TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Mechatronický systém pro třídění LEGO dílků

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Petr Kaspar**
Vedoucí práce: doc. Ing. Josef Chaloupka, Ph.D.

Liberec 2018

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petr Kaspar**
Osobní číslo:	**M15000171**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Mechatronický systém pro třídění LEGO dílků**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s problematikou zpracování a rozpoznávání obrazu a s mechatronickou stavebnicí LEGO NXT.
2. Navrhněte a vytvořte komplexní mechatronický systém ze stavebnice LEGO NXT, který bude umožňovat třídění různých LEGO dílků na základě vyhodnocení obrazové informace.
3. Pro mechatronický systém naprogramujte ovládací program pro třídění dílků, který bude využívat algoritmy pro zpracování a rozpoznávání obrazu, kde budou využity především algoritmy pro rozpoznávání tvaru, velikosti a barvy.
4. Výsledný systém otestujte a upravte tak, aby byla minimalizována chyba špatného rozpoznání a zatřídění LEGO dílku.

Rozsah grafických prací:	**Dle potřeby dokumentace**
Rozsah pracovní zprávy:	**cca 40-50 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **Boogaarts, M., et al: The LEGO MINDSTORMS NXT Idea Book: Design, Invent, and Build, In No Starch Press; 1 edition, ISBN 978-1593271503, 2007**

[2] **Davies, E., R.: Machine Vision - Theory, Algorithms, Practicalities. Morgan Kaufmann Press. UK, ISBN 0-12-206093-8, 2005**

[3] **Šonka, M., Hlaváč V., Boyle. R.: Image processing, analysis, and machine vision. 3rd ed. Toronto: Thomson, 829 s. ISBN 978-0-495-08252-1, 2008**

Vedoucí diplomové práce:	**doc. Ing. Josef Chaloupka, Ph.D.** Ústav informačních technologií a elektroniky
Konzultant diplomové práce:	**Ing. Karel Paleček, Ph.D.** Ústav informačních technologií a elektroniky
Datum zadání diplomové práce:	**12. září 2016**
Termín odevzdání diplomové práce:	**15. května 2017**

Technická univerzita v Liberci - 14 -

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 12. září 2016

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 9. 5. 2018

Podpis:

Poděkování

Rád bych tímto poděkoval svému vedoucímu doc. Ing. Josefu Chaloupkovi, Ph.D. za odborné vedení diplomové práce a poskytování konzultací a cenných připomínek. Dále pak rodině, která mi umožnila studia a podporovala mě v nich a své přítelkyni za trpělivost.

Abstrakt

V rámci diplomové práce byl navržen a sestrojen mechatronický systém, který třídí LEGO dílky na základě zpracované obrazové informace. Snímky kostek jsou pořizovány kamerou připojenou k PC. Činnost třídícího mechanismu a správnou klasifikaci jednotlivých dílků zajišťuje vytvořený ovládací program. Určování barvy a tvaru LEGO kostky se provádí na základě zpracování získaného obrazu a natrénovaného klasifikačního modelu neuronové sítě. Mechatroniský systém byl vytvořen ze stavebnice LEGO MINDSTORMS NXT.

V technické dokumentaci jsou představeny některé využívané segmentační procesy a metody strojového učení. Metoda podpůrných vektorů (SVM) a konvoluční neuronové sítě (CNN) byly v závěru práce podrobeny experimentům, aby se následně nejoptimálnější klasifikátor LEGO dílků implementoval do ovládacího programu. Byl zkoumán i vliv různé datové sady obrázků, které byly za tímto účelem vytvořeny a sloužily jako vstup pro učící se algoritmy.

Klíčová slova

LEGO MINDSTORMS NXT, zpracování obrazu, strojové učení, SVM, CNN, segmentace obrazu

Abstract

This thesis includes design and build a mechatronic system that sorts LEGO pieces using image processing. Images are taken by a camera connected to a PC. The operation of the sorting mechanism and the correct classification of the individual pieces is ensured by the created control program. Determination of the color and shape of the LEGO cube is based on the processing of the acquired image and the trained model of the neural network.

The technical documentation presents some used segmentation processes and machine learning methods. SVM and CNN were experimented at the end of the work and the most optimal LEGO classifier was implemented into the control program. The influence of different sets of images on the learning algorithms was studied.

Key words

LEGO MINDSTORMS NXT, image processing, ML, SVM, CNN, image segmentation

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

NXT	Řídící jednotka robota Mindstorm NXT
ML	Machine Learning
SVM	Support Vector Machines
ReLU	Rectified linear unit
CNN	Convolutional Neural Nnetworks
RGB	Red–Green–Blue, barevný model
CMY	Cyan–Magenta–Yellow, barevný model
HSV	Hue–Saturation–Value, barevný model
COM	Communication port
HOG	Histogram of Oriented Gradients
RBF	Radial Basis Function
CPU	Central Processing Unit
GPU	Graphic Processing Unit
USB	Universal Serial Bus
2D	Dvourozměrný obraz
ROI	Region Of Interesting

Úvod

V dnešním světě se zpracování obrazu využívá v mnoha oborech lidské činnosti. Ať už se jedná o data z bezpečnostního kamerového systému nebo z lékařského tomografu, často se s nimi nějakým způsobem dále manipuluje (provádí se komprese, aplikují se vyhlazovací filtry, zvýrazňují se objekty zájmu atd.). Cílem diplomové práce bylo vytvořit třídící systém, jenž může simulovat průmyslového robota, který za pomoci kamerového systému zasahuje do výrobního procesu.

Díky dostupnosti vysokého výpočetního výkonu a množství nashromážděných dat se v reálném světě stále více prosazují metody strojového učení. Učící algoritmy jsou využívány pro zpracování textu, zvuku, hraní her i ke zpracování obrazu. Právě tomu se věnuje podstatná část práce. Kapitola věnovaná strojovému učení seznamuje s jeho základními principy, představuje algoritmus SVM, ale zejména umělé neuronové sítě. Aby se tyto metody daly prakticky použít a dosahovaly požadované úspěšnosti při klasifikaci, bylo nutné zajistit rozsáhlou množinu trénovacích a testovacích dat.

V rámci diplomové práce byl navrhnut a sestrojen třídící systém, který je ovládán pomocí vytvořeného programu přímo z PC. Využita k tomu byla stavebnice LEGO MINDSTORMS NXT, jejíž hardwarové vybavení a programovací možnosti jsou představeny v první kapitole. Hlavní úlohou mechatronické třídičky je dopravit pod kameru samostatný dílek a po vyhodnocení jej zařadit do příslušné krabičky podle barvy.

Hlavním tématem práce je zpracování obrazu. To je zde rozebíráno ve dvou úrovních. Tou první je proces detekce objektu v obraze. Ten začíná už při získávání obrazu. Následně jsou v kapitole věnované především předzpracování obrazu přiblíženy některé segmentační postupy vedoucí k nalezení LEGO dílku v obraze, který je získán z kamery.

Cílem poslední části diplomové práce bylo navrhnout optimální metodu pro rozpoznání LEGO dílku a implementovat ji do ovládacího programu. Ten by měl být schopen vyhodnotit obraz z kamery a na základě predikce předem naučeným modelem dílek klasifikovat a zařadit ho do správné přihrádky. Několik různých architektur neuronové sítě bylo podrobeno experimentům na vytvořených datových sadách, díky čemuž mohlo být nalezeno optimální nastavení pro natrénování množiny všech využívaných typů LEGO dílků.

Pro úspěšné vytvoření aplikace bylo nutné seznámit se s danou problematikou a v jazyce Python vytvořit dostatečně robustní klasifikační model.

1 Stavebnice LEGO MINDSTORMS

Dánská společnost LEGO, vyrábějící dětskou stavebnici, přivedla na trh v roce 1998 sérii LEGO MINDSTORMS. Do té doby stavebnice obsahovala pouze pasivní prvky. Kostičky s různorodým tvarem se mohly libovolně skládat dohromady. Nyní se však díky aktivním prvkům v podobě motorů mohly výtvory z Lega rozhýbat a doplnit vlastním programem. Pomocí této stavebnice si mohou nejen děti osvojit základy robotiky a programování a rozvíjet tvůrčí a logické myšlení. Tuto hračku tak začaly školy i univerzity používat jako doplněk při výuce.

Od té doby tato série prošla mnoha vylepšeními a dostala další nové prvky v podobě senzorů. Její zatím poslední verze je z roku 2013 LEGO MINDSTORMS EV3. V diplomové práci byla použita souprava LEGO MINDSTORMS NXT, vydaná v roce 2006 a nesoucí produktové označení 9797. Obsahuje řídící kostku NXT, ve které může běžet program a ovládat servomotory, zajišťující pohyb a senzory pro sběr informací z okolí.

Podle dodávaných návodů ke stavebnici a díky mnoha novým dílkům lze postavit různé pohybující se roboty. A to od těch jednodušších, manipulujících s barevnými míčky nebo robotických vozítek sledující čáru vyznačenou na zemi, až po velmi komplexní konstrukční řešení, jako je „Great Ball Contraption“ od Japonského autora Akiyuky.

1.1 Hardwarové vybavení robota

Jedna souprava obsahuje řadu stavebních prvků, hlavně pak ale moduly, které umožňují vykonávat pohyby a získávat informace z okolí. Patří sem tři interaktivní servomotory, dva dotykové senzory, ultrazvukový senzor, optický senzor a zvukový senzor. Další snímače kompatibilní se systémy LEGO MINDSTORMS vyrábí společnost HiTechnic a patří mezi ně například barevný snímač, gyroskop, akcelerometr a kompas. Tyto motory a snímače by však samy o sobě nefungovaly. Nejdůležitějším prvkem je NXT “inteligentní” kostka.

Všechny elektronické prvky jsou kompatibilní s klasickými kostkami stavebnice LEGO a lze je tak jednoduše vestavět do konstrukce.

Obrázek 1.1: NXT brick se třemi servomotory, dotykovým, zvukovým a světelným senzorem a ultrazvuk (převzato z [37])

1.1.1 Řídicí jednotka NXT brick

NXT brick (kostka) je řídicí jednotkou celé robota a umožňuje řídit a využívat ostatní moduly. Disponuje 32bitovým procesorem Atmel ARM. Díky paměti typu RAM se mohou do NXT kostky ukládat soubory, které zůstanou uloženy i po odpojení od zdroje napětí. Těmito soubory se myslí zvukové nahrávky, jednoduché obrázky, ale hlavně programy, ovládající celého robota. K dispozici jsou čtyři vstupy a tři výstupy pro snímače a motory (Obrázek 1.1). Snadné ovládání kostky umožňují čtyři tlačítka a grafický LCD display. Připojení NXT kostky k počítači lze realizovat buď pomocí USB kabelu, nebo je možné použít bezdrátové rozhraní Bluetooth. K napájení jednotky slouží Li-Ion baterie, ale v případě potřeby lze použít i tužkové baterie (6× 1,5V AA). Pro podrobnější přehled technických parametrů NXT kostky je zde tabulka. [2]

Tabulka 1.1: Technické parametry řídící jednotky NXT [2]

Parametr	**Specifikace**
Procesor	**Atmel ARM, AT91SAM7S256** • 48 MHz • 32 bit • 256 KB FLASH • 64 KB RAM

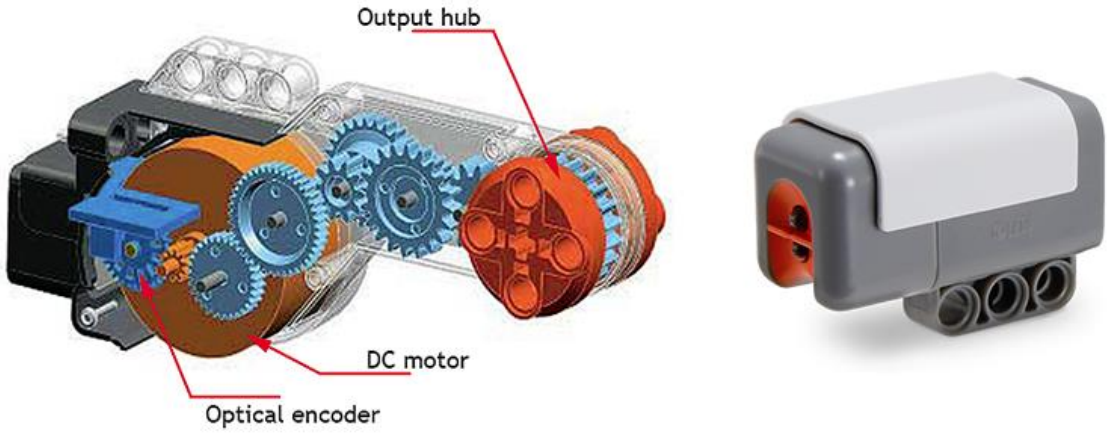
Co-procesor	**Atmel AVR, ATmega48** • 8 MHz • 8 bit • 4 KB FLASH • 512B RAM
Připojení k PC	**USB 2.0** • 12 Mbit/s **Bluetooth, CSR BlueCore 4 v 2.0 + EDR System** • podpora Serial Port Profile (SPP) • interní 47 KB RAM • externí 8 Mb FLASH • 26 MHz • 460,8 Kbit/s
Vstupy / výstupy	**4 vstupní porty (1, 2, 3, 4)** • 6 žilové rozhraní, konektor RJ12 • podpora digitálního i analogového rozhraní • 1 vysokorychlostní port IEC 61158 Typ 4/EN 50170 **3 výstupní porty (A, B, C)** • 6 žilové rozhraní, konektor RJ12 • podpora vstupu pro enkodéry
Display	**Grafický LCD** • rozlišení 100×64, černobílý • viditelná oblast 26×40,6 mm
Reproduktor	**Výstupní kanál s 8bit rozlišením** • podporované vzorkování 2–16 KHz
Ovládání	4 tlačítka
Napájení	Li-Ion baterie nebo 6× 1,5 V AA tužkové články

1.1.2 Servomotor a světelný senzor

Motory lze využít buď jako pohon nebo jako snímače, protože každý motor má zabudovaný optický rotační senzor, který je schopen rozeznat pootočení s přesností až

jednoho stupně. Při jeho ovládání mu lze zadat příkaz k pootočení ve stupních, nebo v počtu otáčkách a k tomu přidat požadovanou rychlost. Díky zabudované převodové soustavě je každý motor schopný vyvinout značnou sílu k rozpohybování požadované soustavy (Obrázek 1.2).

Světelný senzor obsahuje fototranzistor a měří intenzitu světla v okolí. Disponuje i červenou diodou, která se může rozsvítit a podle odraženého světla od sebe rozeznat některé barvy. To ale závisí na mnoha faktorech, například na druhu materiálu, od kterého se světlo odrazí zpět ke snímači.



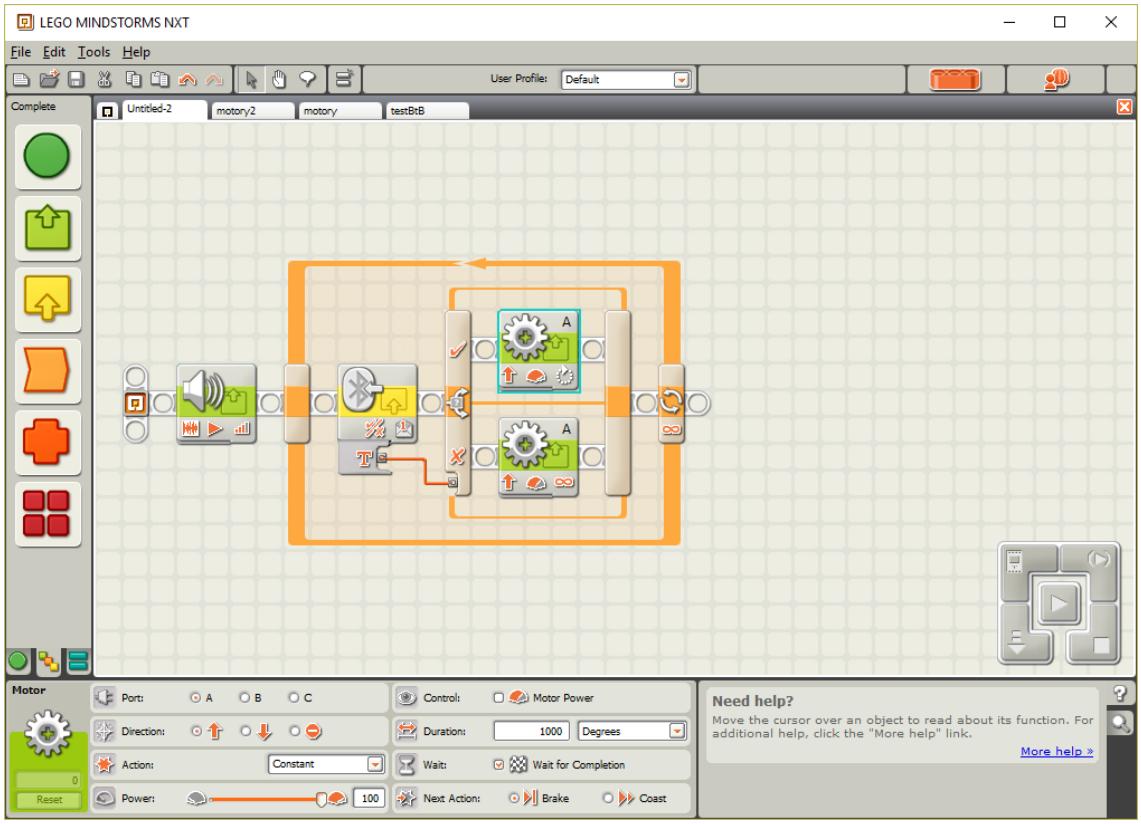
Obrázek 1.2: Ukázka vnitřních převodů servomotoru a světelný senzor [4]

1.2 Možnosti programování NXT

K práci s robotem je zapotřebí mimo jiné základy programování. Nejjednodušeji však lze přimět robota k nějaké řízené činnosti spuštěním základních prvků (motorů, zvukových souborů atd.) přímo na NXT kostce, za pomoci displaye a tlačítek. Můžou se manuálně spustit programy, které byly do zařízení předem zkompilovány a nahrány z PC. Takto připravené programy ve správném formátu mikropočítač pouze vykoná.

K dispozici je celá řada možností, jak takové programy vytvářet. Ukázka uživatelského rozhraní softwaru, který je dodávaný přímo se stavebnicí je vidět na obrázku (Obrázek 1.3). Zde se pracuje v grafickém jazyce NXT-G, který spočívá ve skládání připravených bloků do schématu, které odpovídají požadovaným funkcím. Práce v tomto vývojovém prostředí je principem velmi podobné softwaru LabVIEW a je rovněž

od firmy National Instruments. Tento dodaný software je také užitečný pro případný update NXT firmwaru.



Obrázek 1.3: Ukázka prostředí NXT-G vytvořené společností National Instruments

Pro práci s NXT lze využít i další programovací prostředí, například RoboLab, LabVIEW, Microsoft Robotics Studio, MATLAB a Simulink. Pro komplexnější programy založené také z části na interakci s uživatelem, bude lépe využít rozšířenější textové jazyky, jako třeba Java, C, C++, C# a Python.

Jak je vidět v tabulce nahoře (Tabulka 1.1), NXT kostka se může s PC spojit přes Bluetooth a USB. Pomocí Bluetooth mohou komunikovat i kostky mezi sebou. Tato technologie umožňuje také komunikaci pře mobilní telefon. Pokud je řídicí jednotka NXT s počítačem spojena, jsou k dispozici dva způsoby řízení. Prvním je tak zvané On-Board řízení, kdy je program po napsání uložen do paměti NXT. Zde se daná sekvence příkazů ovládající běh robota může kdykoliv spustit. Druhým způsob řízení je posílání přímých příkazů do řídicí jednotky. V této práci je využita druhá možnost, kdy se za pomocí knihoven v jazyce C# zapínají a vypínají motory v celém třídícím systému.

Použit je open source C# framework AForge.NET, který obsahuje řadu nástrojů pro strojové vidění a umělou inteligenci. Zajímavá třída tohoto framewokru je NXTBrick, umožňující komunikaci přes Bluetooth s NXT kostkou. Tímto propojením lze snadno řídit servomotory a číst hodnoty ze snímačů. Potřeba je znát pouze číslo COM portu. [5]

2 Strojové učení

Strojové učení, vyskytující se často pod zkratkou ML (Machine Learning), je vědní disciplína, zabývající se algoritmy, které dávají počítači schopnost „učit se". Nejčastěji ze vstupních dat, které mu byli předloženy. Zjednodušeně lze říci, že učení je proces, kdy se v tréninkových datech, které jsou vstupem pro algoritmus, identifikují matematické vzory a nabyté zkušenosti se promění ve znalosti. Nalezené vzory se následně aplikují na neznámá data a podle nalezených vzorů se nová data klasifikují do příslušné třídy. [6]

Strojové učení se využívá u takových úloh, u kterých nejsme schopni navrhnou efektivní algoritmus, který by obsáhl všechny možné situace, které mohou nastat.

Například říci o emailu, zda se jedná o spam, či jde o normální zprávu. V takovém případě je vstupem pro rozpoznávací program e-mailový dokument, který v nejjednodušším případě představuje soubor znaků. Jako výstup chceme mít odpověď typu ANO/NE, tedy zda se jedná o spam, či nikoliv. Pro získání správné odpovědi na zcela novou zprávu musíme nejprve systému předložit tisíce jiných zpráv o kterých dopředu víme, do jaké třídy spadají a na těchto datech se systém učí správnému rozpoznání.

Dalšími konkrétními příklady využití strojového učení je rozpoznávání ručně psaných číslic pro automatické třídění poštovních zásilek, dále rozeznání tváří, dopravních značek, strojový překlad. Obchodní řetězce analyzují, kdy a co zákazníci nakupují, banky zase odhadují rizika při poskytování půjček a mnoho dalších. Strojové učení se dostalo do mnoha oblastí lidské činnosti, ve kterých je k dispozici mnoho dat, která jsou zapotřebí analyzovat. [7]

Základní rozdělení algoritmů ML lze provést podle způsobu učení do následujících kategorií:

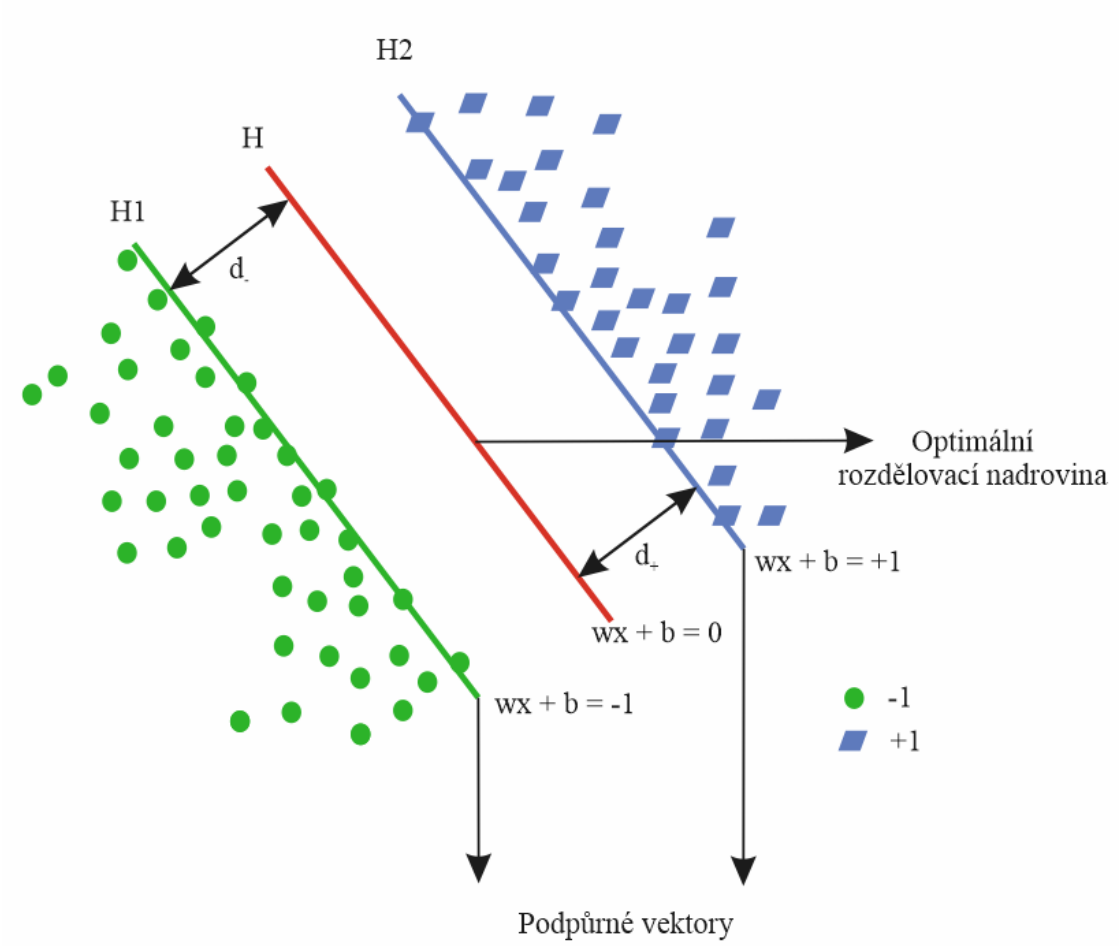
- **Učení s učitelem:** Učení probíhá nad datovou sadou, u které je předem známá její příslušnost ke třídám. Algoritmus hledá vztahy mezi daty, spadající do stejné třídy. Při každém kroku se učící systém upravuje tak, aby se třída aktuálního vstupu shodovala s třídou, do které má správně spadat. Výstup takového systému je tedy klasifikace do konkrétní třídy. [8]
- **Učení bez učitele:** Pro vstupní trénovací data není známý správný výstup. Cílem systému, který se učí bez učitele, je shlukování dat, které pojí nějaké výrazné prvky. Shlukování se tedy provádí na základě podobnosti a nemusí se shodovat s kategoriemi, do který jsou data reálně rozdělena, ale nejsou známa.

- **Kombinace učení s učitelem a bez učitele:** Jedná o kombinaci obou předchozích metod, přičemž jistá část ze vstupních dat je kategorizovaná, ale u ostatních dat správný výstup znám není.
- **Zpětnovazebné učení:** Učící systém je nasazen na úlohu, o které nic neví, ale musí se v ní naučit chovat a vyřešit ji. Systému se dává zpětná vazba, zda daný krok byl správný a vedl blíže k cíli, či špatný a třeba v případě her mohl tah způsobit prohru. Tento typ učení se využívá hlavně při řešení rozhodovacích problémů, jako například vnímání robota a jeho následující pohyb, automatické hraní her (pong, bludiště, šachy, go, atd.), automatické řízení vozidla. [8]

2.1 Support Vector Machine (SVM)

SVM (česky algoritmy podpůrných vektorů) je metoda Strojového učení s učitelem, určená pro klasifikaci předložených dat do jednotlivých tříd.

Základ nejjednodušší metody SVM tvoří lineární klasifikátor, který rozděluje prostor příznaků do dvou tříd. Cílem algoritmu je nalézt takovou optimální nadrovinu, která by rozdělovala trénovací data náležející odlišným třídám, do opačných poloprostorů. Optimální nadrovina by měla být taková, že hodnota minima vzdáleností bodů od roviny je co největší. Hraniční pásmo okolo nadroviny na obě strany je co možná nejširší pruh bez bodů. Toto pásmo je důležité z hlediska kategorizace budoucích dat. Podpůrné vektory popisují nadrovinu a jsou to body ležící na okraji hraničního pásma (Obrázek 2.1). Pokud není možné data plně lineárně separovat (může se jednat o zašuměná data, nebo se jednotlivé třídy částečně překrývají a není proto možné najít jednoznačnou hranici), jedná se o tak zvaný neseparabilní případ. [9]



Obrázek 2.1: Lineární metoda SVM (podle [27])

Obrázek nahoře ukazuje optimální nadrovinu *H*, rozdělující body do dvou tříd a lze ji popsat rovnicí

$$H: x_i w + b = 0 \tag{2.1}$$

kde *w* je normálový vektor nadroviny a *b* je konstanta. Nadroviny H_1 a H_2 jsou rovnoběžné s *H* a mohou být vyjádřeny jako

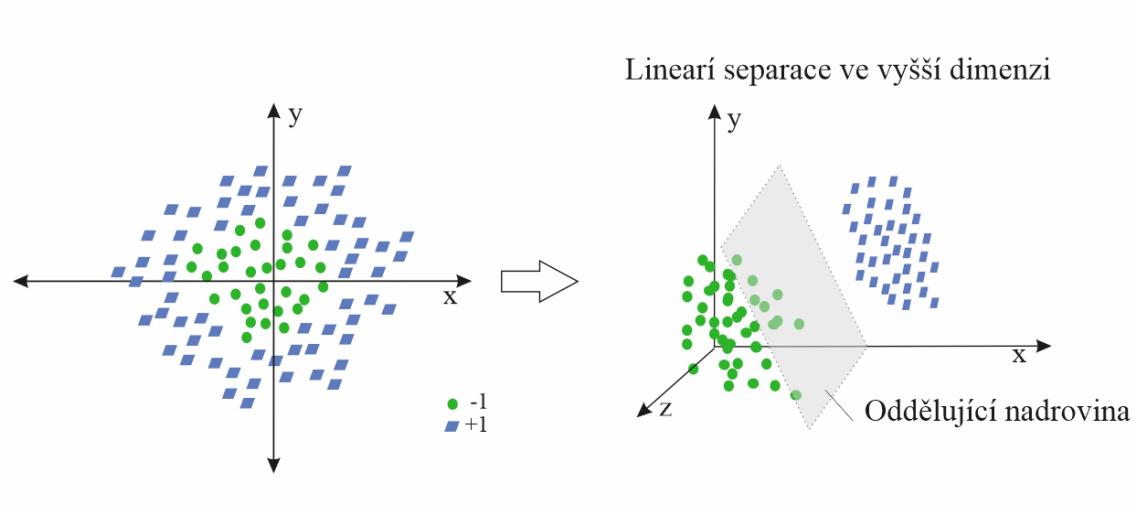
$$H_1: x_i w + b = -1 \tag{2.2}$$

$$H_2: x_i w + b = 1 \tag{2.3}$$

Body, které leží na nadrovině a platí pro ně jedna z předchozích rovností se nazývají podpůrné vektory (support vectors). Vzdálenosti d_+ a d_- jsou měřeny od nadroviny *H* k nejbližším bodům a jejich součet představuje šířku hraničního pásma.

Často však nastávají případy, kdy data lineárně separovat najednou, a proto se začalo využívat nelineární SVM. Algoritmy, které umí najít nelineární oddělovací funkce se obecně učí obtížně a hrozí jim uvíznutí v lokálním extrému, daleko od optima. Hledání nelineární funkce může být také reálně neproveditelné, kvůli vysoké výpočetní náročnosti dané mnoha umělými dimenzemi. [28]

Jádrové algoritmy se snaží využívat efektivní algoritmy pro nalezení lineární hranice a zároveň jsou schopny reprezentovat vysoce složité nelineární funkce. Jedním ze základních principů je převod daného původního vstupního prostoru do jiného, vícedimenzionálního, kde již lze od sebe třídy oddělit lineárně, jak ukazuje následující obrázek (Obrázek 2.2). [29]



Obrázek 2.2: Technika jádrové transformace pro SVM (podle [27])

Na obrázku (Obrázek 2.2) je ilustrace, kde se na data ve 2D prostoru aplikuje polynomiální jádrová metoda, čímž se data převedou do vyšší dimenze a mohou se tak klasifikovat komplexněji a efektivněji. Polynomiální jádrovou funkci lze vyjádřit následovně:

$$f(x) = w\Phi(x) + b \tag{2.4}$$

kde Φ(x) je zobrazení funkce do vyšší dimenze.

2.2 Neuronové sítě

Model umělých neuronových sítí je výpočetní strukturou inspirován neuronovou sítí v mozku. Zjednodušený model mozku se skládá z velkého množství základních

výpočetních jednotek (neuronů), které jsou navzájem propojeny v jednu komplexní komunikační síť, která umožňuje mozku provádět velmi složité „výpočty“. Umělé sítě se snaží tento výpočetní model napodobit. První takové matematické modely byly navrženy již v polovině 20. století. Lidský mozek má přibližně 85 miliard neuronů a každý z nich může mít až tisíc synapsí. Biologické síť tedy obsahuje přibližně 10^{14} až 10^{18} propojení a je tak složitá, že její napodobení ani současné technologie zatím nedovolují. [6]

Umělá neuronová síť (ANN) může být popsána jako orientovaný graf, jehož uzly představují neurony a hrany odpovídají vazbám mezi nimi. Sítě lze rozdělit do dvou hlavních skupin podle struktury: na sítě s dopředním šířením signálu a na sítě se zpětnou vazbou. Při popisu sítě jako orientovaného grafu by zpětná vazba znamenal přítomnost cyklů v grafu. V současnosti se nejčasněji používají struktury s dopředním šířením signálu, kde výstupy z jedné vrstvy jsou vedeny na vstup následující vrstvy. Výstupy z poslední, výstupní vrstvy jsou výstupy z celé sítě. [30]

Síť lze matematicky vyjádřit jako uspořádanou šestici $M = (N, C, I, O, w, t)$, kde:

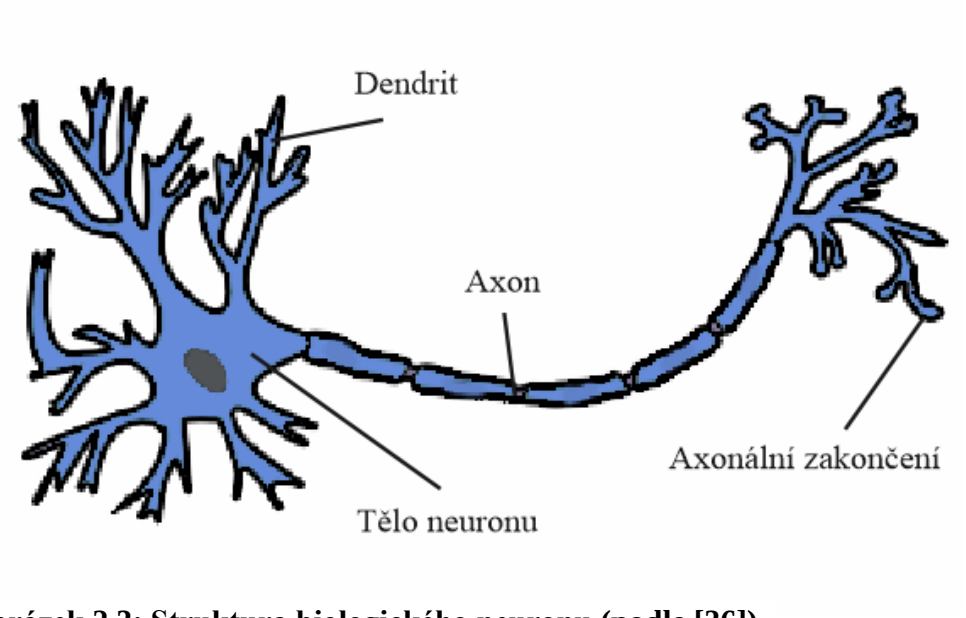
- N je konečná neprázdná množina neuronů (uzlů),
- $C \subseteq N \times N$ je neprázdná množina orientovaných spojů mezi neurony,
- $I \subseteq N$ je neprázdná množina vstupních neuronů,
- $O \subseteq N$ neprázdná množina výstupních neuronů,
- $w: C \rightarrow \mathbb{R}$ je váhová funkce,
- $t: N \rightarrow \mathbb{R}$ je prahová funkce.

Algoritmy neuronových sítí se mohou vymodelovat tak, že zvládnou každou základní metodu učení, tedy s učitelem, bez učitele a zpětnovazební učení. Vhodný typ se vybírá na základě úlohy a dat, které jsou pro ni k dispozici.

2.2.1 Architektura

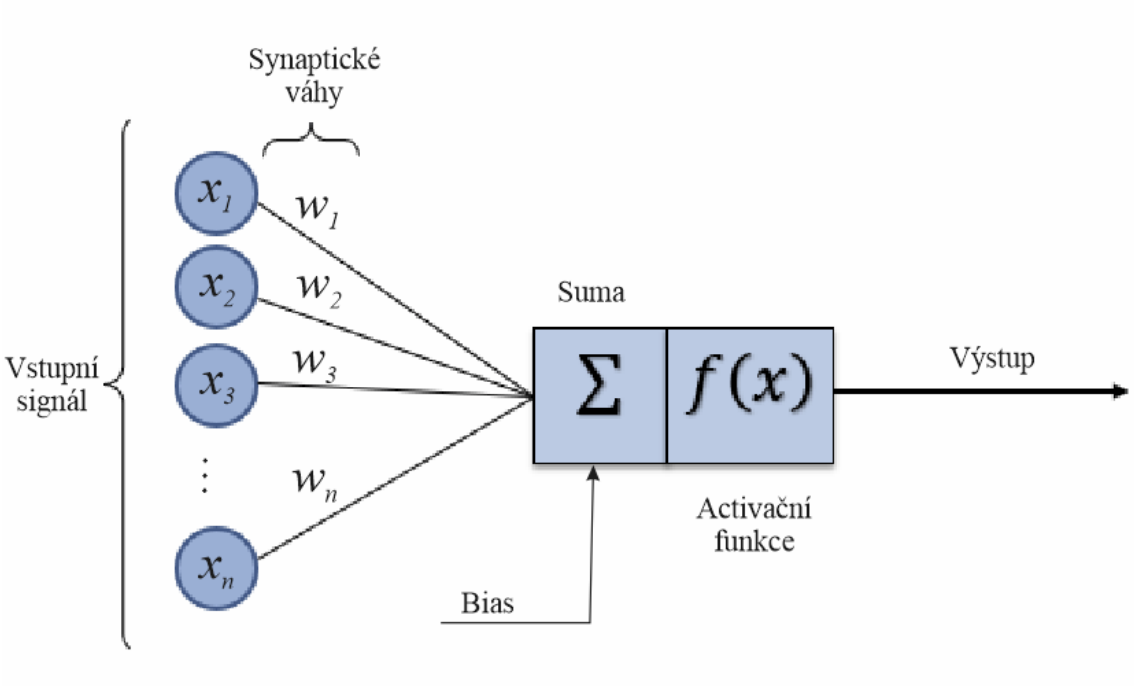
Neuron

Základní stavební jednotka biologické i umělé neuronové sítě je neuron. Struktura toho biologického je ukázána na obrázku dole (Obrázek 2.3). Typicky z jeho těla vystupují kratší výběžky (dendridy) a jeden výběžek delší (axon). Axon je rozvětvený do mnoha takzvaných terminálů, které jsou zakončeny membránou, aby se dostaly do kontaktu s dendrity jiných neuronů. Tomuto propojení se říká synapse a do neuronu se skrz ně přenášejí vzruchy. [31]



Obrázek 2.3: Struktura biologického neuronu (podle [36])

Umělý neuron není úplně analogický k neuronu skutečnému, protože je příliš složitý. Je od něho převzata pouze základní funkcionalita. Každý neuron na vstupu přijímá vážený součet výstupů předchozích neuronů, připojených k jeho vstupním bodům. Na každou příchozí informaci zareaguje výstupní odezvou.



Obrázek 2.4: Model umělého neuronu (podle [32])

Matematický model umělého neuronu se skládá ze tří hlavních částí. Obsahuje vstupní, výstupní a funkční část. Vstupní část se skládá ze vstupů a z přiřazených, nastavitelných vah (synaptické váhy). Na obrázku (Obrázek 2.4) jsou vidět vstupní hodnoty X_1 až X_n. Ty jsou vynásobeny příslušnými váhovými koeficienty W_1 až W_n. Na základě váhových koeficientů mohou být jednotlivé vstupy zvýhodňovány či potlačeny. Následující částí je výkonná jednotka, která zpracuje informace ze vstupu a vygeneruje výstupní odezvu. Předchozí obrázek ukazuje, jak se na výsledek součtu aplikuje funkce (obecně nelineární) a výsledná hodnota této funkce je přivedena na vstup jiných neuronů pomocí třetí, výstupní části. Častěji využívaný model neuronu obsahuje ještě navíc jeden zvláštní vstup, který není připojený k výstupu žádného předchozího neuronu, ale přivádí do něj konstantní veličinu. Tato veličina funguje jako prahová hodnota W_0 (bias) při aktivování výstupu. Když suma váženého součtu vstupů nepřesahuje prahovou hodnotu, tak se neuron neaktivuje a jeho výstup zůstane nezměněný. [30]
Matematicky můžeme formální neuron chápat jako vztah

$$y = F\left(\sum_{i=1}^{n} x_i w_i + \Theta\right) \tag{2.5}$$

kde:

- x_i je hodnota na i–tém vstupu,
- w_i je váha i–tého vstupu,
- Θ je prahová hodnota,
- n je celkový počet vstupů,
- F je aktivační funkce neuronu (obecná nelineární funkce),
- y je hodnota výstupu.

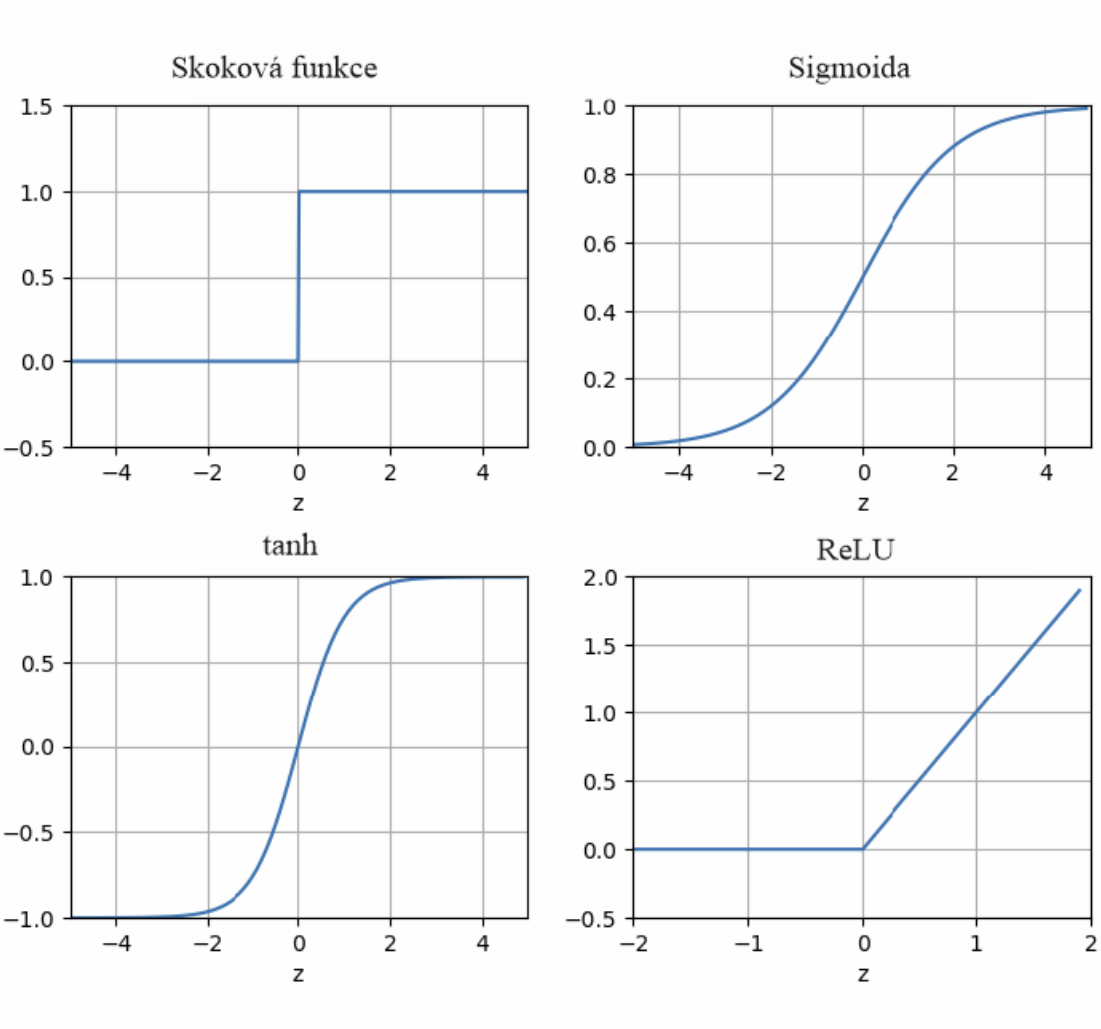
Aktivační funkce

Aktivační neboli přenosová funkce nejvíce ovlivňuje výstup neuronu. Upravuje výstup na hodnoty, které se dále šíří v síti. Nejčastěji se jedná o nelineární funkci. Mezi nejjednodušší patří skoková funkce (někdy též binární), u které práh definuje pouze zlom mezi 0 a 1 (rovnice 2.6). Velmi podobná je funkce signum, která přiřazuje vstupnímu číslu hodnotu z množiny {1, -1, 0}, podle toho, zda vstup funkce je > 0, < 0, nebo = 0.

Mezi další často využívané patří tanh a ReLU (rovnice 2.7), jejichž grafové znázornění je na obrázku dole (Obrázek 2.5).

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}. \tag{2.6}$$

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}. \tag{2.7}$$



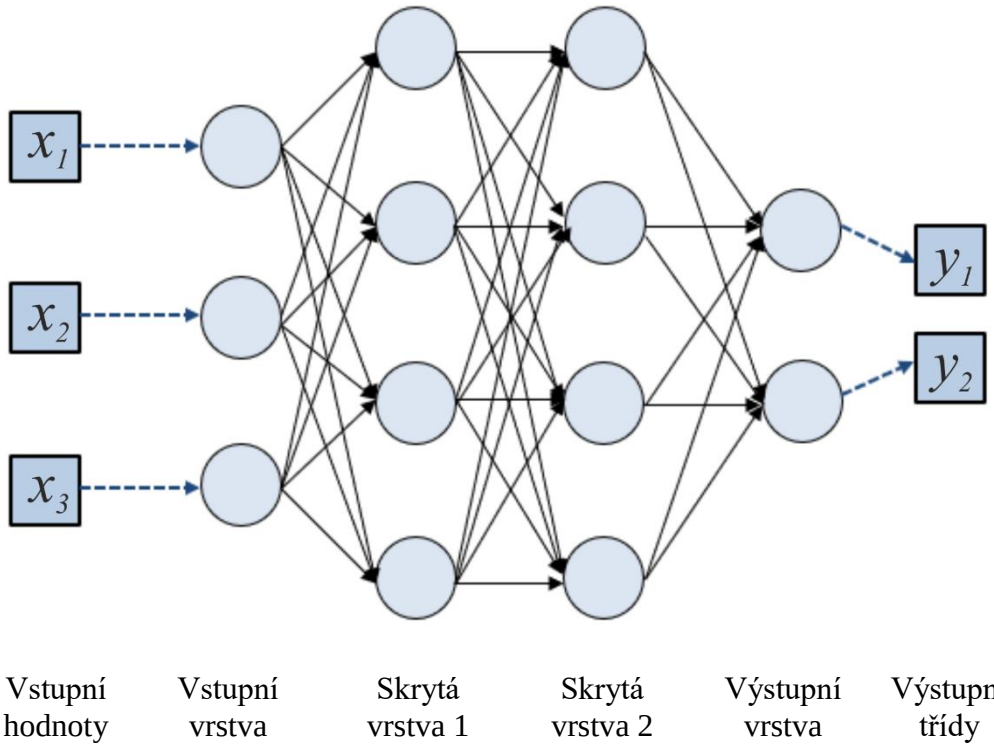
Obrázek 2.5: Graf skokové aktivační funkce, sigmoidy, hyperbolického tangensu a ReLU

Vrstvová struktura umělé neuronové sítě

Vícevrstvá neuronová síť je taková, ve které jsou neurony rozděleny do několika vrstev. Spojené jsou neurony mezi dvěma sousedními vrstvami, ale nejsou spojeny mezi sebou ve vrstvě. První vrstvě se říká vstupní, poslední se říká výstupní a ostatním se říká skryté. [10]

Často se topologie vícevrstvé neuronové sítě zapisuje zkráceně. Na následujícím obrázku (Obrázek 2.6) je síť 3-4-4-2, což znamená, že síť má na vstupu 3 neurony, dvě skryté vrstvy, v každé 4 neurony a ve výstupní vrstvě 2 neurony. Počet synapsí je $3 \cdot 4+4 \cdot 4+4 \cdot 2=36$. Každý neuron, kromě první vrstvy, má ještě vstup s váhou bias, který ale na obrázku vidět není. [10]

Teoreticky na vyřešení libovolného problému stačí síť s dvěma skrytými vrstvami, v praxi se ale často používá vrstev méně. Obecně se doporučuje používat sítě s menším počtem vrstev, jelikož se rychleji učí. Naopak sítě s více vrstvami umí mnohem lépe zobecňovat. [10]



Obrázek 2.6: Příklad architektury vícevrstvé (tří vrstvé) sítě 3-4-4-2 (podle [34])

2.2.2 Konvoluční neuronové sítě

Tento druh hlubokých sítí je obvykle využíván pro klasifikaci obrazových snímků, jejich shlukování podle podobností a rozpoznávání objektů v rámci scény. Jedná se o algoritmy, které dokáží identifikovat obličeje, dopravní značky, ručně psaný text a mnoho dalšího. Konvoluční neuronové sítě (CNN) mohou být aplikovány i na zvukovou stopu, pokud je vizuálně prezentována jako spektrogram. Účinnost konvolučních sítí v rozpoznávání obrazu je jedním z hlavních důvodů pokroku ve strojovém vidění, autonomních vozidlech, robotice, bezpečnosti, lékařské diagnostiky a léčby. [11]

Konvoluční sítě vnímají obrázky jako třírozměrné objekty. Kromě šířky a výšky mají třetí složku a tou je zpravidla barva. Nejčastěji používaný barevný model je RGB. Kombinací tří barev, ze kterých je odvozen i název modelu, tedy červená, zelená a modrá (Red, Green, Blue), se dají získat téměř všechny barvy barevného spektra. Mezi další modely patří třeba CMY, HSV nebo YCbCr. Vstupem pro konvoluční sítě ale mohou být i monochromatické obrázky, tedy takové, jejichž hodnota každého pixelu nese pouze informaci o jeho intenzitě. Hodnota je v rozmezí 0 až 255 a mluví se o šedotónových obrazech. V případě binárních obrazů se jedná pouze o hodnotu 1 nebo 0.

Obvykle se v konvolučních sítích vyskytuje jedna nebo více následujících vrstev:

- Konvoluční vrstva
- Nelineární funkce ReLU
- Podvzorkovací vrstva (pooling)
- Klasifikační vrstva (fully connected layer)

Konvoluční vrstva

Mozky savců zpracovávají obrazy ve vrstvách s rostoucí složitostí. První vrstva rozlišuje pouze základní atributy, jako jsou linie a křivky. Na vyšších úrovních již mozek rozezná barvy a hrany a určí, zda se jedná například o dům nebo ptáka. V konvolučních sítích je proces rozpoznávání podobný. Využívají se k tomu matice vah nazývané filtry, které detekují specifické atributy, jakými mohou být diagonální či vertikální hrany apod. Při průchodu obrazu dalšími vrstvami jsou filtry schopny rozpoznat složitější útvary. [35]

Výstupem konvoluční vrstvy je aktivační mapa. Ta se získá tak, že se provede konvoluce na vstupním obrazu a jako konvoluční jádro se použije pole vah neboli filtr. Obvykle je filtr menších rozměrů a má stejný počet dimenzí, jako vstupní pole. Větší filtry podstatně zvyšují výpočetní náročnost celé úlohy.

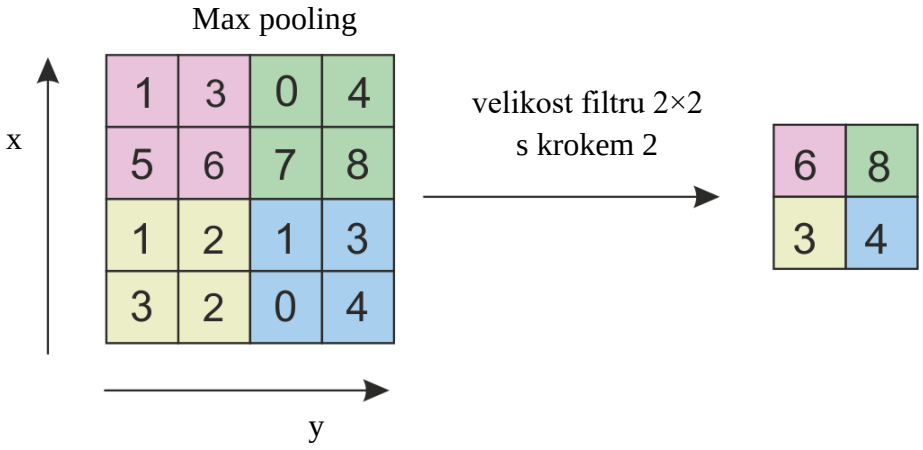
ReLU

Tato nelineární operace běžně následuje za konvoluční vrstvou a nahrazuje všechny záporné hodnoty pixelů na aktivační mapě funkcí nulou.

Podvzorkovací vrstva

Tato vrstva (častěji pod anglickým názvem „pooling layer“) pracuje nejčastěji s výstupem konvoluční vrstvy, zmenšuje velikost pole obrazových hodnot a zjednodušuje tak výpočet pozdějším vrstvám. Pooling layer také přispívá ke schopnosti síti, lokalizovat prvky bez ohledu na jejich umístění v obraze. Síť je tak méně citlivá na malé změny v umístění objektu. [35]

Existuje více způsobů, jak tuto vrstvu implementovat, avšak v praxi se ukazuje jako nejjednodušší a nejúčinnější maximální sdružování (angl. max pooling). Tato funkce může například shrnout oblast velikosti 2×2 neuronů z předchozí vrstvy, do jednoho čísla, které bylo ve vstupní oblasti největší. Pokud by tedy vstup měl velikost 224×224, tak po výstupu z této vrstvy by měl rozměr 112×112. Princip je znázorněn na následujícím obrázku.



Obrázek 2.7: Max pooling

Klasifikační vrstva

Klasifikační vrstva (ang. fully-connected layer) je umístěna na konci sítě a její výstupem je rozhodnutí o klasifikaci vstupního obrazu. Výsledkem je 1D vektor, který udává informace o pravděpodobnosti klasifikace vstupního obrazu pro každou třídu.

3 Zpracování digitálního obrazu

Digitální zpracování a analýza obrazových dat je v dnešní době aktuální, velmi využívaná a rozvíjející se vědeckotechnická disciplína. Data jsou získávána z nejrůznějších zdrojů, například fotoaparátu, kamery, lékařských zobrazovacích technik (ultrazvuk, tomograf) atp. Využití má při rozpoznávání textů, v kamerových bezpečnostních systémech, řídících systémech průmyslových robotů a autonomních strojů, pro porozumění objektům reálného světa zachycených v obraze počítačem a mnohém dalším. Díky dostupnosti a malým rozměrům výkonných výpočetních čipů se může zpracování obrazu implementovat i do různé spotřební elektroniky. [17]

Zpracování lze provádět na dvourozměrných signálech z různých důvodů. Může jít o snahu vylepšit digitální obraz, upravit ho, nebo z něj získat užitečné informace. Průběh zpracování lze rozdělit do několika základních kroků, které se můžou měnit v závislosti na zadání konkrétní aplikace:

- Snímání a digitalizace
- Předzpracování
- Segmentace
- Rozpoznávání
- Klasifikace

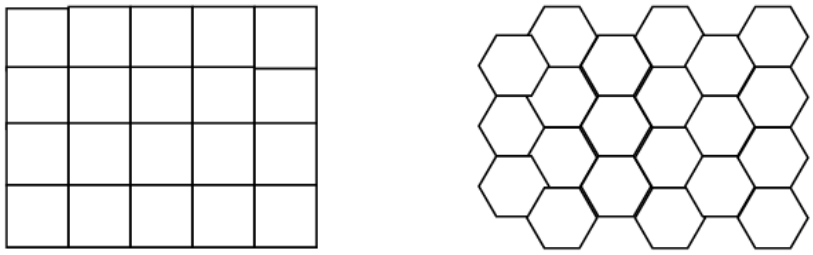
3.1 Snímání a digitalizace

Pokud prováděná úloha začíná pořízením obrazu, vyplatí se věnovat snímání obrazu patřičné úsilí. Správné nasnímání sledovaného objektu pro danou úlohu vede ke zjednodušení následujících kroků, při dalších fázích zpracování. Základem je mít vhodné světelné podmínky.

Digitalizace spočívá ve vzorkování obrazu do diskrétního rastru, tedy do matice o $M \times N$ bodech. Hodnota vzorku přitom zůstává reálné číslo. Tyto spojité jasové úrovně každého vzorku rozdělí kvantování do K intervalů. Díky kvantování nabývá jasová funkce v digitalizovaných obrazech celočíselných hodnot. Čím jemnější je vzorkování (čím větší M, N) a kvantování, tím lépe je aproximován původní spojitý obrazový signál. Určení intervalu vzorkování se řídí podle Shannonovy věty, která říká, že nejmenší detail v digitálním obraze musí být minimálně dvojnásobkem vzorkovací frekvence. Vzorkovací frekvence je v tomto případě rychlost střídání intenzit barev. [1]

Digitální obraz se rozděluje podle způsobu reprezentace obrazových informací na dva základní typy. Prvním je vektorový obraz, který je popsaný pomocí základnách geometrických objektů, jako jsou body, přímky, křivky a polygony. S tím se ale v této práci pracovat nebude. Druhým způsobem popisu obrazové informace je pomocí rastrové grafiky. Zde je obraz složen ze základních bodů – pixelů, které jsou uspořádány do pravoúhlé mřížky. Každý bod nese informaci o své poloze a barvě. Důležitými parametry bitmapy jsou rozlišení a barevná hloubka.

Součástí digitalizace je volba vzorkovací mřížky. Používané mřížky jsou čtvercové, trojúhelníkové a hexagonální (Obrázek 3.1). V praxi se nejvíce používá čtvercová mřížka. Její nevýhodou je především měření vzdáleností a spojitosti objektů. Může v ní také vznikat tak zvaný paradox protínajících se úseček. Hexagonální mřížka většinu těchto problémů řeší, ale není zase vhodná pro některé operace, jakými jsou např. Fourierova transformace. [12]



Obrázek 3.1: Čtvercová matice a hexagonální matice

V této práci se používá pouze čtvercová mřížka. Obecně je vzdálenost dvou bodů v této mřížce, ležících na souřadnicích *(i, j)* a *(x, y)*, chápána jako Euklidovská vzdálenost D_E, definována následujícím vztahem. [12]

$$D_E = \sqrt{(x-i)^2 + (y-j)^2}\,. \tag{3.1}$$

Nevýhodou tohoto způsobu měření vzdálenosti je složitější výpočet kvůli odmocnině a skutečnost, že výsledkem je neceločíselná hodnota. Vzdálenost mezi dvěma body si lze také představit jako nejmenší počet kroků jednotkové vzdálenosti nutných pro přesun z výchozího do cílového bodu v pravoúhlé mřížce. Čtvercová vzorkovací mřížka nám umožňuje mít 4-sousedství a 8-sousedství (Obrázek 3.2). [1][12]

Obrázek 3.2: Okolí obrazového elementu při 4-sousedství a 8-sousedství

Vzdálenost dvou bodů v těchto mřížkách na souřadnicích *(i, j)* a *(x, y)* jsou pak definovány podle vztahů dole. D_4 pro kroky ve svislém a vodorovném směru a D_8 po přidání kroků po diagonálách.

$$D_4 = |x - i| + |y - j| \tag{3.2}$$

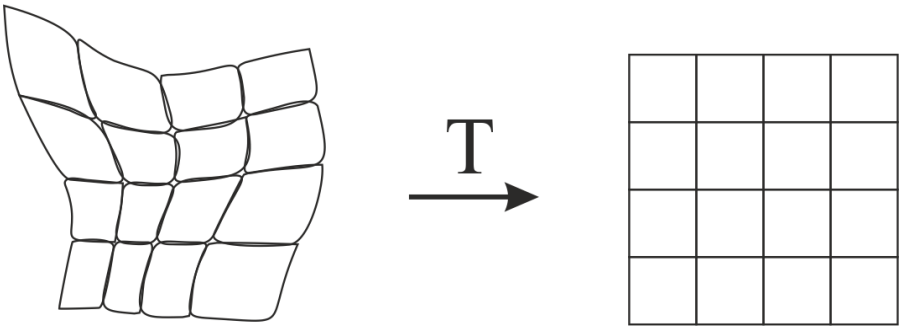
$$D_8 = max\{|x - i|, |y - j|\}. \tag{3.3}$$

3.2 Předzpracování obrazu

Pro další kroky rozpoznání obrazu je potřeba odstranit důsledky nevhodných podmínek při průběhu snímání. Mezi operace, které se u předzpracování nejčastěji aplikují, patří různé úpravy jasu, kontrastu a histogramu. Další významné metody jsou geometrické transformace a aplikace filtrů. Výstupem předzpracování je obraz, u kterého je potlačen šum a rozmazání, odstranění zkreslení a utlumení nebo zvýraznění konkrétních rysů v obraze.

3.2.1 Geometrická transformace

Geometrické transformace hraje při práci s obrazem významnou roli. Využívá se všude tam, kde je třeba provést úpravy obrazu, jako je zvětšování, posouvání, pootáčení, odstraňování geometrických zkreslení apod. Geometrickou transformaci si lze představit jako matici funkcí, které transformují každý pixel obrazu *(x,y)* do nové pozice *(x',y').* [13]



Obrázek 3.3: Geometrická transformace v rovině (podle [1])

3.2.2 Lokální filtrace obrazu

Při lokálním předzpracování se používají lineární nebo nelineární metody transformace. Lineární operace pracují s hodnotou výstupního obrazu *g(x,y)* jako s lineární kombinací hodnot vstupního obrazu *f(x,y)* v malém okolí zvoleného pixelu *(x,y).* [13]

Podle účelu použití se mohou metody lokální filtrace obrazu dále obecně rozdělit do dvou skupin. První skupina, vyhlazování obrazu, provádí potlačení šumu a osamocených fluktuací hodnot obrazové funkce. Aplikace této metody vede k potlačení vyšších frekvencí obrazové funkce. Druhá skupina, detekce hran, nazývána také gradientní operátory, se snaží z hodnot v okolí reprezentativního pixelu odhadnout derivaci obrazové funkce. [14]

Významným nástrojem pro metody lineární transformace je konvoluce. Konvoluce se pro diskrétní funkce může psát jako posloupnost

$$g(x,y) = f(x,y) * h(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x-i, y-j) \cdot h(i,j) \qquad (3.4)$$

kde *g(x,y)* je výstupní obraz, *f(x,y)* je vstupní obraz, a *h(x,y)* je konvoluční maska. Součty se provedou přes všechna čísla i a j popisující souřadnice pixelů v okolí sledovaného pixelu. Konvoluční maska h ve tvaru dvourozměrné matice se rovněž označuje jako konvoluční filtr nebo konvoluční jádro. Obvykle se používá lichý počet řádků a sloupců, pak může být sledovaný pixel uprostřed takto definovaného okolí a současně také uprostřed konvoluční masky. [13]

Lineární metody vyhlazování

Některý druhy šumu, který se v obraze vyskytují, lze odstranit metodou obyčejného průměrování, která se provádí na nejbližším okolí každého bodu v obraze za pomoci konvoluce. Sledovaný bod leží uprostřed masky, o velikosti například 3×3, a spolu s nejbližším okolím je roznásoben skupinou devíti koeficientů, tvořící konvoluční jádro. Příklad konvoluční masky h s okolím 3×3 je následující:

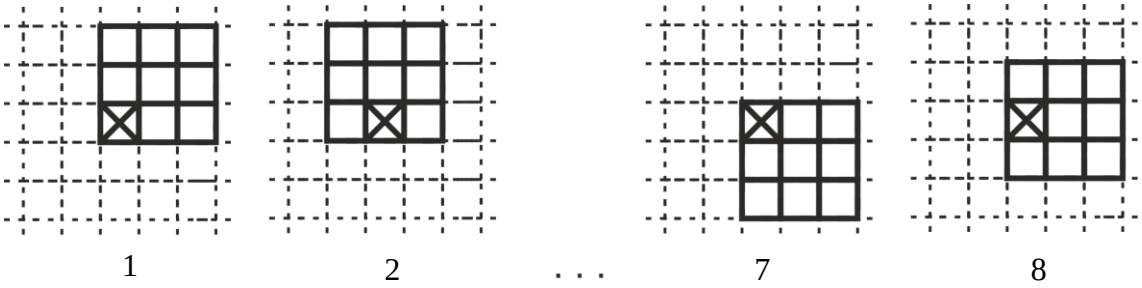
$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.5}$$

Konvoluční jádro může mít libovolnou velikost a hodnoty. Vždy však záleží, na jaká data a za jakým účelem se aplikuje. Nejčastěji má 3×3, 5×5, 7×7 a podobně. Někdy je vhodné přidat váhu na středová bod masky, nebo jeho 4-okolí. Příklad takových masek h je

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.6}$$

Nelineární metody vyhlazování

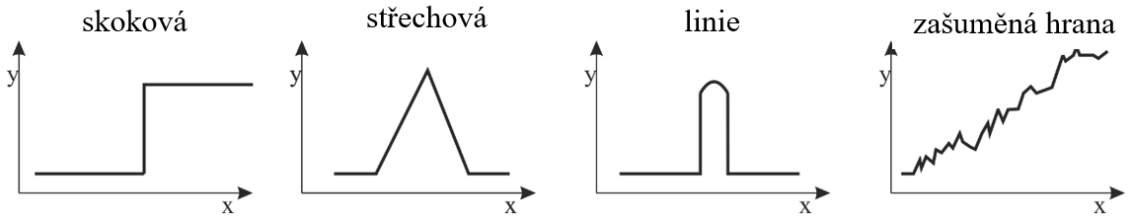
Nelineární metoda vyhlazování obrazu určuje novou hodnotu aktuálního bodu pomocí rotující masky. Nelineární filtry bývají robustnější a v některých případech určí lépe správnou hodnotu reprezentativního pixelu. Rotující maska velikosti 3×3 dává 8 možností své polohy. V každé poloze se vypočítá jasový rozptyl bodů v masce a ta s nejmenším rozptylem se použije pro určování hodnoty bodu, kolem kterého rotuje (Obrázek 3.4). Zkoumané okolí aktuálního pixelu má tedy velikost 5×5. [13] [15]



Obrázek 3.4: Osm možných poloh rotující 3×3 masky (podle [15])

Detektory hran

Detekce hran je postup v digitálním zpracování obrazu, sloužící k nalezení oblastí pixelů, ve kterých se výrazně mění hodnota jasu. Hrana v objektu se nemusí krýt s hranicí mezi objekty ve scéně, hrany mohou vznikat a zanikat v závislosti na úhlu pohledu. Hrany jsou na hranici objektů nebo rozhraní dvou barev či světla a stínu tzn. skoková hrana. Výrazné jsou i u trojrozměrných objektů tzv. trojúhelníková hrana. Tenká linie je čára, která má z obou stran plochu s přibližně stejnou jasovou hodnotu. Typická hrana na rozdíl od teoretické bývá ovšem zašuměná (Obrázek 3.5). [16]



Obrázek 3.5: Jasové profily nejběžnějších hran (podle [14])

Gradientní metody hledání hran využívají skutečnosti, že v místě hrany má absolutní hodnota první derivace průběhu jasu vysokou hodnotu. Velikost hrany, nebo též intenzitu kontury popisuje hodnota derivace v daném bodě. Je-li hrana definovaná jako náhlá změna jasových hodnot, bude v místě hrany velká hodnota derivace jasové funkce. Maximální hodnota derivace bude ve směru kolmo na hranu. Kvůli jednoduššímu výpočtu se ale hrany detekují jen ve dvou, resp. ve čtyř směrech. Velká skupina metod na detekci hran aproximuje tuto derivaci pomocí konvoluce s vhodným jádrem. Operátory, které velikost hrany stanovují a které se obvykle aplikují na každý bod obrazu, jsou nazývány hranovými operátory. [16] [17]

Hojně používaným gradientním operátorem je Laplacův operátor, který aproximuje druhou derivaci. Je invariantní vůči rotaci, ale udává pouze velikost hrany, nikoliv její směr. Konvoluční jádro o velikosti 3×3 využívané pro výpočet, je definované pro 4-okolí (L_4) a 8-okolí (L_8) jako:

$$L_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad L_8 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{3.7}$$

Některé verze Laplacova operátoru kladou větší váhu na středový bod masky. Následující masky již nemají vlastnost invariance vůči otočení.

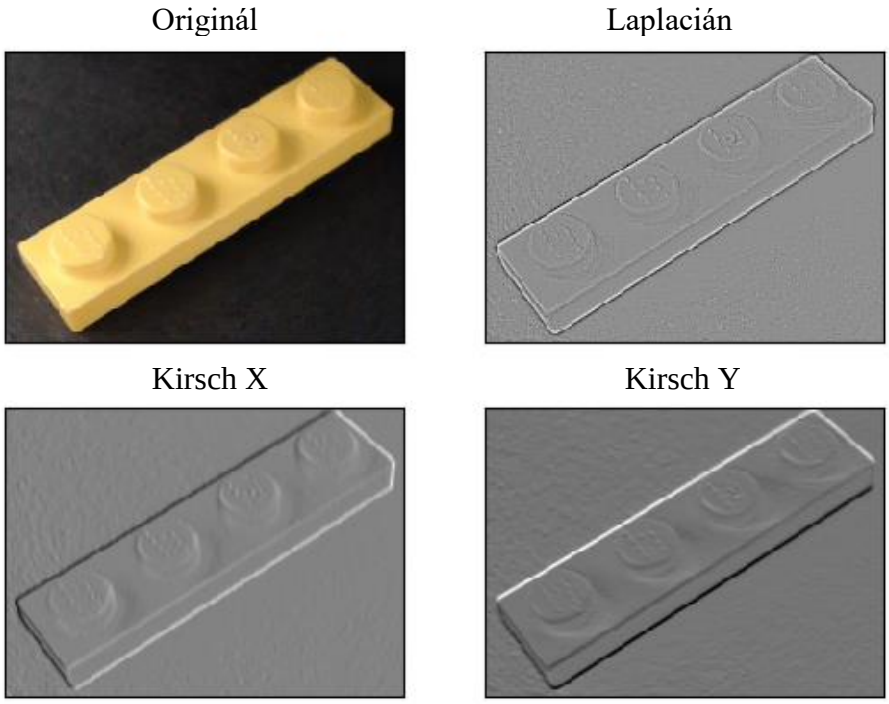
$$h = \begin{bmatrix} 2 & -1 & 2 \\ -1 & 4 & -1 \\ 2 & -1 & 2 \end{bmatrix}, \qquad h = \begin{bmatrix} -1 & 2 & -1 \\ 2 & 4 & 2 \\ -1 & 2 & -1 \end{bmatrix} \tag{3.8}$$

Metoda založená na použití Laplacova operátoru je výrazně citlivá na šum. I při malém zašumění obrazu je detekováno značné množství falešných hran. Problém lze řešit filtrací šumu, která se provede ještě před tím, než jsou hledány hrany. Další nevýhodou mohou být dvojité odezvy na hrany, které odpovídají tenkým liniím v obraze. [16] [14]

Mezi Další zajímavé hranové detektory patří například Prewittové, Sobelův či Kirschův operátor, aproximující první derivaci. Gradient je v těchto případech odhadován pro osm různých směrů. Výsledek konvoluce při použití masky 3×3, je největší hodnota reprezentující směr hrany. Následující masky velikosti 3×3 jsou pro Kirschův operátor. Znázorněny jsou první tři, ostatní lze snadno vytvořit jejich rotací. [15]

$$h_1 = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix}, h_2 = \begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix}, h_3 = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix}, \ldots \tag{3.9}$$

Na následujících obrázcích (Obrázek 3.6) je znázorněn výsledek konvoluce za použití Laplacova gradientního operátoru pro osmi okolí. Dále je vidět rozdíl mezi Kirschovým operátorem pro detekci horizontálních a vertikálních hran. Síla hrany je indikována světlostí pixelu. Silné hrany jsou skoro bílé, což lze využít při dalším zpracování, například použitím prahování.



Obrázek 3.6: Výsledky aplikace Laplacova a Kirschova operátoru

3.3 Segmentace obrazu

Cílem zpracování digitálního obrazu většinou není zpracovávat obraz jako celek, ale soustředit se pouze na jeho vybranou část. Touto částí bývají objekty, které chceme podrobit dalšímu zpracování a říká se jim také někdy regiony zájmů. Proto je jedním z nejdůležitějších kroků vedoucích k analýze obsahu zpracovávaných obrazových dat segmentace.

Jedná se tedy o proces extrakce, v němž jsou objekty, které mají úzkou souvislost s předměty či oblastmi reálného světa, separovány od nezajímavého pozadí. Výsledkem segmentace má být soubor vzájemně se nepřekrývajících oblastí, které buď jednoznačně korespondují s objekty vstupního obrazu, pak jde o kompletní segmentaci, nebo vytvořené segmenty nemusí přímo souhlasit s objekty obrazu a pak jde o částečnou segmentaci. Úspěšnost segmentace závisí na vhodně provedeném předzpracování obrazu. [16]

Segmentace bývá nejčastěji založena na detekci hran, které ohraničují jednotlivé objekty a na detekci celých oblastí, kterými jsou jednotlivé objekty v obraze

reprezentovány. Mezi hojně využívané metody patří hranové detektory, které byly představeny již v předchozí kapitole. Pokud je objekt zájmů v obraze reprezentován souvislou oblastí, která je obklopena hranicí, lze tyto detektory úspěšně využít. [17]

Mezi další přístupy k segmentaci objektů patří prahování, metoda založená na prostorových souvislostech (narůstání oblastí) a segmentace založená na srovnávání se vzorem. [18]

3.3.1 Segmentace obrazu prahováním

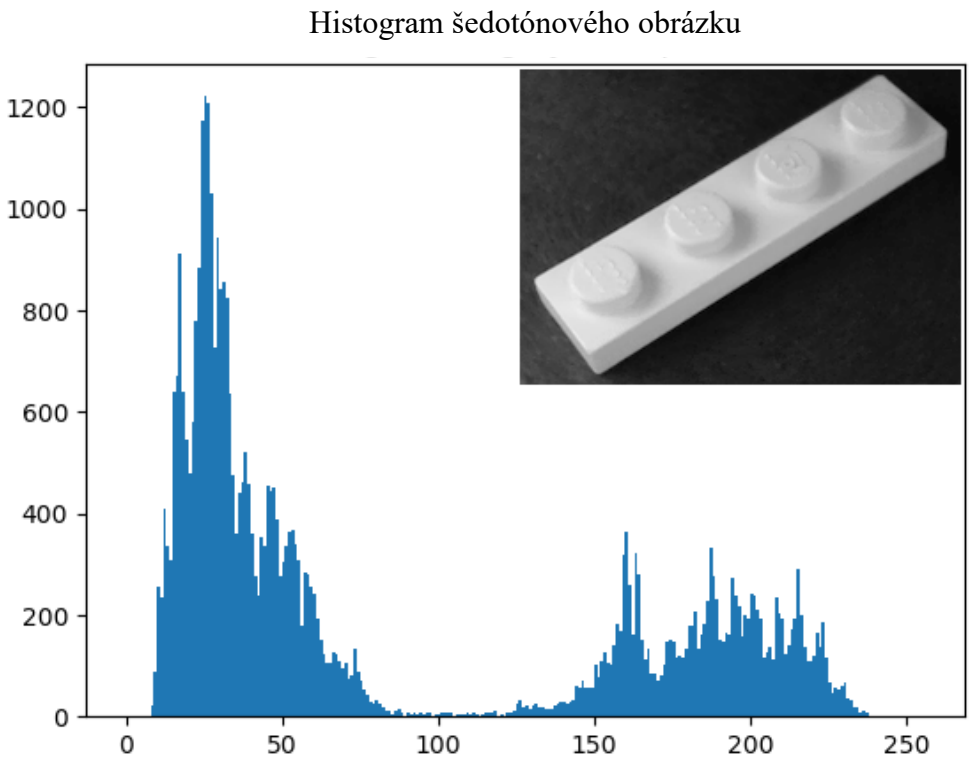
Rychlou detekci celých oblastí v obraze lze realizovat při vhodných podmínkách pomocí metody prahování. Jejím výsledkem je zpravidla binární obraz Pro prahování se zpravidla využívá jasového histogramu obrazu. Histogram intenzity jasu znázorňuje pro každou úroveň šedi což je hodnota mezi 0 a 255, počet bodů v obraze, které odpovídají dané jasové úrovni. [18] [17]

$$N = \sum_{i=0}^{k} h_i. \tag{3.10}$$

V rovnici nahoře reprezentuje *N* počet řádků a sloupců v obraze, h_i je počet pixelů odpovídající *i* úrovním šedi a celkový počet úrovní je *k*.

Je-li v obraze světle šedý objekt na tmavém pozadí, bude mít histogram dva výrazné vrcholy a mezi nimi údolí (Obrázek 3.7). Hodnotu na v lokálního minima na dně tohoto údolí lze považovat pro tuto modelovou úlohu za optimální segmentační práh. Pixely (i, j) z obrazu f, které mají hodnotu jasu nad hodnotou prahu *T*, patří do objektu, a naopak pixely (i, j) s jasem pod prahem *T* patří do pozadí. Výsledkem je tedy binární obraz g.

$$g(x, y) = \begin{cases} 0, & f(i,j) < T \\ 1, & f(i,j) \geq T \end{cases}. \tag{3.11}$$



Obrázek 3.7: Šedotónový obrázek s jasovým histogramem

V reálných aplikacích se příliš často výrazné údolí neobjevuje a práh je tedy obtížnější určit tak, aby ve výsledném obraze bylo co nejméně chybně zařazených obrazových bodů. Chybou při prahování se rozumí pixely, které patří objektu, ale které jsou ve výsledném binárním obrazu klasifikovány jako pozadí, nebo jsou naopak do regionu zájmů přiřazeny pixely patřící do pozadí. Pro zmírnění vlivu špatného osvětlení objektu lze využít metodu dynamické neboli adaptivní prahování. Jedná se o metodu, při kterém je původní obraz rozdělen na menší části a pro každý takový podobrázek se použije optimální prahovací úroveň.

3.3.2 Matematická morfologie

Výsledkem metod provádějících segmentaci obrazu bývají zpravidla binární obrazy. V každém bodě nabývají jedné ze dvou možných hodnot. Obvykle v pixelech náležející objektům nabývá obrazová funkce hodnoty 1 a v pixelech pozadí nabývá hodnoty 0.

Matematická morfologie využívá vlastností bodových množin, výsledky z integrální geometrie a topologie. Nejčastěji se aplikuje na binární obrazy (tzv. binární

matematická morfologie), ale lze ji snadno zobecnit i na šedotónové a barevné obrazy. Morfologické operace se používají často pro předzpracování. Pokud je binární obraz k dispozici až ve fázi segmentace, lze z něj odstranit šum, zjednodušit tvar objektů, zdůraznit struktury objektů (kostra, ztenčování, zesilování, konvexní obal, označení objektů) a popsat objekty číselnými charakteristikami (plocha, obvod, projekce). [1] [19]

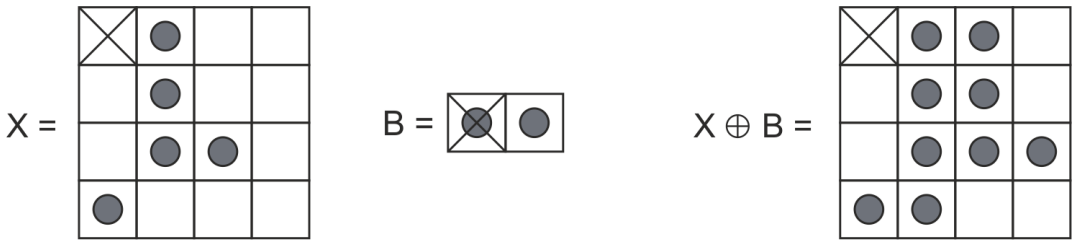
Dvě nejzákladnější operace matematické morfologie jsou eroze a dilatace. Jejich kombinací pak vznikají operace otevření a uzavření.

Morfologická transformace je dána relací mezi bodovou množinou X (typicky vstupním binárním obrazem) s jinou, menší bodovou množinou B, která se nazývá strukturní element. Ten má definovaný lokální počátek O, kterému se říká reprezentativní bod a postupně se přikládá na všechny body v obraze. Některé typické strukturní elementy jsou čtvercové, nebo křížové. Výsledek relace mezi obrazem X a strukturním elementem B se zapíše do výstupního binárního obrazu v reprezentativním pixelu. [1]

Dilatace a eroze

Dilatace skládá body dvou množin pomocí vektorového součtu a značí se $\oplus$. Používá se k zaplnění děr a zálivů. Dilatace $X \oplus B$ je bodovou množinou všech možných vektorových součtů pro dvojice pixelů, vždy pro jeden z množiny X a jeden z množiny B. Jedná se o metodu kumulativní a také asociativní. Lze jí vyjádřit následující definicí, ve které platí, že p je bod v binárním obrazovém prostoru ε^2. [1]

$$X \oplus B = \{p \in \varepsilon^2 : p = x + b,\ x \in X, b \in B\}. \qquad (3.12)$$



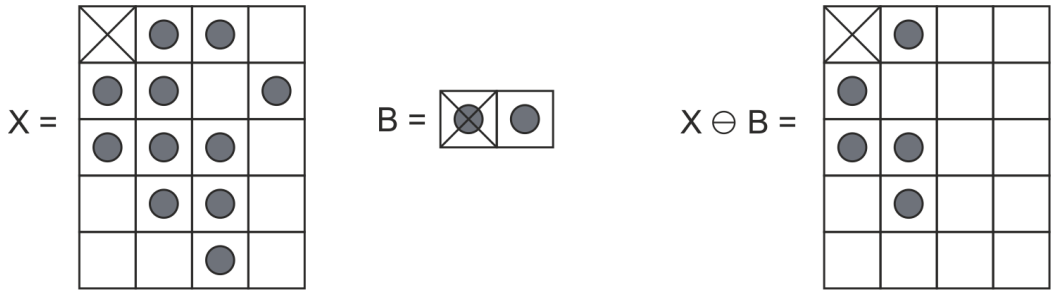
Obrázek 3.8: Ukázka dilatace (podle [20])

Na obrázku (Obrázek 3.8) je ukázka dilatace bodové množina X strukturním elementem B.

Eroze se používá pro zjednodušení struktury objektů. Jedná se o duální operaci k dilataci, ale není její inverzní transformací. Operace eroze $\ominus$ skládá dvě bodové

množiny pomocí rozdílu vektorů. V rovnici dole je *p* bod v binárním obrazovém prostoru ε^2.

$$X \ominus B = \{p \in \varepsilon^2 : p + b \in X \ \forall b \in B\}, \tag{3.13}$$



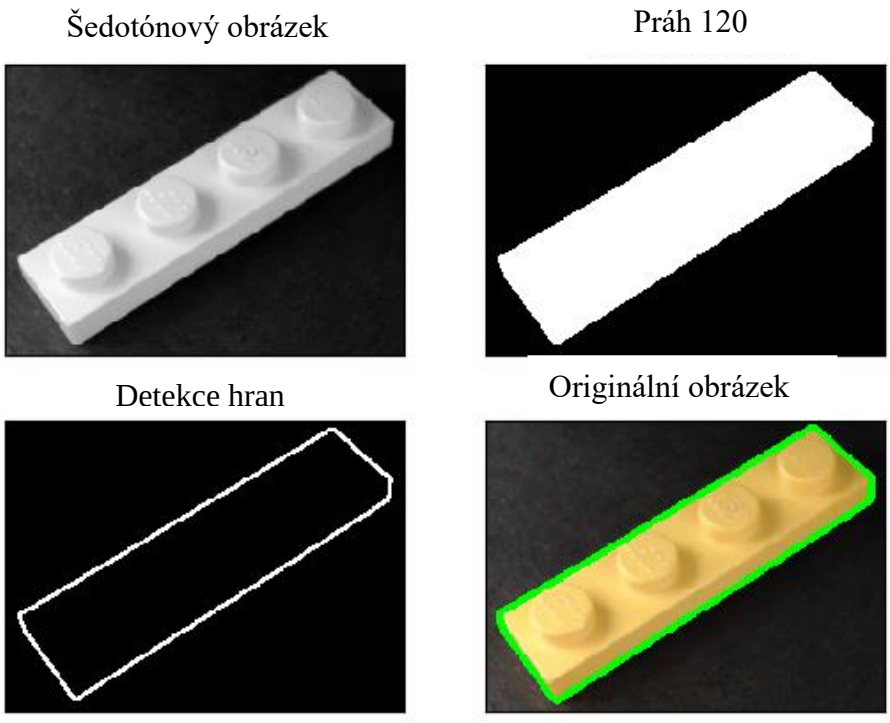
Obrázek 3.9: Ukázka eroze (podle [20])

Příklad eroze na obrázku (Obrázek 3.9) ukazuje výsledek eroze bodové množina *X* strukturním elementem *B*.

Jak je vidět na obrázku (Obrázek 3.10), dilatace objekty zvětšuje, zatím co eroze je zmenšuje. Pro zachování původní velikosti se obě metody kombinují.

Obrázek 3.10: Porovnání operací dilatace a eroze (podle [16])

Při odečtení erodovaného obrazu od původního binárního, může vzniknout obrys původního objektu. Morfologické operace se tedy může využít jako jednoduchý hranový detektor (Obrázek 3.11).



Obrázek 3.11: Hranový detektor využívající erozní metody

Otevření a uzavření

Další morfologické operace jsou otevření a uzavření a jsou to operace, které vzniknou vzájemnou kombinací elementárních operací dilatace a eroze. Výsledkem obou kombinací je zjednodušený obraz, obsahující méně detailů. Celkový tvar objektu se nezmění, ale odstraní se detaily, které jsou menší než strukturní element.

Eroze následovaná dilatací se nazývá morfologické otevření. Oddělí objekty spojené úzkou šíjí, a tak zjednoduší strukturu objektů. Dilatace následovaná erozí se naopak nazývá morfologické uzavření. Uzavření naopak spojí objekty, které jsou blízko u sebe, zaplní díry a vyhladí obrys. [19]

Otevření množiny X strukturním elementem B se označuje $X \circ B$ a uzavření se značí $X \bullet B$. Operace jsou definovány jako

$$X \circ B = (X \ominus B) \oplus B, \tag{3.14}$$

$$X \bullet B = (X \oplus B) \ominus B. \tag{3.15}$$

4 Konstrukční řešení mechanického systému

Účelem mechatronického systému je třídit jednotlivé dílky ze stavebnice LEGO. Tento problém se může rozložit na dílčí části, které jsou potřeba splnit pro správné fungování celé soustavy.

Systém musí mít vstupní místo, kam se budou dávat nezařazené dílky. Takovým místem je zde dopravníkový pás, který dopraví dílek do druhé části. Tou je podložka, na kterou je namířená kamera, která získává obraz pro vyhodnocování. Třetí částí jsou koncové boxy, do kterých je rozpoznaný dílek zařazen.

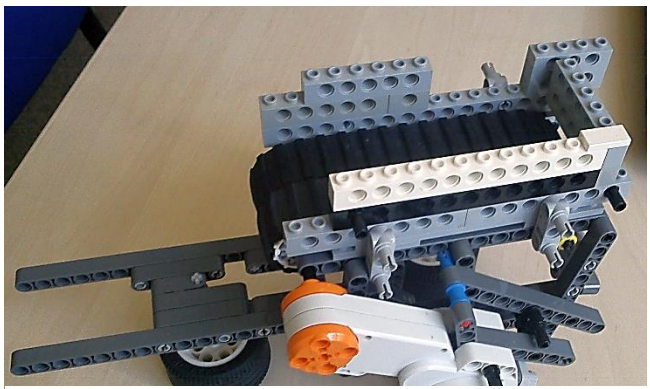
V každé části je použit servomotor, který je připojen k jednotce NXT. Ovládání všech motorů má na starosti ovládací program, který za použití knihovny s řídící jednotkou NXT komunikuje a zasílá do ní příkazy ohledně činnosti celé soustavy.

Celková konstrukce by měla být navržena takovým způsobem, aby byla zajištěna stabilita a robustnost. Při stavbě však bylo k dispozici jen omezené množství stavebních prvků, a tak by některé části mohli být vyztuženy lépe.

4.1 Dopravník

Základ dopravníku tvoří dvojice gumových pásů, které byly součástí balení stavebnice a patrně měli primárně sloužit pro pásové robotické vozítko. Pásy jsou poháněny jedním servomotorem, který je připojen do řídící jednotky přes port A.

Princip činnosti je takový, že pokud z konce dopravníku spadne kostka pod kameru, která zaznamená nový objekt, pás se zastaví, aby mohlo dojít k zařazení kostky do správného koncového boxu. Pás se pohybuje dostatečně pomalu, aby ovládací program měl dostatek času na detekci kostky a zastavení systému. Pokud ale na páse budou kostky spojené, nebo ležet na sobě a pod kameru jich spadne více najednou, způsobí to jejich chybné zařazení.



Obrázek 4.1: Dopravníkový pás

4.2 Sklápěcí mechanizmus s kamerou

V tomto prvku je využit opět jeden servomotor, připojen k NXT na port B. Nosná konstrukce je postavena z lega a je zde použita také kamera a černá papírová podložka. Pod sklápěčkou je ještě umístěn světelný senzor, který je využíván při přistavování koncových boxů na místo, kam se kostka vyklopí.

Využitá webová kamera je značky Logitech s označením HD PRO WEBCAM C92 a je připojena k počítači rozhraním USB, kde její obraz zpracovává ovládací program. Přesto že umožňuje pořizování videa ve vysokém rozlišení Full HD 1080p, pro potřeby diplomové práce je nastavená na pořizování snímků v rozlišení 320 × 240 pixelů. Úloha zpracování obrazu má mnoho kroků a je výpočetně náročnější. Pokud by obraz měl příliš vysoké rozlišení, docházelo by k výrazné latenci, která by narušovala plynulí chod třídičky a včasné zastavení pásu.

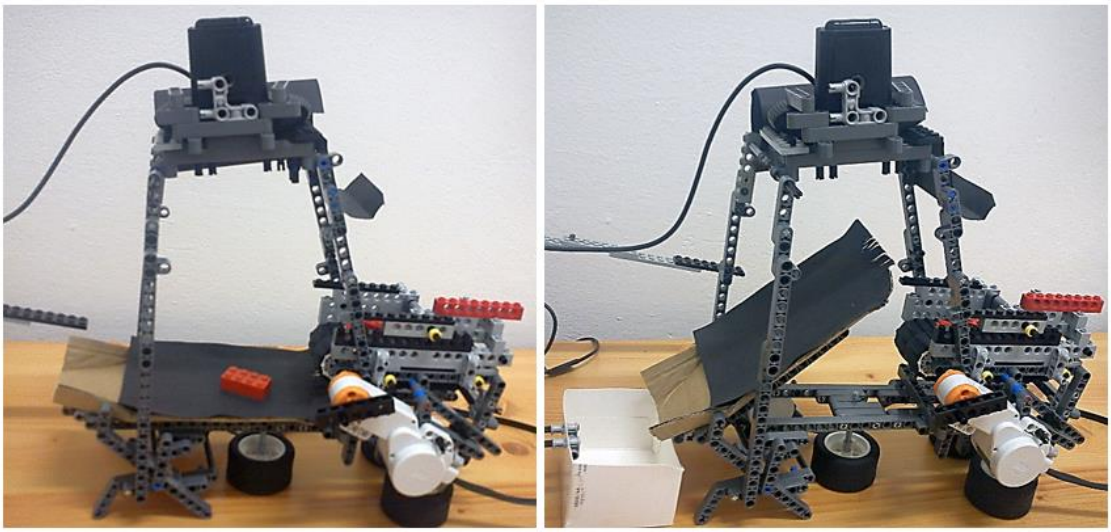
Pokud z horního dopravníku spadnou 2 nebo více kostek, pak se pro následnou klasifikaci vybere ten největší samostatný objekt. Jestli že se ale LEGO kostky budou překrývat a identifikují se tak jako jeden objekt, pak není možné jednoznačně odhadnout, jak se neuronová síť zachová a objekt klasifikuje.

Kamera pořizující obrázky objektů snímá černou plochu, na kterou jednotlivé kostičky padají. Všechny obrázky LEGO kostek tak mají stejné jednobarevné pozadí, které usnadňuje jejich automatické nalezení. Jsou dva způsoby, jak může být kamera v konstrukci uchycena:

A. Kamera je namířena kolmo na podložku a získává tak důležitou informaci o obrysu dílku. Z něho je pak různými segmentačními metodami zjištěna velikost kostky (zda se jedná o lego 2×2, 3×1 atd.). Dále jsou zde řetězové kódy pro získání podrobnější informace o tvaru. Kvalita pořízeného snímku je u této metody velmi důležitá. Při horním pohledu nelze jednoznačně vyhodnotit dílek, který stojí nebo leží na boku, pokud má takto položený stejnou základnu jako jiný dílek v jiné pozici. Spadne-li například kvádrový dílek tvaru 4x2 na podložku tak, že stojí na své nejmenší stěně, má stejný půdorys jako dílek 2x1, ležící na své největší podstavě. V takovém případě je správně zjišťována alespoň jeho barva.
B. Kamera snímá LEGO dílky pod určitým úhlem (přibližně 50°–60°), čímž je do obrazu přidána informace o třetím prostoru. Z obrázku je poznat, jestli LEGO dílek

neleží na boku, nebo lze rozlišit, zda se jedná o kostičku klasické tloušťky, nebo zda je zúžená (plate). Takto sejmutý obrázek je určen pro metody strojového učení SVM a CNN.

Na následujícím obrázku (Obrázek 4.2) je vidět sklápěcí mechanizmus společně s předešlým dopravním pásem. Pod kameru, která je umístěna na vršku konstrukce a snímá kolmo podložku se na prvním obrázku nachází červená LEGO kostka. V druhém případě je sklápěčka v poloze pro vyklopení do koncové krabičky, určené pro všechny červené objekty. Vyklopení se provede v okamžiku, kdy se po dokončení správné detekce dílku vykoná posun příslušného boxu pod sklápěčku. Po vrácení podložky do výchozí vodorovné pozice se opět rozpohybuje pás a systém čeká na příchod dalšího objektu.



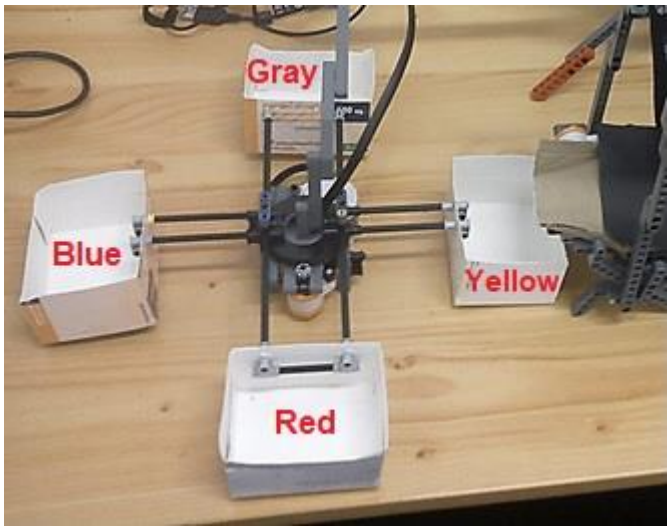
Obrázek 4.2: Konstrukce sklápěčky v dolní a horní poloze

4.3 Koncové úložné boxy

Servomotor připojený k portu C, otáčí se čtyřmi rameny. Každé rameno má na konci připevněnou krabičku, do které jsou umísťovány kostky podle barvy.

Knihovna určená pro ovládání NXT umožňuje rotaci motorů o konkrétní počet stupňů. Zajištění přesné polohy krabičky ale napomáhá určit světelný senzor schovaný pod vyklápěcí rampou. Ten je vhodné při změně světelných podmínek překalibrovat v ovládacím programu. Pokud box dorazí na požadované místo, senzor se zastíní a program dá pokyn k zastavení motoru. Při absenci tohoto čidla a konstantní počtu úhlů narůstala chyba polohy boxu vedlo k neschopnosti systému kostku do boxu zařadit.

Korektní poloha je na obrázku (Obrázek 4.3). Je na něm vidět také základní poloha jednotlivých boxů při prvotním zapnutí. Program má jejich polohy nastaven staticky, takže po spuštění bude pod sklápěčkou vždy přistavěn box na žluté dílky, a naproti němu na modré. Krabička na šedé dílky slouží zároveň pro všechny ostatní barvy, které nemají zastoupení ve zbývajících třech.



Obrázek 4.3: Koncové úložné boxy

5 Tvorba ovládacího programu

Hlavní program je spuštěný na PC a má za úkol ovládat robotický systém a vyhodnocovat obraz pořízený připojenou kamerou. Aplikace je vytvořená v objektově orientovaném jazyce C# využívající platformu .NET Framework a je spustitelná ve Windows. Ze všech použitých knihoven je nejzajímavější AForge.NET, která obsahuje kromě nástrojů pro komunikaci s NXT také velké množství funkcí pro zpracování obrazu a strojové učení. Kromě uživatelského programu bylo v rámci diplomové práce vytvořeno množství scriptů v jazyce Python, ve kterých jsou zpracovávána testovací data a implementovány metody SVM a NN.

Program nepracuje zcela samostatně a alespoň při jeho spuštění je vyžadováno nastavení od uživatele. Ten musí v první řadě vybrat kameru a zadat číslo sériového portu COM, přes který je možné komunikovat s NXT.

Celý třídící systém umožňuje dva různé způsoby, jak se budou obrazy zpracovávat. Uživatel si zvolí, zda se bude jednat o segmentační technikou (při umístění kamery shora pro získání obrysu dílku) nebo metodu strojovým učením (a pohled pod úhlem, který umožňuje získat informaci o prostorovém tvaru).

5.1 Snímaná scéna

Kamera snímající plochu, na kterou padají jednotlivé dílky se může v systému uchytit ve dvou různých pozicích. Výběr její pozice je ovlivněn zvolenou metodou rozpoznávání. Na kvalitu osvícení snímané scény je kladen velký důraz v obou případech. Při nevhodně zvoleném osvětlení u předmětů, které mají lesklé plochy, dochází ke značnému ztížení aplikace algoritmů pro zpracování obrazu. Falešné odlesky lze částečně odstínit použitím polarizačních filtrů. Rušivým elementem může být jak denní světlo, tak umělé osvětlení na pracovišti. Pro eliminaci světelného šumu a získání neměnné jasové úrovně obrazu lze vyřešit technickým odstíněním, nebo volbou vhodného osvětlovače, který svou intenzitou nežádoucí světlo překryje. V práci je tento problém vyřešen částečným zastíněním prostoru, kde ke snímání dochází. V ovládacím programu má uživatel možnost nastavení některých parametrů pro algoritmy, které objekt ve snímané scéně hledají.

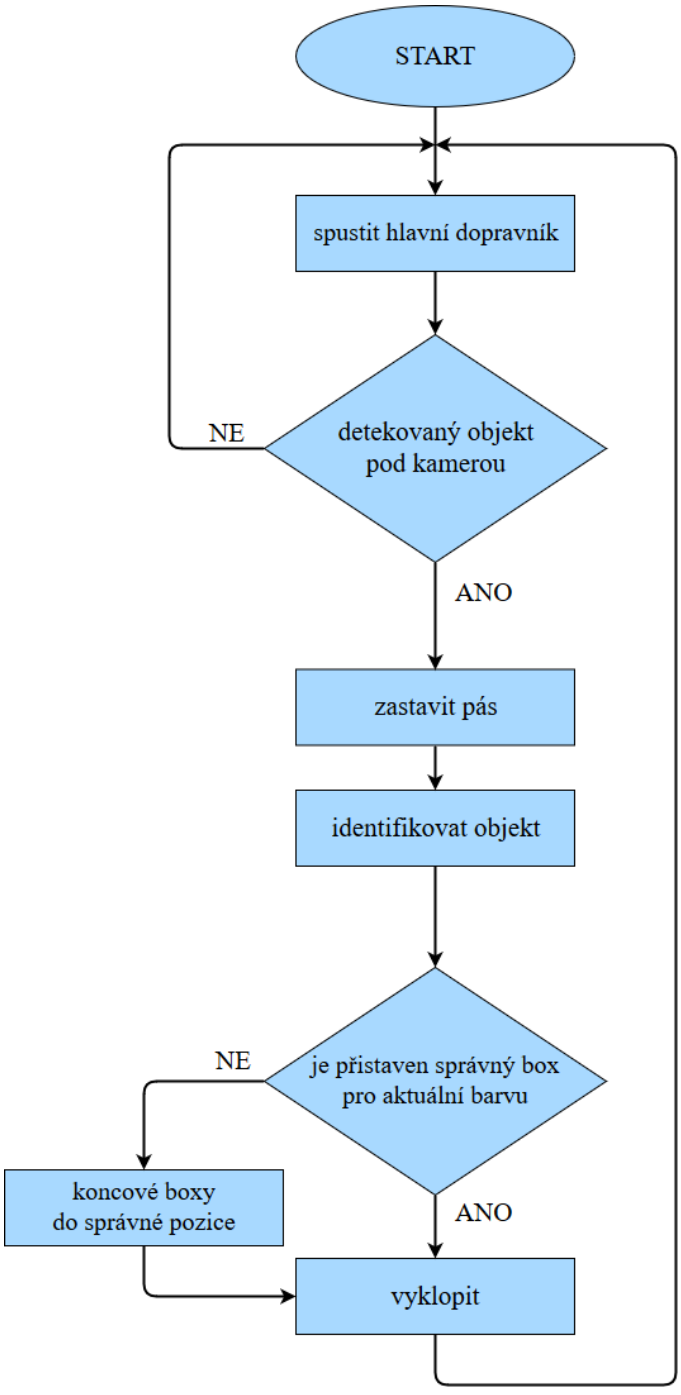
Podložka, na které LEGO dílek leží, by měla být nasvícena rovnoměrně ze všech stran tak, aby nebyla část dílku tmavá a část světlá. Rovněž by neměly vznikat ostré stíny. Nejdůležitější je najít korektně obrysové hrany objektu. Případná tmavá místa uvnitř

kostky, která nedosáhnou k segmentačnímu prahu, nebudou mít příliš negativní vliv na úspěšné zařazení.

5.2 Řízení mechanického systému

Jak již bylo zmíněno dříve, program komunikuje s řídící jednotou NXT bezdrátově přes Bluetooth. U NXT jsou obsazeny všechny výstupní porty připojenými motory a jeden vstupní port ke kterému je připojen světelný sensor. Pokud při spuštění program dojde k úspěšnému spojení, vypíšou se do konzole informace týkající se NXT, jako je stav baterie, verze firmwaru, stav paměti atd. Program ale může pracovat i bez připojení třídícího robota. V hlavním okně aplikace se bude v textové podobě zobrazovat simulovaný stav všech motorů. Uživatel tak může mít přehled o tom, který motor je aktuálně spuštěn a co vykonává za činnost. Objekty určené k vyhodnocení se ale budou muset pod kameru umísťovat ručně. Výsledek rozpoznání se opět zobrazí v okně aplikace.

Na následujícím diagramu (Obrázek 5.1) je přiblížena činnost systému, která je spojena s pohybem motorů. Hlavní dopravník se zastaví v okamžiku, kdy se detekuje nějaký objekt pod kamerou. U objektu se vyhodnotí jeho barva a tvar, případně příslušnost ke třídě a přistaví se odpovídající koncový box. Poté se sklápěcím mechanismem do boxu vyklopí a pás se opět rozjede.

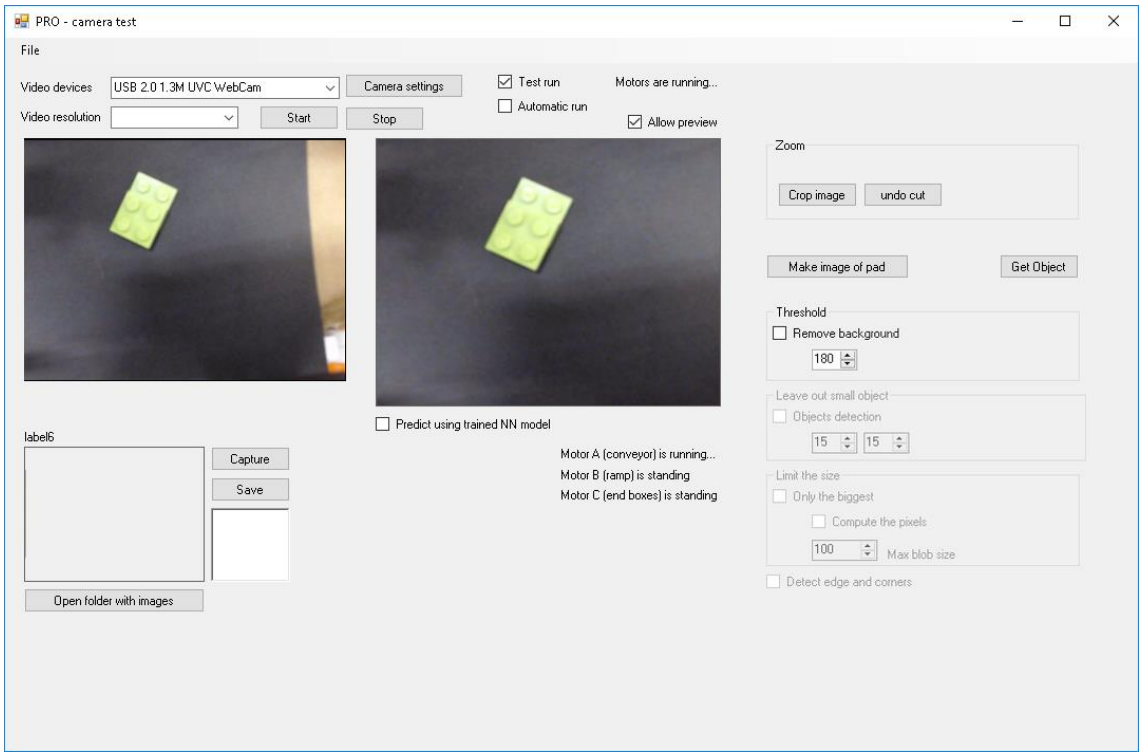


Obrázek 5.1: Vývojový diagram pro práci s motory

5.3 Nalezení LEGO dílku v obraze

Pro použité metody rozpoznání objektu v obraze je důležité určit jeho přesnou polohu na obraze pořízený kamerou. S tím úzce souvisí nastavení programu, které musí provést uživatel před uvedením soustavy do plného chodu.

Prvním krokem úspěšné segmentace je omezení regionu, ve kterém se dílek může vyskytovat. Na obrázku (Obrázek 5.2) je ukázáno hlavní okno GUI. Poté, co se z kombinovaného pole (angl. combo box) vybere dostupná kamera a stiskne se tlačítko start, začne se v levé ohraničené oblasti promítat obraz snímaný kamerou v reálném čase. Ve druhém rámečku jsou na tentýž obraz aplikovány postupy vedoucí k nalezení LEGO dílku. Tím je dána možnost uživateli pozorovat, jak se bude obraz měnit při změnách proměnných, aby bylo možné nastavit ideální podmínky pro bezobslužný chod. Na obrázku dole je vidět, že snímaná černá podložka nezasahuje až do kraje obrazu, což by mohlo vést k falešné detekci objektu. Do pravého okna s obrázkem byla zabudovaná funkce oříznutí oblasti, kterou si uživatel definuje jednoduchým táhnutím myši po obraze.

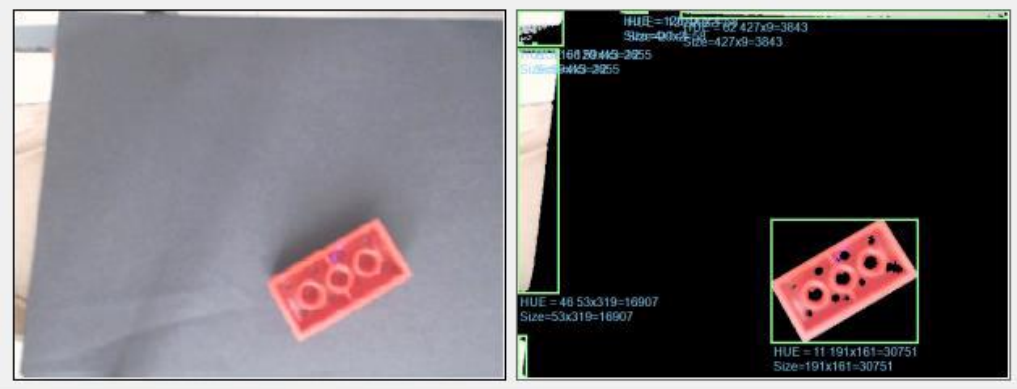


Obrázek 5.2: GUI ovládacího programu

V pravé části hlavního okna jsou ještě tři zaškrtávací boxy, které na sebe navazují a po zaškrtnutí umožní nastavovat hodnotu prahu, velikost nežádoucích malých objektů a zobrazení pouze největšího nalezeného objektu v obraze. K nalezení osamocených objektů se používá třída *BlobDetection* z knihovny *AForge.Imaging*. Za osamocený objekt se považuje vše, co nemá černou barvu, což je barva pozadí. Jednotnou barvu podložky zajistí částečné prahování, které všechny body obrazu f, které jsou pod prahem,

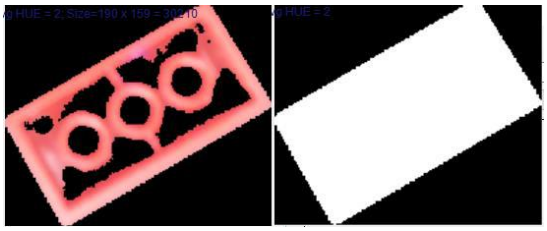
nahradí nulou (černou barvou) a ostatní pixely ponechá nezměněné tak jak je vidět v následujícím vzorci. Platí to pro všechny tři barevné složky.

$$g(x,y) = \begin{cases} 0, & f(i,j) < T \\ f(i,j), & f(i,j) \geq T \end{cases}. \tag{5.1}$$



Obrázek 5.3: Ohraničení detekovaných objektů

Obrázek (Obrázek 5.3) ukazuje způsob, jakým se nalezené objekty ohraničí. Pod každý vykreslený zelený obdélník se vypisují jeho rozměry a průměrnou hodnotu barevného odstínu z modelu HSV. Pro vybrání té správné oblasti zájmů tzv. ROI (angl. region of interesting) poslouží výše popsané prvky v grafickém rozhraní. Nejprve se vyřízne daná oblast tak, aby bylo dále zpracováváno jen to, co leží na černé podložce a následně se definuje minimální velikost ohraničených oblastí, které se mají brát v úvahu pro následné zpracování. V této fázi se binární obraz objektu získá pouhým nastavením hodnoty 1 pro všechny pixely, které neměli hodnotu 0 z předchozí operace prahování. Případné vzniklé díry se vyplní snadno, protože jsou ohraničeny body, které nenáleží pozadí (Obrázek 5.4).



Obrázek 5.4: Získaný binární obraz

Pokud jsou známy souřadnice rohů obdélníku, který ohraničuje hledaný objekt, je takto vymezená bitmapa vyříznuta z původního obrázku, na který ještě nebylo provedeno

prahování. Jedná se o to, aby okolo kostky nebyly žádné zbytečné okraje, ale také aby nebyl nijak upraven. Poté může sloužit jako vstup pro natrénovaný model konvoluční neuronové sítě.

Může se stát, že pod kameru spadnou z pásu dvě kostičky. V takovém případě je proces segmentace úplně stejný. Předem definovaná oblast pomocí vyříznutí odstraní z obrazu části stolu, které nenáleží podložce a pomocí prahování se detekují dva samostatné objekty. Pokud se tak stane, k dalšímu zpracování postupuje ten největší z nich (pokud ovšem nepřesahuje maximální přípustnou velikost definovanou uživatelem).

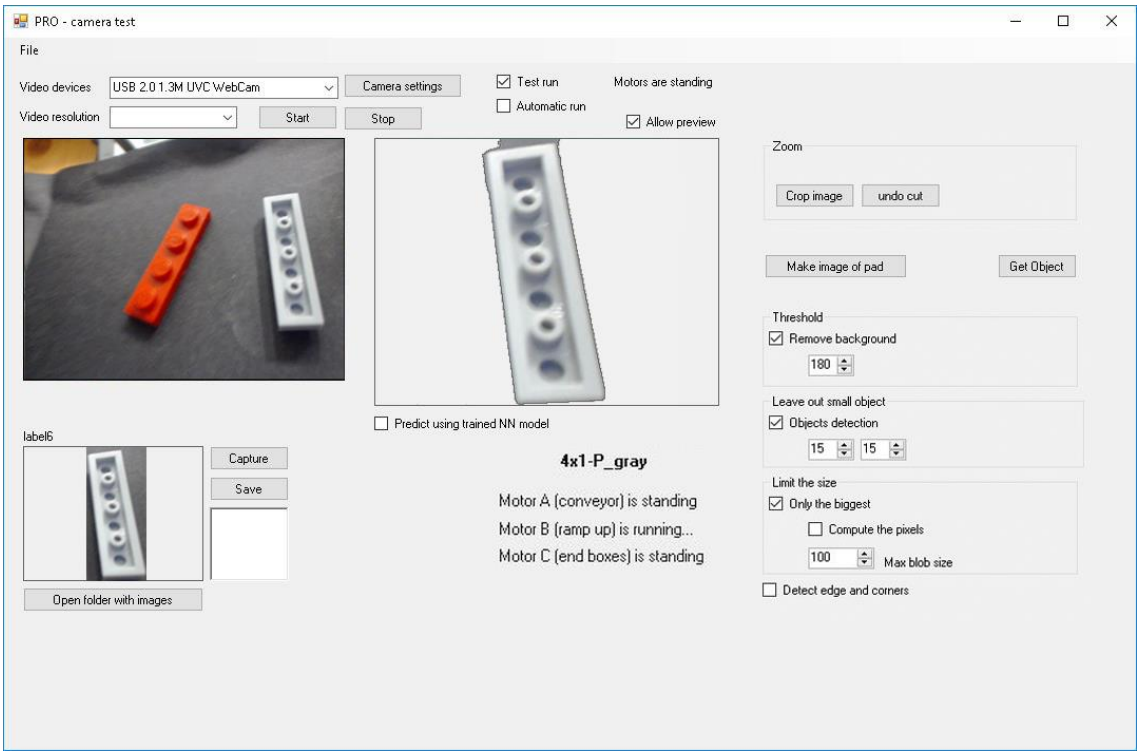
V ukázce (Obrázek 5.5) je GUI, ve kterém je vidět reálný obraz z kamery (vlevo) a samostatný dílek (vpravo), který byl z něho vyextrahován. Pod rámečkem s tímto dílkem je textový popis všech motorů a stavů, ve kterých se aktuálně nacházejí. Jedná se o jednoduchou simulaci mechatronického systému pro případ, že by z nějakého důvodu nebyla připojena NXT jednotka. Uživatel tak může mít představu o jeho chování. Na obrázku dole je zachycen stav, kdy se motor A (dopravní pás) i motor C (koncové boxy) stojí a motor B vyklopí kostky do boxu.

Menší obrázek vlevo dole tvoří vstup pro CNN a po uložení se může použít pro rozšíření datové sady. Aby bylo možné jej do natrénovaného modelu vložit a získat informace o predikovaných třídách s procentuální přesností, je nutné bitmapu převést na přesný tvar, se kterým model dokáže pracovat. V ukázkovém příkladu je výsledek klasifikace správný tedy „4x1-P_gray“, jak je možné vidět ve vypisovaném textovém štítku pod segmentovaným obrázkem.

Takto nastavený program již mohl pracovat zcela samostatně. Uživatel pouze mohl doplňovat dílky na pás a ty se sami třídily do příslušných krabiček podle barvy. V GUI navíc mohl pozorovat výsledky rozpoznání. Zatím co horní dvě okénka zobrazují pohyblivý obraz přímo z kamery, dolní obrázek se v levém rámečku zobrazí, pouze v momentě, kdy je systém zastaven a objekt pod kamerou vyhodnocován. Správnost klasifikace systémem lze zkontrolovat i zpětně, protože každý obrázek se automaticky ukládá s predikovaným názvem a časem pořízení.

V práci bylo nejčastěji v neuronové síti pracováno s obrázky velikosti 128×128 a 64×64. Pro získání správného formátu museli být ještě převedeny na jednodimenzionální pole.

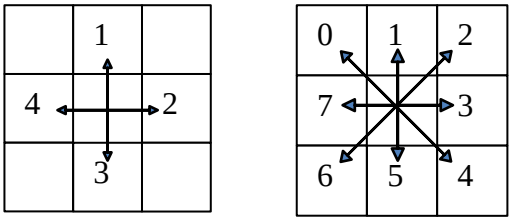
Pro manipulaci s modelem navrhnutým a natrénovaným v Pythonu byla v řídící aplikaci použita knihovna TensorFlowSharp.



Obrázek 5.5: Ukázka GUI se správnou detekcí objektu

5.4 Popis tvaru pomocí řetězového kódu

Na získaný binární objekt byl aplikován detektor rohů a dílek byl přesně popsán pomocí obdélníku, jehož délka stran v pixelech umožňovala rozhodnout, o jakou LEGO kostku se jedná. Tento způsob vyžadoval pevné umístění kamery nad podložkou a neměnné nastavení jejího rozlišení. Experimentálně byl zkoumán i další způsob, jak získat popis složitějšího tvaru objektu, ten ale nebyl do ovládacího programu implementován a zůstal v podobě samostatných scriptů v Pythonu. Jednalo se o Freemanovy řetězové kódy, což jsou deskriptory založené na hranici objektu (nejčastěji binárního).

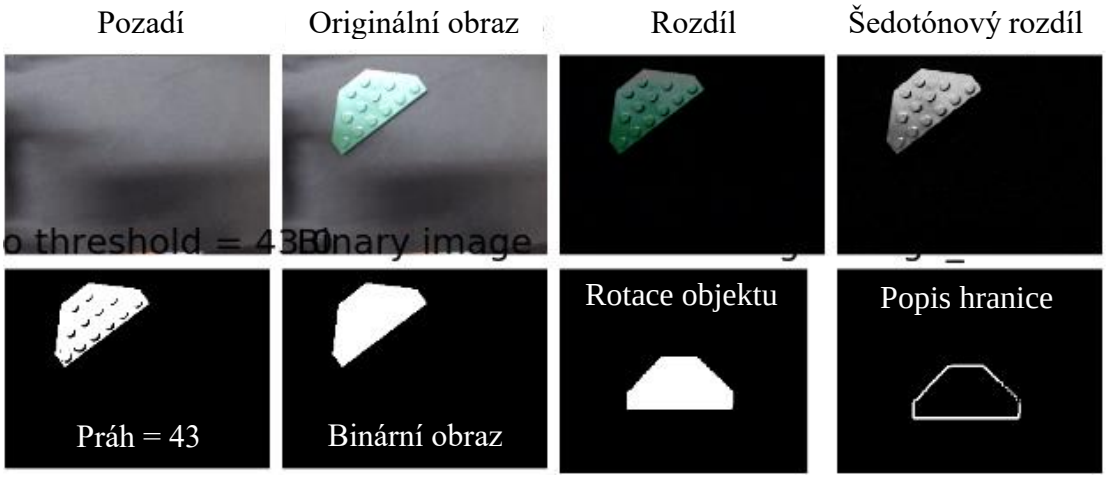


Obrázek 5.6: Freemanův řetězový kód pro 4-okolí a 8-okolí

Metoda spočívala v tom, že se od počátečního bodu postupovala podél hranice binárního objektu a jednotlivým směrům úsečky tvořící jeho hranu se přiřadil symbol podle obrázku nahoře (Obrázek 5.6). Počáteční bod byl vždy pixel nacházející se v levém horním rohu objektu.

Protože se LEGO kostky vyskytovaly i takového tvaru, že jejich obrys neměl jen pravé úhly, používaly se pro popis jejich hranice řetězové kódy pro 8-okolí.

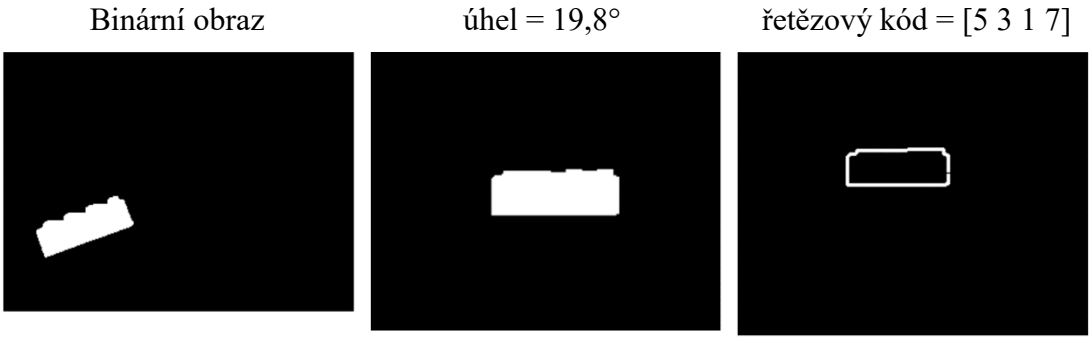
Aby bylo možné tuto metodu úspěšně využívat, bylo potřeba mít co možná nejpřesnější binární obraz. Jeho nalezení pomocí obyčejné funkce prahování bylo v tomto případě nedostačující. Problémy při prahování můžou vzniknou nevhodně zvoleným prahem, nerovnoměrným nasvícením, nebo odleskem od objektu. Všechny tyto problémy se dají řešit tím, že se pořídí snímek podložky před tím, než se na ní objeví objekt. Oba obrázky pak lze od sebe odečíst a po odstranění drobného šumu se získají pixely, které na prvním obrázku nebyly. To jsou body, které tvoří právě hledaný objekt. Takto lze získat i černý dílek, který by jinak bylo velmi obtížné v obraze nalézt. Ukázky tohoto procesu jsou na obrázcích dole (Obrázek 5.7).



Obrázek 5.7: Postup získání hranice objektu pro popis řetězovým kódem

První řada obrázků (Obrázek 5.7) znázorňuje proces odečtení originální obrazu s objektem od pozadí a výsledku převedeného do stupňů šedi. Následně je automaticky zvolen práh, který nemusí být nijak vysoký, protože pozadí objektu by již mělo být po odečtení černé. Ve výsledném binárním obrázku jsou zaplněny díry. Nyní přijde na řadu důležitý krok, kdy se výsledný objekt otočí tak, aby jeho nejdelší hrana byla rovnoběžná s dolním okrajem bitmapy. Díky tomu lze získat stejný řetězový kód pro tentýž tvar, libovolně natočený. V ukázce došlo k automatické rotaci o −36,9°. Proces přiřazování kódů podle směru hrany vždy začíná v levém horním rohu nalezeného objektu, takže řetězový kód může ideálně vypadat v tomto případě [6,6,6,6,6,6,5,5,5,5,5,3,3,3,3,3,3,3,3,3,3,3,3,3,1,1,1,1,1,0,0,0,0,0,0,7,7,7,7,7,7,7,7]. Tento tvar se následně zredukuje odstraněním po sobě jdoucích duplicitních znaků, čímž se dostane výsledný kód [6,5,3,1,0,7].

Před začátkem popisu hranice musí binární objekt podstoupit operaci matematické morfologie uzavření, aby se jeho obrys vyhladil. K tom je použit dostatečně velký strukturní element. Výsledek vyhlazeného objektu je na prostředním obrázku (Obrázek 5.8). Pokud se i přes vyhlazení objeví v popisu nežádoucí hodnota, dojde k její odstranění pomocí mediánového okénka, které se posouvá po hodnotách získaného kódu a nežádoucí hodnoty eliminuje. Pro příklad v poli [7,7,7,7,7,5,5,7,7,7,7] se vhodným mediánem hodnoty 5 odstraní.



Obrázek 5.8: Rotace a vyhlazení objektu před popisem hranice

6 Algoritmy pro klasifikaci LEGO dílků

Úspěšnost aplikace metod strojového učení s učitelem je závislá na dostatečně obsáhlé, kvalitní a správně anotované datové sadě (dataset). V případě této práce bude dataset obsahovat vlastní obrázky lego kostek, na kterých se použité metody může učit. Tato kapitola popisuje vytváření rozsáhlé sady obrázku, které budou tvořit trénovací množinu pro použité metody SVM a CNN. Následně byla prováděna řada experimentů, který měli nalézt nejpřesnější způsob rozpoznání dílku. Všechny tesy byly prováděny na procesoru Intel Core i5-7500 CPU @ 3.40 GHz s operační pamětí 8 GB.

6.1 Tvorba datové sady obrázků

Na internetu je k dispozici množství datových sad, které jsou volně dostupné a kdokoliv je může využít jako základ pro svoji vlastní úlohu. Pro velké společnosti jako je Google či Facebook jsou data velmi ceněnou komoditou a k jejich rozšiřování mnohdy přispívají sami uživatelé. Přehled nejzajímavějších, volně dostupných datových sad je k vidění například zde [21]. Co se týče obrazových datasetů, nejznámější je třeba MNIST, obsahující ručně psané číslice, nebo CIFAR10/CIFAR100 s barevnými obrázky z 10/100 kategorií. Vhodná databáze LEGO dílků však pravděpodobně neexistuje, a tak byla pro účely diplomové práce vytvořena.

6.1.1 Sestrojení snímací soustavy

Při prvotních pokusech bylo pro navrhnutou neuronovou síť použito několik málo tříd a každá obsahovala desítky, ručně nasnímaných obrázků. To se podle předpokladu ukázalo jako příliš malý počet a chybovost rozpoznání byla příliš vysoká. Takto dosažené výsledky měli minimální hodnotu a pro třídění za různých podmínek se nedaly použít.

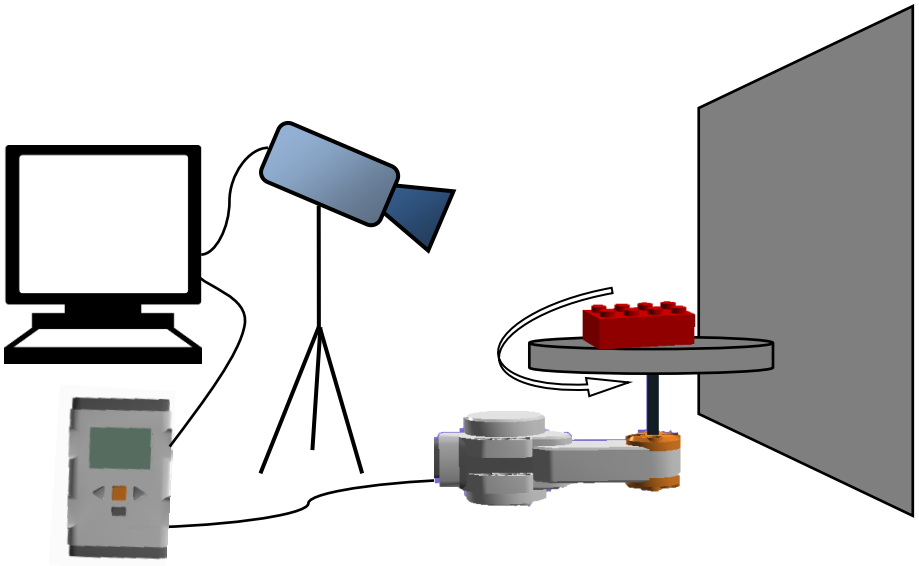
Proces pořizování trénovacích obrazových dat byl tedy zautomatizován, aby bylo možné získat stovky až tisíce dat. Pro srovnání databáze MNIST obsahuje celkem 70000 snímků pro 10 kategorií a CIFAR-10 má 6000 různých obrázků pro každou kategorii.

Řekněme, že základní tvar LEGO kostky je kvádr a lze ho položit čtyřmi různými způsoby. Kromě základní pozice, kdy spojovací body směřují nahoru, může být kostka postavena obráceně, tedy body dolů, na boku, nebo na čelní stěně. Pokud se bude jednat o zúženou kostku tzv. plate, budou možné pozice položení pouze dvě.

Za účelem získání většího množství snímků byla sestrojena rotující podložka snímaná kamerou ze statické pozice (Obrázek 6.1). Pomalou rotaci zajišťoval servomotor připojený k NXT, ovládaný jednoduchými příkazy z PC. Z kamery se pořizovaly obrázky

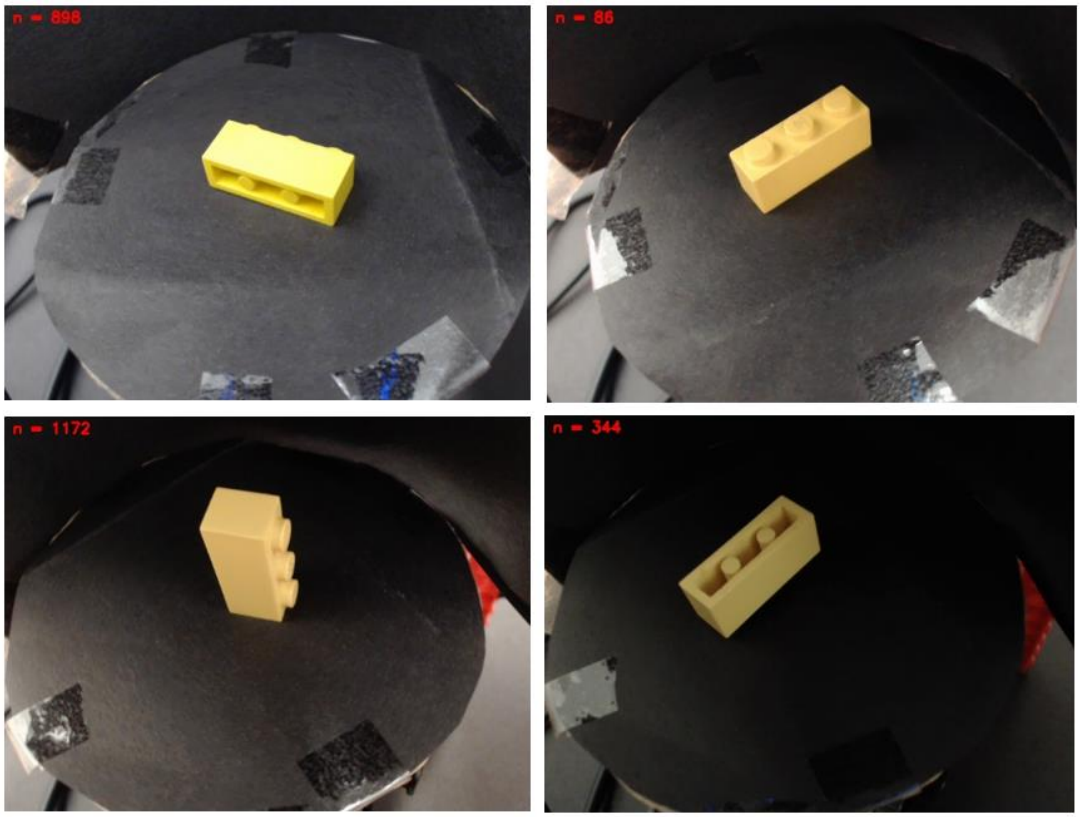
s frekvencí jednoho za 300ms. Pro pokrytí co možná největšího množství úhlů pohledu na dílek, byla po provedení otočky o 360° poloha kostky změněna a provedla se další otáčka.

Důležitým faktorem je pozadí objektu. Pokud má být jeden druh kostky klasifikován správně, nemělo by se příliš lišit pozadí u trénovací množiny a reálné aplikace. Protože je podložka v třídícím stroji pokryta černým papírem, je i při tvorbě trénovací obrazů pozadí objektů černé. Významným element přispívající k variabilitě snímků patří nasvícení, vzdálenost od kamery a ostrost.



Obrázek 6.1: Modelové schéma snímací soustavy pro tvorbu datasetu

Tímto procesem se zdokumentovalo 36 různých LEGO dílků. Jednodušší tvary (obvykle plate) měli 400 až 600 obrázků, ty složitější jich měli 800 až 1000. Zkoumané objekty však v obraze zaujímali pouze minoritní část a zbytek tvořilo pozadí. To navíc nebylo dáno jednolitou barvu a vyskytovaly se v něm artefakty, které mohly proces učení narušovat. Ukázka typických snímků pořízených tímto systémem ja k vidění na následujícím obrázku (Obrázek 6.2).



Obrázek 6.2: Ctyři různé polohy LEGO kostky

Script obsluhující mechanismus tvorby snímků ukládal každou třídu do samostatného adresáře, jehož umístění a jméno si uživatel sám předem definoval. Přitom byla navrhnutá a dodržovaná taková jmenná konvence, aby z názvu adresáře bylo jazné, o jaký druh LEGO dílku se jedná. Například všechny obrázky žluté kostky, která je na předchozím obrázku, byly ukládány do složky „3x1_yellow“. Stejná kostka, ale s třetinovou výškou (plate) je v adresáři „3x1-P_yellow“ atd.

6.1.2 Normalizace získaných dat

V předchozí podkapitole byla ukázka snímků, které byly vytvořeny poloautomatickým procesem a tvoří hlavní trénovací obrazovou množinu. Jejich pozadí

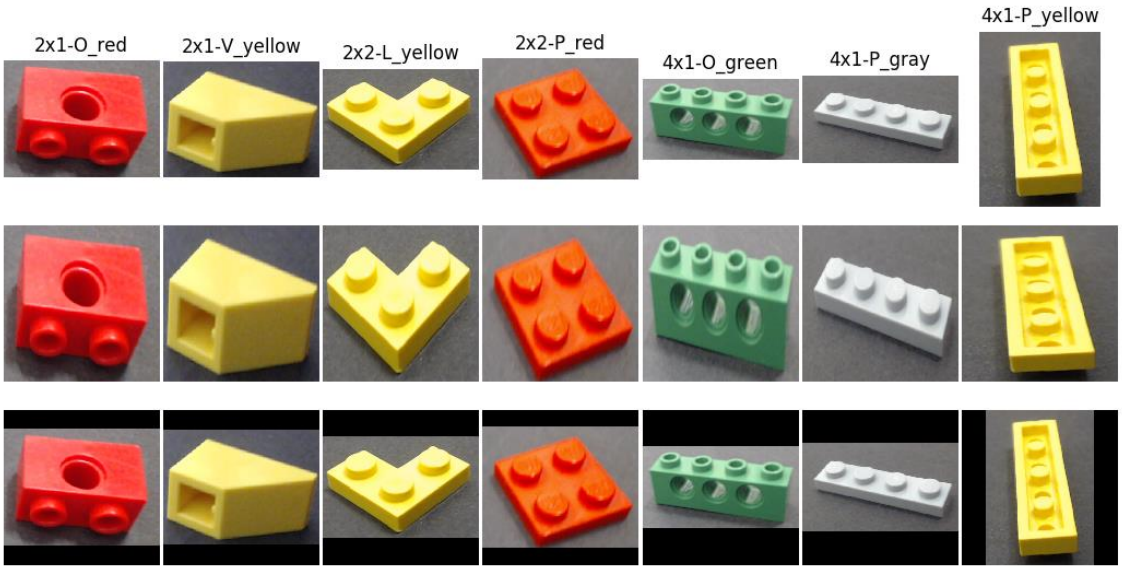
je však odlišné od těch snímků, které pořizuje kamera zabudovaná v třídícím robotovi. Proto byl vytvořen script, kterému se jako vstup určí adresář, ve kterém se nacházejí obrázky určené k oříznutí a jako výstup pak adresář, kam se mají výstupní bitmapy ukládat. Po spuštění se uživateli popořadě zobrazují obrázky a táhnutím myši si vybere oblast s objektem, který se má vyříznou. Po stisknutí klávesy „c“ dojde k uložení označeného úseku jako samostatného snímku do výstupního adresáře.

Neuronová síť se obvykle přijímá ve vstupní vrstvě obrazy pevně daných rozměrů. Standardně se používá čtvercový tvar $N \times N$. Pro neuronovou síť využívané v diplomové práci se používaly bitmapy velikosti 128×128 a 64×64. Při menších rozměrech se ztrácely důležité informace o objektu a větší rozměry výrazně prodlužovaly dobu potřebnou k natrénování modelu.

Při prvotních experimentech se upravovala velikost vstupních dat při každém spuštění trénovacího i testovacího procesu. Pro urychlení už takto časově náročných operací byly všechny obrazy upraveny na požadovanou velikost předem. Jako vhodné varianty se jevily 2 způsoby:

- bez zachování poměru stran, kdy se obrázek transformuje do čtvercového tvaru a následně se vzorkuje na danou velikost,
- poměr stran je zachován, čímž nedochází k deformaci zachyceného objektu a podle kratší strany je na obě strany bitmapy přidán černý pruh. Šířka pruhů je volena taková, aby se srovnala dálka stran obrazu a vznikl tak čtverec. Nakonec přichází na řadu opět vzorkování.

Obě tyto metody úpravy rozměrů byly použity a výsledky trénování porovnány. Ukázka obou způsobů převedení variabilních obrazů na jednotnou velikost je na obrázku dole (Obrázek 6.3).



Obrázek 6.3: Ukázka nové datové sady

i. **první řádek obsahuje obrázky s oříznutými nežádoucími částmi**
ii. **ve druhém řádku jsou tyto obrázky převedeny do čtvercového tvaru**
iii. **ve třetím řádku jsou převedeny na jednotnou velikost přidáním černých pruhů**

6.1.3 Rozšíření datové sady

Pro zlepšení generalizace neuronové sítě se v některých případech využívá umělé rozšíření obrazových dat tzv. augmentace datasetu. Přistupuje se k tomuto kroku v případech, kdy je reálných snímků stále nedostačující počet a není způsob, jak jednoduše získat další originální data.

Augmentace je proces, při kterém se na snímky existujícího datasetu aplikují různé transformace, čímž vzniknou snímky zcela nové. V této práci byly existující datové sady rozšířeny o následující transformace s cílem zjistit vliv takto vytvořených dat na chování natrénovaného modelu:

- oříznutí obrazu z každé strany náhodně o 10 až 20 pixelů
- horizontální překlopení
- rotace o 5°, 10°, 15°, 20°, 25°, 30°, 35°, 40° a 45°

Z jednoho obrázku tak vzniklo 11 nových. Mezi další často používané metody patří barevné transformace, zanesení libovolného šumu, aplikace průměrovacích filtrů, filtrů zvýrazňující hrany, transformace jasu, zrcadlové překlápění a mnoho dalších. Volba vhodných variant je spojená s typem úlohy a charakteru dat. K využívání těchto metod napomáhá knihovna „imgaug“ pro Python, která navíc umožňuje náhodné zvolení

některých těchto augmentačních kroků. Příklad získaného rozšíření napomáhá robustnosti sítě a je na obrázku dole (Obrázek 6.4) pro dílek 4x2_red.

Obrázek 6.4: Obrázky získané augmentací

6.1.4 Statistiky

Výsledný dataset obsahoval dostatečné množství snímků LEGO kostek, zachycených v rozličných polohách, za proměnlivých světelných podmínek a různé kvality. V následující tabulce je uvedeno, kolik originálních obrázků (bez obrázků uměle rozšiřovaných) je v každé kategorii. Ty byly získávány třemi hlavními zdroji:

- snímacím systémem popsaným v kapitole 6.1.1

- ručním snímáním nejběžnějších tvarů z důvodu rozšíření jejich datasetu a rozlišení od tvarově podobných dílků
- obrázky pořízené při testovacím provozu mechatronického systému a ručně zařazené do správných tříd

Tabulka 6.1: Četnost jednotlivých druhů (tříd) obsažených v základním datasetu

Třída	Počet	Třída	Počet
2x1_gray	894	**3x2-V_green**	868
2x1_red	1012	**4x1-M-P_gray**	701
2x1-O_gray	1079	**4x1-O_green**	1152
2x1-O_yellow	523	**4x1-P_gray**	547
2x1-P_gray	807	**4x1-P_red**	1252
2x1-V_black	864	**4x1-P_white**	438
2x1-V_yellow	713	**4x1-P_yellow**	1286
2x2_black	871	**4x2_red**	889
2x2_gray	694	**4x2-P_blue**	1038
2x2_red	902	**4x2-strange_blue**	234
2x2-3L-P_yellow	295	**4x4-P_blueGray**	430
2x2-Ox_black	469	**6x3-A-P_green**	422
2x2-Ox-P_gray	422	**8x1-P_blue**	447
2x2-P_lightGreen	877	**8x1-P_gray**	219
2x2-V_black	843	**8x1-P_yellow**	841
3x1_yellow	1161	**8x2-P_green**	643
3x1-P_yellow	1105	**10x6-P_gray**	190
3x2_blue	842	**wing_orange**	219

Při množství experimentů prováděných při návrhu architektury konvoluční neuronové sítě a při hledání optimálního nastavení jejích parametrů nebylo pracováno se všemi nasbíranými daty. To by vyžadovalo velký výpočetní výkon a spoustu času. Na takovou úlohu se nehodí běžný CPU, ale mnohem výhodnější je využít GPU, která je navržena pro vysokou pracovní zátěž a součin mezi vstupní hodnotou a váhami na každém neuronu může provádět paralelně. Navíc LEGO dílky dostupné při tvorbě diplomové práce měli velkou variabilitu tvarů a velikostí cílem aplikování strojového

učení na danou úlohu bylo získat schopnost rozeznat od sebe dílky tvarově podobné. Z toho důvodu se pro různé experimenty využívala pouze vybraná podmnožina tříd. Ve všech případech se data náhodně rozdělila na 3 části:

- trénovací sada – tvoří 60 % až 70 % z celkového počtu
- validační sada – tvoří 20 % až 25 % z celkového počtu
- testovací sada – tvoří 10 až 15 % z celkového počtu

6.2 Implementace SVM

Při konstrukci kvalitního klasifikátoru jsou klíčové kroky návrh trénovacího algoritmu a příprava vstupních dat. V práci jsou experimentálně porovnány výsledky učícího se algoritmu s různými vstupními daty, aby bylo možné nalézt pro danou úlohu optimální řešení.

6.2.1 Histogram orientovaných gradientů

Vstupem do SVM klasifikátoru může být kromě klasické obrazu i histogram orientovaných gradientů (HOG). Ve stručnosti lze říci, že HOG zachycuje obrysové informace šedotónových obrazů pro účely detekce objektů. Gradientní informace se shromažďují do 1D histogramu orientací, čímž se 2D obraz transformuje na mnohem menší 1D vektor. Ten tvoří vstup pro algoritmy strojového učení, jako jsou náhodné rozhodovací lesy, algoritmy podpůrných vektorů (SVM) nebo logistické regresní klasifikátory (LDA). [22]

Ze vstupního obrazu, na který byla aplikována maska pro výpočet diskrétního gradientu vzniknou obrazy I_x a I_y, tak že I_x je výsledkem konvoluce Gaussovsky filtrovaného obrazu I s maskou K, a I_y s transponovanou maskou K^T. Poté je pro každý pixel vypočtena velikost gradientu $m(x, y)$ a směr gradientu $\Theta(x, y)$ podle následujících vzorců:

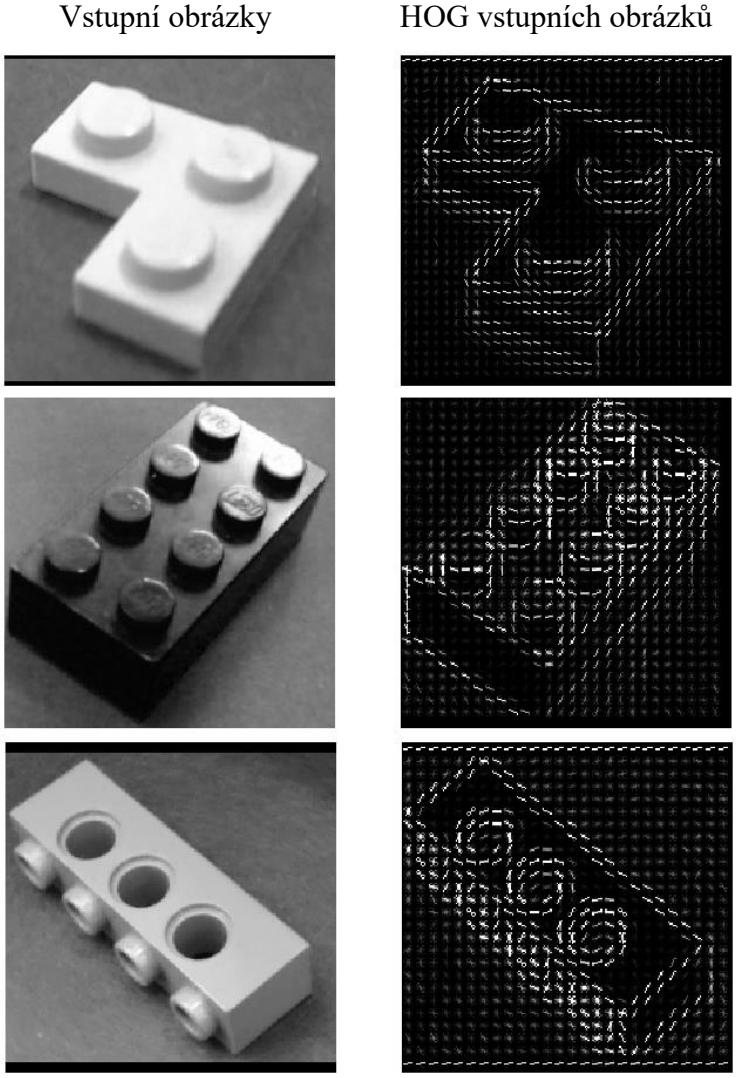
$$m(x, y) = \sqrt{I_x^2 + I_y^2}, \tag{6.1}$$

$$\Theta(x, y) = \tan^{-1}\left(\frac{I_x}{I_y}\right). \tag{6.2}$$

Zkonstruovaný histogram orientací má počet tak zvaných binů určený intervalem $\mathrm{i} = \langle 0{,}2\pi \rangle$, udávající rozsah, ve kterém se zjišťuje směr gradientu $\Theta(x, y)$.

Sektory *s* definují počet stejně velkých výsečí, na které je daný interval *i* rozčleněn. Obvykle jsou ještě získané histogramy orientací normalizovány. [23]

Obrázek (Obrázek 6.5) vizualizuje histogram pro tři různé objekty. Počet binů byl zvolen 9 a velikost výsečí 8×8.



Obrázek 6.5: Vlevo jsou šedotónové obrázky a vpravo jejich popis pomocí HOG

Implementace HOG je v jazyce Python velmi jednoduchá. Využila se k tomu knihovna OpenCV ve verzi 3.1.0. V kódu je funkce, která ve zvolené složce načte obrazové soubory, převede je do stupňů šedi, sjednotí jejich velikost a spočítá histogram orientovaných gradientů. funkce vrátí celé pole hodnot spolu s počtem obrazů, které jsou v tomto poli uložené. Jejich počet je důležitý mimo jiné pro to, že další funkce, která má na starosti vytvoření vstupních polí pro metodu SVM, vytváří pole tříd. Pro jednoduchost,

pokud budou ve složce "2x2_red" 3 obrázky a v "4x1-P_yellow" budou 2, pak pole tříd bude mít následující podobu: ['2x2_red', '2x2_red', '2x2_red, '4x1-P_yellow', '4x1-P_yellow'].

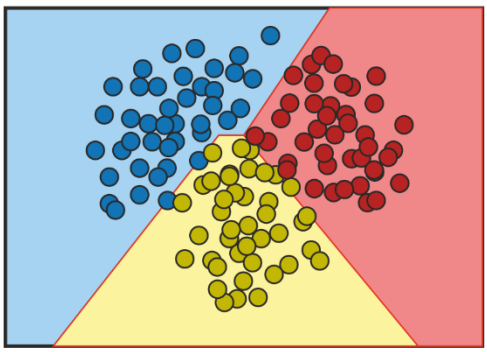
Algoritmus, který převede barevný obraz z prostoru RGB na obraz ve stupních šedi pro každý obrazový bod se řídí podle následujícího vzorce:

$$Y = 0.2989 * R + 0.587 * G + 0.114 * B. \qquad (6.3)$$

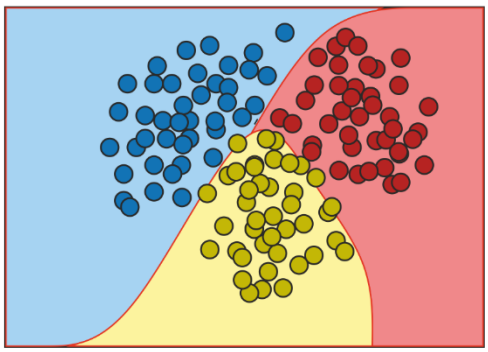
6.2.2 Testování SVM

Následují odstavce přibližují některé výsledky, které byly získány při pokusech s různými vstupními datasety a různým nastavením SVM algoritmu. Datové sady byly voleny tak, aby byly zastoupeny pouze nejpodobnější tvary kostek. Získá se tak dobrý přehled o síle daného klasifikátoru za přijatelně dlouhou dobu. Pro práci s SVM byla využita volně dostupná knihovna strojového učení *scikit-learn* pro Python. Důležitý parametr při definování klasifikátoru je typ jádra, který nabýval nejčastěji dvou hodnotu, *linear* nebo *rbf* a měl velký vliv na výsledný natrénovaný model. Jedná se o lineární a nelineární způsob oddělení dat různých kategorií. Ilustrace (Obrázek 6.6) přibližuje rozdíly zvoleného jádra pro 3 kategorie.

Funkce SVC s lineárním jádrem



Funkce SVC s RBF jádrem



Obrázek 6.6: Znázornění rozdílu lineárního a nelineárního jádra (podle [24])

Před spuštěním trénovacího procesu je ve scriptu nutné zadat cestu k adresářům pro trénovací a testovací obrázky. Tato data jsou následně načtena a převedena na HOG, nebo stupně jasu a spolu s názvy tříd použita jako vstup pro algoritmus.

První experiment probíhal pouze na dvou třídách, a to na kostce tvaru 3×1 a 3×1 plate. Cílem bylo ověřit, zda lze rozeznat od sebe stejné kostky o různé tloušťce, pokud

je obrázek snímán ze šikmého úhlu. Je to i jeden z důvodů nasazení strojového učení na úloho třídění LEGO dílků. Výsledná tabulka (Tabulka 6.2) ukazuje, že nejúspěšnější a zároveň jedna z nejrychlejší metoda dopadla SVM se vstupními daty HOG.

Počet tříd	2
Počet trénovacích dat	1708

Tabulka 6.2: Popis úspěšnosti klasifikace SVM pro vstupní data HOG a šedotónových obrázků

Klasifikátor	Úspěšnost	Doba trvání
SVM *(linear kernel)* + HOG	95,53 %	1 min
SVM *(rbf kernel)* + HOG	91,62 %	3 min
SVM *(linear kernel)* + GRAY	89,38 %	1 min
SVM *(rbf kernel)* + GRAY	74,86 %	2 min
SVM *(signum kernel)* + HOG	68,15 %	2,5 min
SVM *(polynomial kernel)* + HOG	44,69 %	2,5 min

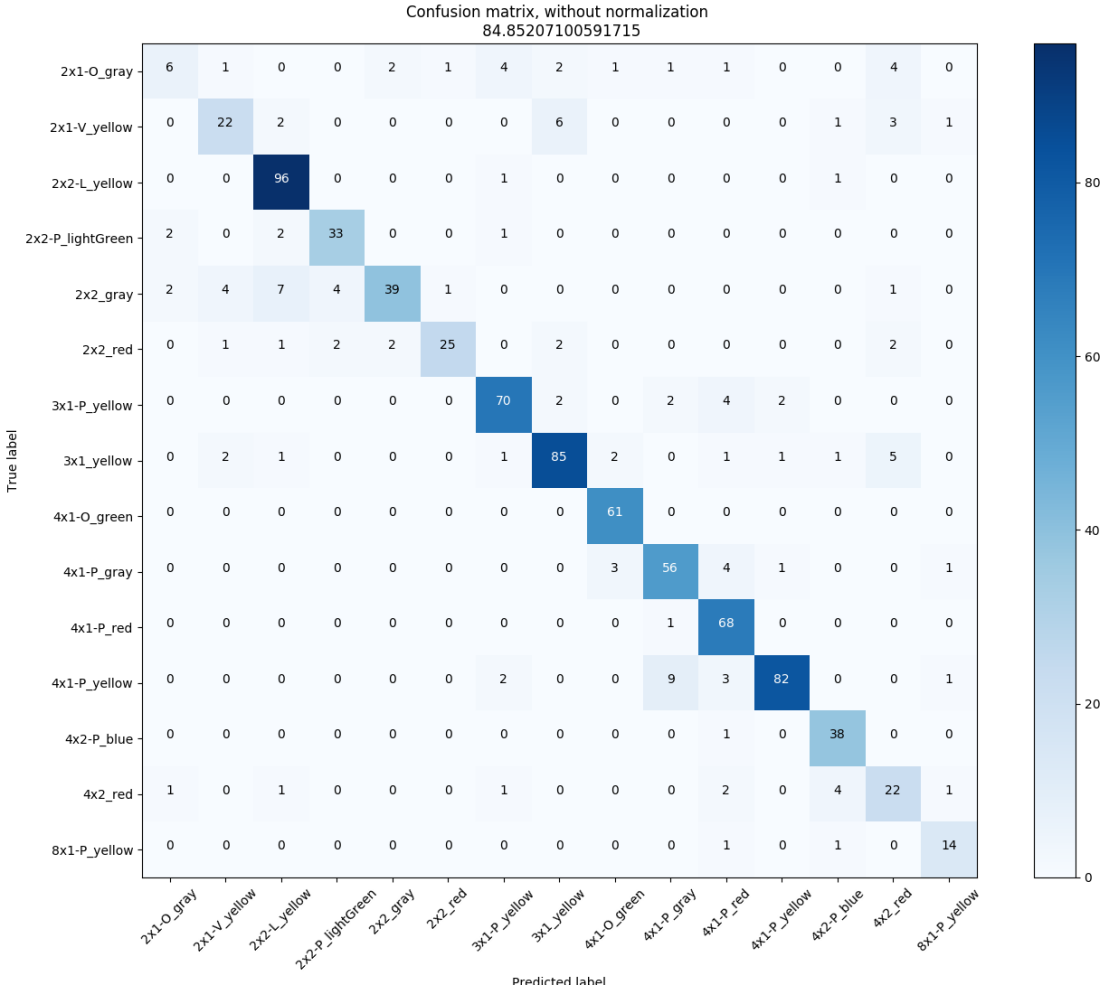
Výsledky pokusů s navyšujícím se počtem tříd dopadaly velmi podobně. Jako další je uveden výsledek s použitím datasetu o 15 kategoriích, opět poskládaný, pokud možno z podobných dílů. Záměrně vynechány byly tedy dílky velikostně i tvarově velmi odlišné. Výsledky z předchozích pokusů pomohly postupně vyloučit některá neoptimální nastavení. Po porovnání úspěšného klasifikování testovacích dat, které nebyly součástí trénovací sady vede k výsledku, že optimální nasazení SVM zahrnuje výpočet HOG pro každý snímek a zvolení lineární jádrové funkce.

Počet tříd	15
Počet trénovacích dat	10433
Počet testovacích dat	845

Tabulka 6.3: Výsledky SVM pro 15 kategorií

Klasifikátor	Úspěšnost	Doba trvání
SVM *(linear kernel)* + HOG	84,85 %	45 min
SVM *(linear kernel)* + GRAY	78,10 %	20 min

SVM *(rbf kernel)* + HOG	41,18 %	2 h



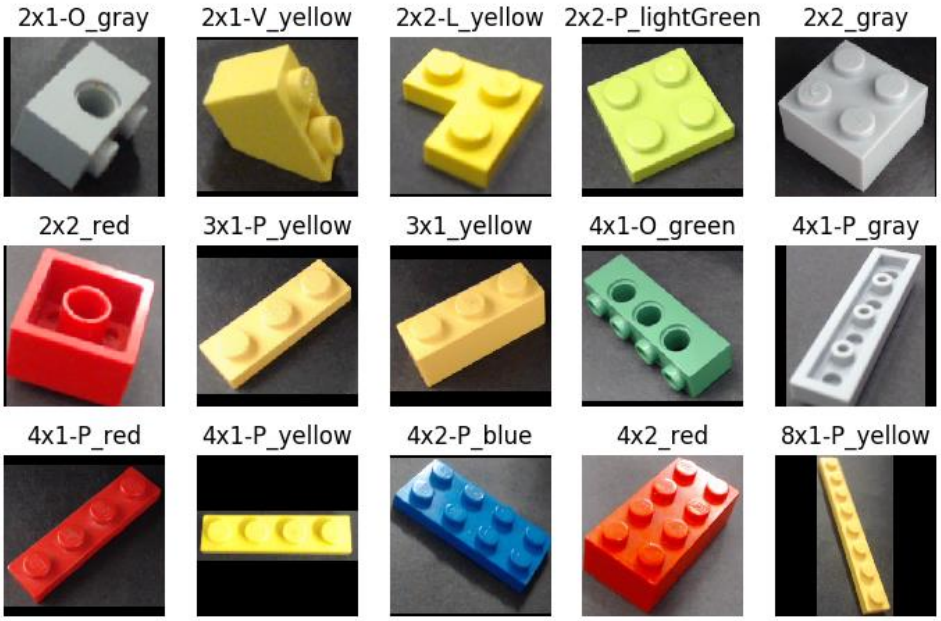
Obrázek 6.7: Matice zobrazující počty úspěšně a neúspěšně predikovaných obrázků

Matice na obrázku (Obrázek 6.7) je speciální druh tabulky, který vizualizuje počet chyb a počet úspěšně zařazených objektů pro každou třídu. Každý řádek představuje testovanou třídu a každý sloupec pak její předpovídané zařazení natrénovaným modelem. Pokud by úspěšnost byla bez chyby, vyplněná by byla pouze hlavní diagonála. Takovouto matici vykreslí a uloží do souboru navrhnutý trénovací algoritmus automaticky po dokončení procesu učení.

Třídící systém zařazuje LEGO kostky do koncových boxů podle barvy. Zajímavé tedy bylo zjistit, jaká je úspěšnost předpovězení barvy i za cenu nesprávné celkové klasifikace objektu. Pokud se bude zkoumat pouze správná předpověď barvy a bude se

vycházet z nejlepšího výsledku pro 15 tříd, očekávaná procentuální úspěšnost musí být z principu lepší nebo stejné již předešlým 84 %. Výsledek 88,3 % však není nijak výrazně lepší.

V tabulce (Obrázek 6.7) jsou používány popisky pro jednotlivé kategorie, u kterých nemusí být vždy zcela jasné o jaký tvar LEGO kostky se vlastně jedná. Pro snazší orientaci v názvech jsou na obrázku (Obrázek 6.8) ukázány nejčastěji používané tvary.



Obrázek 6.8: Ukázka druhů objektů, které tvořili trénovací a testovací dataset

Test rozpoznání kostky stejného tvaru, ale 4 různých barev dopadl velmi dobře. Jak je vidět v tabulce (Tabulka 6.4), úspěšnost dosáhla až na 100 procent. V třídách bylo celkem 5531 obrázků a test dokázal, že ačkoliv se při vstupu každá barevná bitmapa převede na matici s hodnotami intenzity jasové funkce, rozeznání od sebe různobarevných dílků se provede spolehlivě. Jediná kostka, která byla ve stejném tvaru zastoupena 4 barvami byla 4x1 plate.

Počet tříd	4
Počet trénovacích dat	5531

Tabulka 6.4: Výsledky pro SVM se 4 třídami

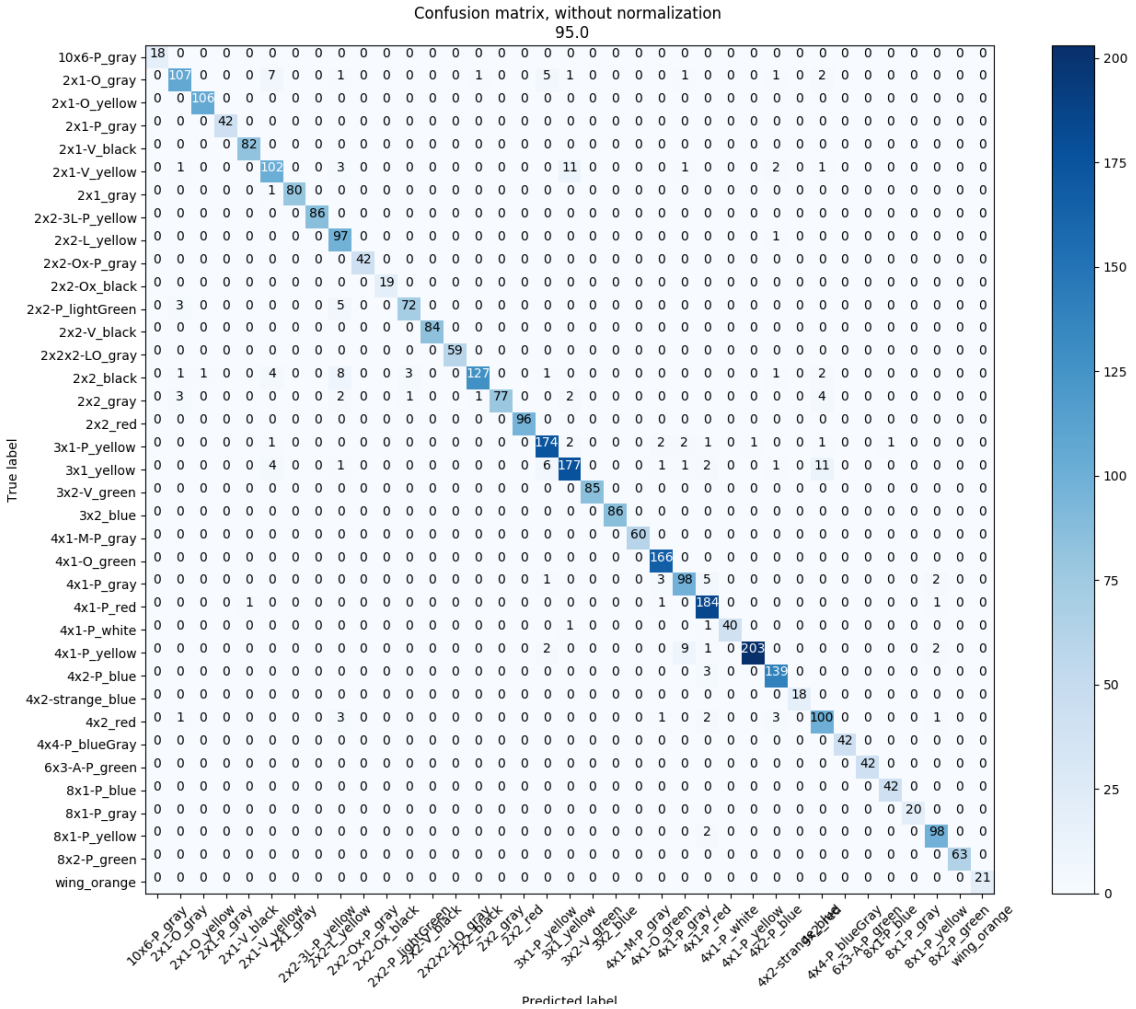
Klasifikátor	Úspěšnost	Doba trvání
SVM *(linear kernel)* + HOG	99,07 %	10 min
SVM *(linear kernel)* + GRAY	100 %	3 min

Poslední experiment, který je zde uvedený se týkal všech dostupných tvarů. Celkem bylo použito přes 32000 snímků pro trénovací množinu a 3300 pro množinu testovací. Procentuální úspěšnost je zde přibližně o 10 % větší než v předchozí sadě o 15 třídách. Je to pravděpodobně dáno přítomností některých nezaměnitelných tvarů. V konečné tabulce úspěšností pro metodu SVM (Tabulka 6.5) se vstupními obrázky popsaných pomocí HOG (Obrázek 6.9) se může nalézt 21 kategorií, jejichž testovací obrázky byly klasifikovány bez chyby.

Počet tříd	37
Počet trénovacích dat	32647
Počet testovacích dat	3320

Tabulka 6.5: Klasifikátor SVM pro 37 tříd

Klasifikátor	Úspěšnost	Doba trvání
SVM *(linear kernel)* + HOG	95,00 %	7,5 h
SVM *(linear kernel)* + GRAY	92,86 %	2 h



Obrázek 6.9: Matice úspěšnosti 37 tříd pro metodu SVM + HOG

6.3 Implementace CNN

Cílem této části práce bylo zjistit, jaké je vhodná architektura neuronové sítě a jaké parametry mají pro danou úlohu nejzásadnější vliv na výsledný model. Trénování bylo opět prováděno na různých datových sadách. Část reprezentativních výsledků je uvedeno v těchto kapitolách. Pro trénování a vytváření modelu byl použit nástroj TensorFlow a jazyk Python. K manipulaci s obrazovými daty opět posloužila knihovna OpenCV. Při psaní kódu posloužila předloha z [25], kde jsou dobře popsány základní operace v TensorFlow pro návrh a implementaci konvoluční neuronové sítě.

6.3.1 Framework TensorFlow

TensorFlow je knihovna s otevřeným zdrojovým kódem určená pro složité numerické výpočty, především pak pro práci s multidimenzionálními datovými poli (tenzory). Výpočty jsou založené na grafové struktuře, ve které jsou jednotlivé operace reprezentovány uzly a hrany mezi nimi představují tenzory. Kromě jazyka Python je dostupná i pro C++. Flexibilní architektura umožňuje provádět výpočty i na více CPU nebo GPU. TensorFlow byl původně vyvinut inženýry z týmu Google Brain v rámci organizace Google AI za účelem provádění strojního učení a výzkumu hlubokých neuronových sítí. Systém je dostatečně obecný, aby byl použitelný i v celé řadě dalších oblastí. [26]

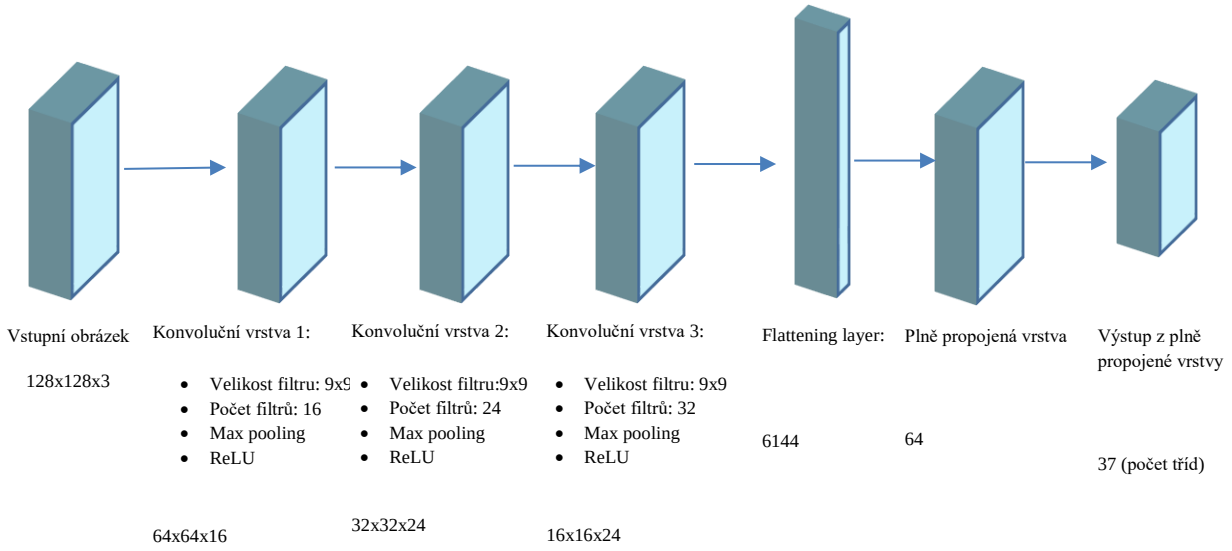
6.3.2 Vstupní data

Stejně tak jako při implementaci SVM i zde byly jednotlivé třídy v datasetu odděleny v samostatných adresářích, přičemž při načítání vstupních dat (pro trénování i pro testování) se jako název třídy automaticky zvolil název daného adresáře. Každý celkový dataset byl rozdělen na dva podadresáře, jeden pro trénování a druhý určený pro testovací obrázky, pro které byla po každém procesu trénování sítě spočítána celková úspěšnost a vytvořena matice chybovosti (angl. Confusion matrix). Tato testovací data tvořila asi 15 % z celkového počtu nasbíraných obrázků. Ze zbylých trénovacích dat se 20 % použilo pro validaci, aby bylo možné pomocí nich počítat přesnost během procesu učení.

Konvoluční vrstva nepřijímá obrazová data v klasickém formátu jako dvoudimenzionální pole pro každou ze tří barev, ale musí se nejdříve transformovat na jednodimenzionální tenzor. Z jednoho obrázku se tak stane vektor velikosti *šířka*výška*počet_kanálů*. V práci byly nejdříve používány obrázky velikosti 128×128, později se však přistoupilo k omezení velikosti na 64×64. Za cenu ztráty některých detailů se tím výrazně urychlil adaptační algoritmus.

6.3.3 Návrh architektury neuronové sítě

Prvotní návrh sítě (Obrázek 6.10) měl vstupní konvoluční vrstvu, která přijímala obrázky již v transformované podobě, jak bylo zmíněno v předchozí kapitole. Následovali další dvě konvoluční vrstvy, po kterých se tvar dat musel opět transformovat v tzv. flattening layer (zplošťovací vrstvě). Na konci byly dvě plně propojené vrstvy, z nichž druhá měla *x* výstupů, reprezentující *x* různých tříd.

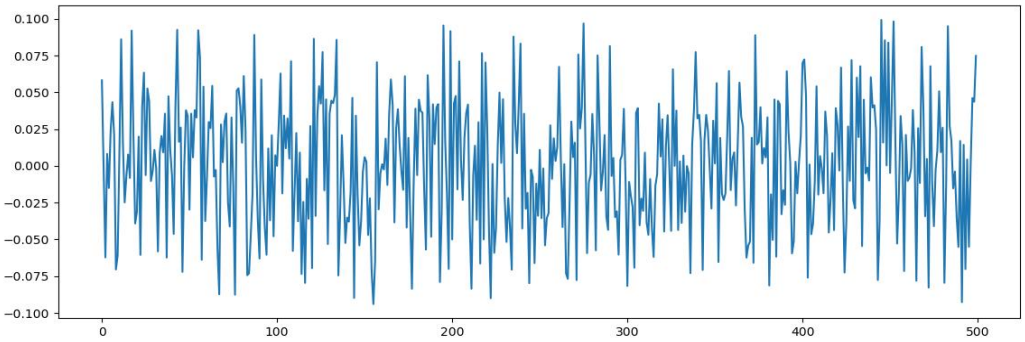


Obrázek 6.10: Příklad architektury CNN

Zatím co u biologických sítí se získané znalosti ukládají v dendritech, kam je signál přiváděn z předchozích neuronů, u neuronů umělých je při trénování zkušenost ukládána ve vahách. Tyto váhy tedy představují paměť neuronu a jejich hodnotu nastavuje adaptační algoritmus.

Prvotní inicializace vah je důležitým krokem, protože přímo ovlivňuje schopnost učení. Jestliže je proces učení založen na zpětné propagace chyby (angl. Backward propagation), tak se během něho váhové koeficienty postupně mění tak, aby se pro každý trénovací obrázek získal správný výstup.

Při jejich nastavování je potřeba pouze dbát na to, aby jejich hodnoty nebyly všechny stejné, nebo nulové. Také je dobré, aby vektor vah byl tvořen malými čísli okolo nuly. V práci jsou tyto hodnoty generovány náhodně v rozmezí přibližně od −0,1 do 0,1. V grafu (Obrázek 6.11) je pro ilustraci vygenerována náhodná sada váhových koeficientů (weights) s 500 hodnotami, které mohou být použity jako váhy při počáteční inicializaci.



Obrázek 6.11: Náhodně generované váhy

Počet vah v jedné konvoluční vrstvě je odvozen od velikosti a počtu filtrů. Pokud bude mít vrstva 16 filtrů o velikosti 5×5 a vstupní obraz bude mít 3 barevné kanály, počet parametrů bude součin těchto čtyř čísel, tedy 1200.

Samotná 2D konvoluce má zajímavý parametr „*strides*“, který definuje, o kolik se filtr během konvoluce posouvá. Je-li nastaven na [1, 1, 1, 1], posun je prováděn po jednom pixelu. Pokud je [1, 2, 2, 1], filtr se posouvá o 2 pixely po ose *x*, i po ose *y*, atd. Toto přeskakování je implementováno hlavně z důvodu snížení počtu prováděných konvolucí a urychlení výpočtů.

Jestliže se ve vrstvě využívá funkcí ReLU i maxpooling, je výhodné provést nejdříve podvzorkování, aby se snížil počet pixelů, nad kterými se aktivační funkce ReLU bude vykonávat.

Postupu použitý při trénování konvoluční sítě shrnutý do pěti kroků:

1. V prvním kroku dochází k normalizaci obrazových dat, která jsou následně rozdělena na trénovací, testovací a validační množinu tak, aby byl průnik všech tří množin prázdný.
2. Před spuštění učení s učitelem je definována struktura celé sítě a inicializovány počáteční hodnoty vah a biasů.
3. Provede se epocha, což je jeden průchod všech trénovacích dat sítí. U každého vstupního obrazu, který je převeden na tenzor, je známá požadovaná odezva sítě. Pokud se výsledná odezva liší od správné kategorie, jsou podle toho upraveny váhové vektory.

4. Jednotlivé epochy z bodu 3 jsou prováděny tak dlouho, dokud se nevyrovná jejich předem definovaný počet, nebo dokud se vyhodnocená chyba na validační množině nesníží pod nastavenou hranici.
5. Po dokončení adaptace je vyhodnocena úspěšnost predikce na testovací množině.

6.3.4 Testování CNN

Experimenty probíhaly obdobně jako u SVM. Nicméně počet parametrů, které měli nezanedbatelný vliv na výsledek bylo mnohem více, a tak i prováděné testy byly mnohem obsáhlejší. V této kapitole jich je uvedeno pouze zlomek z celkového počtu a jsou vybrány tak, aby přibližovaly vliv nastavení sítě na konečnou úspěšnost predikce.

Prví může být uveden příklad se dvěma třídami žluté kostky 3×1 a 3×1 plate. Do obrázků je již předem zanesen černý okraj pro získání čtvercového formátu. Počet trénovacích dat byl 1367, validačních 341 a testovacích 179. Průměrný dosahovaný výsledek byl 93 %. Doba potřebná pro natrénování sítě a dosažení takovýchto výsledků byla po 95 epochách 1 hodina, což se nedá srovnávat s metodou SVM, které stačila 1 minuta.

Test na detekci barvy u kostek stejných tvarů dopadl s úspěšností mezi 85 % a 97 %. Vzniklé chyby byly u klasifikace šedého dílku 4×1, který byl opakovaně označen jako bílý. Někdy byl takto zaměněn i dílek žlutý.

Nejvíce experimentů probíhalo na datové sadě s 15 druhy kostek, protože měla dobrou vypovídající hodnotu a obsahovala dílky, které se často používaly v třídícím mechanismu. Jestliže jsou v tabulce (Tabulka 6.6) filtry popsány jako *filtry[(3,32),(3,32),(3,64)])* pak to znamená, že v první a druhé konvoluční vrstvě bylo použito v každé 32 filtrů velikosti 3×3 a v poslední jich bylo 64. Z výsledných záznamů je jasné, že využití tří konvolučních vrstev nebylo optimální a pro danou úlohu se hodí počet vrstev zredukovat.

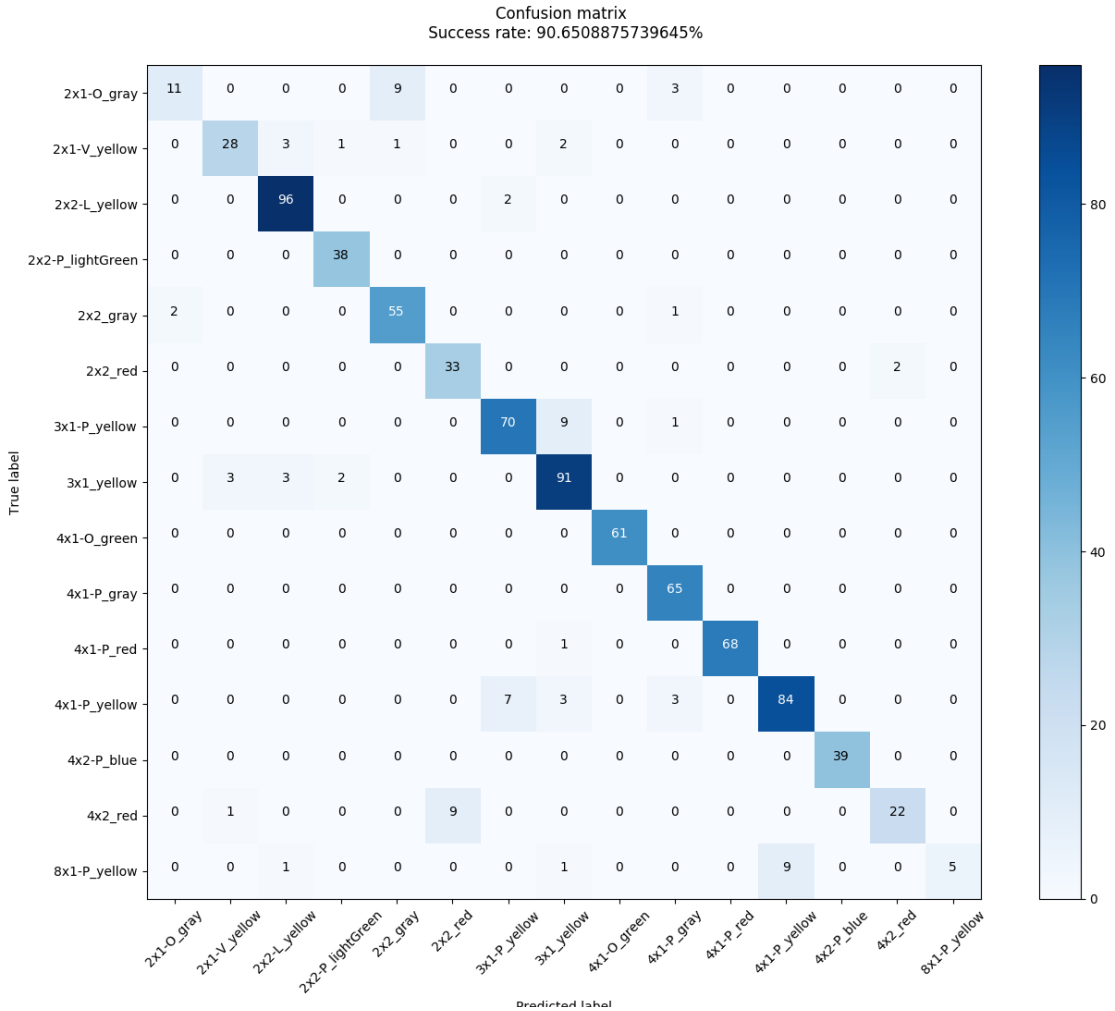
Poslední dva řádky v tabulce zachycují jev, kterému se říká přetrénování. Překročil se okamžik, kdy byla minimální chybovost na validačních datech a tím i vhodná doba pro ukončení trénování. Od té doby ztrácel trénovaný model schopnost generalizace, tedy predikovat neznámé obrázky na základě podobnosti s naučeným modelem a příliš se orientoval na validační množinu. Proto se ani po mnohonásobné prodloužení doby trénování nezlepšila procentuální úspěšnost správného zařazení.

Počet tříd	15
Počet trénovacích dat	8347
Počet validačních dat	2086
Počet testovacích dat	845

Tabulka 6.6: Výsledky CNN pro 15 tříd

Klasifikátor	Počet epoch	Doba trvání	Úspěšnost
CNN ***(1 kon. vrstva; filtry [(13,32)])***	22	2 h	90 %
CNN ***(1 kon. vrstva; filtry [(17,64)])***	20	2 h	90 %
CNN ***(1 kon. vrstva; filtry [(21,32)])***	21	1,5	90 %
CNN ***(1 kon. vrstva; filtry [(15,32)])***	20	3	89 %
CNN ***(2 kon. vrstvy; filtry [(21,32), (15,32)])***	20	3 h	89 %
CNN ***(2 kon. vrstvy; filtry [(15,32), (11,32)])***	25	2,5 h	89 %
CNN ***(2 kon. vrstvy; filtry [(5,32), (5,32)])***	81	2,5 h	88 %
CNN ***(1 kon. vrstva; filtry [(9,16)])***	23	0,5 h	86 %
CNN ***(1 kon. vrstva; filtry [(3,32)])***	23	0,5 h	85 %
CNN ***(3 kon. vrstvy; filtry [(3,16),(3,16),(3,32)])***	117	2,5 h	85 %
CNN ***(3 kon. vrstvy; filtry [(8,16),(16,32),(24,32)])***	26	5,5 h	84 %
CNN ***(3 kon. vrstvy; filtry [(17,24),(9,24),(9,32)])***	197	10	82 %
CNN ***(3 kon. vrstvy; filtry [(16,16),(16,16),(24,32)])***	72	3 h	81 %
CNN ***(3 kon. vrstvy; filtry [(16,32),(16,32),(16,64)])***	72	5 h	79 %
CNN ***(3 kon. vrstvy; filtry [(8,16),(16,24),(24,32)])***	36	1,5 h	76 %
CNN ***(3 kon. vrstvy; filtry [(8,16),(16,24),(24,32)])***	358	12,5 h	75 %

Z tabulky (Obrázek 6.12) zachycující počty chybě a správně klasifikované snímků jednotlivých kategorií lze vyčíst, kolik bylo správně rozpoznaných barev. Chyb, kdy se nějaký dílek zařadil jako jiný, který měl kromě tvaru i jinou barvu, bylo pouze 10. Po vyjádření v procentech vyjde správnost predikce barvy 98,8 %, což je o 10 % více, než v případě nejlepšího výsledku za použití metody SVM a HOG.

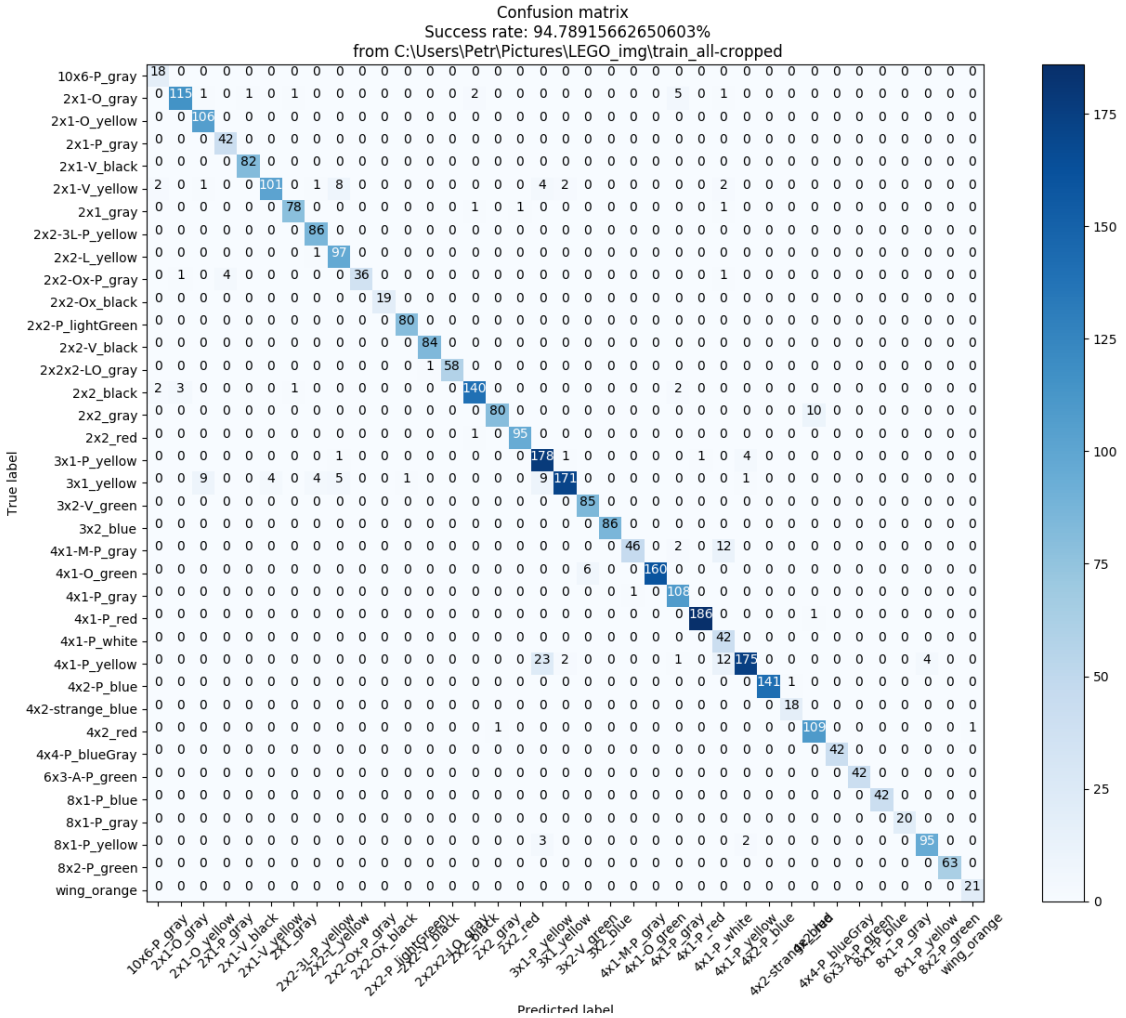


Obrázek 6.12: Matice chybovosti pro CNN

Předešlá datová množina byla použitá i pro porovnání různých aktivačních funkcí. Za pomoci jediné konvoluční vrstvy a 16 filtry velikosti 15×15 vyšla nejlépe funkce ReLU s 85 %. Při zachování stejných parametrů sítě byl výsledek s funkcí hyperbolický tangens 80 % a sigmoid pouhých 63 %.

Následující experiment byl proveden na celkové množině 37 tříd. Po mnoha neuspokojivých pokusech s několika konvolučními vrstvami a při laborování se změnami parametrů vyšla nejlépe jediná konvoluční vrstva s velikostí filtru 15×15. Trénování trvalo 3 hodiny, ale úspěšnost se vyrovnala metodě SVM, konkrétně 94,79 %.

Z tabulky (Obrázek 6.13) je zřejmé, že výraznějších chyb síť dosahovala při predikci kostky *4x1-P_yellow*, kterou několikrát klasifikovala jako *3x1-P_yellow* a také *4x1-P_gray*. Dále došlo k záměně *4x1-M-P_gray* s *4x1-P_gray*. Tyto tvary se liší pouze povrchem horní strany a z jistého úhlu je není možné bezpečně rozeznat.



Obrázek 6.13: Matice chybovosti kompletního datasetu pro CNN

Testy provedené nad uměle rozšířeným datasetem dopadly podle očekávání lépe, než bez přidaných dat. Dataset, na který byly aplikovány augmentační techniky byl použit již v předchozích testech a měl obrázky rozdělené do 15 různých tříd. Nejlepší dosažený výsledek pro něj byl 90 %.

Augmentace spočívala v oříznutí obrazu ze všech stran, změně kontrastu, lokální náhodné přeskupení pixelů, aplikace průměrovacího filtru a rotaci obrazu kolem středu o 15°, 30°a 45°. Z jednoho obrazu vzniklo pět nových. Takto rozšířená byla trénovací a validační sada. Přesné počty a některé výsledky jsou ukázány v následujících tabulkách.

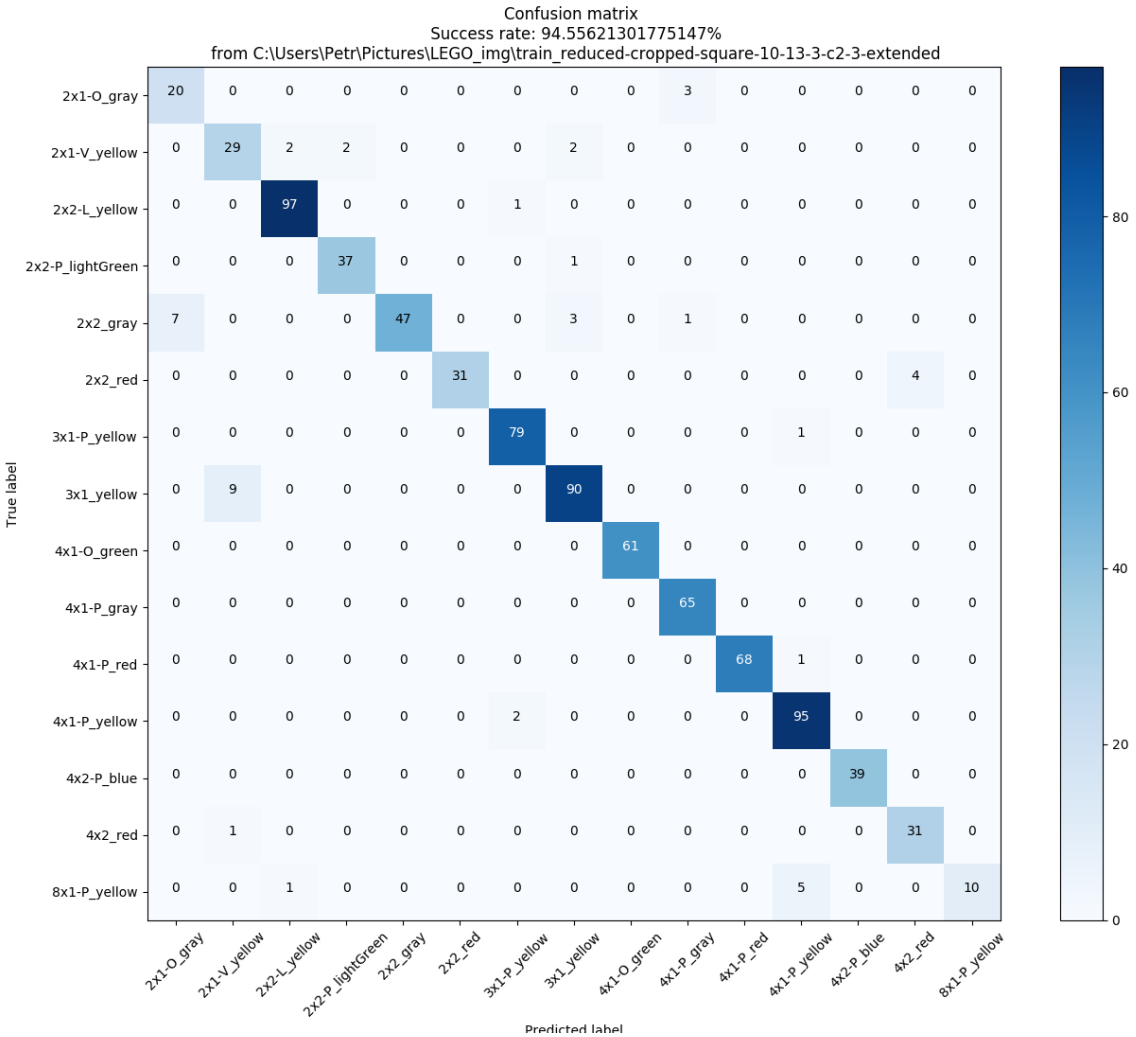
Počet tříd	15
Počet trénovacích dat	50079

Počet validačních dat	12519
Počet testovacích dat	845

Tabulka 6.7: Výsledky CNN po umělém rozšíření datasetu

Klasifikátor	Počet epoch	Doba trvání	Úspěšnost
CNN ***(2 kon. vrstvy; filtry [(15,32),(3,32)])***	33	9 h	94,55 %
CNN ***(2 kon. vrstvy; filtry [(15,32),(3,32)])***	11	3 h	92,42 %
CNN ***(3 kon. vrstvy; filtry [(3,16),(3,16),(3,32)])***	117	3,5	85,27 %
CNN ***(3 kon. vrstvy; filtry [(5,16),(9,32),(13,32)])***	15	8,5	73,36 %

Podrobnější tabulka s výsledky nejúspěšnějšího testu je na obrázku (Obrázek 6.14). Takto navržený model lze bezpečně využít v ovládacím programu pro klasifikaci reálných objektů.



Obrázek 6.14: Výsledky experimentu s uměle rozšířenou datovou sadou

Poslední uvedený experiment měl za cíl zjistit, zda je možné natrénovat síť uměle vytvořenými daty. K této metodě se přistupuje, pokud nejsou z nějakého důvodu k dispozici data reálná, ale nelze ji použít vždy. Typicky v úlohách rozeznání tváře, zvířat, ručně psaného textu a mnoho dalších hraje významnou roli detail a různorodost prostředí. V takových případech není možné pořízené fotografie rozumně nahradit. Stavebnice LEGO má v tomto směru určitou výhodu, protože jednotlivé kostky mají přesný rozměr a jednoduchý tvar.

Pro vymodelování konkrétních kostek v trojrozměrné grafice by bylo nutné znát jejich přesné rozměry a poměry stran. Pro zjednodušení byl k tomuto účelu vybrán program „LEGO Digital Designer“ přímo od tvůrců LEGO. Ten umožňuje uživatelům stavět modely z virtuálních LEGO kostek dostupná je většina možných tvarů a barev.

Program nabízí pohled na jednotlivé dílky a měnit jejich úhel a natočení. Získávání snímků touto cestou neomezuje jejich množství, ale věrohodnost. Reálné snímky jsou zatíženy šumem a stíny tvořené osvětlením. Umělým datům tyto zdánlivé vady chybí a jsou v tomto ohledu příliš přesné. Částečným řešením může být například augmentace datasetu, což bylo využito i v této práci.

Uměle vytvořená a rozšířená data spadala do 7 kategorií a v každé bylo okolo 1000 obrázků. Testování pro určení celkové úspěšnosti klasifikace probíhalo na reálných snímcích. Architektura sítě vycházela z předchozích uvedených pokusů nad reálnými daty. Bylo vyzkoušeno použití jedné i dvou skrytých konvolučních vrstev, ale úspěšnost predikce se stále pohybovala okolo 45 %. I když dílky několika tříd byly rozpoznány téměř bezchybně, bylo to dáno spíše jejich unikátním tvarem nebo barvou. Výsledkem tohoto experimentu bylo zjištění, že reálná data nelze těmi umělými zcela nahradit.

Závěr

V rámci diplomové práci byl vytvořen komplexní mechatronický systém pro třídění LEGO dílků. Použita k tomu byla stavebnice LEGO MINDSTORM NXT. Systém lze rozdělit na tři dílčí části, z nichž každá má specifickou funkci. První částí je dopravníkový pás, na který se vloží dílky, které se mají roztřídit. V prostřední části je použita kamera, snímající černou podložku pod sebou. Jakmile na podložku spadne dílek, pás se zastaví a celý systém má tak dostatek času na rozpoznání nového objektu a jeho následné zařazení do koncové části systému.

Důležitým úkolem mechatronického systému je separovat od sebe nesetříděné množství vstupních LEGO dílků tak, aby pod kameru padaly vždy po jednom. Práce byla ale omezena nedostatkem stavebních prvků a z důvodu realizovatelnosti a zachování robustnosti celé soustavy nakonec separaci zajišťuje pouze jediný dopravník.

Pro mechatronický systém byla naprogramována ovládací aplikace, která po zpracování a vyhodnocení obrazových dat pořízených z kamery ovládá třídičku.

Metody strojového učení, na kterých byly postaveny klasifikátory LEGO dílků, vyžadovaly velké množství obrazových dat. Pro tyto účely byl vytvořen rozsáhlý dataset obsahující 36 kategorií a v každé 400 až 1200 různých obrázků. Velká část obrazových dat LEGO kostek byla pořízena poloautomaticky pomocí rotující podložky a kamery.

Testovaná metoda SVM dosahovala na všech třídách úspěšnosti 95 %. Vstupem pro tuto metodu nebyly samotné obrázky, ale jejich HOG. Výsledek na redukovaném datasetu čítající 15 tříd byl 84,85 %. Tato datová sada měla lepší vypovídající hodnotu díky tvarové podobnosti dílků v jednotlivých třídách.

Navržená CNN dosahovala na redukovaném datasetu úspěšnost predikce až 90 %. Umělé rozšíření datové množiny přineslo zlepšení přibližně o dalších 5 %. Experimentálně byla nalezena optimální architektura sítě pro danou úlohu, využívající pouze jednu konvoluční vrstvu a aktivační funkci ReLU.

Navržený mechatronický systém třídí dílky podle barvy do koncových boxů. Nejúspěšnější CNN na redukované množině určuje barvu s přesností 98,8 %, proto je tento natrénovaný model implementován i do ovládacího programu, ve kterém se využívá ke klasifikaci jednotlivých LEGO dílků.

Literatura

[1] HLAVÁČ, Václav a Miloš SEDLÁČEK. *Zpracování signálů a obrazů*. 2. přeprac. vyd. Praha: ČVUT, 2007, 255 s. ISBN 978-80-01-03110-0.

[2] FOJTÍK, David, Jaromír ZAVADIL a Petr PODEŠVA. Návody ke stavebnici LEGO MINDSTORMS pro týmová cvičení v předmětu výpočetní technika. Ostrava, 2010.

[3] Lego Mindstorms NXT. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-03-24]. Dostupné z: https://en.wikipedia.org/wiki/Lego_Mindstorms_NXT

[4] Getting Started with Mindstorms NXT. In: *Smashing Robotics* [online]. Dan Mihai, 2014 [cit. 2018-03-24]. Dostupné z: https://www.smashingrobotics.com/getting-started-with-mindstorms-nxt/

[5] Lego Robotics. *AForge.NET :: Computer Vision, Artificial Intelligence, Robotics* [online]. [cit. 2018-03-25]. Dostupné z: http://www.aforgenet.com/framework/features/lego_robotics.html

[6] SHALEV-SHWARTZ, Shai. a Shai. BEN-DAVID. *Understanding machine learning: from theory to algorithms*. New York, NY, USA: Cambridge University Press, 2014. ISBN 978-1-107-05713-5.

[7] ALPAYDIN, Ethem. *Introduction to machine learning*. 2nd ed. Cambridge, Mass.: MIT Press, c2010. Adaptive computation and machine learning. ISBN 978-0-262-01243-0.

[8] CHAO, Wei-Lun. *Machine Learning Tutorial* [online]. National Taiwan University: DISP Lab, 2011, December, 2011 [cit. 2018-03-25].

[9] Support vector machines. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-03-25]. Dostupné z: https://cs.wikipedia.org/wiki/Support_vector_machines

[10] KAČENKA, Petr. Neuronové sítě. *Historie, využití neuronových sítí a často používané typy sítí* [online]. Mariánská, 1998 [cit. 2018-03-25]. Dostupné z: https://mks.mff.cuni.cz/library/NeuronoveSitePK/NeuronoveSitePK.pdf

[11] A Beginner's Guide to Deep Convolutional Networks. *Deep Learning for Java* [online]. [cit. 2018-04-03]. Dostupné z: https://deeplearning4j.org/convolutionalnetwork.html

[12] FIŘT, Jaroslaf a Radek HOLOTA. Digitalizace a zpracování obrazu. In: *Digitální mikroskopie a analýza obrazu v metalografii: sborník z 1. mezinárodní konference, Plzeň, 25.9.2002*. Plzeň: Západočeská univerzita, 2002. ISBN 80-7082-917-6.

[13] VLACH, Jaroslav. *Metody zpracování obrazu pro časově náročné úlohy*. Liberec, 2012. Dostupné z: http://www.mujweb.cz/jvlach/Disertace_Vlach_2012.pdf. Disertační práce. Technická univerzita v Liberci. Vedoucí práce doc. Ing. Milan Kolář, CSc.

[14] HLAVÁČ, Václav a Miloš SEDLÁČEK. *Zpracování signálu a obrazu* [online]. Elektrotechnická fakulta ČVUT v Praze, 1999 [cit. 2018-04-03]. Pracovní verze skripta v tisku pro studenty. FEL ČVUT.

[15] MILAN ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image Processing, Analysis, and Machine Vision*. 4. USA: Cengage Learning, 2014. ISBN 978-285981444.

[16] HÁJOVSKÝ, Radovan, Radka PUSTKOVÁ a František KUTÁLEK. *Zpracování obrazu v měřící a řídící technice*. Vyd. 1. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2012. 1 DVD-ROM. ISBN 978-80-248-2596-0. Diplomová práce. Mendelova univerzita v Brně Provozně ekonomická fakulta. Vedoucí práce prof. RNDr. Ing. Jiří Šťastný, CSc.

[17] SOJKA, Eduard. *Digitální zpracování a analýza obrazů*. Ostrava: VŠB-Technická univerzita, 2000. ISBN 80-7078-746-5.

[18] FRIBERT, Miroslav. *Základy zpracování obrazu* [online]. Pardubice, 2006 [cit. 2018-03-12]. Dostupné z: http://docplayer.cz/19212230-Zpracovani-obrazu-v-merici-a-ridici-technice.html. Skripta. UNIVERZITA PARDUBICE FAKULTA CHEMICKO-TECHNOLOGICKÁ.

[19] HORAK, Karel. Cvičení 10 - Morfologické operace. *Multimedia Interactive Didactic System* [online]. VUT FEKT Brno: Computer Vision Group, 2010 [cit. 2018-03-16]. Dostupné z: http://midas.uamt.feec.vutbr.cz/ZVS/Exercise10/content_cz.php

[20] ŽELEZNÝ, Miloš. *Zpracování digitálního obrazu* [online]. [cit. 2018-04-06]. Dostupné z: http://www.kky.zcu.cz/uploads/courses/zdo/ZDO_aktual_130215.pdf. Učební text. Západočeská univerzita, Fakulta aplikovaných věd, Katedra kybernetiky.

[21] List of datasets for machine learning research. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-04-10]. Dostupné z: https://en.wikipedia.org/wiki/List_of_datasets_for_machine_learning_research

[22] Histogram of Oriented Gradients (HOG) Boat Heading Classification. *Https://medium.com/* [online]. Adam Van Etten, 2016 [cit. 2018-04-16]. Dostupné z: https://medium.com/the-downlinq/histogram-of-oriented-gradients-hog-heading-classification-a92d1cf5b3cc

[23] *LBP, HoG* [online]. 2015 [cit. 2018-04-11]. Dostupné z: http://www.kky.zcu.cz/uploads/courses/mpv/05/materialy05.pdf. Učební text. Západočeská univerzita, Fakulta aplikovaných věd, Katedra kybernetiky.

[24] Support Vector Machines. *Scikit-learn Machine Learning in Python* [online]. scikit-learn developers, 2017 [cit. 2018-04-11]. Dostupné z: http://scikit-learn.org/stable/modules/svm.html#svm-kernels

[25] Tensorflow Tutorial 2: image classifier using convolutional neural network. *CV-Tricks.com Learn Machine Learning, AI & Computer vision* [online]. Ankit Sachan [cit. 2018-04-11]. Dostupné z: http://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/

[26] About TensorFlow. TensorFlow [online]. [cit. 2018-03-13]. Dostupné z: https://www.tensorflow.org/

[27] MANTHIRA MOORTHI, S., Indranil MISRA, Rajdeep KAUR, Nikunj P DARJI a R. RAMAKRISHNAN. Kernel based learning approach for satellite image classification using support vector machine. In: *2011 IEEE Recent Advances in Intelligent Computational Systems* [online]. IEEE, 2011, 2011, s. 107-110 [cit. 2018-04-12]. DOI: 10.1109/RAICS.2011.6069282. ISBN 978-1-4244-9478-1. Dostupné z: http://ieeexplore.ieee.org/document/6069282/

[28] ŽIŽKA, Jan. *SUPPORT VECTOR MACHINES* [online]. 2009 [cit. 2018-04-18]. Dostupné z: https://akela.mendelu.cz/~zizka/Machine_Learning/Presentace/SVM-introduction.pdf. Prezentace. Ústav informatiky, PEF, Mendelova universita v Brně.

[29] *Support vector machines (SVM)* [online]. Brno, 2006 [cit. 2018-04-02]. Dostupné z: https://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf. Studijní materiály předmětu FI:PA034. Masarykova univerzita.

[30] MOLNÁR, Karol. Úvod do problematiky umělých neuronových sítí. *Elektrorevue* [online]. 2000, **2000**(13) [cit. 2018-04-11]. ISSN 1213-1539. Dostupné z: http://www.elektrorevue.cz/clanky/00013/index.html

[31] ÇELEBİ, Ömer Cengiz. Celebi Tutorial: Neural Networks and Pattern Recognition Using MATLAB. *Neural Networks Tutorial* [online]. [cit. 2018-04-18]. Dostupné z: https://www.byclb.com/TR/Tutorials/neural_networks/ch7_1.htm

[32] JACOBSON, Lee. Introduction to Artificial Neural Networks - Part 1. *Neural Networks Tutorial* [online]. 2013 [cit. 2018-03-28]. Dostupné z: http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7

[33] *CS231n Convolutional Neural Networks for Visual Recognition* [online]. [cit. 2018-03-28]. Dostupné z: http://cs231n.github.io/neural-networks-1/]

[34] GIBSON, Adam a Josh PATTERSON. *Deep Learning* [online]. August 2017. O'Reilly Media, 2017 [cit. 2018-04-18]. ISBN 9781491924570. Dostupné z: https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/ch04.html

[35] ANIEMEKA, Ifu. A Friendly Introduction to Convolutional Neural Networks. *Hashrocket* [online]. 2017, August 22, 2017 [cit. 2018-03-28]. Dostupné z: https://hashrocket.com/blog/posts/a-friendly-introduction-to-convolutional-neural-networks

[36] Neuron. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-04-22]. Dostupné z: https://commons.wikimedia.org/wiki/File:Neuron.svg

[37] A Guide to LEGO Robotics. *Razor Robotics* [online]. 2017 [cit. 2018-04-30]. Dostupné z: https://www.razorrobotics.com/robot-kits/lego/

A Obsah přiloženého DVD

- Petr_Kaspar-DP.pdf: tato práce ve formátu pdf
- Dataset15: adresář s 15 třídami obrázků
- Dataset37: adresář s 37 třídami obrázků
- DP-control_program: projekt ovládacího programu z IDE Visual Studia
- Scripts: zdrojové kódy pomocných scriptů v Pythonu