9 Mapa natrénovaných klasifikátorů

Poslední částí webového rozhraní je stránka pro definování rozlišovacích regionů kompatibilních tříd (Příloha 1). Po vybrání konkrétního zařízení a způsobu rozlišování mezi třídami, vybere uživatel žádanou třídu, následně její konkrétní variaci a tahem na zobrazeném obrázku označí oblast. V případě textového rozlišování ještě vyplní do textového pole, jaký text se v označené oblasti nachází, a tím i jaký text má být při rozlišování hledán. Stejným postupem vyplní druhou třídu a tlačítkem *Save* může tuto množinu uložit do databáze. Zatím je implementace omezena na 2 třídy, ale nebude problém rozšíření na *N* tříd.

9 VÝSLEDKY

Řešení bylo vyzkoušeno na celkově šesti zařízeních: Konica Minolta C364, eDee POC-1, Sharp MX 3060N, Terminal Professional 4, Epson WF-C869R a Xerox Versalink C7020. Zařízení eDee POC-1 představuje nejobtížnější klasifikaci, jelikož obsah jednotlivých obrazovek je často chudý a velké množství různých tříd si je podobná, proto budou výsledky prezentovány převážně na tomto zařízení.

Pro ověření funkčnosti klasifikátoru byla ručně vytvořena nová testovací množina pro toto zařízení složena ze 61 snímků, které pokrývají veškeré klasifikační třídy. Rovněž u této nové množiny byly navozovány různé okolní podmínky, aby se množina přiblížila reálným podmínkám při běžném provozu. Výsledky naučeného klasifikátoru pro zařízení eDee POC-1 shrnuje následující tabulka (Tabulka 4), ve které jsou zobrazeny přesnosti jednotlivých slabých klasifikátorů, přesnost silného klasifikátoru a časy, za které se celá testovací galerie na daném modelu spočítala. V rámci silného klasifikátoru se slabé klasifikátory vyhodnocují paralelně, proto doba výpočtu zahrnující všechny slabé klasifikátory není součtem dílčích dob, ale v podstatě doba nejpomalejšího slabého klasifikátoru navýšena o časové náklady spojené s agregací výsledků a dodatečných operací.

Tabulka 4 Tabulka přesností a dob výpočtů slabých klasifikátoru a silného klasifikátoru

SURF NN		**SIFT NN**		**SURF SVM**		**FREAK SVM**		**Silný klasifikátor**	
[%]	[s]	[%]	[s]	[%]	[s]	[%]	[s]	[%]	[s]
90.16	81.36	95.08	78.24	98.37	80.88	88.52	81.06	100.00	88.26

V tabulce dosažených přesností (Tabulka 4) mají slabé klasifikátory minimální přesnost 88.52%, což bohatě splňuje podmínku slabých klasifikátorů, která požaduje alespoň 50% přesnost. Silný klasifikátor dosáhl čistých 100%, z čehož lze usoudit, že normalizační a agregační mechanismus slabých klasifikátorů funguje správně. Nutné podotknout, že na každý obraz byl aplikován algoritmus potlačující šum i typu moaré, který je podstatně časově náročný. Následující tabulka (Tabulka 5) porovnává přesnosti a doby výpočtů všech klasifikátorů v závislosti na použití metody pro odstranění šumu. Obrazy, které byly klasifikátorům předloženy včetně šumu, měly obecně horší výsledky, avšak doba výpočtu se razantně snížila. Zda využívat metodu na odstranění šumu, která sice zvyšuje přesnost systému, ale vyžaduje pětinásobek času, je otázkou na operátory robotického systému a bude souviset i s typem prováděných testů. U testů, které běží v noci, je tento problém zanedbatelný. U prezentačních účelů však může

hrát velkou roli, z toho důvodu by byl asi ideální přístup mít možnost metodu na odšumění vypnout i za doby běhu aplikace.

Tabulka 5 Porovnání výsledků klasifikace slabých klasifikátoru a silného klasifikátoru při použití metody na odstranění šumu a její vliv na dobu výpočtu

Šum	SURF NN		SIFT NN		SURF SVM		FREAK SVM		Silný klasifikátor	
-	[%]	[s]	[%]	[s]	[%]	[s]	[%]	[s]	[%]	[s]
Ne	90.16	81.36	95.08	78.24	98.37	80.88	88.52	81.06	100.00	88.26
Ano	83.33	7.02	95.00	9.81	85.00	9.88	91.67	1.37	98.33	16.38

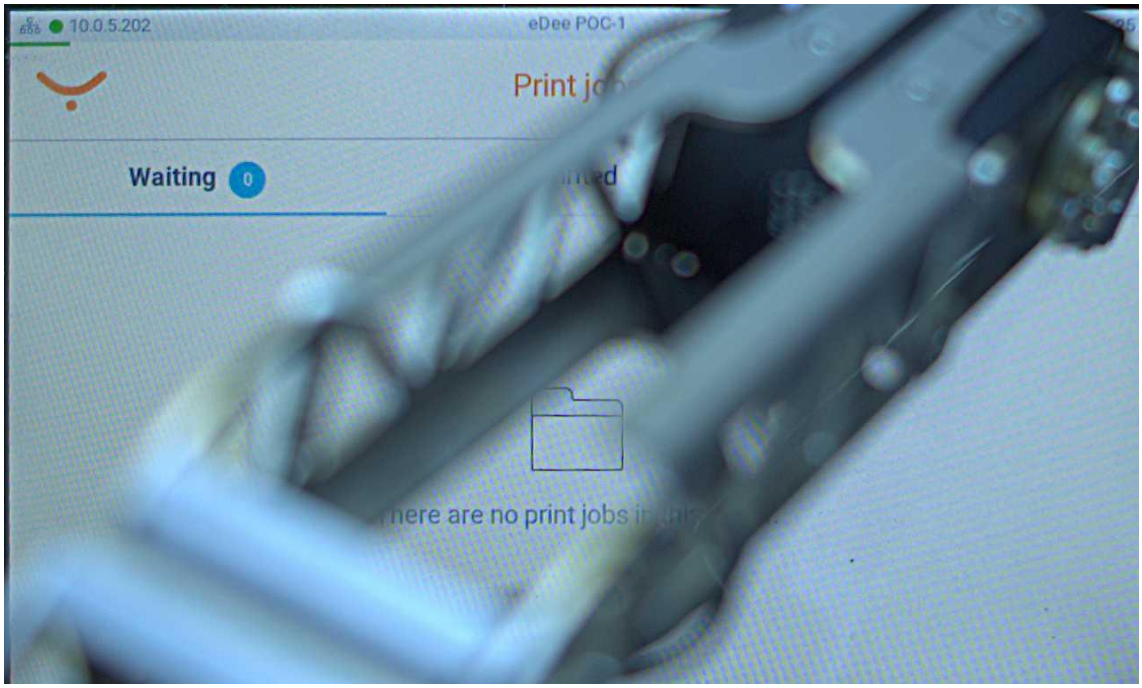
Další neznámou byl vliv velikosti simulované trénovací galerie na celkovou přesnost slabých klasifikátorů i silného klasifikátoru. Tabulka 6 takovou závislost zobrazuje, kde multiplikátor je znásobená velikost celé galerie oproti standartní velikosti. Multiplikátor 1 tedy znamená standartní galerii a multiplikátor 2 dvojnásobnou velikost trénovací galerie, tedy 2x tolik vygenerovaných simulačních obrazů. Obecně lze v tabulce sledovat trend, který se zvětšující se galerií trénovacích obrazů snižuje přesnost všech slabých klasifikátorů. Na výsledný silný klasifikátor nemá zvětšování galerie podstatný vliv, krom dvou případů snižující přesnost až na 95.08%. Lze tedy usoudit, že simulováním více obrazů nemá pro zvýšení přesnosti smysl.

Tabulka 6 Tabulka závislosti velikosti simulované trénovací galerie na přesnost klasifikátorů

Multiplikátor	SURF NN	SIFT NN	SURF SVM	FREAK SVM	Silný klasifikátor
-	[%]	[%]	[%]	[%]	[%]
1	96.72	98.36	88.52	95.08	100.00
2	96.72	98.36	95.08	85.25	100.00
3	86.69	83.61	90.16	91.80	100.00
4	91.80	93.44	88.52	90.16	98.36
5	90.16	91.80	91.80	90.16	100.00
6	81.97	83.61	86.89	83.60	95.08
7	90.16	95.08	85.25	91.80	100.00

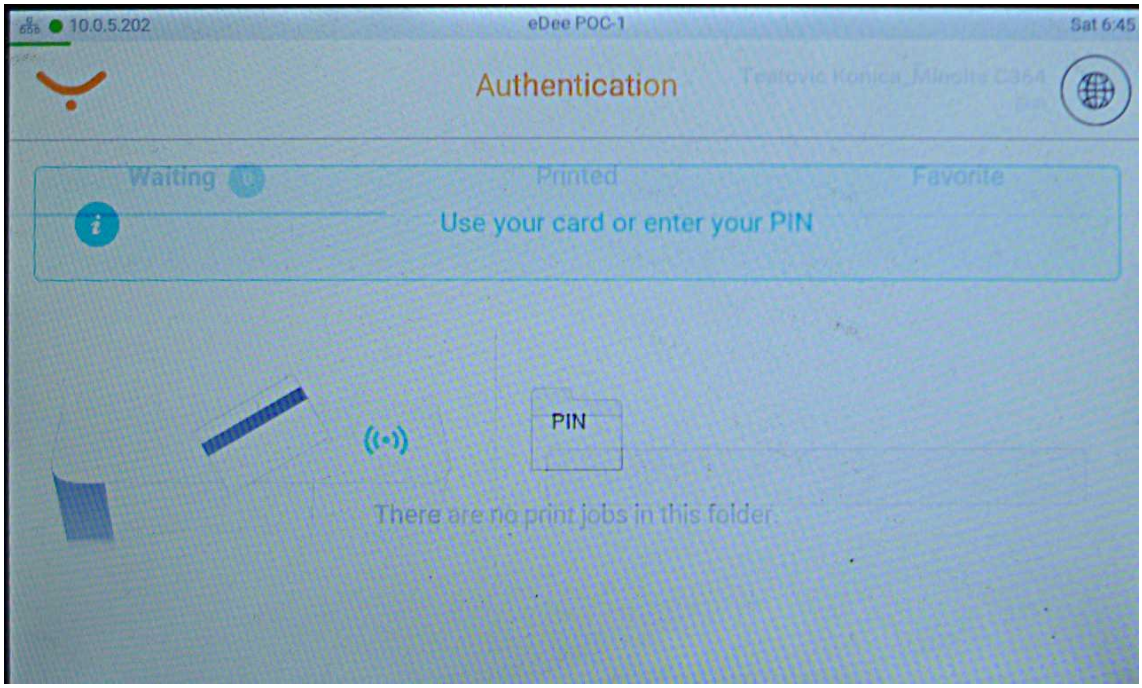
K vyhodnocení výkonnosti klasifikátoru se často využívá matice záměn (confusion matrix), která srovnává požadované a skutečné klasifikace. Definuje tak čtyři segmenty: správně pozitivní, správně negativní, nesprávně pozitivní a nesprávně negativní. U této konkrétní úlohy je však velice obtížné určit hranice, kdy je obraz správně či nesprávně klasifikován. Občas se totiž můžou vyskytnout

takové obrazy, které jsou z menší či větší části zastíněné robotickým manipulátorem (Obrázek 9-1). Takový případ může nastat v reakčním měření, kdy je sledována odezva systému, ve kterém se nahrává sekvence snímků s vysokou rychlostí snímání. Systém se snaží nalézt nejbrzčí výskyt požadované třídy. Pokud se daná třída objeví dříve, než se stihne robotický manipulátor pohnout mimo záběr kamery, nastává případ s překrytím obrazovky robotickým manipulátorem. Při běžných testech je před každou akcí explicitně požadován čistý záběr kamery.



Obrázek 9-1 Obraz pro klasifikaci zastíněný robotickým manipulátorem

Dalším problematickým určením, zda byl obraz správně klasifikován, může být případ zobrazený na následujícím obrázku (Obrázek 9-2). Kamera zachytila obraz přímo v okamžiku překreslování displeje zařízení, nejedná se o chybu kamery či její závěrky, ale na konkrétním displeji zařízení je možné tento jev pozorovat. Obraz tak obsahuje pro klasifikátor známé informace dvou různých tříd. Zda považovat jednu či druhou klasifikaci za správnou, nebo považovat jakoukoliv klasifikaci za nesprávnou, může být předmětem delších debat. Nicméně klasifikátor není na takovéto rozpoznání uzpůsoben a je tak možné, že obrazy tohoto typu budou zanášet jistou chybu do celkové přesnosti.



Obrázek 9-2 Obraz zachycení při překreslování mezi dvěma třídami

Naučený silný klasifikátor pro zařízení eDee POC-1 byl otestován v produkčním provozu po dobu 24 hodin. Za tuto dobu se klasifikovalo 11801 snímků pokrývajících 10 klasifikačních tříd z celkových 19. Zbylé třídy jsou chybové hlášky nebo třídy, které se tak často nevyskytují v běžném běhu testu. Drtivá většina nesprávně klasifikovaných snímků byla částečně kryta robotickým manipulátorem (Obrázek 9-1), složena z více různých obrazů (Obrázek 9-2) nebo byla překrytá načítacím popředím. Veškeré obrazy bez těchto jevů byly klasifikovány správně, včetně obrazů vyžadující rozlišování kompatibilních tříd, a to i s malým translačním posunem, který se v době 24 hodinového testování objevil. Obecně má systém problém, pokud se mu předloží obraz, který zobrazuje neznámou klasifikační třídu. Modely použitých typů neumí na výstupu uvést výsledek typu *neznám,* a tak jsou takové obrazy problematické a vyžadují speciální přístup, který by mohl být budoucím rozšířením této práce.

Závěr

Byla provedena rešerše dostupných metod strojového učení, ve které se probraly metody rozhodovacích stromů, učení založených na instancích, podpůrných vektorů a neuronových sítí. Z těchto metod byly metody podpůrných vektorů a neuronových sítí označeny za nejvíce vyhovující. Obě metody se zdály být schopné k naučení nelineárních závislostí ve vstupních atributech a mají rychlý proces vybavování. Pro modely SVM byly vybrány vstupní data ve formě histogramu zastoupení slov ze slovníků vizuálních slov naučených na deskriptorech SURF a FREAK. Pro neuronové sítě byly vybrány jako deskriptory významných bodů metody SURF a SIFT. Detektory významných bodů pro neuronové sítě je pak agregace výsledků detektoru konvexních tvarů doplněné o významné body nalezené Harrisovým detektorem seřazené podle velikosti jejich odezvy.

Byla sestavena velká galerie trénovacích a testovacích obrazů, které pokrývalo jedno zařízení o 9 klasifikačních třídách. Tento přístup vytváření galerie byl v prostředí Matlab otestován na modelu neuronové sítě a prokázal se funkční. Po bližším přezkoumání principu tvoření galerie, bylo usouzeno, že ač je přístup správný a funkční, tak není vhodný pro danou úlohu, jelikož vyžaduje po obsluze neúnosnou časovou investici. Z toho důvodu byl přednesen druhý přístup ke tvoření galerie, který spoléhá na dostupnost služby pro poskytování existujících referenčních obrazů pokrývající veškeré klasifikační třídy. Pro každý referenční obraz je vygenerovaná sada syntetických obrazů simulující jevy změn okolního prostředí, jako je změna intenzity osvětlení, změna teploty barev, omezený translační posun či rotaci.

V rámci návrhu řešení byla vyřešena struktura silného klasifikátoru, architektura hlavní aplikace a následně i komunikace mezi jednotlivými zúčastněnými komponentami.

Řešení se zabývá procesem učení nového klasifikátoru, které je automatizováno skrze webové rozhraní. Uživatel má možnost vytvořit požadavek na výpočet nového klasifikátoru konkrétního zařízení, který se zařadí do fronty všech požadavků a je časem odbaven. Tímto je splněn pátý bod zadání o doučování na novou třídu nebo zařízení. Nejedná se tedy úplně o postupné doučování, to však pro řešení této úlohy stejně není vhodné, jelikož by se musely znovu vytvářet slovníky vizuálních slov a tím i přeučovat celé SVM modely. Neuronové sítě by při přidání nové třídy byly schopné postupnému doučování, to by však bylo logisticky příliš náročné a praktičtější je tak výpočet celého nového klasifikátoru, který se průměrně vytvoří do jedné hodiny. Proces učení je rozdělen do logických částí obstarávající vytvoření galerie, vytvoření slovníků vizuálních slov, naučení SVM modelů, naučení neuronových sítí a následného uložení do databáze. Dále je v rámci

řešení probrán způsob evaluace příchozí obrazovky již naučených silným klasifikátorem a způsob agregace výsledků slabých klasifikátorů. Je představena metoda normalizace výsledků, která spoléhá na normalizaci pomocí průměrného skóre na testovacích datech po naučení modelu slabého klasifikátoru. V poslední řadě se řešení zabývá tématem kompatibilních tříd a rozlišování mezi nimi, představuje tak dvě techniky rozlišování založené na textu nebo vizuální informaci.

Implementační část je rozdělena do čtyř logických částí: samotné hlavní aplikace, databázového modelu, integrace do existující vizuální zpětné vazby a do existujícího webového rozhraní. V prvé řadě jsou představeny technologie, jež byly použity pro řešení jednotlivých částí. Implementace hlavní aplikace představuje jednotlivé části, které jsou popsány v kapitole Řešení. Zmiňuje také návrhový vzor, který byl při tvoření hlavní aplikace použit. Implementace databázového modelu se věnuje návrhu struktury databáze, a tedy vhodnému uložení dat, které jsou následně používána ve vizuální zpětné vazbě i webovém rozhraní. Integrace do existující vizuální zpětné vazby byla provedena vytvořením NuGet balíčku, který potřebný kód zpřístupní cílovému projektu. Tento kód byl zaintegrován a vyzkoušen ve 24 hodinovém produkčním provozu. Poslední implementační částí byla integrace do existujícího webového rozhraní, které slouží jakožto centrální bod pro ovládání celého systému této práce. Uživatel má možnost zadávat požadavky na učení nových klasifikátorů, rovněž může definovat topologie a parametry neuronových sítí, prohlížet natrénované klasifikátory v uživatelsky přívětivé mapě naučených klasifikátorů a v neposlední řadě umožňuje uživateli definování regionů pro rozlišování kompatibilních tříd.

V kapitole shrnující výsledky výsledného klasifikátoru bylo vybráno referenční zařízení eDee POC-1, které se zdálo být nejproblematičtější pro klasifikaci z důvodu obsahově podobných rozdílných klasifikačních tříd. Toto zařízení podstoupilo sadů testů, které měly za cíl navrženou metodu vyzkoušet a vyjádřit její výkonnost. K tomu byla manuálně vytvořena testovací galerie, která zastupovala reálné změny okolních vlivů. Každý obraz z této galerie byl poskytnut naučenému klasifikátoru a byly zjištěny hodnoty přesností jak slabých klasifikátorů, tak i výsledného silného klasifikátoru. V případě ošetřování šumu klasifikoval silný klasifikátor všechny obrazy správně. Následně byl porovnán vliv použití metody pro odstranění šumu na přesnost modelů a dobu výpočtu. Bylo tak zjištěno, že obecně kvalita klasifikace klesla, avšak čas potřebný k výpočtům se zmenšil na pětinu oproti přístupu s ošetřením šumu. Další test srovnával vliv velikosti generované trénovací galerie na přesnost predikce, kde bylo zjištěno, že zvětšování galerie spíše kvalitu výsledné klasifikace zhoršuje, a tak je nejlepší možností standartní minimální velikost galerie. Posledním testem byl 24 hodinový test v produkčním provozu, za který se klasifikovalo 11801 snímků, kde všechny

snímky bez zmíněných defektů byly správně klasifikovány, včetně snímků, které potřebovali rozlišování kompatibilních tříd.

Tato práce dodala funkční a kompletní řešení pro zadaný problém, tedy klasifikaci obrazovek dotykového zařízení jedním algoritmem, který se bude schopen automatizovaně naučit a bude moct být okamžitě po naučení použit. Metoda má své limitace, jakožto požadavek na odbornou obsluhu při nastavování parametrů neuronových sítí, které jsou však převyšovány výhodami systému. Rovněž má metoda aktuálně problém s klasifikací úplně neznámých tříd, což je pro zvolené typy modelů typické. Řešitelným problémem jsou monotónní obrazovky nacházející se jako načítání mezi jednotlivými definovanými třídami, na to by mohl být vytvořen předřazený algoritmus, který by monotónnost v obraze detekoval a do klasifikátoru by se pak ani nedostal, což může být budoucím rozšířením této práce.

Literatura

[1] HONZÍK, P. *Strojové učení* [online]. Brno: FEKT Vysoké učení technické, 2006, 47-61 s. [cit. 2017-12-21]. Dostupné z: http://midas.uamt.feec.vutbr.cz/STU/Lectures/Honzik%20-%20Strojove_uceni_S.pdf

[2] HONZÍK, P. *Strojové učení* [online]. Brno: FEKT Vysoké učení technické, 2006, 62-75 s. [cit. 2017-12-21]. Dostupné z: http://midas.uamt.feec.vutbr.cz/STU/Lectures/Honzik%20-%20Strojove_uceni_S.pdf

[3] *Image processing, analysis, and machine vision*. 3rd ed. Toronto: hompson Learning, c2008, s. 396-401. ISBN 978-0495082521.

[4] VOLNÁ, Eva. *Neuronové sítě 1* [online]. Druhé, 2008. Ostrava: Ostravská univerzita v Ostravě, 2008, 87 s. [cit. 2017-12-18]. Dostupné z: http://www1.osu.cz/~volna/Neuronove_site_skripta.pdf

[5] JIRSÍK, Václav. Umělá inteligence: Perceptron. 2017. FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

[6] JIRSÍK, Václav. Umělá inteligence: Vícevrstvá neuronová síť algoritmem učení backpropagation. 2017. FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

[7] HORÁK, Karel, KALOVÁ, Ilona, PETYOVSKÝ, Petr, RICHTER, Miloslav. *Počítačové vidění* [online]. Brno: VUT, 2008 [cit. 2017-12-18]. Dostupné z: http://www.uamtold.feec.vutbr.cz/vision/TEACHING/MPOV/Pocitacove_videni_S.pdf

[8] JAN, Jiří. Číslicová filtrace, analýza a restaurace signálů. 2. upr. a rozš. vyd. Brno: VUTIUM, 2002. ISBN 80-214-1558-4

[9] WALLENIUS, HENRIK. Feature point description and classification for urban street scenes using convolutional neural networks [online]. Gothenburg, Sweden, 2016 [cit. 2018-04-24]. Dostupné z: http://publications.lib.chalmers.se/records/fulltext/238494/238494.pdf. Diplomová práce. CHALMERS UNIVERSITY OF TECHNOLOGY.

[10] HORÁK, Karel a Jan KLEČKA. Deskriptory oblastí. Midas [online]. Brno: VUT, 2010 [cit. 2018-04-24]. Dostupné z: http://midas.uamt.feec.vutbr.cz/ROZ/Lectures/05_Deskriptory_oblasti.pdf

[11] ZHANG, Hongyang, Jianfeng HAN, Hui JIA a Yan ZHANG. Features Extraction and Matching of Binocular Image Based on SIFT Algorithm. In: 2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS) [online]. IEEE, 2018, 2018, s. 665-668 [cit. 2018-04-24]. DOI: 10.1109/ICITBS.2018.00173. ISBN 978-1-5386-4201-6. Dostupné z: http://ieeexplore.ieee.org/document/8332857/

[12] ZHANG, Hongyang, Jianfeng HAN, Yunlong GUAN a Yan ZHANG. A SIFT Algorithm Based on DOG Operator. In: 2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS) [online]. IEEE, 2018, 2018, s. 609-612 [cit. 2018-04-24]. DOI: 10.1109/ICITBS.2018.00159. ISBN 978-1-5386-4201-6. Dostupné z: http://ieeexplore.ieee.org/document/8332843/

[13] YANHAI, Wu, Zhang CHENG, Wang JING a Wu NAN. Image registration method based on SURF and FREAK. In: 2015 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC) [online]. IEEE, 2015, 2015, s. 1-4 [cit. 2018-04-24]. DOI: 10.1109/ICSPCC.2015.7338825. ISBN 978-1-4799-8918-8. Dostupné z: http://ieeexplore.ieee.org/document/7338825/

[14] SINGHAL, Neetika, Nishank SINGHAL a V. KALAICHELVI. Image classification using bag of visual words model with FAST and FREAK. In: 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT) [online]. IEEE, 2017, 2017, s. 1-5 [cit. 2018-04-24]. DOI: 10.1109/ICECCT.2017.8117861. ISBN 978-1-5090-3239-6. Dostupné z: http://ieeexplore.ieee.org/document/8117861/

[15] HONZÍK, P. Odhad přesnosti modelu. Midas [online]. Brno: VUT, 2010 [cit. 2018-04-24]. Dostupné z: http://midas.uamt.feec.vutbr.cz/STU/Lectures/07_Odhad_presnosti_modelu.pdf

[16] ERGENE, Mehmet Celalettin a Akif DURDU. Robotic hand grasping of objects classified by using support vector machine and bag of visual words. In: 2017 International Artificial Intelligence and Data Processing Symposium (IDAP) [online]. IEEE, 2017, 2017, s. 1-5 [cit. 2018-04-24]. DOI: 10.1109/IDAP.2017.8090228. ISBN 978-1-5386-1880-6. Dostupné z: http://ieeexplore.ieee.org/document/8090228/

[17] JIANG, Jiale, Duoming WU a Zhen JIANG. A correlation-based bag of visual words for image classification. In: 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)[online]. IEEE, 2017, 2017, s. 891-894 [cit. 2018-04-24]. DOI: 10.1109/ITOEC.2017.8122482. ISBN 978-1-5090-5363-6. Dostupné z: http://ieeexplore.ieee.org/document/8122482/

[18] ACUNA, Gonzalo a Hans MOLLER. Indirect training of Gray-Box Models using LS-SVM and genetic algorithms. In: 2016 IEEE Latin American Conference on Computational Intelligence (LA-CCI) [online]. IEEE, 2016, 2016, s. 1-5 [cit. 2018-04-24]. DOI: 10.1109/LA-CCI.2016.7885719. ISBN 978-1-5090-5105-2. Dostupné z: http://ieeexplore.ieee.org/document/7885719/

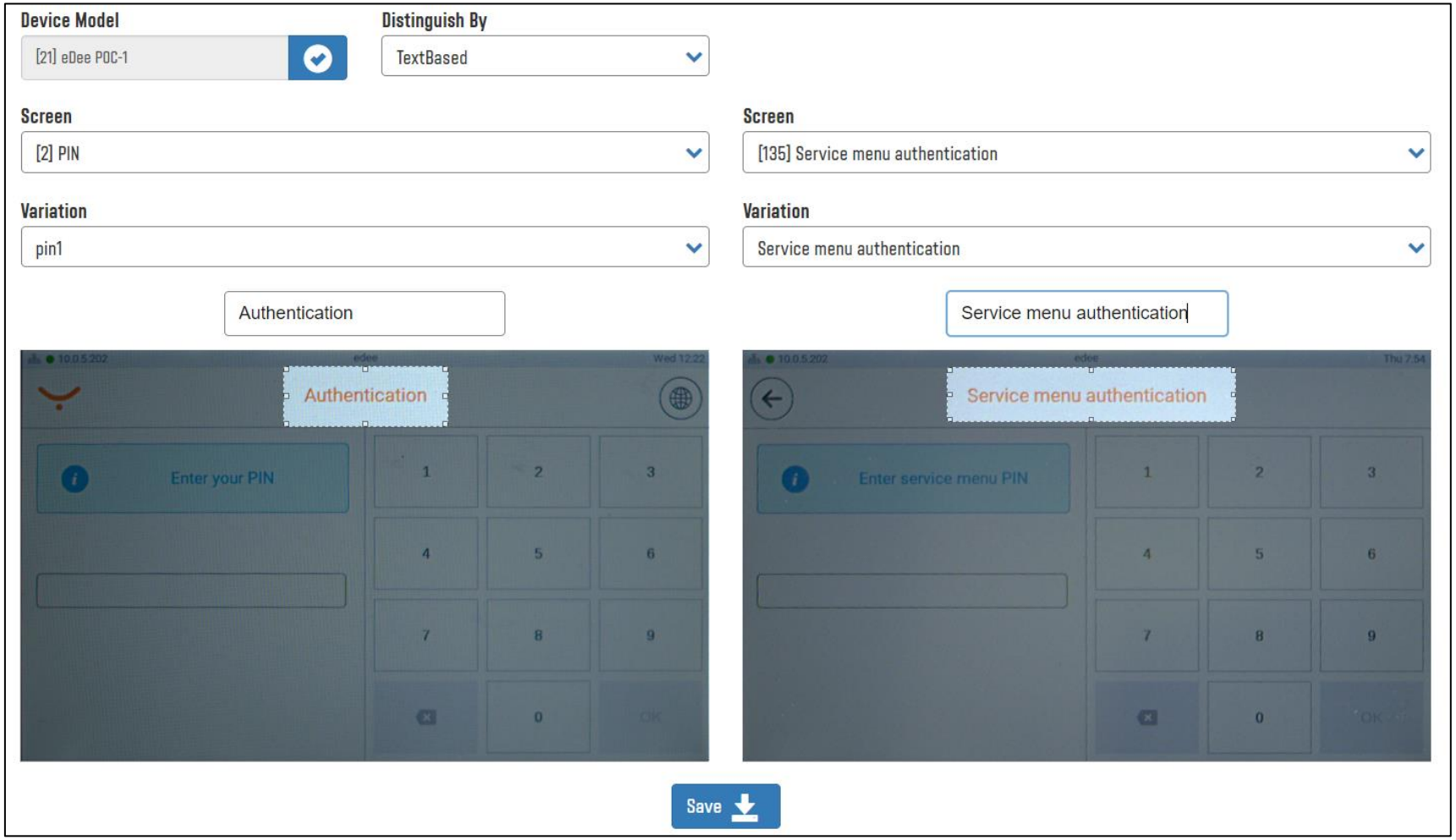
[19] HONZÍK, P. *Strojové učení* [online]. Brno: FEKT Vysoké učení technické, 2006, 29-45 s. [cit. 2018-04-24]. Dostupné z: http://midas.uamt.feec.vutbr.cz/STU/Lectures/Honzik%20-%20Strojove_uceni_S.pdf

[20] TANG, Lei, Payam REFAEILZADEH a Huan LIU. Cross-Validation [online]. 2008, , 1-4 [cit. 2018-04-25]. Dostupné z: http://leitang.net/papers/ency-cross-validation.pdf
[21] OpenCV: About. OpenCV [online]. [cit. 2018-04-26]. Dostupné z: https://opencv.org/about.html
[22] SHIMAT. OpenCvSharp. OpenCvSharp: Github Home [online]. [cit. 2018-04-26]. Dostupné z: https://github.com/shimat/opencvsharp
[23] Introduction. The Accord.NET Image Processing and Machine Learning Framework [online]. [cit. 2018-04-26]. Dostupné z: http://accord-framework.net/intro.html
[24] RIGANTI, s.r.o. Dotvvm: Home. Open source MVVM framework for Web Apps [online]. [cit. 2018-04-26]. Dostupné z: https://www.dotvvm.com/
[25] An introduction to NuGet. *Microsoft docs*[online]. [cit. 2018-04-28]. Dostupné z: https://docs.microsoft.com/en-us/nuget/what-is-nuget
[26] Netjsongraph.js. *GitHub: NetJSON NetworkGraph visualizer based on d3.js* [online]. [cit. 2018-04-28]. Dostupné z: https://github.com/netjson/netjsongraph.js/tree/master

Seznam symbolů, veličin a zkratek

IoT	-	Internet of Things
CI	-	Continous Intergration
RS	-	Rozhodovací stromy
IBL	-	Instance based learning, učení založené na instancích
OCR	-	Optical character recognition, optické rozpoznání znaků
SIFT	-	Scale invariant features transform
SURF	-	Speeded up robust features
FREAK	-	Fast retine keypoints
BOVW	-	Bag of visual words, slovník vizuálních slov
K-NN	-	K-Nearest neighbor, k nejbližších sousedů
SVM	-	Support vector machines, podpůrné vektory
NS	-	Neuronová síť
BP	-	error Back propagation, zpětné šíření chyby
CSV	-	Comma-separated values, souborový formát
IoC	-	Inversion of control
DI	-	Dependency injection

Přílohy



Příloha 1 Webové rozhraní pro definování nových rozlišovacích regionů kompatibilních tříd