Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Identifikace ovoce metodami strojového učení

Autor: Bc. Patrik Dítě
Studijní obor: IM-2 K

Vedoucí práce: doc. RNDr. Kamila Štekerová, Ph.D.

Hradec Králové 04/2020

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 30.4.2020 Patrik Dítě

Poděkování:

Rád bych poděkoval své vedoucí diplomové práce doc. RNDr. Kamile Štekerové, Ph.D. za metodické vedení práce, její odborné rady a podporu při řešení problematiky.

Anotace

Diplomová práce se zabývá metodami strojového učení používanými pro identifikaci druhu ovoce z fotografie. Cílem práce bylo zhodnotit možnosti využití metod, navrhnout a vytvořit ukázkovou aplikaci pro rozpoznávání ovoce. Z provedeného průzkumu byly získány informace o používaných modelech a metodách a na jejich základě byly navrženy 4 konvoluční neuronové sítě. Ty byly pomocí experimentů s architekturou, normalizací, daty a parametry testovány a laděny. Bylo dosaženo 98,09% přesnosti na testovací sadě Fruits-360 obsahující 120 druhů ovoce.

Klíčová slova

Strojové učení; Konvoluční neuronové sítě; Neuronové sítě; Hluboké učení; Reprezentativní učení; Strojové vidění; Rozpoznávání ovoce

Annotation – Title: Identification of Fruit Types Using Machine Learning

The master thesis deals with machine learning methods for identification of fruit types from photographies. The aim of the work was to create an application for fruit recognition and identification. A literature reivew was conducted to summarize the state of the art. Then four convolutional neural networks were proposed, implemented and tuned using experiments with architecture, normalization, data and parameters. 98,09 % accuracy was achieved on the Fruits-360 test set, containing 120 types of fruit.

Keywords

Machine learning; Convolutional neural network; Neural networks; Deep Learning; Representation learning; Machine vision; Fruits identificaiton

Obsah

Seznam obrázků

Seznam tabulek

Seznam rovnic

Seznam grafů

Seznam zdrojových kódů

Seznam použitých zkratek

AUC – Area Under Curve – Plocha pod křivkou

AVG – Average – Průměr

BN – Batch normalization – Dávková normalizaceBoC – Bag-of-Curvature

BoSV – Bag-of-Shape-Vocabulary

BoW – Bag-of-Words

CCV – Color Coherence Vector – Vektor soudržnosti barev

CNN – Convolutional Neural Network – Konvoluční neuronová síť

DL – Deep learning – Hluboké učení

DT – Distance transformation – Transformace vzdálenosti

EOAC – Edge Orientation Autocorrelogram

FC – Fully connected – Plně propojená vrstva

FN – False negative – Nesprávně negativní

FNN – Feedforward Neural Networks – Dopředné neuronové sítě

FP – False positive – Nesprávně pozitivní

GAN – Generative Adversarial Network – Generativní kompetitivní síť

GD – Gradient Descend – Klesání podle gradientu

GPU – Graphics processing unit – Grafická výpočetní jednotka

HOG – Histogram of Oriented Gradient – Histogram orientovaných gradientů

HSI – Hue, Saturation, Intensity – barevné schéma; barevný tón, sytost, intenzita

HSV – Hue, Saturation, Value – barevné schéma; barevný tón, sytost, jas

CHT – Circular Hough Transform

ILSVRC – ImageNet Large Scale Visual Recognition Challenge

IoU – Intersection over Union -

KNN – k-Nearest Neighbours – Algoritmus nejbližších sousedů

LBP – Local Binary Patterns – Místní binární vzory

LRPBP – Loacl Relative Phase Binary Pattern – Místní relativní fázový binární vzor

MADM – Multi Attribute Decision Making – Algoritmus rozhodování o více atributech

MAX – Maximum

ML – Machine learning – Strojové učení

MLP – Multilayer Percepton – Vícevrstvé perceptony

NN – Neural Network – Neuronová síť

PCA – Principal Component Analysis – Analýza hlavních komponent

PReLU – Parametric Rectified Lineaur Unit - Parametrická usměrňovací lineární jednotka

R-CNN – Reginal CNN – Regionální konvoluční neuronová síť

ReLU – Rectified Lineaur Unit – Usměrněná lineární jednotka

RGB – Red Blue Green – Základní barevné schéma červená, modrá, zelená

RHT – Random Hough Transform

ROC – Receiver Operating characteristic Curve – Operační křivka

ROI – Region of Interest – Průnik sjednocení

SGD – Stochastic Gradient Descend – Stochastický gradientní sestup

SGDM – Spatial Gray-level Dependence Matrix – Prostorová matice závislosti na šedé úrovni

SIFT – Scale Invariant Feature Transforms

SLIC – Simple Linear Iterative Clustering

SSD – Single Shot Detectors

SVM – Support Vector Machines – podpůrné vektory

TF – Tensorflow

TN – True negative – Správně negativní

TP – True positive – Správně pozitivní

VAE – Variational Autoencoder – Variační autoenkodér

WDCCN – Weakly Dense Connected Convolution Network – Slabě hustě propojená konvoluční síť

YIQ – barevné schéma

YOLO – You Only Look Once

YUV – barevné schéma

1 Úvod

Počátky strojového učení sahají do 50. let 20. století, kdy byl tento pojem poprvé zaveden v odborné literatuře. Vývoj prošel mnoha úskalími. Použitelnost a popularita rostly v důsledku zvyšování výpočetního výkonu a digitalizace společnosti, kdy bylo k dispozici čím dál více dat, jejichž autory nejsou dnes už jen lidé, ale i stroje, modely, algoritmy produkující obrovské množství dat každý den. Problémem je zpracování dat do takové podoby, aby mohla být použita pro strojové učení. To bývá stále ještě lidská práce, nicméně již nyní je možné najít několik algoritmů strojového učení, které dokážou například oštítkovat předem neviděné fotografie. V minulosti se postupně rozšiřovala odvětví aplikující strojové učení. Jednou takovou oblastí je i zemědělství, kde strojové učení přispívá k automatizovanému obdělávání půdy, sběru plodů či řízení plně autonomních farem. K dosažení takového cíle je důležité správné rozpoznávání plodů, rostlin, škůdců, nemocí, zralosti nebo kvality z obrazu. Tato práce se zaměřuje na identifikaci druhu ovoce z fotografie. Pomáhá tak v rozvoji automatizovaných systémů pro třídění a identifikaci.

Toto téma bylo autorem vybráno z důvodu zájmu o rozvoj profesní kariéry v oblasti strojového učení při aplikaci v zemědělství, kde je důležitá optimalizace procesů a možné zvýšení produkce při optimálních podmínkách a provedení úkonů ve správný čas.

Práce řeší problematiku přístupu strojového učení k rozpoznávání ovoce. Teoretická část představuje východiska strojového učení a konceptu rozpoznávání obrazu. Popisuje, jaké metody jsou používány a s jakými výsledky. Má zároveň upozornit na komplexnost dané domény. První kapitola teoretické části byla věnována popisu základních metod strojového učení, aby se čtenář orientoval v navazujícím textu. Popsány byly hlavní přístupy ke strojovému učení, klíčové algoritmy, podoba experimentů, vyhodnocovací metriky a jsou nastíněny výchozí podmínky pro úspěšné sestavení modelu. Druhá kapitola byla věnována rozpoznávání obrazu, jeho specifikám, podmínkám a obtížím a jejich řešení. Dále je uveden obecný návrh modelu a zpracování obrazových dat. Popsány jsou zejména rozdíly mezi dvěma hlavními přístupy ke strojovému vidění. Tyto přístupy jsou podstatnými determinanty strategie tvorby modelu. V poslední kapitole byly analyzovány a srovnány state-of-the-art přístupy k rozpoznávání ovoce v různých doménách použití. Každá tato doména má svá specifika a algoritmy jsou jim uzpůsobeny.

Praktická část je členěna na tři kapitoly. V první je rozebrán návrh několika architektur modelu strojového učení na základě zjištěných informací v teoretické části a postup, parametry experimentů. Ve druhé kapitole je popsáno technické zázemí a vývojové prostředí pro učení modelu, stejně jako samotná implementace s vysvětlením důležitých částí kódu. Poslední kapitola je věnována interpretaci získaných výsledků experimentů a jejich vyhodnocení.

Odborná literatura nemá ustálený český slovník pojmů pro strojové učení. Byl proto převzat většinový překlad, originální názvy jsou uváděny v závorce při prvním výskytu a v seznamu požitých zkratek. Zároveň během překladu pokročilých metod a algoritmů bylo obtížné najít český překlad, a proto jsou některé názvy uváděny v anglickém jazyce.

2 Cíl práce

Hlavním cílem práce bylo zhodnotit možnosti využití metod strojového učení při identifikaci druhů ovoce z fotografií, navrhnout a realizovat ukázkovou aplikaci. Taková aplikace měla využívat poznatků a informací zjištěných v odborné literatuře. Vytvořený model strojového učení by se měl držet standardů a obecného modelu učení. Nemělo by chybět předzpracování dat a jejich následné vyhodnocení. Je zde předpoklad, že by úspěšnost modelu měla být srovnatelná s aktuálními modely při použití metod z odborné literatury.

Cíl lze dále rozvrhnout na dílčí úkoly:

- Zmapovat metody strojového učení používané pro identifikaci druhů ovoce.
- Vyhodnotit získané informace a vytvořit shrnutí.
- Navrhnout model strojového učení na základě získaných poznatků a informací včetně technických náležitostí, zajištění výpočetního výkonu a datové sady.
- Realizovat navržený model a optimalizovat řešení pro co nejvyšší úspěšnost.
- Vyhodnotit realizaci a získané výsledky.

3 Metodika zpracování

Pro průzkum metod a technik strojového učení využívaných pro identifikaci ovoce bylo využito vědeckých databází dostupných v rámci univerzity – Science Direct, Web of Science, Springer a Scopus. Byly hledány zmínky o strojovém učení v souvislosti s rozpoznáváním ovoce, případně zeleniny. Hledání probíhalo několikafázově, nejprve byla hledána různá klíčová slova (strojové učení, ovoce, identifikace ovoce, klasifikace ovoce, strojové vidění ovoce atp.) podle relevance, poté podle data, kde primárně byly zkoumány články maximálně 2 roky staré a poté podle nejvyššího citačního indexu. Některé články se v jedné knihovně vyskytovaly jako placené, v jiné byly k dispozici. Pokud se objevil článek, který se podle abstraktu jevil jako přínosný, a byl placený na všech dostupných knihovnách, byl hledán i mimo. Občas se takové články povedlo najít na portálu *researchgate.com* nebo *arxiv.org*. Pro organizaci článků, dokumentů a zdrojů v rámci práce byl použit citační program Mendeley. Získané zdroje byly členěny do složek, popisovány a zvýrazňovány. Články byly tříděny, celkem jich bylo nalezeno přes 420. Z nich bylo v práci použito kolem 40. K vědeckým knihovnám se autor vracel v průběhu práce, pokud bylo třeba získat informace na konkrétní téma, byl vyhledán článek s odpovídající hodnotou.

Další část práce se opírá o publikaci *Deep Learning Book* dostupnou na adrese *deeplearningbook.com,* jako o primární zdroj informací o hlubokém učení a konvolučních neuronových sítích a jejich optimalizacích a o knihu *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms. Worked Examples, and Case Studies,* ze které bylo čerpáno pro část metod strojového učení.

Pro praktickou část bylo hojně využito oficiálních dokumentací Keras a Tensorflow [1], [2], hlavně při implementaci kódu. V práci nejsou uváděny odkazy na dokumentaci při použití jednotlivých funkcí a popisů jejich parametrů, vše je možné dohledat ve zmíněných zdrojích. Samotný kód byl psán vlastnoručně s využitím oficiálních informací. Pro návrh modelu bylo využito informací jednak získaných z článků, jednak dále dohledaných, jelikož na ně bylo v článcích jen odkazováno, či nebyly v odborných pracích řešeny vůbec. Byly vytvořeny varianty modelů a následně seznam parametrů a variant, které mají být testovány pro získání nejlepších výsledků. Byla provedena série testů. Poté se pokračovalo v testování dle seznamu s nejlepší architekturou. Během experimentů bylo zjištěno, že ne vždy nejlepší architektura má lepší výsledky na následném testu, a proto bylo experimentováno s více než jedním modelem pro jejich

srovnání a ověření správného směru. Některé experimenty nepřinesly pozitivní výsledek. V některých případech bylo třeba zpětně otestovat variantu z předchozí iterace testů s mírně pozměněnými parametry na základě nově získané informace z pozdějšího experimentu. Celkem bylo provedeno 69 experimentů. S celkovou časovou náročností na výpočetní zařízení 346 hodin, tj. 14 dní, průměrně 5 hodin na experiment. Mnoho experimentů skončilo s chybou, nebo nepodaly odpověď na otázku (chyba byla v nastavení parametrů nebo v architektuře). Vyskytovaly se problémy s alokací paměti grafické karty, proto muselo být poté přidáno omezení pro hladký průběh testování. Stroj na provádění experimentů byl zapůjčen. Výsledky získané z experimentů byly zpracovány v tabulkovém procesoru Excel, kde se vedla evidence provedených experimentů. Průběhy učení byly vykresleny do grafů pomocí knihovny matplotlib v jazyce Python.

4 Teoretická část

První část této diplomové práce slouží jako podklad pro část praktickou. Představuje teoretická východiska, která čtenáře informují o řešeném problému, a zároveň o vybraných prvcích konceptu rozpoznávání obrazu. Tato část má zároveň upozornit na komplexnost dané problematiky. Dále jsou v teoretické části analyzovány *state-of-the-art* přístupy k rozpoznávání ovoce a nastíněny výchozí podmínky pro úspěšné sestavení modelu. Popsány jsou zejména rozdíly mezi dvěma hlavními přístupy ke strojovému učení. Tyto přístupy jsou podstatnými determinanty strategie tvorby modelu.

4.1 Metody strojového učení

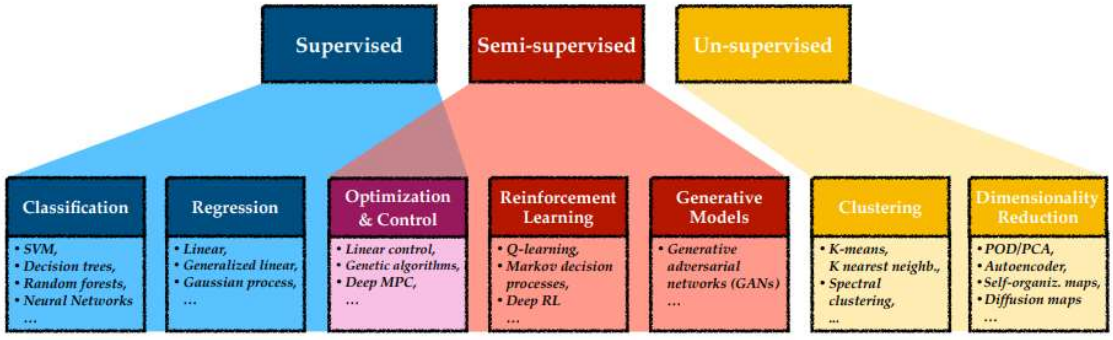
V této kapitole bude nastíněna disciplína strojového učení, její rozdělení a metody. Dále budou rozvedeny techniky používající se primárně pro klasifikaci a rozpoznávání obrazu, následně bude popsáno vyhodnocení kvality modelu. V závěru kapitoly bude popsána optimalizace modelů strojového učení.

4.1.1 Strojové učení

Strojové učení je definováno jako automatický proces, který extrahuje vzorce z dat [3]. Souvisí úzce s lineární algebrou, pravděpodobnostní teorií, statistikou a matematickou optimalizací. Obvykle jsou stavěny modely strojového učení založené na prvních třech zmíněných přístupech a následně probíhá optimalizace modelů. Model přijímá data jako vstup (tzv. datová sada, dataset) – mohou to být číselná, textová, obrazová nebo audiovizuální data. Obvykle má model i výstupy – může jimi být číslo s plovoucí desetinou čárkou (např. zrychlení automobilu) či celé číslo (např. označující kategorii či třídu při klasifikaci). Hlavním úkolem strojového učení je zkoumat a konstruovat algoritmy, které se mohou učit z historických dat, a následně vytvářet predikce z nových vstupních dat. Pro řešení založené na datech je třeba definovat (nebo nechat definovat algoritmem) vyhodnocovací funkci zvanou ztráta (loss function, někdy také nákladová funkce, cost function), která měří, jak dobře se modely učí. Poté se vytváří optimalizační problém s cílem minimalizovat nákladovou funkci a získat tím nejefektivnější učení [4]. Model strojového učení zkoumá data na základě příznaků (charakteristik, v odborné literatuře features) – například v doméně rozpoznávání obrazu jsou příznaky hrany, linie, přechody. Složitějšími jsou potom např. při úloze rozpoznávání tváře oči, nos, pusa atp. Model

strojového učení poté zjišťuje, která charakteristika koreluje s jakým výsledkem pro následnou predikci [5].

Spousta zadání strojového učení může být vyřešena správným návrhem sady příznaků a jeho interpretací přenesení do modelu. Naopak pro některá zadání je velmi těžké určit, které charakteristiky by měly být extrahovány. Jedno z řešení tohoto problému je nechat model najít tyto charakteristiky samostatně a poté se naučit reprezentaci vztahu mezi nimi a výstupy. Tento přístup se nazývá reprezentativní učení (representation learning). Naučená reprezentace charakteristik má často za následek mnohem lepší výkon než jakého lze dosáhnout s ručně navrženými charakteristikami. Umožňují také modelům strojového učení rychle se přizpůsobit novým úkolům s minimálním zásahem člověka. Takový model může objevit sadu příznaků pro jednoduchý úkol v několika minutách nebo pro složitý úkol v hodinách až měsících. Ruční navrhování příznaků pro složitý úkol vyžaduje hodně lidského času a úsilí [5].



Obrázek 1 *Rozdělení strojového učení*

Zdroj: [61], *strana 3, obr. 1*

4.1.2 Rozdělení strojového učení

Text této kapitoly je zpracován na základě publikace od S. Raschka [4]. Strojové učení lze dělit podle různých pohledů. Metody strojového jsou rozdělovány do tří základních kategorií, kde každá je vhodná pro určitý typ problému a jiná vstupní data (Obrázek 1). *Učení s učitelem* je základním stavebním kamenem strojového vidění a také se používá při rozpoznávání ovoce, proto mu bude věnována samostatná podkapitola, a dále budou popisovány metody a techniky spadající do této kategorie.

Učení bez učitele (unsupervised) - učící datová sada neobsahuje popisky s požadovaným výstupem a je tedy na učícím algoritmu, aby našel strukturu dat a objevil skryté vzory, informace a souvislosti mezi daty, nebo určil, jak je popsat. Učící data se nazývají jako neoznačená (unlabelled). Tento druh učení se používá pro detekci anomálií

(podvod, vadné vybavení atp.) nebo seskupení objektů do skupin podle společných atributů (zákazníci s podobným chováním pro marketingovou kampaň). Učení bez učitele lze dále dělit na primární dvě aplikace, tj. shlukování a redukce dimenze. Používanými algoritmy jsou:

- Shlukování třídící objekty do skupin sobě podobných objektů.
    1. Shlukování metodou k-průměrů (*k*-means clustering).
    2. Hierarchické shlukování (agglomerative clustering).
- Redukce dimenze je proces snížení počtu náhodných proměnných, znaků, které charakterizují objekt, na co nejmenší číslo při zachování stejné informace.
    1. Analýza hlavních komponent (Principal Component Analysis, PCA).
    2. Nezáporné matice aproximace (Non-negative Matrix Factorization, NMF).

Kombinace učení s učitelem a bez učitele (semi-supervised) - učící datová sada obsahuje jak označená (labelled), tak neoznačená data. Využívá velké množství neoznačených dat pro učení, vedle malého množství označených. Tento druh učení se aplikuje v případech, kde je finančně nebo časově nákladné či nemožné získat plně označenou datovou sadu, a proto je praktičtější označit pouze malou podskupinu. To často vyžaduje doménového experta. Získání neoznačených dat bývá snadné. Příkladem aplikace tohoto druhu učení je klasifikace proteinových sekvencí, kde odvození funkce proteinů obvykle vyžaduje aktivní lidský zásah.

Učení posilováním (reinforcement Learning) – Při učení je poskytována přímá zpětná vazba, takže model adaptuje výstup podle dynamicky se měnících podmínek pro dosažení určitého cíle. Používá se zde systém odměn. Model vyhodnocuje svůj výkon na základě zpětné vazby a adekvátně reaguje. Známým příkladem aplikace je řízení autonomního vozidla nebo modely hrající šachy, poker a jiné hry. Používanými algoritmy jsou:

- Q-učení (Q-learning),
- SARSA (State-Action-Reward-State-Action),
- kritérium optimality (criterion of optimality),
- metoda Monte Carlo.

4.1.3 Učení s učitelem

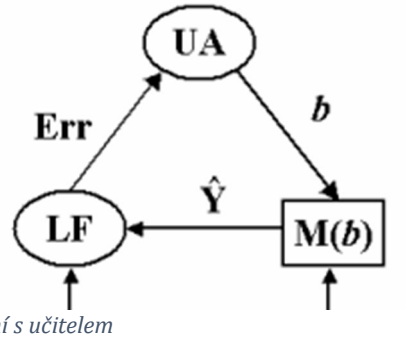
V učení s učitelem (supervised Learning) mají data pro učení popisek s požadovaným výstupem. Cílem učení je najít obecné pravidlo, které mapuje vstupy na výstupy. Data určená pro učení se nazývají jako označená (labelled). Naučené pravidlo poté slouží k označení nových dat. Označení dat je často prováděno systémy, které data generují, a lidmi. Tento druh učení je často používán v každodenních aplikacích, jako je rozpoznávání obličejů, řeči, produktů, obrázků či doporučení filmů a předpovědi prodeje nebo cen. Učení s učitelem lze rozdělit podle hlavních řešených problémů [4]. Lze jej dělit na dvě primární kategorie podle řešeného problému [3]:

- *Klasifikace* se snaží najít příslušnou třídu pro vstupní data, například určení, co je na obrázku, nebo zda e-mail zařadit mezi spamy.
- *Regrese* se na druhé straně učí z průběžně hodnocených dat, jako jsou ceny nemovitostí, akcií, a snaží se předpovídat jejich hodnotu.

Níže jsou uvedeny používané algoritmy a některé z nich budou popsány více v dalších kapitolách.

- Algoritmus nejbližších sousedů (k-Nearest Neighbours, KNN),
- lineární modely,
- naivní Bayesův klasifikátor (Naive Bayes Classifiers),
- rozhodovací stromy (Decision trees),
- algoritmy podpůrných vektorů (Support Vector Machines, SVM),
- neuronové sítě (Neural Networks, NN).

Základní schéma na Obrázek 2 níže popisuje komponenty modelu učení s učitelem. Skládá se ze vstupní veličiny X, výstupní veličiny Y, predikované výstupní hodnoty Y, modelu s parametry $M(b)$, nákladové funkce LF a učícího (optimalizačního) algoritmu UA. Nejprve jdou vstupní data X do modelu $M(b)$, poté je výstup modelu Ŷ porovnáván



Obrázek 2 Schéma modelu učení s učitelem

Zdroj: [6], *strana 9, obr. 1.2*

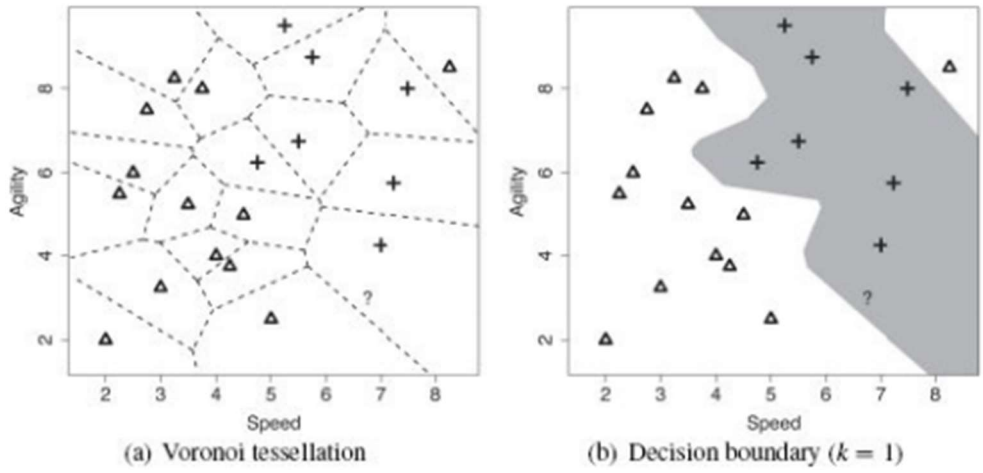
s požadovaným výstupem Y, kde nákladová funkce kvantifikuje míru vzniklé odlišnosti jako chybu Err. UA následně upraví parametry $M(b)$ na základě informace o chybě Err tak, aby její hodnota byla v příštím cyklu co nejmenší [6].

4.1.4 Algoritmus k-nejbližších sousedů (KNN)

KNN je jedním z nejjednodušších klasifikačních algoritmů [7]. Pro vytvoření předpovědí na nových datech algoritmus najde nejbližší bod(y) z tréninkové datové sady. Kolik bodů bere algoritmus v úvahu, uvádí parametr k. Algoritmus při hledání nejbližšího souseda používá Eukleidovskou metriku danou vzorcem [8]

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2} \qquad 1$$

a následně rozdělí prostor příznaků do tzv. Voroného diagramu (teselace) a rozhoduje se, do jakého Voroného regionu predikovaná hodnota náleží. Klasifikační hranici lze vygenerovat agregací Voroného regionů, které odpovídají stejné třídě, jak lze vidět na Obrázek 3 [3].



Obrázek 3 Voroného regiony (a) a klasifikační hranice v 1-NN klasifikátoru (b)

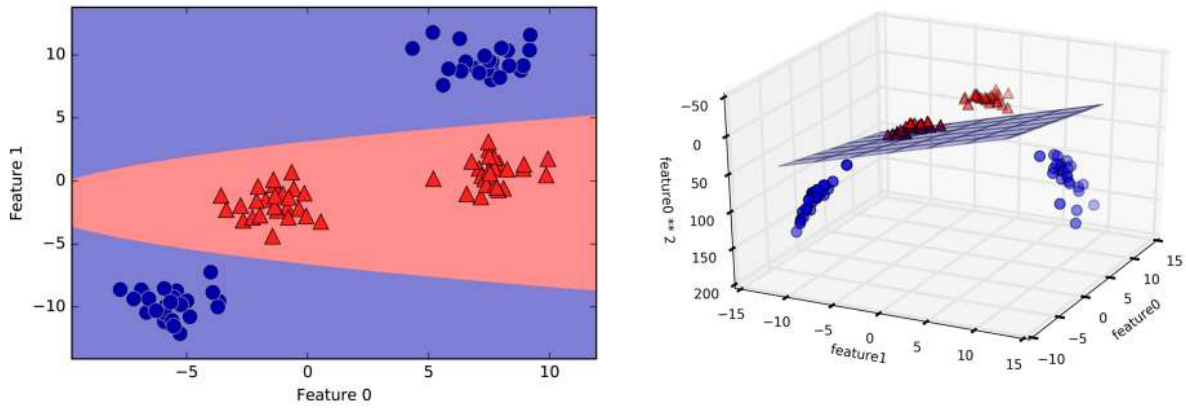
Zdroj: [3], *strana 244, obr. 5.4*

KNN klasifikátor lze použít pro klasifikaci do více tříd. Parametr k se určuje nejlépe laděním modelu pomocí křížové validace (cross-validation). Pro některé úlohy $k = 1$ může vést ke špatné generalizaci, jelikož se bere v úvahu jen první (nejbližší) soused. Z praxe je vhodné 5 nebo 7 sousedů [8].

4.1.5 Algoritmus podpůrných vektorů (SVM)

SVM je binární klasifikační metoda, která hledá nadrovinu v prostoru příznaků pro optimální rozdělení učících dat. Optimální je taková nadrovina, jejíž body leží v opačných poloprostorech a hodnota minima vzdálenosti bodů od nadroviny je co největší. Pro popis nadroviny jsou dostačující pouze nejbližší body, tzv. podpůrné vektory (support vectors). Rozdělující nadrovina je v prostoru příznaků lineární. Optimální lineární oddělovač se hledá pomocí kvadratického programování [3].

Součástí techniky SVM je jádrová transformace prostoru příznaků dat do prostoru vyšší dimenze (Kernel SVM, KSVM). Umožňuje transformovat původně lineárně neseparovatelnou úlohu na lineárně separovatelnou, kterou lze optimalizací vyřešit. KSVM však nejsou dobře škálovatelné. Na Obrázek 4 je vizualizováno rozšíření do další dimenze k nalezení nadroviny pro lineární separaci dvou příznaků [7].



Obrázek 4 SVM klasifikace

Zdroj: [7], *strana 95-96, obr. 2-38,2-39*

SVM lze rozšířit pro více klasifikačních tříd podle schématu „jedna versus zbytek“ nebo „jedna versus jedna“. Takový SVM se pak označuje jako MSVM (Multi SVM) [9].

4.1.6 Hluboké neuronové sítě

Rodina algoritmů známých jako neuronové sítě mají spoustu druhů a odvětví. Zde budou popsány pouze základní principy a typy používané pro klasifikaci, protože se o nich mluví dále v této práci.

Vícevrstvé perceptony (Multilayer Percepton, MLP), také známé jako dopředné neuronové sítě (Feedforward Neural Networks, FNN), často referované jen jako NN,

se skládají z vrstev, jejichž počet se označuje jako hloubka sítě, a ty z neuronů, jejichž počet ve vrstvě je označován jako šířka sítě. Neurony jsou na sebe napojeny – výstup z jedné vrstvy je vstupem do další. Pokud jsou všechny vstupy propojené se všemi výstupy, nazývá se vrstva plně propojená (fully connected, FC). První vrstva se nazývá vstupní, neprobíhá na ní žádný výpočet, pouze předává vstup na další vrstvu. Další vrstvy se nazývají skryté, protože z nich není primární výstup, a poslední vrstva se nazývá výstupní. Spoje mezi neurony mají své váhy (weights) a neurony mají tzv. bázi (bias) [10].

Neuron je lineární jednotka, tzn., provádí sumu součinů vstupů neuronů a jejich vah a následně přičte bázi. Matematicky to lze zapsat následovně:

$$\theta = W \cdot X + b\,, \qquad 2$$

kde θ je výstup, W je vektorem vah, X vstup a b báze. Tato jednotka může provádět pouze lineární transformace a pro popis složitějších funkcí musí být spárována s nelineární aktivační funkcí $g(\theta)$. Často používanými aktivačními funkcemi jsou logistická (sigmoid), usměrněná lineární (Rectified Lineaur Unit, ReLU) či hyperbolický tangent (tanh) [10].

Sítě se učí na základě minimalizace (dosáhnutí globálního minima) ztrátové funkce pomocí úpravy vah a bází mezi neurony díky algoritmu zpětné propagace (back-propagation algorithm). K optimalizaci funkce se používá klesání podle gradientu (Gradient Descend, GD), nyní spíše stochastický gradientní sestup (Stochastic Gradient Descend, SGD) díky jeho menší výpočetní náročnosti [5].

4.1.7 Vyhodnocení úspěšnosti modelu

Pro vyhodnocení úspěšnosti modelu existuje několik metrik, vzniklých na základě požadavků různých aplikací. Pro každou aplikaci je nezbytné zvolit takovou metriku, která bude nejlépe reflektovat schopnost modelu řešit daný problém. Měl by to být jeden z prvních kroků při realizaci modelu, protože zvolená metrika povede všechny budoucí kroky optimalizace modelu. Dalším předpokladem je alespoň hrubá představa o úrovni požadovaného výkonu modelu. V akademickém prostředí obvykle máme určitý odhad míry chyb, která je dosažitelná na základě dříve publikovaných výsledků. V aplikacích reálného světa máme určitou představu o míře chyb, která je nezbytná k tomu, aby aplikace byla bezpečná, nákladově efektivní nebo přitažlivá pro spotřebitele [5].

Zde budou popsány pouze standardní, obecně uznávané metriky pro klasifikaci a strojové vidění.

Rovněž je požadováno, vedle vhodných metrik, aby byl použit adekvátní návrh vyhodnocení experimentu. Cílem je zajistit, aby byl vypočten co nejlepší odhad toho, jak bude model fungovat, když bude skutečně nasazen v realitě. V podkapitole budou popsány také časté návrhy experimentů [3].

4.1.8 Návrhy experimentů

Tato kapitola je zpracována na základě publikace od J. D. Kellehera [3] Prvním zmíněným návrhem je jednoduchá metoda zádrže (hold-out). Jedná se o rozdělení datové sady na učící, validační a testovací sadu. Na validační sadě je při každé iteraci kontrolován výkon modelu a podle toho upravovány hyperparametry, a je vidět průběh učení ve zlepšování či zhoršování úspěšnosti modelu. Lze tak získat více informací pro jeho ladění. Výhodou také je, že je možné získat poslední iteraci (pokud je model ukládán průběžně), ve které se nevyskytuje přeučení. Úspěšnost na validační sadě totiž začne s přeučením modelu klesat. Testovací sada slouží jen k finálnímu testování plně naučeného modelu a musí to být data, která model nikdy neviděl, jde o simulaci reálných dat. Běžné rozdělení sad je 50: 20: 30 nebo 40: 20: 40, nicméně toto nejsou fixní doporučení. Tato metoda má dva problémy.

1. Je zapotřebí velké množství dat, aby samotná učící datová sada byla stále dost veliká. Malá datová sada může snížit úspěšnost.
2. Měření výkonnosti modelu může být zavádějící, pokud dojde k tomu, že složitá data (např. obrázek, který je těžko rozlišitelný i pro člověka; data jsou velmi podobná dalším kategoriím) jsou v trénovací sadě a jednoduchá (např. data jasně rozlišitelná) v testovací. Model vykazuje poté lepší výsledky, než ve skutečnosti podává.

Řešením těchto dvou problémů je křížová validace (cross-validation). Datové sady jsou rozděleny do N stejně velikých podmnožin, které se střídavě využívají k trénování, testování a validaci.

Každé nastavení se měří a poté se vybere nejlepší nastavení modelu, nebo se výsledky zprůměrují a dostane se celkové hodnocení modelu. Jakýkoliv počet podmnožin je možný, nejpoužívanější je však 10. Extrémním případem je tzv. jackknifing, kdy podmnožin je tolik, kolik je prvků v datové sadě.

Jeden datový záznam je ponechán jako testovací, zbytek slouží jako učící. Pro vyhodnocení modelu se výsledky průměrují. Tento postup je vhodné pro velmi malé datové sady.

Poslední zmíněnou metodou je v originále tzv. bootstrap. Je preferována, pokud není datová sada dost velká pro použití křížové validace (přibližně méně než 300 záznamů). Datová sada se rozdělí do několika částí, ale na rozdíl od předchozích metod se testovací data vyberou náhodně. Celý proces má opět několik iterací.

4.1.9 Klasifikační metriky

Nejprve bude zavedena matice změn, pomocí které budou vysvětleny další metriky. Matice ve sloupcích obsahuje skutečnou hodnotu znaku a v řádcích hodnotu odhadovanou klasifikačním modelem [11].

		Skutečná hodnota	
		Pozitivní	Negativní
Předpověď	Pozitivní	TP	FP
	Negativní	FN	TN

Tabulka 1 Matice záměn pro binární klasifikaci

Zdroj: Vlastní zpracování

Hodnoty tabulky jsou definovány pro binární klasifikaci jako:

- TP (True positive) – správně identifikované pozitivní příklady.
- TN (True negative) – správně identifikované negativní příklady.
- FP (False positive) – nesprávně identifikované pozitivní příklady.
- FN (False negative) – nesprávně identifikované negativní příklady.

U následujících metrik platí, že čím vyšší hodnota z intervalu ⟨0; 1⟩, tím lepší [12], [13]. První metrikou je ***přesnost, správnost (accuracy),*** která určuje poměr správně identifikovaných příkladů oproti celkovému počtu příkladů. Spočítá se jako:

$$P\text{řesnost} = \frac{TP + TN}{TP + TN + FP + FN} \qquad 3$$

Druhou metrikou je ***precisnost (precision),*** která určuje podíl správně klasifikovaných pozitivních příkladů. Spočte se:

$$Precisnost = \frac{TP}{TP + FP} \quad 4$$

Třetí metrikou je ***citlivost, senzitivita, (recall, sensitivity)***, která určuje, kolik hodnot příkladů model označil za pozitivní. Spočte se jako:

$$Senzitivita = \frac{TP}{TP + FN} \quad 5$$

Čtvrtou metrikou je ***specificita, (specificity)***, která určuje, kolik hodnot příkladů model označil za správně negativní ku všem pozitivním případům. Uvádí schopnost identifikace správně negativních případů. Spočte se jako:

$$Specificita = \frac{TN}{TP + FP} \quad 6$$

Často je vhodné vyjádřit výkon modelu pouze jedním číslem. K tomu slouží ***F-míra, F-skóre, (F-measure, F-score)***, která je harmonickým průměrem precisnosti a senzitivity. Nastavení $\beta^2 < 1$ zvýší důležitost přesnosti a naopak. Často se počítá s $\beta = 1$, aby se precisnosti i senzitivitě přikládal stejný význam. Odvození pro $\beta = 1$:

$$F1 = (1 + \beta^2) * \frac{Precisnost * Senzitivita}{\beta^2 * Precisnost + Senzitivita} = \\ = \frac{2 * TP}{2 * TP + FN + FP} \quad 7$$

Je dobré zmínit, že vždy existuje kompromis mezi precisností a citlivostí. Pokud je požadována příliš vysoká přesnost, dochází k poklesu míry senzitivity a naopak [14].

Standardním ohodnocením výkonu binárního klasifikátoru je **operační křivka *(ROC curve)***, případně ***AUC-ROC*** (Area Under Curve, ROC Index) – ta se spočte jako plocha pod křivkou. Graf má na *ose x* hodnoty specificity (často označováno jako False Positive Rate, FPR) a na *ose y* hodnoty sensitivity (True Positive Rate, TPR).

Lze také sestavit ***PR křivku (PR curve)***, s precizností na *ose y* a senzitivitou na *ose x*. Používá se jako jiný pohled na výkon modelu. PR křivka na rozdíl od ROC lépe vystihuje výkon modelu, pokud jsou třídy v datové sadě nerovnoměrně zastoupené a dokáže odhalit rozdíly výkonu algoritmů, které ROC křivka nedokáže [13].

Předchozí metriky lze počítat pouze pro binární klasifikaci v podmínkách dichotomie (když se kategorie nepřekrývají). V ostatních situacích se používají následující metody. Jsou používány především pro výpočet F-skóre [15], [16].

- ***Makro-průměr (macro average)*** spočívá v tom, že všechny třídy mají stejnou váhu. Nejprve jsou spočteny metriky pro každou třídu a poté zprůměrovány mezi třídami.
- ***Mikro-průměr (micro average)*** spočívá naopak v tom, že všechny prvky v datové sadě mají stejnou váhu. Společně se spočítají výsledky napříč všemi třídami, jako by to byla jedna, a poté se spočítají požadované metriky. Nejpočetnější třídy mají největší vliv na výsledek.
- ***Vážený průměr (weighted average)*** spočte výsledky uvnitř třídy podobně jako makro-průměr a následně jsou hodnoty vynásobeny počtem prvků ve třídě a poté zprůměrovány celkovým počtem prvků.

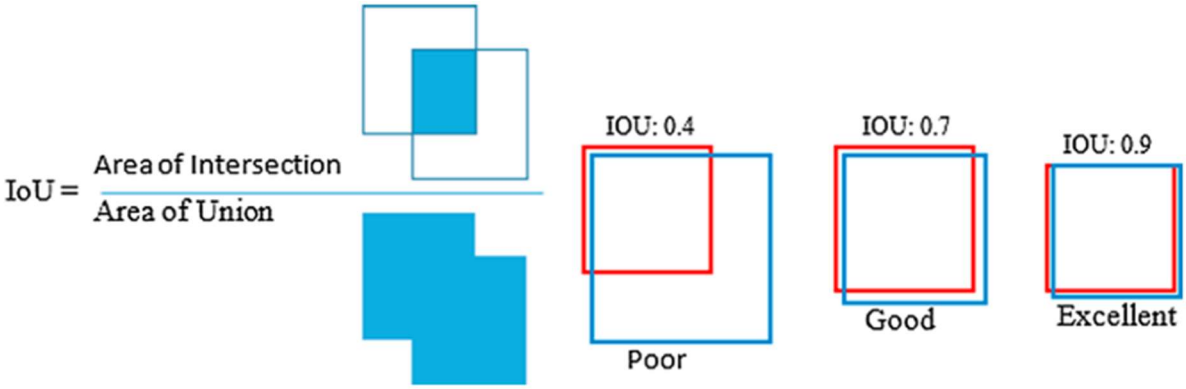
Pokud mají třídy zhruba stejně příkladů, mikro i makro průměr budou velmi podobné. Jestliže mikro je o hodně nižší než makro, tak mají početnější třídy nevyhovující hodnotu pozorované metriky (a naopak). V případě, že jsou více vážené větší třídy, je vhodné používat mikro průměr a naopak. [17]

Poslední zmíněnou metrikou je ***pokrytí (coverage)***. Pokrytí je podíl příkladů, na které je systém strojového učení schopen reagovat. V některých aplikacích je možné, že algoritmus strojového učení nemůže učinit rozhodnutí. Je užitečné, když algoritmus dokáže odhadnout jistotu rozhodnutí, zejména pokud by nesprávné rozhodnutí mělo negativní důsledky a tedy přenechá rozhodování na člověku [18].

4.1.10 Metriky strojového vidění

Metriky zmíněné v předchozí kapitole jsou univerzální pro celou doménu strojového učení. Pro některé specializace je však vhodné odvodit a definovat příbuzné metriky, které by lépe odrážely potřeby v dané doméně. Takovým případem je i strojové vidění, kterému je dále věnována tato práce.

Používanou metrikou při detekci objektů je ve strojovém vidění především Jacardův index, známý také jako ***průnik sjednocení*** (dále jen „*IoU*“). Je to poměr oblasti překrytí a oblasti spojení mezi detekovaným a skutečným ohraničením objektu, jak lze vidět na Obrázek 5.

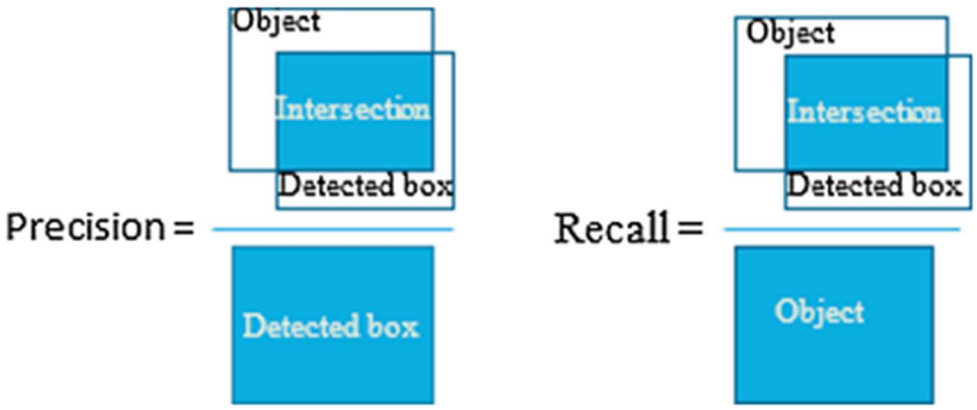


Obrázek 5 Definice IoU v detekci objektů za použití anotace ohraničení

Zdroj: [19], *strana 227, obr. 11*

IoU může nabývat hodnot z intervalu $\langle 0; 1 \rangle$, kde 0 znamená žádné a 1 úplné překrytí. Obecný práh je nastaven na 0,5, kde hodnoty vyšší znamenají úspěšné detekování. V některých případech je práh upraven, aby odpovídal aplikaci – detekce malých plodů ovoce apod.

Při detekci objektu je jiná interpretace preciznosti a senzitivity, než bylo popsáno, jak je vidět na Obrázek 6 níže. Preciznost je podíl detekovaných boxů, které odpovídaly skutečnosti, a detekovaného boxu. Senzitivita je podíl detekovaných boxů, které se shodovaly se skutečností, a celkového objektu. [19]



Obrázek 6 Interpretace preciznosti a senzitivity při detekci objektů za použití anotace ohraničení

Zdroj: [19], *strana 227, obr. 12*

4.2 Rozpoznávání obrazu

Rozpoznávání obrazu je klasifikační problém, řešený jak klasickými metodami strojového učení, tak reprezentativním učením. Před vítězstvím modelu AlexNet [20] v soutěži ImageNet v roce 2012 bylo standardem ruční vytváření charakteristik. Metody lze dělit do několika skupin dle úlohy. Používá se detekce objektů, klasifikace, ohraničení objektu, sémantická segmentace, odhad hloubky a další. Reálných aplikací je nepřeberné množství, např. autonomní vozidla, OCR, autonomní zemědělství atp. [14].

4.2.1 Výzvy

Rozpoznávání a klasifikace ovoce z obrazu je podskupinou detekce objektů. Ovoce má zásadní charakteristické vlastnosti, od kterých se odvíjí aplikace identifikace a detekce. Klíčové úkoly spojené s klasifikací ovoce se podle [21] dělí na:

Vhodný senzor –Výběr senzoru pro získávání dat je klíčem pro klasifikaci. Ke klasifikaci ovoce byly použity senzory od černobílých kamer po nevizuální senzory, jako jsou akustické a hmatové senzory. Ne všechny senzory jsou stejně vhodné pro všechny aplikace. Akustické i hmatové senzory jsou méně vhodné pro nedestruktivní klasifikaci a rozpoznávání. Vizuální senzory jsou navíc velmi citlivé na mnoho variačních faktorů, tj. stav osvětlení a prostředí v pozadí. Tyto základní faktory jsou kombinací mnoha složitých faktorů, včetně odrazu, lomu, velikosti a rotace, které je třeba hlouběji zvážit. V dnešní době se používají primárně RGB kamery či fotoaparáty s vysokým rozlišením [22].

Volba příznaků – Příznaky jsou vlastnosti objektu, které jej mohou odlišit od ostatních objektů. Ovoce a zelenina mají mnoho fyzikálních charakteristik, tj. barvu, strukturu, tvar a velikost, které lze použít jako prvky pro účinnou klasifikaci. Ovoce má četné variace a podobnosti uvnitř i vně tříd. Variace mezi třídami jsou hlavní změny, tj. změny barvy, textury a tvaru, zatímco variace uvnitř třídy jsou obecně mnohem jemnější a těžko rozlišitelné, např.: u jablek jsou pouze malé změny ve vlastnostech. Správný výběr charakteristik umožní systému vypořádat se s klasifikací uvnitř i mezi třídami lépe.

Přístup strojového vidění – Soubor algoritmů strojového učení používaných pro klasifikaci a rozpoznávání obrázků. Rozsáhlý výzkum byl prováděn od počátku 80. let, a navržené algoritmy mohou být kategorizovány mnoha způsoby. Obvykle jsou

to algoritmy založené na neuronové síti a na ručně vytvářených příznacích. Výběr vhodného algoritmu v jakékoli aplikaci strojového učení je vždy kritickým úkolem.

4.2.2 Konvoluční neuronové sítě

Konvoluční neuronové sítě (Convolutional Neural Networks, CNN) jsou speciálním druhem neuronových sítí pro zpracování dvoudimenzionálních dat, vhodné pro vizuální data [5]. První CNN vytvořil Y. Lecun [23] v roce 1989 a komerčně se používaly již v devadesátých letech [24]. Velkou popularizaci zažily v roce 2012, kdy A. Krizhevsky [20] vyhrál soutěž ImageNet v rozpoznávání obrázků, a to s velkou převahou nad ostatními algoritmy strojového učení. Je zřejmé, že konvoluční sítě byly výpočetně efektivnější než plně propojené sítě, takže bylo snazší provádět s nimi více experimentů a vyladit jejich implementaci a hyperparametry. Větší sítě se také zdají být snadněji trénovatelné, navíc s dnešní dostupností výkonných grafických karet a velkým množstvím datových sad.

Vstupní obraz je převeden na vektor pixelů, který je vstupem do modelu. Velikost první vrstvy a vstupního vektoru musí být shodná. Proto všechny vstupní data procházejí předzpracováním. CNN se skládají z konvoluční vrstvy, podvzorkovací vrstvy, případně normalizační a plně propojené vrstvy [8].

Základní operací je konvoluce. Je to matematická operace dvou funkcí reálných argumentů. Značí se symbolem $*$. Rovnice je následující:

$$s(t) = (x * \mathrm{w})(\mathrm{t})\,, \qquad 8$$

kde w je platná funkce hustoty pravděpodobnosti (jinak výstup nebude váženým průměrem) nazývaná jako konvoluční jádro, a x je vstup. Výstup je někdy označován jako mapa příznaků (charakteristik) či konvoluční filtr [5].

V aplikacích strojového učení je vstupem převážně multidimenzionální pole, stejně tak je jádro, které je učeno. Tato vícerozměrná pole se nazývají tenzory. Oba tenzory je potřeba uložit za předpokladu, že obsahují konečný počet nenulových prvků. V praxi to znamená, že vstupní obrázek lze převést na dvourozměrné pole I, výsledek bude dvourozměrné pole K a podle vzorce níže lze spočítat diskrétní konvoluci dvourozměrných polí.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \quad (9)$$

Vzorec v této podobě je snazší implementovat v knihovně strojového učení [5].

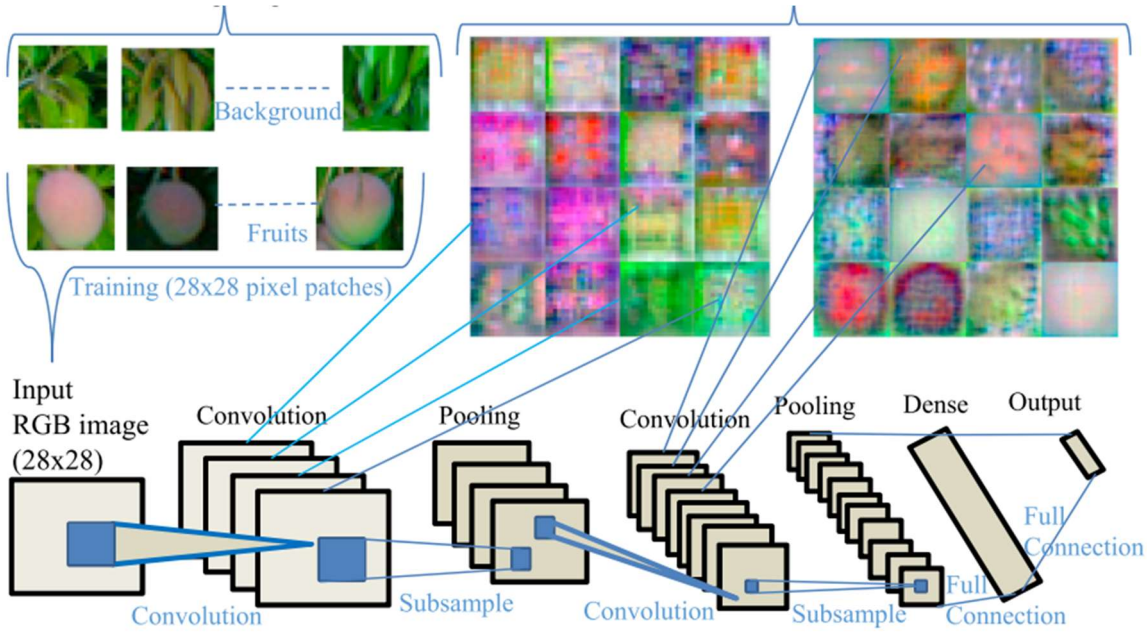
Základem CNN je konvoluční vrstva. Na rozdíl od běžné neuronové sítě mají vrstvy CNN neurony uspořádané do 3 dimenzí: šířka, výška, hloubka. Hloubka zde odkazuje na třetí dimenzi vrstvy (filtry), nikoli na hloubku celé neuronové sítě. Neurony ve vrstvě jsou spojeny pouze s malou oblastí vrstvy před ní, místo toho, aby byly plně propojeny se všemi neurony. Všechny filtry v jedné vrstvě tvoří stejný počet neuronů. Všechny neurony ve stejném filtru mají stejnou množinu vah. Vychází se z úvahy, že neurony ve filtru slouží k detekci stejné charakteristiky. Každý neuron provádí skalární součin mezi jeho vstupem a množinou vah roviny, ve které se nachází. Výstupem filtru je mapa příznaků. Výstupy všech filtrů tvoří výstup konvoluční vrstvy. Běžnou praxí je aktivační funkce ReLU [5], [8], [25].

Druhou velmi používanou vrstvou je podvzorkovací (také sdružovací). Je běžné periodicky vkládat podvzorkovací vrstvu (pooling layer) mezi konvoluční vrstvy. Její funkcí je postupné zmenšení prostorové velikosti vrstev, aby se snížilo množství parametrů a urychlil výpočet modelu. Tato vrstva nemá učící parametry, určuje se pouze velikost zmenšení. Prostorově mění velikost tenzoru pomocí operace MAX nebo AVG [8]. Ve všech případech, podvzorkovací vrstva pomáhá učinit model nezávislým na malých variacích vstupu. Invariance k místní translaci může být užitečnou vlastností, pokud nám záleží na tom, zda jsou některé funkce přítomny, a ne na tom, kde přesně jsou. To je třeba zvážit při návrhu modelu podle zadání [5]. Podle J. T. Springenberga [26] je možné tuto vrstvu vynechat a nahradit ji další konvoluční vrstvou s větším jádrem pro zachování redukce parametrů. Vynechání této vrstvy je důležité pro vývoj generačních modelů jako jsou GAN nebo VAE [26].

CNN také používají plně propojené vrstvy, jako ostatní neuronové sítě. Neurony v plně propojené vrstvě jsou spojeny se všemi aktivacemi v předchozí vrstvě, jak je vidět v běžných neuronových sítích. Jejich aktivace lze tedy počítat s násobením matice s přičtenou bází [25].

4.2.3 Architektury

Na Obrázek 7 je znázorněna celá architektura CNN. Vstupem je obrázek, probíhá střídavě konvoluce a podvzorkování až k plně propojeným vrstvám a výstupní vrstvě.



Obrázek 7 Architektura CNN

Zdroj: [19], *strana 222, obr. 3*

Dále budou popsány používané architektury.

AlexNet – byl první model, který zpopularizoval CNN ve strojovém vidění. Byl odeslán na výzvu ImageNet ILSVRC a významně překonal druhé místo v žebříčku (top 5 chyb ve výši 16 % ve srovnání s vítězem s chybou 26 %). Síť představovala konvoluční vrstvy naskládané na sebe (dříve bylo běžné mít pouze jednu konvoluční vrstvu vždy bezprostředně následovanou podvzorkovací vrstvou) [8], [20].

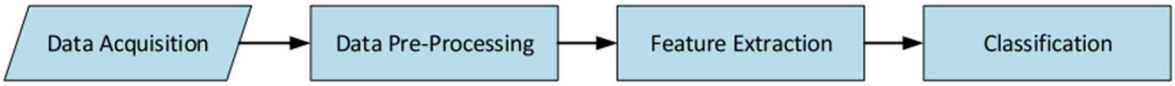
GoogleNet – od společnosti Google byl vítězem ILSVRC 2014. Jeho hlavním přínosem byl vývoj Inception architektury, která výrazně snížila počet parametrů v síti (4M, v porovnání s AlexNet s 60M). Také používá AVG podvzorkovací vrstvu namísto plně propojených vrstev na konci sítě, čímž eliminuje velké množství parametrů [8], [27].

VGGNet – Jejím hlavním přínosem bylo prokázání, že hloubka sítě je rozhodující pro dobrý výkon. Konečná nejlepší síť obsahuje 16 konvolučních vrstev a vyznačuje se extrémně homogenní architekturou, která má konvoluční jádra pouze o velikosti 3 a podvzorkování 2x2. Nevýhodou je náročnost modelu na výpočetní výkon a jeho velikost (140M) [8], [28].

ResNet – Reziduální neuronová síť se vyznačuje speciálními přeskočenými spoji a intenzivním využíváním dávkové normalizace. Zavádí speciální reziduální bloky. Ty výstup z jednoho neuronu přenáší nejen do následující vrstvy, ale i do 2 až 3 dalších vrstev pro zamezení problému s mizejícím gradientem u hlubokých sítí. Architektuře také chybí plně připojené vrstvy na konci sítě [8], [29].

4.2.4 Obecný postup

Klasifikace ovoce a zeleniny je relativně složitým problémem vzhledem k velké rozmanitosti (tvaru, barvě a struktuře uvnitř třídy) a podobnosti mezi třídami. Tato omezení způsobila nedostatek automatizovaných klasifikačních systémů pro třídění ovoce. Automatizovaný systém pro takovou klasifikaci s komplexnějšími informacemi o ovoci může být užitečný hned v několika úlohách. Aplikace jsou popsány v kapitole 4.3. Obecně lze rozpoznávání ovoce i obecné úlohy rozpoznávání obrazu, rozdělit do 4 fází (Obrázek 8). Níže bude každá popsána jednotlivě [21].



Obrázek 8 Schéma rozpoznávání obrazu

Zdroj: [21], *strana 26, obr. 1*

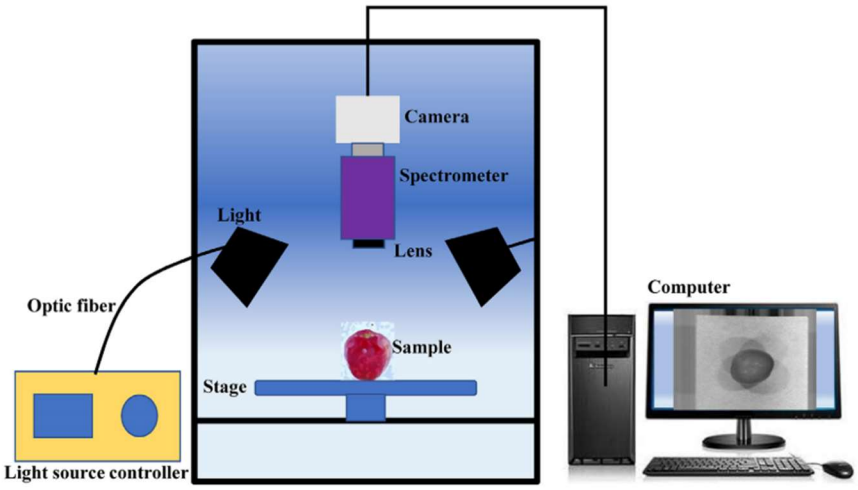
4.2.5 Sběr dat

Při návrhu modelu i charakteristik je často třeba oddělit nesledované variační faktory (factors of variation) od pozorovaných dat. Variačními faktory se rozumí například tvar objektu podle pozorovacího úhlu při pořizování fotografie, světelné podmínky – červená barva může v noci vypadat jako černá, pozice objektu na fotografii atp. Je to jeden z důvodů provádění přípravy dat (data preparation). Prováděné metody závisí na zadání řešeného problému a datové sadě [5].

Je důležité si uvědomit, že složitost modelu je úzce spjata s variací vstupů obsažených v datové sadě. Čím větší množství dat, tím složitější model může být vytvořen bez přetrénování, díky jejich rozmanitosti. Duplikování stejných dat nebo shromažďování velmi podobných dat nepomůže. Vytvoření složitého modelu s velkým objemem dat může vyřešit zásadní problémy v některých úlohách. Ve skutečném světě je často možnost rozhodnout, kolik dat se bude shromažďovat, a je potřeba zvážit, zda je výhodnější sběr dat než ladění a optimalizace modelu [7]. Pokud je datová sada příliš malá, model

se pravděpodobně nenaučí všechny vztahy mezi příznaky a výsledky a bude mít nízkou přesnost. V tu chvíli je vhodné zvážit použití již předučeného klasifikačního modelu (transfer learning) a znovu naučit pár jeho posledních vrstev. Model už rozpozná hrany, linky, tvary a pouze se přeučením specializuje v dané kategorii [30]. Je dobré tvořit datovou sadu rovnoměrnou, aby nevznikala různá zkosení a nerovnováhy, výsledky mohou být poté zkreslené a je s tím třeba počítat ve vyhodnocování modelu a použít správné metriky [3].

Již při přípravě datové sady je možné redukovat variační faktory. Na následujícím obrázku je znázorněna tvorba datové sady pomocí spektrometru v laboratorních podmínkách. Při implementaci tohoto přístupu je třeba brát v úvahu generalizaci modelu z pořízených dat – data z laboratoře nemusí poskytovat tak dobré výsledky na datech z reálného prostředí, záleží na optimalizaci modelu a typu úlohy.



Obrázek 9 Hyperspektrální systém pro tvorbu obrázků

Zdroj: [31], *strana 2 obr. 1*

Obrázek jablka je pořizován za konstantního osvětlení, objekt je centrován a focen ze stejné pozice a vzdálenosti [31]. Dalším příkladem je H. Mureşan a M. Oltean [32], kde fotografie ovoce byly pořizovány nahráváním videa, zatímco motorek otáčel ovocem dokola. Poté byly extrahovány jednotlivé snímky pořízené z různých pohledových stran. Při těchto experimentálních podmínkách nebyly splněny požadavky na jednotnost všech variačních faktorů, a proto data musela být dále zpracovávána.

4.2.6 Předzpracování dat

Data pořízená vizuálními senzory obsahují určitou úroveň šumu a zkreslení. Prvotní obrázky jsou většinou nevhodné pro extrakci příznaků pro rozpoznávání obrazu. Proto je role předzpracování dat významná. Matice RGB zachycují zbytečné, nezpracované informace, které musí být statisticky vyhodnoceny, aby byly odstraněny nechtěné informace a určeny chybějící informace kvůli šumu, zkreslení a proměnné citlivosti senzoru na stejný fyzický vstup z prostředí. Prvotní obrazy jsou zpracovány na holistické nebo elementární úrovni, přičemž pixel je považován za nejnižší úroveň abstrakce [19].

Jako první se většinou mění velikost obrazu, aby odpovídala vstupním parametrům modelu. Bylo zjištěno, že použité barevné schéma má velký vliv na výslednou přesnost. Proto se často zmenšený obraz převádí na různá jiná barevná schémata [21].

Segmentace je extrahování oblasti zájmu (Region of Interest, ROI). Jedná se o kritický krok ve strojovém vidění, který ovlivňuje celkovou efektivitu analýzy obrazu. V praxi je používáno, a v literatuře popsáno, obrovské množství segmentačních technik, založených na jasu, barvě, stupnici šedi, textuře a hranách. S postupným vývojem výpočetního výkonu neustále přibývají nové, efektivnější techniky [21].

Jednou z používaných technik jsou superpixely. Bylo ověřeno, že přesnějších výsledků lze dosáhnout pomocí větších oblastí za cenu délky výpočtu [33]. Další technikou je matice sousednosti intenzit v barevném prostoru HSI (Co-occurrence Matrix, CCM). Analyzuje se struktura povrchu a jeho anomálie [21]. Dále segmentace pomocí techniky v originále zvané jako watershed, založená na transformaci vzdálenosti (Distance Transformation, DT). Euklidovská vzdálenost má v binárních obrazech významnou účinnost [21]. Naopak watershed založená na zaplavovací metodě byla použita v práci H. Mureşana [32] pro odstranění pozadí se značným úspěchem. Hodnoty parametru vzdálenosti sousedů byly nastavovány stylem pokus-omyl. Kombinace vlastností místních binárních vzorů (Local Binary Patterns, LBP), histogramu orientovaných gradientů (Histogram of Oriented Gradient, HOG), globálních barev a tvarů byla použita v Otsu prahování, pro optimální výběr ROI a bylo zjištěno, že je velmi efektivní [21]. Použito bylo také shlukování metodou nejbližších středů pro zvýšení kontrastu barevných obrázků [21]. Pro přehled metod je níže uvedena Tabulka 2.

Rok	Aplikace	Segmentační technika
2006	Hodnocení kvality	Superpixely s variační velikostí sousedství
2006	Hodnocení kvality	HSI CCM
2012	Detekce	DT a watershed
2016	Klasifikace	Záplavový watershed
2018	Detekce	LBP, HOD, globální barvy a tvary s Otsu
2018	Detekce nemocí	Shlukování metodou nejbližších středů

Tabulka 2 Souhrn segmentačních technik

Zdroj: Upravená tabulka z [21]*, strana 29, tabulka 4*

Rozšíření trénovací množiny

Malá datová sada může vést ke špatným výsledkům modelu. Jedním z řešení je vytvořit falešná data z dostupných dat a přidat je do učící sady. Model poté díky větším variacím dat lépe generalizuje a není tolik náchylný k přetrénování [34].

Často prováděnými metodami jsou:

- Rotace – obraz je natáčen dle úhlu a osy. Může vzniknout situace, kdy část obrazu je třeba doplnit. V závislosti na typu vstupních dat se pak doplní chybějící místo barvou pozadí nebo barvou nejbližšího pixelu, či se obraz ořízne.
- Zrcadlení – rozměry obrazu zůstávají stejné, jedná se o otočení podle $osy\ x$ nebo $osy\ y$. Jednoduchá, ale účinná metoda.
- Škálování – mění se měřítko obrazu při zachování stejné velikosti a rozměrů pomocí interpolace. Podle směru škálování se obraz ořízne, nebo doplní.
- Posun – důležitá metoda pro obrázky s objekty v jedné oblasti, posunutím se předejde naučení vzoru v jednom místě.
- Barevná editace – při správném nastavení může simulovat měnící se světelné podmínky. Důležité pro rozpoznávání obrazu v reálném světě. Posouvají se hodnoty jednotlivých kanálů v obraze (sytost, odstín, barvy, ...).
- Šum – poslední zmíněnou metodou je přidání šumu do obrazu. To má za následek lepší generalizaci modelu, protože si model musí umět poradit i s drobnými variacemi v obraze. Používanými algoritmy jsou například pepř a sůl nebo Gaussův bílý šum. [19], [24]

Často se zmíněné metody kombinují a data se nechávají generovat automaticky s předem stanovenými parametry (někdy i náhodnými variacemi) pro jednotlivé metody přímo při učení. Upravována jsou pouze trénovací, nikoli testovací data [25].

4.2.7 Extrakce příznaků

Extrakci příznaků a charakteristik z obrazu lze dělit na klasický přístup (ručně vytvářené) a reprezentativní učení, které si příznaky vytváří samostatně [5]. V rozpoznávání ovoce mají oba tyto přístupy své zastoupení. Podle autorů K. Hameed a L. Zhou [21], [22] mají modely reprezentativního učení lepší výsledky než modely klasické. Dříve byly tradiční přístupy ke klasifikaci obrazu založeny na charakteristikách vytvořených ručně, jejichž kvalita výrazně ovlivnila celkové výsledky. Je to komplexní, časově náročný proces a úpravy jsou třeba vždy, když se objeví problém nebo se změní datová sada. Tento přístup představuje vynaložení značného úsilí, které závisí na znalostech odborníků a negeneralizuje dobře [12]. V dnešní době tenhle problém řeší především hluboké učení (Deep Learning, DL), kde je jednou z nejdůležitějších výhod ve zpracování obrazu samostatná lokalizace charakteristik při učení. Nevýhodou DL je však delší doba tréninku a potřeba velkého objemu dat [6]. Nejprve je popsán klasický přístup k extrakci příznaků a poté extrakce formou reprezentativního učení.

Klasický přístup k extrakci příznaků – Ovoce má několik výrazných vizuálních charakteristik. Nejčastěji používanými příznaky pro rozpoznávání ovoce a zeleniny jsou barva, tvar a textura. Deskriptor příznaku je buď globální, nebo lokální v závislosti na komplexní nebo částečné reprezentaci. Zejména u rozpoznávání objektů globální příznak popisuje objekt jako celek ve zobecněné formě, například tvaru. Lokální příznak popisuje mnoho zajímavých kousků obrazu, například defekty textury. Lokální příznaky nejsou konzistentní a mohou se lišit mezi daty. K obvyklým postupům patří kombinace místních a globálních deskriptorů pro vyšší účinnost klasifikace. Mezi inherentní omezení při získávání vizuálních dat patří především šum, částečné informace (v důsledku úhlu, světelných podmínek atp.) a ztráta dat během převodu např. RGB na šedou stupnici [21].

Barva má z hlediska strojového zpracování významné výhody oproti jiným charakteristikám. Výhodou je snadnost extrakce, nezávislost na velikosti, tvaru a orientaci. Barvy jsou znázorněny v různých barevných prostorech, které jsou navrženy pro konkrétní účel. Obvyklý prostor je RGB. Obraz generovaný stejnými pixely v prostoru RGB může mít různé hodnoty RGB pro různá zařízení – je třeba standardizace. Barva

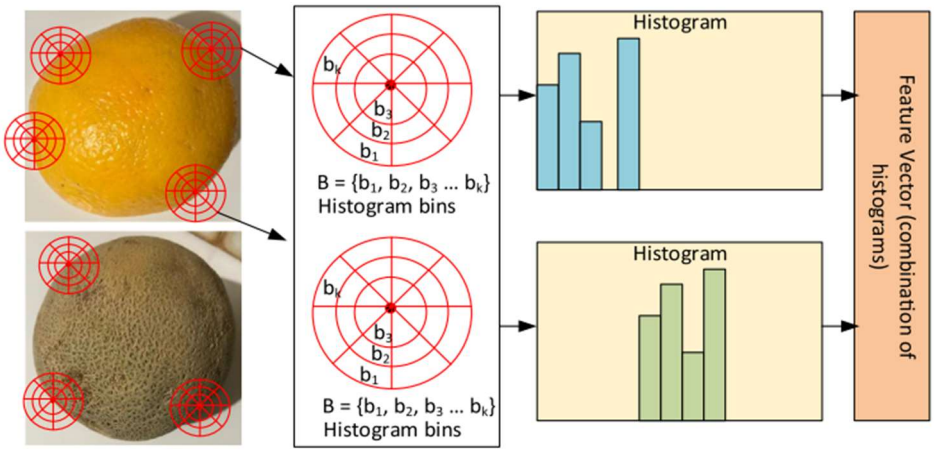
ovoce se řídí fyzickými, biochemickými a mikrobiálními změnami během zrání a růstu. Avšak fotometrické změny, tj. orientace, měřítko a osvětlení, mohou mít významný účinek na barvu ovoce, jak je znázorněno na Obrázek 10. Pro snížení těchto účinků musí mít deskriptor barvy významnou invariační vlastnost. Používanými metodami jsou histogramy, invariantní momenty, (Scale Invariant Feature Transforms, SIFT) a koherentní vektory. Histogramy celého barevného prostoru jsou skládány z histogramů jednotlivých kanálů. SIFT je invariantní k fotometrickým změnám, protože gradienty obrazu jsou též invariantní k těmto změnám. Variantami SIFT jsou HSV SIFT, HUE SIFT, oponent SIFT, CSIFT, rgSIFT a RGB-SIFT pro použití v různých barevných prostorech. Vektor soudržnosti barev (Color Coherence Vector, CCV) popisuje holistické rozložení barev s prostorovou relevancí okolních pixelů rozdělením obrazu na propojené části. Byly také analyzovány různé barevné modely pro rozpoznání ovoce během dne a noci, kdy byly pro reprezentaci použity statistické rysy barevného prostoru YIQ, RGB, HSV a YUV s pozitivními výsledky [21].

Obrázek 10 Různorodost textury pomeranče (a) a chleba (b) při změně osvětlení a úhlu pohledu

Zdroj: [21], *strana 33, obr. 3*

Tvar – Jednou z nejintuitivnějších kategorizací deskriptorů tvaru je obrys a oblast založená na geometrii tvarů. Elementárnější formou kategorizace může být prostorová a transformační doména, kde použití konkrétního druhu deskriptoru závisí na aplikaci. Základní geometrické parametry, jako jsou zakřivení, rohy, regiony, těžiště, konvexnost, poměr kruhovitosti a excentricita spojené s tvarem ovoce, dokážou diferencovat tvary pouze s velkými rozdíly, ale jejich kombinace odliší i jemnější detaily. Novější metodou v popisu tvaru je použití v originále tzv. Bag-of-Curvature (BoC) a Bag-of-Shape-Vocabulary (BoSV) jako varianty Bag-of-Words (BoW) – metoda zaměřující se na hledání určité sady „slov“, v případě vizuálních dat, tedy kontur a hran v testovaném obrazu, a měření jejich výskytu. Metoda v originále zvaná jako Edge Orientation Autocorrelogram (EOAC) se používá pro odhad orientace hran a prostorovou korelaci mezi pixely. Algoritmy, jako je v originále tzv. Circular Hough Transform (CHT) a Random Hough Transform (RHT) se široce využívají k rozpoznání kruhových a oválných tvarů kolem

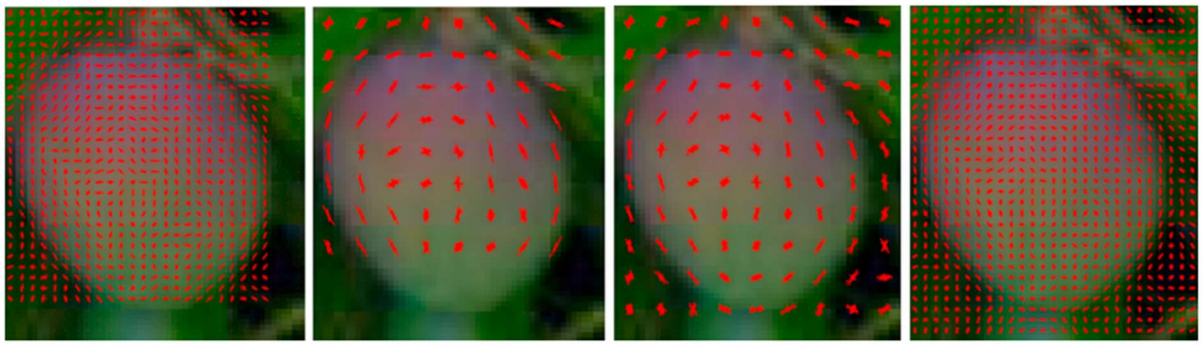
možných oblastí zájmu. Na Obrázek 11 lze vidět histogram detailů okolních pixelů na identifikovaném klíčovém bodě na hranici objektu. Je pořizován v kontextu zjištění tvaru, kde kombinace všech histogramů popisuje tvar objektu [9], [21].



Obrázek 11 Extrakce vektoru charakteristiky tvaru

Zdroj: [21], *strana 30, obr. 2*

Textura je prostorové uspořádání primitivů nazývaných textony, což jsou základní struktury na mikroskopické úrovni – pixely ve strojovém interpretování obrazu a atomy ve vizuálním vnímání člověka. Textura v digitálních obrazech má určitou statistickou vlastnost periodické rekurze s určitou mírou rozptylu. Tato odchylka se může pohybovat od statistických po stochastické funkce [14]. Aby se snížila citlivost deskriptorů textury na měřítku, úhlu pohledu a osvětlení (Obrázek 10), používají se SIFT a LBP. V případě klasifikace ovoce založené na texturách byly hlášeny významné výsledky za využití metod založených na filtru Gabor s jádrem PCA s nízkými výpočetními náklady a prostorovou reprezentací v transformační doméně. Statistická analýza založená na prostorové matici závislosti na stupních šedi (Spatial Gray-level Dependence Matrix, SGDM) bývá použita k nalezení 13 statistických prvků definovaných k popisu textury. V současných metodách reprezentace textury byl použit místní relativní fázový binární vzor (Local Relative Phase Binary Pattern, LRPBP) [21]. Také byla použita matice sousednosti intenzit (také Gray Level Co-occurrence Matrix, GLCM), s kterou byla prezentována významná míra přesnosti [35]. Velmi používanou metodou je HOG s různými parametry (velikost oblasti buňky a bloku a pokrytí obrazu), jak lze vidět na Obrázek 12 [19], [28].



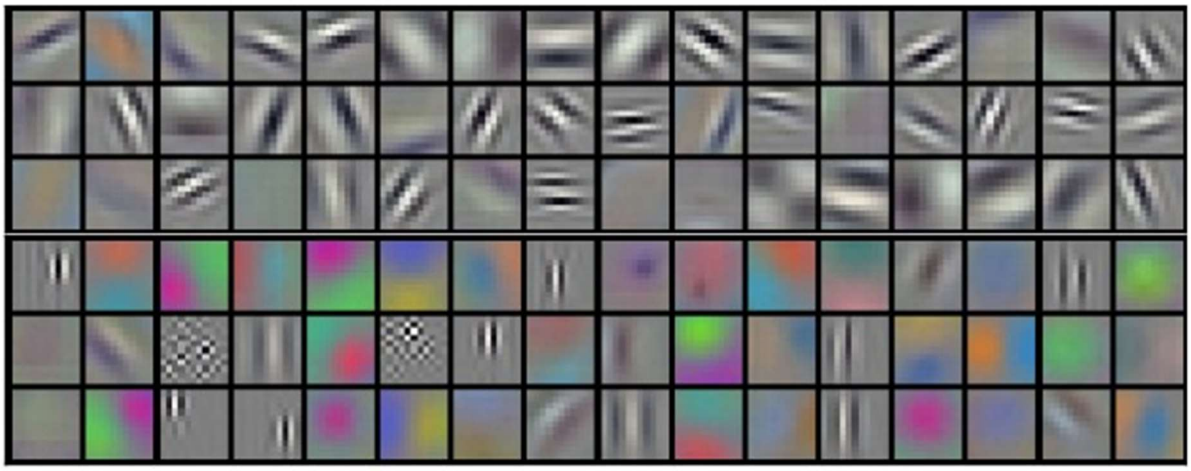
Obrázek 12 HOG charakteristiky s proměnlivými parametry

Zdroj: [19], *strana 221, obr. 1*

Extrakce příznaků pomocí reprezentativního učení – U metod reprezentativního učení, především hlubokého učení, není díky skrytým vrstvám vyžadována ruční extrakce příznaků. Pro zpracování obrazových dat se primárně využívají konvoluční neuronové sítě. Během učení CNN automaticky vytváří příznaky a mohou tak být vyvinuty modely pro detekci mnoha tříd. Například J. Redmon a A. Farhadi [36] uvedli model YOLOv2, založený na CNN rozpoznávající 9 000 běžných kategorií objektů. Množství práce spojené s takovým množstvím tříd za použití tradičních metod segmentace a tvorby příznaků by bylo enormní.

Obrazová data přiváděná do vstupní vrstvy CNN prochází prostředními vrstvami, kde se vytvoří vektor příznaků. CNN filtry (někdy uváděné také jako jádra) v konvolučních vrstvách se učí jednotlivé příznaky, které extrahují užitečné informace pro různé třídy objektů [13]. Každý filtr je prostorově malý (šířka a výška), ale prochází celou hloubkou vstupu. Například typický filtr na první konvoluční vrstvě může mít velikost 5x5x3 pixelů. Během dopředného průchodu posouvá (přesněji konvoluje) každý filtr napříč vstupním obrázkem a vypočítává skalární součiny mezi vstupy filtru a obrázku v libovolné poloze. Posouváním filtru se vytváří dvourozměrná aktivační mapa, která poskytuje odpovědi tohoto filtru v každé prostorové poloze. Intuitivně se síť naučí filtry, které se aktivují, když uvidí nějaký typ vizuálního prvku, jako je hrana nebo skvrna nějaké barvy na první vrstvě nebo nakonec celé vzory na vyšších vrstvách sítě. Jedna konvoluční vrstva má sadu takových filtrů (podle její hloubky) a každý z nich vytvoří samostatnou dvourozměrnou aktivační mapu. Tyto aktivační mapy se naskládají na sebe a vytvoří výstup vrstvy. Filtry

lze vizualizovat pro jejich lepší pochopení. Obrázek 13 znázorňuje filtry první konvoluční vrstvy modelu [20].

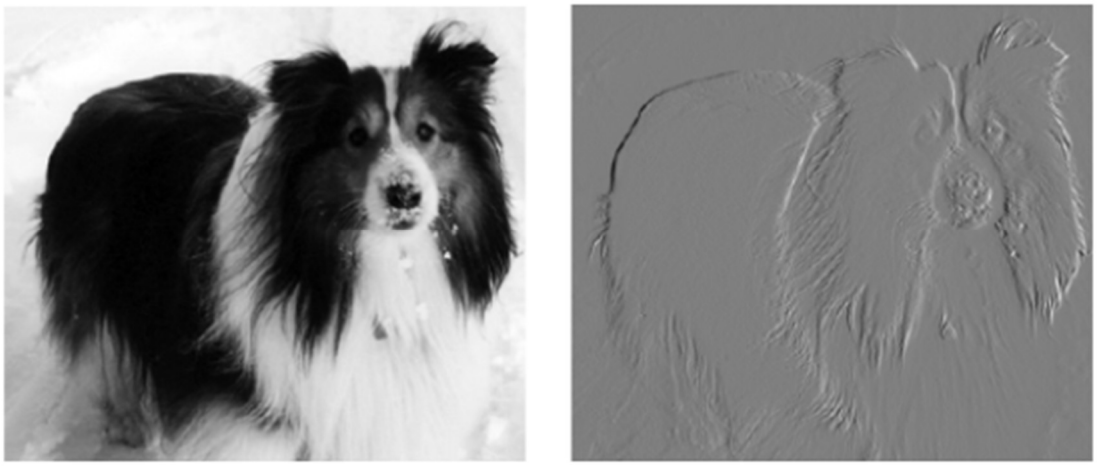


Obrázek 13 Vizualizace první konvoluční vrstvy v CNN trénované na datové sadě ImageNet

Zdroj: [20], *strana 89, obr. 3*

Jak lze vidět, filtry zachycují základní rozložení barev a hran. Komplexnější a sémantické rysy, jako je tvar a vzory, se model učí v hlubších vrstvách z těchto naučených základních linií. Velikost a počet filtru závisí na parametrech vrstvy, lze pro každou vrstvu nastavit jiné [3].

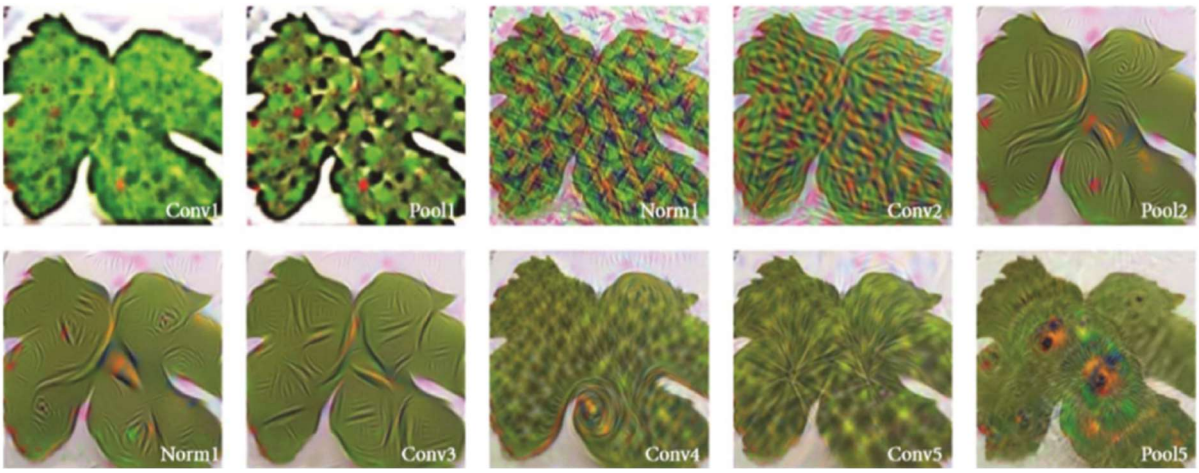
Na Obrázek 14 níže je vidět efektivnost detekce hran. Pravá část obsahuje vertikální hrany obrazu vlevo. Filtry CNN se mohou naučit rozpoznat třídu psa právě díky těmto hranám, které model různě testuje [5].



Obrázek 14 Detekce vertikálních hran

Zdroj: [5], *strana 334, obr. 9.6*

Na Obrázek 15 je vizualizace konvolučních filtrů zaměřující se na texturu v obrázku listu, na kterém se provádí klasifikace nemoci rostliny. Na každé vrstvě se provádí jiná konvoluce, a čím hlubší vrstva, tím jemnější detaily zachytí. Až na poslední (vpravo dole) je vidět zřetelně oblast, kde je zřetelně zachycen a rozpoznán projev nemoci.



Obrázek 15 Vizualizace konvolučních filtrů hlubších vrstev

Zdroj: [12], *strana 3, obr. 2*

4.2.8 Klasifikace

Klasifikace probíhá na základě získaných map příznaků (feature maps) z předešlého kroku, buď v rámci modelu neuronových sítí, nebo klasickými klasifikátory, např. KNN, SVM, rozhodovací stromy a jejich varianty.

Existují tzv. předučené neuronové sítě pro klasifikaci obecných objektů. Mohou se vypořádat s nedostatkem dat u rozpoznávání ovoce, protože využívají podstatnější vlastnosti objektů naučených na jiných datech. Pro klasifikaci pomocí předučených modelů CNN, se používají různé varianty (AlexNet, GoogleNet, VGGM, ResNet) [19], [22].

Ranné CNN užívaly posouvací okno (jádro), kde se prováděla konvoluce postupně po celém obrazu. V každé poloze filtru je generováno velké množství map, přičemž každá mapa je klasifikována jako objekt obsahující nebo neobsahující. Postoupení všech dostupných map do CNN zpomalilo detekci objektů [28]. Metodu posuvného jádra nahradila regionální CNN (Regional CNN, R-CNN), což je relativně rychlejší síť. Používá fixní heuristický algoritmus selektivního vyhledávání [37], který odfiltruje některé regiony a do CNN vloží pouze návrhy 2 000 potenciálních regionů obsahujících objekt. Pro ještě větší zkrácení doby detekce je žádoucí přivést celý obrázek do modelu CNN v jednom průchodu pro vygenerování konečné mapy charakteristik. Tohoto cíle lze dosáhnout

dvoustupňovým přístupem (fáze návrhu regionu, po které následuje fáze klasifikace / detekce). Toho bylo dosaženo v rychlých R-CNN (Fast R-CNN), kdy je do modelu poslán celý obraz pro vygenerování mapy příznaků, z té jsou poté identifikovány ROI (opět selektivním vyhledáváním) a mapy příznaků ROI jsou předány do FC vrstev pro klasifikaci [38]. Rychlejší R-CNN (Faster R-CNN) vynechal selektivní algoritmus a nahradil ho další sítí, která predikuje ROI [39]. Pro dosažení dalšího zlepšení rychlosti byly zavedeny v originále zvané Single Shot Detectors (SSD) a You Only Look Once (YOLO) algoritmy. Odstranily fáze návrhu regionu a CNN byly navrženy tak, že jedna síť identifikovala ROI a zároveň klasifikovala. YOLO modely musí zvážit husté vzorkování možných umístění objektů pro detekci. Obraz se rozdělí do SxS matice a uvnitř každé mřížky se vezme m ohraničujících boxů. Pro každý prvek z ohraničeného boxu síť spočte pravděpodobnost třídy a hodnoty offsetu pro ohraničovací pole. Ohraničovací rámečky, mající pravděpodobnost třídy nad prahovou hodnotou, jsou vybrány a použity k nalezení objektu v obraze [40]. Mají však problémy s detekcí velmi malých objektů, jako jsou například hejna ptáků [19].

4.3 Přehled aplikací

Vývoj výpočetní techniky umožňuje využívat metody strojového učení, zvlášť potom hluboké učení a jejich aplikaci do reálného světa. Tento pokrok má přesahy do ostatních odvětví. Příkladem je medicína, bezpečnost, zemědělství, marketing apod. Především u odvětví s více finančními zdroji je tento pokrok rychlejší [22]. V posledních letech bylo vyzkoušeno velmi mnoho klasických technik a metod klasifikace ovoce, např. podle barvy, tvaru, textury, ale ty nebyly schopny zvládnout velkou datovou sadu. To vyústilo v malou přesnost a dlouhou dobu rozpoznávání. K odstranění těchto nedostatků se začalo s používáním hlubokého učení, konkrétně CNN s jejími variacemi. V roce 2018 přes 50 % state-of-the-art modelů používalo hluboké učení [12].

Strojové učení se ve vztahu k rozpoznávání ovoce používá mnoha způsoby. Modely popisované v následujících kapitolách mnohdy spadají do více než jedné vymezené domény. V našem přehledu budou zařazeny podle primárních cílů autorů.

4.3.1 Identifikace druhu

První aplikační úlohou je identifikace druhu ovoce. Popišme si některé modely, ve kterých je určení druhu prioritou a nerozlišují další atributy.

Typickou aplikací je rozpoznávání lokálního ovoce v Bangladéši, viz [35]. Bangladéš má tropické podnebí, vyskytuje se zde přes 70 druhů tropického a subtropického ovoce, ale pouze 9 druhů je považováno za běžné ovoce a počet obyvatel, kteří nedokážou místní ovoce rozpoznat, stále roste. Autoři mobilních aplikací reagují na problém obyvatel určovat vzácné druhy na fotografiích. Bylo vybráno 6 druhů ovoce a vytvořena datová sada celkem 480 barevných obrázků ovoce různých velikostí, zachycených z různých vzdáleností a úhlů. To má simulovat různorodost fotek od uživatelů aplikace. Fotografie byly pořízeny za denního osvětlení v reálných podmínkách včetně běžného pozadí. Tok dat celé aplikace jde od pořízení fotky, její odeslání na server, zpracování expertním systémem a následné zaslání výsledku zpět. Konkrétními kroky jsou:

- pořízení obrázku,
- změna jeho velikosti na 350x300 pixelů a zvýšení kontrastu,
- konverze RGB barevného schématu na CIELAB pro lepší segmentace,
- segmentace obrázku pomocí shlukování metodou nejbližších středů pro odstranění pozadí,
- extrakce příznaků textur pomocí matice sousednosti intenzit a statistických příznaků v souvislosti s vadami na textuře jako průměr, derivace, rozptyl, špičatost a šikmost,
- klasifikace pomocí SVM,
- sdělení výsledku.

Segmentace je klíčová pro zvýšení úspěšnosti rozpoznání ovoce, protože zvyšuje kvalitu extrakce příznaků pro klasifikaci. Data byla rozdělena do dvou částí metodou zádrže. Dvě třetiny tvoří učící data, zbytek testovací. Učící data byla ještě rozdělena na dvě části (trénovací a validační), aby se zabránilo vzniku problému generalizace a přetrénování. S popsaným expertním systémem autoři dosáhli přesnosti 94,79 %. Na stejných datech testovali i další klasické modely strojového učení jako je rozhodovací strom a Bayes. SVM vyšlo nejlépe v hodnocení ROC křivkou s 90,79 %, druhý v pořadí byl rozhodovací strom s 89,4 % a poslední Bayes s 82,9 %. Další rozvoj aplikace tkví ve zvýšení počtu rozpoznaného ovoce.

Ve srovnání s dalšími klasickými metodami strojového učení v rozpoznávání lokálního ovoce je tento SVM klasifikátor jeden z nejúspěšnějších, a to díky své segmentaci ovoce. Úspěšnosti téměř 100 % dosáhl SVM klasifikátor [41], ale pouze na výhodnějších

experimentálních datech (jednotné pozadí, centrované ovoce a stejná vzdálenost). Navíc tento klasifikátor řešil pouze binární klasifikaci na základě tvaru ovoce. Se zmíněným SVM klasifikátorem byla v rámci téže práce porovnávána dopředná neuronová síť, která dosáhla pouze 60% úspěšnosti. Autoři však nepoužili konvoluční vrstvy, které jsou pro zpracování obrazu klíčové (proto si modely s použitím CNN obecně vedou lépe).

Jedním z dobrých příkladů je 13-ti vrstvá hluboká konvoluční neuronová síť [34] použitá ke klasifikaci. Datová sada byla vytvořena fotografováním a stahováním z internetu, celkem bylo získáno 3 600 obrázků, 200 pro každý druh ovoce. Obrázky byly upraveny (změněna velikost na 256x256, centrováno ovoce, odstraněno pozadí pomocí tzv. Split & Merge algoritmu) a byl manuálně přidán popisek druhu ovoce. Učící a testovací sada byly stejně velké. K zamezení přetrénování byly použity metody pro rozšíření datové sady (rotace obrázku, gamma korekce, přidání šumu, afinní transformace a škálování). K původní učící sadě bylo přidáno 63 000 vytvořených obrázků. Síť se skládala ze 4 konvolučních bloků (konvoluční vrstva, ReLU – usměrňovací lineární jednotka, aktivační vrstva, sdružovací vrstva – Pooling). Za nimi následovaly 2 plně propojené vrstvy zakončené softmax vrstvou. Jako chybová funkce byla zvolena křížová entropie, velikost dávky 128 a rychlost učení 0,01 se snížením o řád každých 10 epoch. Jako optimalizátor SGD byl použit algoritmus Adam (adaptivní odhad momentu). Byly testovány dvě sdružovací vrstvy – průměrová a maximální. S maximální (Max-pool layer) sdružovací vrstvou dosáhl model úspěšnosti 94,94 %. S průměrovou o 0,11 % menší. Obdobně byl testován počet konvolučních bloků v rozmezí 2 až 7, kde 4 z nich dosáhly nejvyšší úspěšnosti vyšší o celé 1 %. Hotový model byl testován na obrázcích s nedokonalým pozadím s úspěšností 89,6 %. Byl ukázán rozdíl na těchto testovacích datech bez rozšíření dat (model učen na původní sadě 1 800 obrázků) s úspěšností 84,9 %. Do budoucna autoři plánují zlepšit generalizaci modelu a zvětšení počtu rozeznaných druhů.

Na popsaný model navazuje práce autorů S.-H. Wang a Y. Chen [42]. Srovnává šestivrstvý model s osmivrstvou sítí, kterou tvoří 5 konvolučních a 3 plně propojených vrstev. Používá opět konvoluční bloky s tím rozdílem, že aktivační funkcí je parametrická usměrňovací lineární jednotka (PReLU). Dalším rozdílem v architektuře sítě je přidání vrstvy zahazování (droupout layer) s koeficientem 0,5 před každou plně propojenou vrstvu a zmenšení konvolučních jader ze 7 a 5 na 3 za účelem snížení počtu parametrů sítě. Byla použita stejná datová sada jako v předchozím modelu i metody rozšíření dat

s jediným rozdílem – bylo celkem vytvořeno 325 tisíc obrázků. Síť dosáhla 95,67 % úspěšnosti, to je více než předchozí model.

Další model CNN má 6 vrstev (4 konvoluční a 2 plně propojené) [32]. Práce uvádí především datovou sadu o 81 tisících obrázků a 120 druhů ovoce. Datová sada byla pojmenována Fruits-360 a je veřejně dostupná a neustále aktualizovaná. Obrázky jsou o velikosti 100x100 pixelů, zbavené pozadí a ovoce je vycentrované. Tato sada byla pořizována v laboratorních podmínkách, data byla rozšířena převrácením a změnou sytosti. Primární úprava dat probíhala předzpracováním v rámci učení. Po testech vyšlo, že nejlepších výsledků se dosáhlo pomocí převedení na stupně šedi s HSV barevným schématem. Byly použity konvoluční vrstvy s aktivační funkcí ReLU tradičně následované sdružovací vrstvou. Dalšími testy byly zjištěny velikosti konvolučních sítí a jejich jader, stejně jako šířka vrstev a počet výstupů. Nejvyšší úspěšnost na testovacích datech byla 96,13 %. Síť má největší problém se správnou identifikací jablka. Autoři se domnívají, že je to způsobeno ztrátou příliš velkého množství charakteristik převodem do stupňů šedi. Pro další vylepšení se autoři zaměřili na nahrazení ReLU aktivační funkce a změnu struktury sítě za účelem získání větší úspěšnosti. Na této datové sadě bylo dosaženo i 100% úspěšnosti [43]. Vytvořili model o dvou konvolučních vrstvách s jádrem 3 a 64 filtry a aktivační funkcí ReLU následované sdružovací vrstvou a dvěma plně propojenými vrstvami, po nichž následovaly vrstvy zahazování. Poslední vrstvou byla softmax. Jako optimalizér byl použit Adam.

Větší úspěšnost získávají předučené modely na velkých datových sadách. Důvodem je jejich velikost a vytvoření koevolučních filtrů na objemných datových sadách. Následné naučení potřebné klasifikace poté není tak náročné a stačí méně dat. Síť tak dosahuje lepších výsledků. Například za použití 16 předučených vrstev dosáhl model úspěšnosti přes 99 % na experimentálních datech a na testovacích, těžko identifikovatelných datech, přes 96% úspěšnost [44]. Síť EfficeNet na zmíněné datové sadě Fruits-360 [32] dosahuje úspěšnosti 98 %. Síť obsahuje 9 konvolučních vrstev s jádrem 3 a ReLU aktivační funkcí následované sdružovacími vrstvami a končí plně propojenou vrstvou [45]. Ještě lepších výsledků dosáhla předučená síť GoogleNet, 22-vrstvá hluboká konvoluční neuronová síť učená na datové sadě ImageNet. Obsahuje konvoluční vrstvy s jádrem 3 a sdružovací vrstvu za každou druhou konvoluční vrstvou. Síť byla doučena na vlastní datové sadě s 6 druhy ovoce. Výsledkem byla 99,99% úspěšnost [46].

Další aplikací je rozpoznání druhu ovoce v maloobchodech, kde prodavač někdy neumí rozpoznat exotické druhy [18]. Aplikace je navíc přímo propojena s pokladním systémem. Byly vytvořeny dvě sítě se stejnou architekturou, ale jiným rozdělením vah. První síť identifikuje druhy ovoce z celého obrázku, poté síť s architekturou YOLOv3 rozpozná, kde se na obrázku nachází ovoce (ROI), obrázek je oříznut a druhá síť identifikuje každé ovoce zvlášť. Podle koeficientu jistoty je poté určeno, o jaký druh ovoce se jedná. Byla vytvořena vlastní datová sada (z reálného prostředí) o 23 tisících obrázcích. Konvoluční sítě se skládají ze dvou standardních konvolučních bloků následujících dvěma plně propojenými vrstvami. Bylo dosaženo 99,78% úspěšnosti s rychlostí zpracování pod 200 milisekund v reálném užití.

4.3.2 Třídění podle kvality

Třídění ovoce podle kvality je proces, který ovlivňuje kontrolu kvality a proces zpracovávání. Jak zmiňuje J. O. Adigun a Y. Gurubelli [29], [47], vysoce kvalitní ovoce a zelenina jsou základem úspěchu na dnešním vysoce konkurenčním trhu. Existuje zvýšená poptávka po dobrém i čerstvém ovoci a zelenině. Tradiční třídění ovoce a zeleniny zahrnuje manuální práci, je časově náročné a vyžaduje kvalifikovanou pracovní sílu. Navíc chybí standardy v kontrole kvality. Přístupy strojového učení a počítačového vidění poskytují technické, proveditelné, nákladově a časově efektivní řešení výše uvedeného problému.

J. O. Adigun a spol. [29] vytvořili aplikaci pro hodnocení kvality jablek. Byly vytvořeny dva modely za pomocí CNN: Jeden k určení, zda je na obrázku jablko, druhý k posouzení jeho kvality. Data byla získána z datové sady Fruits-360 [32], Golden Apple Database [48] a stažena z internetu. Datová sada obsahovala jak obrázky bez jablek, tak s jablky různé kvality. Datová sada byla rozdělena na učící, validační a testovací část. K rozšíření dat v učící sadě pro zamezení přetrénovaní bylo použito metod úpravy obrazu (změna velikosti, přiblížení, horizontální obrácení a rotace). Změna velikosti na 224x224 pixelů byla použita na všechny sady pro zajištění integrity modelů. Pro oba modely byla použita síť ResNet50 s dvěma přidanými plně propojenými sítěmi. Rozdíl mezi nimi je v poslední plně propojené vrstvě, kde má model pro určení, zda je na obrázku jablko (dále jako „ACM“), 2 prvky. Jeho výstup je binární. Model pro určení kvality (dále jako „AGM“) má v poslední vrstvě prvky 4 – klasifikace do 4 tříd kvality (A, B, C, zkažený). Konvoluční vrstvy extrahují primitivní příznaky jako barva, křivky a hrany. První plně propojená

vrstva extrahuje příznaky vyšší dimenze jako tvar a textura. Poslední vrstva je určena pro klasifikaci. Pro trénování ACM bylo použito 3 500 obrázků celkem, 2 100 trénovacích a 1 400 testovacích. Jako chybová funkce byla použita kategoriální křížová entropie a stochastický sestup gradientu pro optimalizaci. Model dosáhl 99,78% úspěšnosti jak na validační, tak na testovací sadě. To ukazuje, že model dobře zvládá generalizaci. Pro AGM byla datová sada (8,4 tisíc obrázků) rozdělena na trénovací a testovací část v poměru 6:4. Zde byl menší počet epoch, ostatní parametry stejné jako u ACM modelu. S AGM bylo dosaženo 99,89% úspěšnosti.

Y. Gurubelli [47] se zabývá tříděním granátových jablek. Přístup je založený na lineární diskriminační analýze (Linear Discriminant Analysis, LDA) sloužící k redukci vyšších dimenzí příznaku. Klasická forma LDA má několik problémů při práci s obrazem. K jejich odstranění bylo použito dvoudimenzionální LDA (2DLDA) a její modifikace fuzzy 2DLDA podle fuzzy KNN (FKNN). Celkem zde byly použity 4 algoritmy pro extrakci příznaků (2DLDA, F2DLDA, FLDA, FF2DLDA) a posléze SVM pro testování, která technika extrakce příznaku poskytuje nejlepší míru rozpoznávání. Poslední zmiňovaný algoritmus frakční fuzzy 2DLDA je vytvořen autorem modelu. Datová sada tvoří 328 barevných obrázků granátového jablka v různých kvalitách. Jedinou úpravou byla změna velikosti na 386x512 pixelů. Modely dosahují nejvyšší úspěšnosti s nejnižším počtem diskriminačních příznaků – rozdíl až 30 % (65 -> 95). Frakční fuzzy 2DLDA vykazuje nejlepší míru rozpoznávání, 97 % (lepší o 4 %), ve srovnání s ostatními třemi technikami, protože rozptylové matice začleňují informace o distribuci překrývajících se vzorků.

D. Sugumar a spol.[15] se zabývali rozpoznáním a tříděním citrusů podle jejich zralosti, konkrétně barvy, velikosti a textury pomocí KNN klasifikátoru. Hlavním cílem modelu je rozpoznat velmi kvalitní citrusy pro export. Jsou použity příznaky jako výška, šířka, plocha, obvod, výstřednost, těžiště, rozsah a fáze Fourierova popisu. Model rozlišuje dvě kategorie citrusů, od každého je v datové sadě 300 obrázků pro učení a 25 pro testování. Obrázky byly pořízeny na bílém pozadí a ovoce je vycentrované. Jako první jsou převedeny do stupňů šedi a do černo-bílého barevného spektra. Dále jsou spočítány zmíněné charakteristiky. Druhou částí klasifikace je spočítání žlutých, zelených, hnědých a černých pixelů odděleně. Samotná klasifikace pomocí KNN je rozdělena do 3 tříd, každá třída je identifikována pomocí algoritmu rozhodování o více atributech (Multi Attribute Decision Making, MADM). První třídou je druh ovoce, ve kterém klasifikátor dosáhl 99% úspěšnosti. Druhou třídou je kvalita, která byla rozeznávána jako „dobrá“ a „špatná“. Zde

byla úspěšnost 93,5 %, přičemž model chyboval pouze u špatné kvality – označil ji za dobrou. Poslední třídou byla zralost – nedozrálý, zralý a shnilý. Podle barevných ploch lze dopočítat, kdy ovoce dozraje. Zde model chyboval pouze u kategorie „shnilé“, kdy nesprávně označoval za zralé. Celková úspěšnost byla 95,6 %.

Další aplikací je kontrola kvality ovoce v supermarketech, což by mělo zajistit stále čerstvé ovoce v nabídce. S. Ashraf a spol. [30] navrhli systém na architektuře Inception v3, který přeučili na vlastní datové sadě získané ze záznamů kamer v obchodech.

4.3.3 Detekce plodů ovoce

V posledních letech se začalo intenzivně zkoumat uplatnění rozpoznávání ovoce v sadech především ve spojení s automatickým sběrem, monitorováním průběhu růstu, zrání a odhadu sběru ovoce i celých plodin. Nalezení plodu ovoce na obrázku je zásadní pro úspěšný a reálně použitelný sběr robotem. V přirozených podmínkách to má několik obtíží, jako netransparentní pozadí často splývající s ovocem nebo proměnlivé osvětlení [49], [50].

Tématu identifikace a počítání plodů ovoce na stromech se věnuje mnoho prací. Bylo shrnuto 40 studií [12]. Bylo doporučeno používání RGB kamery pro pořizování snímků (především kvůli ceně a snadné implementace) a použití ručně vytvořených vlastností jako barva, textura, tvar v detekci ovoce. A. Koirala [19] dále rozvíjí tuto studii a uvádí, že tyto vlastnosti potřebují upravit při venkovním použití na základě kalibračních podmínek – variace ovoce, vzdálenost a úhel pozorovaného ovoce, osvětlení, a tudíž nejsou ideální. Nejlepší algoritmy dosahují téměř 100% míry rozpoznání v laboratorních podmínkách a 85-99% míry v reálných podmínkách. Tento rozdíl je způsoben především zmíněnými přírodními světelnými podmínkami. Vyřešením této výzvy lze dosáhnout nahrazení lidské pracovní síly ve sbírání ovoce v sadech za mechanickou a ušetřit tak náklady a čas.

S. Sun a D. Liu [50], [51] zkoumají vylepšení rozpoznávání plodů kiwi, přičemž dosáhl 99% úspěšnosti oproti nejlepšímu předešlému algoritmu s 86% přesností za použití homomorfního filtrování. Tomuto tématu se věnuje i G. Wu [49], který uvádí úspěšnost 88 % s 292 milisekundovou odezvou v rozpoznání jednoho kusu broskve v reálných podmínkách sadu pomocí RGB-D kamery kombinované s údaji o barvě a 3D konturách obrázku. Sběru jablek v reálných podmínkách sadu se věnuje S. Sun a spol. [50]

s vylepšeným modelem GrabCut s NCut algoritmem. Dosahují s nimi lepších výsledků než ostatní používané modely (GBVS, AIM, SDSR) a to o desítky procent.

C. Zhao [33] se věnuje rozpoznávání objektů v přírodních světelných podmínkách. Přesněji zeleným citrusovým plodům na jednotlivých stromech. Ve své studii popisuje algoritmus založený na adaptivní červeno-modré chromatické mapě, superpixelech, místních binárních vzorech a KNN klasifikátoru. Obrázky byly pořizovány v RGB barvách, za dne s různými světelnými podmínkami. Pro lepší efektivnost byla jako první změněna velikost obrázku na 912x684 pixelů. Poté byla aplikována chromatická mapa, kterou autor vyvinul již ve své předchozí práci. Používá se pro odfiltrování pozadí – necitrusové pixely. Je komplikované vybrat správné spektrum barev k odstranění, protože listy a plody mají podobné spektrum. Dále se používá metoda superpixelů pro rozlišení zbylých pixelů. Jak název napovídá, tvoří z pixelů různé bloky. Lze to vnímat jako abstrakci základních informací – každá oblast má poté svůj histogram a texturu. Superpixely mají několik algoritmů. V této studii se používá SLIC, který se jevil oproti jiným jako nejspolehlivější a zároveň nejrychlejší. Jako atributy byly zvoleny Tamura textury, hrubost a kontrast. Poté byla použita metoda LBP, která je výpočetně nenáročná a velmi efektivní pro popsání textury histogramem, testuje vztah mezi pixelem a jeho sousedy. Histogram celého obrázku je složen z histogramu každého superpixelu. Z vytvořené LBP matice (jednotlivé histogramy) byly extrahovány statistické veličiny – průměr, odchylka, entropie, pravidelnost a druhý úhlový moment. Ty navíc spolu Tamura a LBP hodnotami jsou atributy neparametrického klasifikátoru KNN pro odstranění nesprávně pozitivních výsledků. Bylo dosáhnuto 83% úspěšnosti. Falešně pozitivních výsledků bylo 15 %. Vysoká míra chybovosti byla způsobena překrýváním ovoce listy tak, že byla vidět jen čtvrtina. Algoritmus dokáže rozpoznat plod, pouze pokud je vidět více než zmíněná čtvrtina. Dalšími úskalími byly světelné podmínky a barevná podobnost, kdy například osvícení listu způsobilo vysoký kontrast a tím falešně pozitivní výsledek.

I. Sa [28] prezentoval hlubokou konvoluční neuronovou síť k nalezení a ohraničení ovoce na obrázku v reálném čase, což je klíčové pro automatizovaný sběr a odhad úrody. Byl přeučen model Faster R-CNN pro rozpoznávání ovoce. Síť se skládá z 13 konvolučních vrstev architektury VGG-D. Síť velmi dobře generalizovala z malé datové sady 122 obrázků. Základem modelu bylo ladění modelu VGG16, který byl předtrénovaný na datové sadě ImageNet. Byl obohacen o použití multimodálních informací pozdní fúze RGN a NIR barevného schématu. Model dosáhl úspěšnosti 0,83 F1 skóre na reálné datové sadě

ze sadu. Udržel si zároveň rychlost detekce. Tento výsledek byl komparativní s předchozí prací autorů, ve které s modelem na základě rozpoznávání pixelů dosáhli 0,8 F1 skóre.

V porovnání s dalšími modely hlubokého učení si tento model nestojí nejlépe. Architektura SSD ResNet měla F1 skóre 0,96 na datové sadě 6 druhů ovoce při rozpoznávání a ohraničení plodů ze stromu v reálných podmínkách. Ještě lépe, 0,97 F1 skóre, měla architektura MangoYOLO při identifikaci a ohraničení pouze 1 druhu ovoce. [19]

4.3.4 Identifikace nemocí a škůdců ovoce

S. Xing [52] se ve své práci věnuje identifikaci škůdců a nemocí u citrusového ovoce. Uvádí, že tyto dvě kategorie nejvíce ovlivňují úrodu citrusů, jelikož mohou způsobit velké škody, pokud nejsou včas určeny. Pro farmáře je obtížné rozeznat škůdce včas. V přírodě se vyskytuje příliš mnoho druhů a některé jsou si velmi podobné. V práci autor uvádí nový algoritmus založený na principu zlepšení využití parametrů modelu – slabě hustě propojená konvoluční síť (Weakly Dense Connected Convolution Network, WDCCN). Design sítě vychází z toho, že některé mapy příznaků vytvořené konvoluční vrstvou nejsou užitečné, a proto se je snaží redukovat. Využívá k tomu křížovou fúzi příznaků, dvě po sobě jdoucí konvoluční vrstvy s jádrem 1. S touto myšlenkou přišla poprvé síť HighwayNet, dále byla rozvíjena architekturou ResNet a DenseNet. WDCCN navíc vynechává některá propojení mezi vzdálenými vrstvami. Byla použita vlastní datová sada 17 druhů škůdců (převážně získaných z internetu) a 7 druhů nemocí (vlastní tvorba) o velikosti 12,5 tisíc obrázků. Všechny obrázky byly převedeny do velikosti 224x224 pixelů. Experiment proběhl i na několika dalších sítích – MobileNet, VGG-16, SENet-16 a NIN-16. WDCCN měla nejlepší výsledky, 93,4 %, na testovací sadě. Podobný výsledek (93 %) měla také síť VGG-16. Rozdíl byl markantní ve velikosti modelu, kde VGG-16 měla 120 MB, a WDCCN pouze 30,5 MB. To bylo pro autory klíčové. Dosáhli lepších výsledků s jednodušší sítí. Místo prohloubení a rozšíření sítě se zaměřili na optimalizaci parametrů modelu. Vytvořili tak optimalizovanou odlehčenou síť.

Další aplikací je identifikace škůdců a nemocí na banánech pomocí hluboké konvoluční neuronové sítě. Banány jsou oblíbené po celém světě a často základní potravina v mnoha rozvojových zemích. Nemoci velmi ovlivňují produkci banánů. Byl proto vytvořen model pro rozpoznání škůdců a nemocí z listů a plodů [16]. Z různých míst v Africe byly experty shromážděny a popsány obrázky nemocí banánů a vytvořena datová

sada 18 druhů nemocí. Byly přeučeny 3 architektury CNN a vytvořeno celkem 6 modelů. Experimenty odhalily, že ResNet50 na bázi InceptionV2 mají nejlepší úspěšnost přes 90 %.

4.3.5 Klasifikace pokrmů

Další, širší aplikací je určení kalorické a výživové hodnoty ovoce, popřípadě celého jídla, a to v jakémkoliv bodě jeho přípravy či servírování. Z fotografie pokrmu síť určí zmíněné hodnoty na základě segmentace jídla k příslušnému druhu – maso, příloha, obloha, nápoj apod. Bylo testováno několik architektur, nejlepších výsledků bylo dosaženo se SegNetVGG16, a to 94,86% úspěšnosti [53]. Další rozšířenou aplikací je řešení problémů a výzev v potravní doméně. Týká se to třídění potravin, zmíněného odhadu kalorií, detekce kvality ovoce a zeleniny, masa a tekutých produktů, optimalizace potravinového řetězce a kontaminace potravin [22]. Jak dále L. Zhou [22] uvádí, využití ocení obzvláště diabetici, alergici apod., kteří by měli přísně sledovat a kontrolovat své stravovací návyky.

4.3.6 Shrnutí

V této kapitole 4.3 byly popsány domény aplikace strojového učení pro rozpoznávání ovoce a byly prezentovány aktuální modely, zjištění jsou shrnuta v Tabulka 3 Shrnutí použitých modelů v aplikačních doménách (Zdroj: vlastní zpracování)

Je nutné poznamenat, že využitelnost strojového učení v zemědělství je mnohem širší: například je možné odhadovat růstu plodin, úrodu či datum sklizně; podle listu nebo tvaru rostliny lze identifikovat druh, nemoci či škůdce [12].

Neuronové sítě, především CNN si vedou v rozpoznávání ovoce výrazně lépe než metody klasického strojového učení [12], [21]. Je to dáno především automatickým vytvořením příznaků konvolučními vrstvami, na rozdíl od klasických metod rozpoznávání obrazu, kde musí být vytvořeny předem ručně. Obecně lze říct, že přesnost neuronových sítí roste s jejich hloubkou za cenu zvýšení potřebného výpočetního výkonu a objemnější datové sady. Byly vyvinuty modely detekce objektů (SSD, YOLO, Faster R-CNN) jako odpověď na potřebu rychlejších sítí pro zpracování obrazu v reálném čase. Dostupnost těchto algoritmů s předtrénovanými vahami a objemné veřejně přístupné datové sady označených obrázků pro trénování dělají hluboké učení dostupnějším a urychlují vývoj.

Za použití pár stovek obrázků lze vytvořit laděný model pro použití v určité doméně detekce objektů [19].

Zdroj	Úloha	Řešení / architektura	Datová sada	Přesnost
[35]	Klasifikace druhů ovoce	Expertní systém, SVM	480 obrázků; reálné prostředí	94,97 %
[41]	Klasifikace druhů ovoce	SVM	Laboratorní data	~ 100 %
[34]	Klasifikace druhů ovoce	13vrstvá CNN	3 600 obrázků; reálné prostředí	94,94 %
[42]	Klasifikace druhů ovoce	8vrstvá CNN	3600 obrázků; reálné prostředí	95,67 %
[32]	Klasifikace druhů ovoce	6vrstvá CNN	Fruits-360; laboratorní prostředí	96,13 %
[43]	Klasifikace druhů ovoce	4vrstvá CNN	18 000 obrázků; laboratorní prostředí	99,79 %
[44]	Klasifikace druhů ovoce	16vrstvá CNN	2633 a 5946 obrázků; reálné prostředí	96 %
[45]	Klasifikace druhů ovoce	EfficeNet (13vrstvá CNN)	Fruits-360; laboratorní prostředí	98 %
[46]	Klasifikace druhů ovoce	GoogleNet (22vrstvá CNN)	ImageNet; reálné prostředí	99,99 %
[18]	Klasifikace druhů ovoce	GoogleNet (22vrstvá CNN)	23 000 obrázků; reálné prostředí	99,78 %
[29]	Klasifikace kvality ovoce	ResNet50 (50vrstvá CNN)	Fruits-360; laboratorní prostředí	99,89 %
[47]	Klasifikace kvality ovoce	FF2DLDA, SVM	328 obrázků; reálné prostředí	97 %
[15]	Klasifikace kvality ovoce	KNN	700 obrázků; laboratorní prostředí	95,6 %
[51]	Detekce plodů	Homomorfní filtrování	Reálné prostředí	99,16 %
[49]	Detekce plodů	RANCAC & PROSAC	Reálné prostředí	88 %
[50]	Detekce plodů	GrabCut	200 obrázků; reálné prostředí	94,12 %
[33]	Detekce plodů	Superpixely, LBP, KNN	80 obrázků; reálné prostředí	83 %
[28]	Detekce plodů	VGG16	122 obrázků; reálné prostředí	83 %
[19]	Detekce plodů	SSD ResNet	Reálné prostředí	96 %
[19]	Detekce plodů	MangoYOLO	Reálné prostředí	97 %
[52]	Identifikace nemocí a škůdců	WDCCN	12 500 obrázků; reálné prostředí	93,4 %
[16]	Identifikace nemocí a škůdců	ResNet50 (50vrstvá CNN)	18 000 obrázků; reálné prostředí	~ 90 %
[53]	Klasifikace pokrmů	SegNetVGG16	236 obrázků; laboratorní prostředí	94,86 %

Tabulka 3 Shrnutí použitých modelů v aplikačních doménách (Zdroj: vlastní zpracování)

5 Praktická část

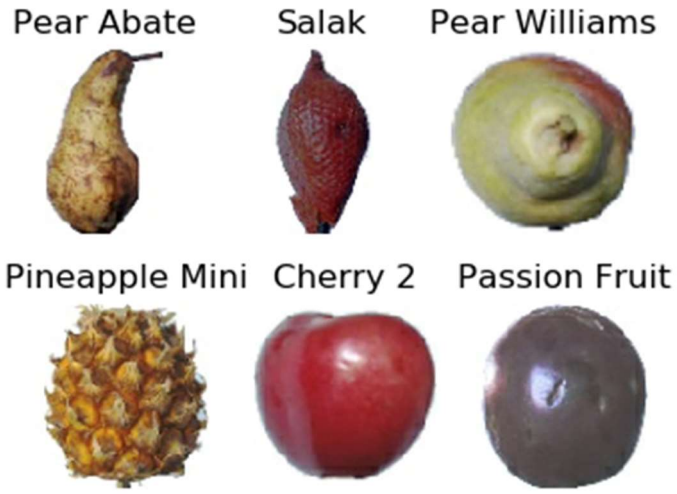
Praktická část pojednává o procesu vytváření ukázkové aplikace pro identifikaci druhu ovoce z obrázku. Přesněji o zkoumání modelu konvoluční neuronové sítě pro kategoriální klasifikaci ovoce. V této části jsou popsány předpoklady modelu, jeho výsledky a implementace. Jsou zhodnoceny dosažené výsledky, graficky znázorněny vlivy zkoumaných parametrů sítě na její úspěšnost.

5.1 Návrh aplikace

Tato ukázková aplikace spadá do učení s učitelem, více-kategoriální klasifikace, representativního učení a hlubokého učení. Jedná se o aplikaci kategorizaci druhu ovoce popsanou v kapitole 4.3.1. V této kapitole byla popsána použitá datová sada, její možnosti a omezení, návrh konvoluční neuronové sítě a postup ladění hyperparametrů modelu.

5.1.1 Datová sada

Byla použita datová sada Fruits-360 o celkovém počtu 120 kategorií a 81 120 obrázků ve formátu JPG, vytvořena v práci od autorů H. Mureşan a M. Oltean [32]. Data byla pořizována v laboratorních podmínkách popsaných v kapitole 4.2.5. Objekty na obrázku jsou vycentrované a na bílém pozadí, které bylo odstraněno specializovaným algoritmem typu záplavové výplně (flood fill). Při aplikování algoritmu se začalo od každého okraje obrázku, označily se všechny pixely, poté všechny sousední pixely k již označeným pixelům, pro které byla vzdálenost mezi barvami menší než prahová hodnota zvolená na začátku testování. Předchozí krok se opakoval, dokud nezbyly žádné pixely k označení. Všechny označené pixely byly považovány za pozadí a zabarveny bílou barvou. Maximální prahová hodnota vzdálenosti mezi barvami byla stanovena na 2 metodou pokus-omyl pro každý druh ovoce. Velikost obrázku je 100x100 pixelů. Autoři uvádí, že při menší velikosti není možné rozeznat například jablko od nektarinky. Naopak větší velikosti obrázků potřebují více času na učení, a proto autoři zvolili kompromis. Data obsahují z větší části ovoce, z menší části zeleninu. Tyto třídy byly ponechány.

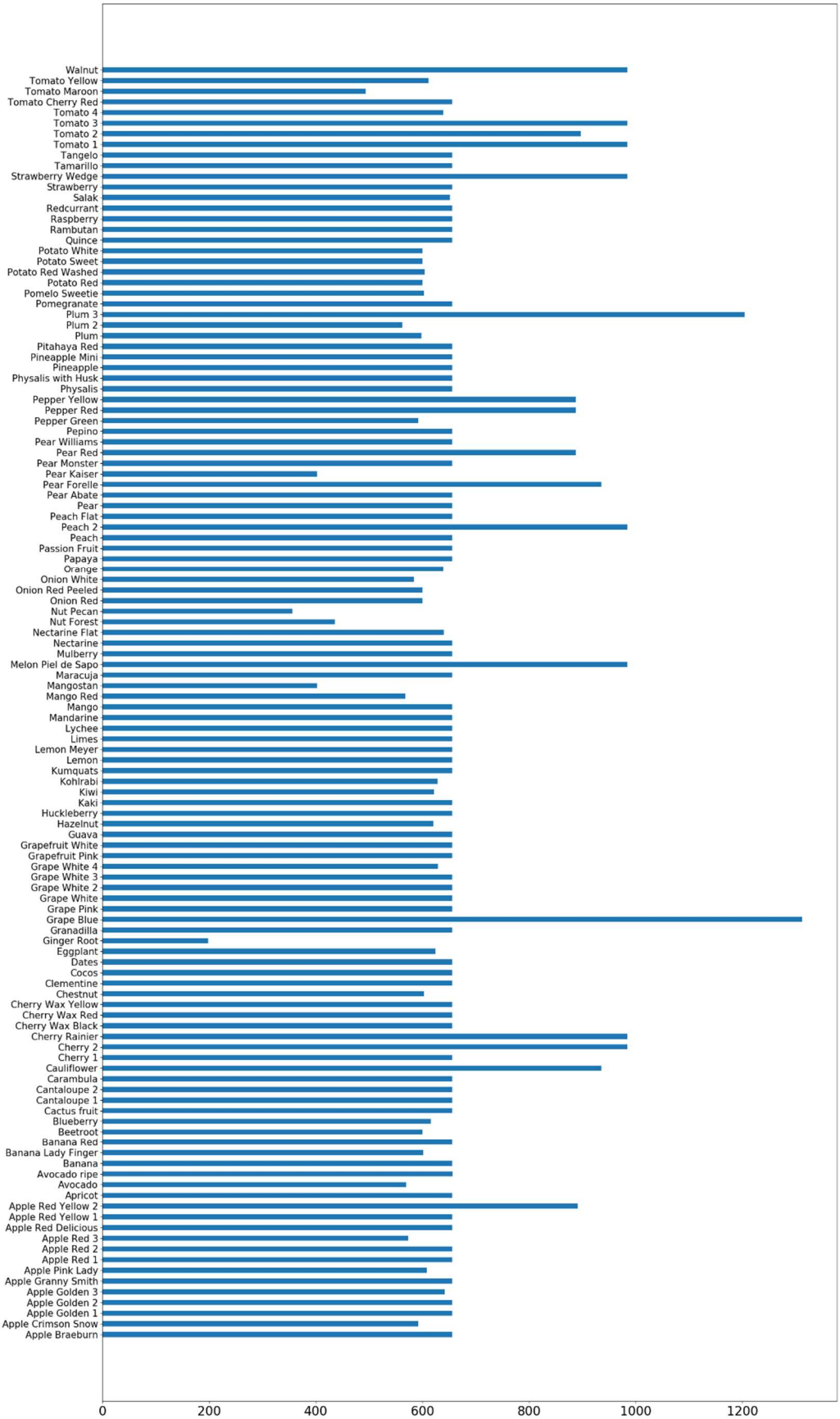


Obrázek 16 Obrázky z datové sady Fruits-360

Zdroj: obrázek vlastní zpracování, obrázky z Fruits-360 [32]

Data byla rozdělena do učící a testovací sady v poměru 75:25. Pro učení neuronové sítě v této práci byla rozdělena i učící sada na učící a validační. Výchozí celkové poměry učící, validační a testovací sady jsou 60:15:25. Byly testovány i jiné poměry, kolem doporučovaných hodnot - 50:25:25 s horšími výsledky.

Výčet kategorií s jejich absolutním počtem obrázků je v příloze 1. Na Graf 1 lze vidět histogram kategorií. Byly ponechány originální anglické názvy kategorií z důvodu transparentnosti při porovnávání. Lze vidět mírnou nerovnoměrnost v zastoupení jednotlivých kategorií. Většina tříd je však zastoupena 656 obrázky a tomu odpovídá i medián zastoupení datové sady. Ve zmíněné publikaci H. Mureşana [32] bylo uvedeno, že vytvořený model na této datové sadě měl problém rozeznat mezi sebou jednotlivé druhy jablek. V průběhu experimentů je sledována schopnost modelů rozeznávat třídy jablek pro porovnání s původním modelem.



Graf 1 Histogram rozdělení četností kategorií v datové sadě Fruits-360

Zdroj: Vlastní zpracování

5.1.2 Model neuronové sítě

Pro modelovaný problém byla zvolena konvoluční neuronová síť na základě zjištění lepších dosahovaných výsledků než u tradiční metody strojového učení. Síť se skládá z:

- Vstupní vrstvy, serializujícího vstupního obrázku,
- několika po sobě jdoucích konvolučních bloků,
- několika FC vrstev, extrahujících složité příznaky,
- výstupní softmax vrstvy o velikosti počtu tříd v datové sadě.

Základní konvoluční blok lze charakterizovat jako konvoluční vrstvu s jádrem 5, posunem 1, následovanou aktivační vrstvou ReLU a maximální sdružovací vrstvou s jádrem a posunem 2.

Základní otázkou je, jak by měl být model hluboký, kolik by měl mít konvolučních bloků, filtrů v konvoluční vrstvě, FC vrstev a počet jejich neuronů. Odpověď na tuto otázku není jednoznačná. Doposud neexistuje deterministický způsob určení optimální velikosti sítě. Hlubší sítě si vedou obecně lépe, pokud mají dostatek dat k učení a jsou optimalizované pro příslušnou hloubku. Na používané datové sadě bylo během jejího tvoření provedeno již několik experimentů s výsledky mezi 96 až 99,79% úspěšností, avšak ne všechny měly k dispozici stejné množství tříd a obrázků jako v této práci (poslední aktualizace datové sady proběhla na konci roku 2019). Hloubka modelů se pohybovala od 2 konvolučních vrstev až po 50 reziduálních bloků. V této práci byla snaha o vytvoření mělké hluboké sítě (2-5 skrytých vrstev). Tato úvaha vychází z výsledků předešlých prací a vstupní velikosti obrázku, která je 100x100 pixelů. Každou použitou sdružovací vrstvou se zmenší vstup o polovinu a po 5 takových vrstvách je výstupem tenzor [3; 3; $počet\ filtr$ů]. Proto byl model směřován spíše do šířky.

V následující tabulce je popsána výchozí architektura 4 modelů, které byly v práci testovány, a dále je pracováno pouze s nejlepší z nich. Model 2 byl odhadnut pomocí Hyperband optimalizačního algoritmu. Měl za úkol nalézt optimální počet konvolučních bloků, filtrů v rozmezí 2-5 a 32-512 a FC vrstev s počtem neuronů v rozmezí 1-2 a 256-1024. Další modely jsou odvozeny z předchozích prací a nalezeného optima. Každý model má jiný počet konvolučních bloků. Parametry konvolučních bloků jsou uvedeny pro jednotlivé bloky. Všechny sdružovací vrstvy mají jádro 2. Výstupní vrstva je FC s aktivační funkcí Softmax o velikosti počtu tříd, kde výstupní hodnota každého neuronu odpovídá pravděpodobnosti, zda obrázek patří do dané třídy.

Pro redukci přetrénování byla použita metoda v originále zvaná jako dropout (Dropout layer) [54], dále je na ni odkazováno jako na dropout vrstvu. Ta snižuje přetrénování. Klíčovou myšlenkou je náhodně vyhodit neurony (spolu s jejich spojeními) z vrstvy s určitou pravděpodobností, a to pouze během tréninku. Tím se zabrání přílišné adaptaci neuronů na určitý vstup. Při testování se použije síť se všemi neurony. Síť má poté menší váhy. To významně snižuje přetrénování a poskytuje vylepšení oproti jiným metodám regularizace, se kterými lze tuto metodu kombinovat. Vrstva má parametr p - míra vynechání neuronů ve vrstvě.

S architekturou modelů je dále experimentováno následovně (písmena odpovídají referencím na úpravu modelu dále v práci):

A. Přidání konvolučních vrstev za sebe v konvolučním bloku pro dosažení lepší extrakce příznaků (s jádrem 1 pro zdůraznění vlivu jednotlivých barevných kanálů).
B. V návaznosti na práci J. T. Springenberga [26], nahrazení sdružovacích pouze konvolučními vrstvami s jádrem 1 a stejným posunem u sdružovací vrstvy.
C. Přidání dávkové normalizační vrstvy.
D. Přidání dropout vrstvy (byla zkoumána otázka, zda použití dávkové normalizace společně s dropout vrstvou má za následek zhoršení kvality modelu, jak popsal X. Li [55], označováno jako varianta CD).

Model 1	Model 2	Model 3	Model 4
Vstupní vrstva (100,100,3)			
2 konvoluční bloky	**3 konvoluční bloky**	**4 konvolučních bloků**	**5 konvolučních bloků**
Konvoluční vrstva (filtr; jádro)			
128; 5 128; 5	192; 5 240; 5 16; 5	32; 5 64; 5 128; 5 32; 5	64; 5 64; 5 64; 5 64; 5 64; 5
Sdružovací vrstva (posun)			
3	2		
Plně propojená vrstva			
512	896	1024	512
Výstupní softmax vrstva			

Tabulka 4 Navržené modely

Zdroj: Vlastní zpracování

Pro konvoluční vrstvy je použit inicializér vah s rozdělením *he_norm*, které bere hodnoty ze zkráceného normálního rozdělení se středem na 0 a $\sigma = \sqrt{\frac{2}{F}}$, kde F je počet vstupních jednotek v tenzoru vah. Báze inicializovány na 0. Ztrátová funkce je pro všechny modely stejná, a to kategoriální křížová entropie (categorical crossentropy). Byl použit regularizér L2 umožňující během optimalizace aplikovat sankce na aktivitu vrstvy. Tyto sankce jsou začleněny do ztrátové funkce, kterou síť optimalizuje [1].

Úprava konvolučního bloku modelu 1 A vypadá tedy takto (pro ostatní modely obdobně):

<table>
<tr><td colspan="2">Varianta 1 A</td></tr>
<tr><td colspan="2">Konvoluční vrstva (128; 5)
Konvoluční vrstva (128; 1)
Sdružovací vrstva (2)
Konvoluční vrstva (128; 5)
Konvoluční vrstva (128; 1)
Sdružovací vrstva (3)</td></tr>
<tr><td>Varianta 1 B</td><td>Varianta 1 C & 1 D</td></tr>
<tr><td>Konvoluční vrstva (128; 5)
Konvoluční vrstva (128; 1; 2)
Konvoluční vrstva (128; 5)
Konvoluční vrstva (128; 1; 3)</td><td>Konvoluční vrstva (128; 5)
Normalizační vrstva
Sdružovací vrstva (3)
(Dropout vrstva)</td></tr>
</table>

Tabulka 5 Architektura variant A, B, C, D, CD modelu 1

Zdroj: Vlastní zpracování

5.1.3 Ladění modelu

Pro vyhodnocení modelu byly použity standardní metriky pro kategoriální klasifikaci zmíněné v kapitole 4.1.9 Vzhledem k nerovnoměrnému zastoupení obrázků v kategoriích jsou použity vážené průměry těchto metrik, tj.: primárně F1 skóre a přesnost pro srovnání s ostatními modely. Dále jsou uvedeny preciznost, senzitivita a matice změn pro hlubší zkoumání výsledků experimentů.

Experimenty využívají metodu zádrže, jak bylo zmíněno výše. Křížová validace vzhledem k velikosti datové sady a časové náročnosti výpočtu není použita.

Laděnými hyperparametry modelu jsou:

- Velikost jader,
- velikost dávky,
- aktivační funkce (ReLU, PReLU, ELU),
- míra zahazování (dropout rate).

Prvotní zjištění optimální velikosti modelu bylo provedeno pomocí algoritmu v originále tzv. Hyperband [56]. Zaměřujeme se na urychlení náhodného vyhledávání prostřednictvím adaptivního přidělování zdrojů. Metoda využívá včasného zastavení, adaptivně přiřadí náhodně definovaný zdroj, např. iteraci, vzorky dat nebo počet funkcí k náhodnému výběru. Rozšiřuje tak algoritmus v originále zvaný jako Siccesive Halving. Hyperband byl zvolen z důvodu lepší optimalizace alokovaných zdrojů (výpočetních i časových), nalezení parametrů oproti Bayesové optimalizaci hyperparametrů (Bayes Hyperparameters Optimalization), GridSearch nebo RandomSearch [57], které jsou výpočetně velmi neefektivní. V prozkoumané literatuře se neobjevuje automatický typ ladění modelu. Autoři tuto skutečnost neuvádějí, nebo jsou modely laděny manuálně – tak, jak tomu je části v této práci. I tak odhad parametrů trval 47 hodin a úspěšnost výsledného modelu byla 60 % na validačních datech po 40 epochách.

Optimální velikost dávky [5] se uvádí 32 až 256. V práci od autorů S. Ashraf a spol., [30] je uvedeno i 15. Menší dávka (v práci *On large-batch training for deep learning: Generalization gap and sharp minima* [58] je definována jako 32 až 512) provede více průchodů zpětně propagačním algoritmem a provede úpravu gradientu s méně přesnou minimalizací směrem. Větší dávka (512+) provede naopak méně průchodů s lepším směrem gradientu, protože optimalizuje gradient na základě více obrázků najednou. Velké dávky špatně generalizují, protože průchodů bývá málo k naučení se vztahů mezi charakteristikami a výstupem. Čas učení se s větší dávkou snižuje. Jde o to optimalizovat velikost dávky vzhledem k velikosti a obsahu datové sady. Každá se může chovat jinak [58]. Dále v práci je rozebrán vliv dávky na použitou datovou sadu Fruits-360.

Dále byla použita technika normalizace dávek. Poskytuje způsob reparametrizace téměř jakékoli sítě. Reparametrizace výrazně snižuje problém koordinace aktualizací gradientu napříč mnoha vrstvami. Normalizaci dávky lze aplikovat na jakoukoli síť. Normalizuje aktivace předchozí vrstvy v každé dávce, tj. aplikuje transformaci, která udržuje průměrnou aktivaci blízko 0 se standardní směrodatnou odchylku blízko 1 [1], [5].

Ve všech modelech byl použit algoritmus s adaptivní mírou učení – Adam. Vykazuje empiricky nejlepší výsledky v praktických aplikacích [5], [21]. SGD udržuje jednotnou míru učení pro všechny aktualizace vah a míra učení se během tréninku nemění. Adam počítá individuální adaptivní rychlosti učení pro různé parametry z odhadů prvních a druhých okamžiků gradientů. Kombinuje výhody dvou dalších přístupů – RMSProp

a AdaGrad. Místo toho, aby přizpůsobil rychlosti učení parametrů na základě průměrného prvního okamžiku (průměr) jako v RMSProp, Adam také využívá průměr druhého momentu gradientů (necentrovaná variance) a metodu hybnosti (momentum), která je navržena tak, aby urychlila učení, zejména s ohledem na vysoké zakřivení, malé, ale konzistentní gradienty nebo rušivé gradienty. Algoritmus hybnosti akumuluje exponenciálně klesající klouzavý průměr minulých gradientů a pokračuje v jeho směru [5].

5.2 Popis implementace

V této kapitole byla popsána implementace modelu, technické zázemí projektu, obtíže, které se vyskytly během procesu vytváření modelu a vysvětleny klíčové části kódu. Celý zdrojový kód pak lze nalézt v příloze 4.

5.2.1 Experimentální prostředí

Hardware – Experimenty byly prováděny na stroji s Intel Core i5-8600 CPU, 64 GB RAM, základní deskou ASUS ROG STRIX Z370-I, grafickou kartou Nvidia GeForce RTX 2080 a OS Windows 10 Pro. Učení bylo realizováno na GPU s CUDA jádry. Učení na GPU bylo přes 120krát rychlejší než učení na CPU.

Software – Programovací jazyk Python ve verzi 3.7 (tato verze z důvodu kompatibility s platformou pro strojové učení). Jako platforma byl použit Tensorflow (dále jen jako „TF“) verze 2.1.0 [2] s nadstavbou Keras verze 2.3.1 [1]. Pro Keras lze využít různé knihovny strojového učení. TF je open source knihovna pro strojové učení vyvinutá společností Google, která se používá k provádění složitých numerických operací a několika dalších úkolů k modelování modelů. Jeho architektura umožňuje nasazení výpočtů na více platformách, CPU, GPU a paralelní distribuci výpočtů. Pracuje primárně s tenzory. Jedná se o matematický objekt, zobecnění vektoru, který má mnoho dimenzí. Platforma funguje na systému výpočetního grafu. Výpočtový graf má síť uzlů, přičemž každý uzel provádí operaci, jako je sčítání, násobení nebo vyhodnocení některých vícerozměrných rovnic. Hrany představují tenzory. TF umí zpracovat data různých typů, jako je celé číslo, booleovské číslo, znak atd. Lze jej použít k vytváření libovolných typů algoritmů hlubokého učení, jako jsou CNN, RNN, dopředné neuronové sítě, pro zpracování přirozeného jazyka atd. Existuje několik programovacích prvků v TF, jako jsou konstanty,

proměnné, zástupné symboly, relace atd. Každý má své vlastní funkce a používá se k sestavení jakéhokoli modelu hlubokého učení.

Od TF verze 2.0 je součástí knihovny implementace na GPU a již není potřeba stahovat samostatný balíček knihovny tensorflow-gpu. Dále bylo třeba nainstalovat Nvidia CUDA pro práci s jádry na grafické kartě ve verzi 10.1 a cuDNN ve verzi 7.6. V době psaní práce byly již dostupné novější verze zmíněných programů, nebyly však kompatibilní s TF 2.1.

Vývojovým prostředím byl PyCharm 2020.01 od JetBrains. K vývoji byl použit i verzovací systém Git. Dále byly použity externí Python knihovny pro usnadnění práce s daty a vykreslování grafů – keras-tuner (Hyperband algoritmus), numpy (práce s vektory, matice a vícerozměrnými poli), sklearn (balíček pro strojové učení, obsahuje některé funkce, které Keras nemá implementované a jsou v práci používané), matplotlib a seaborn (vizualizace dat), pandas (datová analýza a manipulace s velkými daty).

5.2.2 Předzpracování datové sady

Datová sada byla stažena z portálu Kaggle [59]. Klíčové byly složky Train a Test. Podsložky obsahují název třídy, v nich pak jsou jednotlivé soubory. Bylo vyzkoušeno několik variant zpracování souborů do přijatelného formátu pro vstup do sítě. TF nabízí 2 varianty – jako vícerozměrné pole ve formátu (numpy array), nebo TF dataset objekt. Keras pak nabízí třetí variantu – generátor (pracuje na bázi numpy polí, protože operuje s daty mimo TF operační graf). Každá varianta měla svá omezení, dopad byl především na rychlost učení. Načtení obrázků do formátu numpy pole potřebuje několikanásobně více operační paměti než stejná data ve formátu TF dataset. S jeho použitím se zvedla rychlost učení v průměru 6krát. TF tenzor má však nevýhodu v malé možnosti úpravy obrázků. Ta je velmi dobře pokryta v generátoru Kerasu, který je pomalejší než čistě používaný TF dataset, protože data augmentuje za chodu. Data lze převést z generátoru do tenzoru ještě před vstupem do modelu. Tato varianta nepřinesla žádné zrychlení učení. V rámci testování augmentace dat byl použit Keras generátor, jinak TF dataset.

Proces zpracování datové sady byl proveden následovně. Nejprve byla celá datová sada převedena na numpy pole a uložena na disk v převedené podobě, aby převod nemusel být prováděn při každém spuštění (v této podobě jsou uloženy i ukázkové datové sady v TF a Kerasu). Po nahrání dat byla datová sada rozdělena na validační a učící sadu, přivedena do generátoru a následně na vstup modelu. Generátor má možnost načíst data

autonomně ze složky, nicméně to by znemožnilo náhodné rozdělení na učící a validační sadu při zanechání stejného rozdělení prvků ve třídách, a tím zhoršilo možnost ladění modelu.

Uložení dat - Pro nahrání souborů slouží funkce *load_files()*, z balíčku *sklearn* [60], která vytvoří pole s cestami k souborům, názvem kategorie a polem všech kategorií a vše náhodně zamíchá (páry soubor – třída odpovídají). Celá funkce a její použití pak vypadá následovně. Obě složky se nahrávají zvlášť.

```
def load_files_from_directory(path, shuffle=True):
    files = load_files(path, shuffle=shuffle)
    filenames = np.array(files['filenames'])
    target = np.array(files['target'])
    target_names = np.array(files['target_names'])
    return filenames, target, target_names


train_images, train_labels, target_labels = load_files_from_directory(TRAIN_DIR)
test_images, test_labels, _ = load_files_from_directory(TEST_DIR)
```

Zdrojový kód 1

Následuje nahrání obrázků do paměti funkcemi *tf.keras.preprocessing.image.load_img()* a *tf.keras.preprocessing.image.img_to_array()*. Poté jsou data spárována se štítky a uložena pomocí balíčku *pickle*.

```
def load_images_from_filepaths(filepaths):
    images = []
    for filepath in filepaths:
        image = tf.keras.preprocessing.image.load_img(filepath, target_size=(
        IMAGE_WIDTH, IMAGE_HEIGHT))
        images.append(tf.keras.preprocessing.image.img_to_array(image))
    return images


train_images = np.array(load_images_from_filepaths(train_images))
test_images = np.array(load_images_from_filepaths(test_images))

train_dataset = np.array(
    {"images": train_images, "labels": train_labels, "target_labels":
target_labels})
test_dataset = np.array({"images": test_images, "labels": test_labels})

def save_dataset(dataset, filename):
    with open(filename+".npy", 'wb') as pickle_file:
        pickle.dump(dataset, pickle_file, protocol=pickle.HIGHEST_PROTOCOL)
        pickle_file.close()


save_dataset(train_dataset, TRAIN_DATASET_FILENAME)
save_dataset(test_dataset, TEST_DATASET_FILENAME)
```

Zdrojový kód 2

Nahrání dat - funguje pomocí funkce *np.load().* Následuje rozdělení učící datové sady na učící a validační díl pomocí funkce *train_test_split()* ze stejné knihovny jako předchozí funkce, v poměru určeném v konstantě *VALIDATION_DS_SIZE*. Důležité je, že ponechává rozdělení tříd ve stejném poměru ve vytvořených podílech. Znovu míchá data.

```
train_dataset = np.load(TRAIN_DATASET_FILENAME + ".npy", allow_pickle=True)[()]
train_images = train_dataset["images"]
train_labels = train_dataset["labels"]
class_names = train_dataset["target_labels"].tolist()

test_dataset = np.load(TEST_DATASET_FILENAME + ".npy", allow_pickle=True)[()]
test_images = test_dataset["images"]
test_labels = test_dataset["labels"]

train_images, val_images, train_labels, val_labels = sk.train_test_split(
    train_images, train_labels, test_size=VALIDATION_DS_SIZE,
    random_state=random.seed())
```

Zdrojový kód 3

V trénovací sadě probíhá úprava dat. Vstupní obrázky byly načteny ve formátu RGB a nejprve bylo třeba provést normalizaci pixelů z intervalu $< 0; 255 >$ na interval $< 0; 1 >$. Dále se postup liší podle toho, zda jsou data augmentována či nikoli. Nejprve je popsán postup neaugmentovaných dat.

Pro práci v rámci TF dataset tenzoru, je třeba nahrát fotografie jinak. Použila se funkce *load_files_from_directory()*, následně proběhlo rozdělení do validační a učící sady a poté pomocí funkce *tf.data.Dataset.from_tensor_slices()* byly z dat vytvořeny tenzory. Poté byly nahrány obrázky pomocí funkcí *tf.io.read_file()* a *tf.image.decode_jpeg()* – ty pracují přímo s tenzory. Normalizace pixelů bylo dosaženo funkcí *tf.image.convert_image_dtype(image, tf.float32).* Spojení do štítků a obrázků bylo provedeno pomocí funkce *zip().* V průběhu kódu se objevuje konstanta *AUTOTUNE*, která optimalizuje alokaci výpočetních prostředků při paralelismu při iteraci objemných dat (jako jsou tenzory). Tato hodnota může být nastavena ručně, nebo ji *tf.data.experimental.AUTOTUNE*, nastaví automaticky za běhu.

```
def load_files_from_directory(path, shuffle=True):
    files = load_files(path, shuffle=shuffle)
    filenames = np.array(files['filenames'])
    target = np.array(files['target'])
    target_names = np.array(files['target_names'])
    return filenames, target, target_names

train_labels = tf.keras.utils.to_categorical(train_labels, CLASS_COUNT)
train_labels = tf.data.Dataset.from_tensor_slices(train_labels)
train_images = tf.data.Dataset.from_tensor_slices(train_images)
train_images = train_images.map(load_images_to_tensor_from_filepaths,
                                num_parallel_calls=AUTOTUNE)
train_dataset = tf.data.Dataset.zip((train_images, train_labels))
```

Zdrojový kód 4

Takto připravená datová sada před vstupem do modelu musí být ještě upravena. Na to slouží funkce *prepare_dataset().* Nejprve je datová sada načtena do (případně z) mezipaměti, poté zamíchána, nastaveno opakování a vytvořeny dávky. Opakování se nastavuje z toho důvodu, aby byla i poslední dávka plnohodnotná. Kdyby ne, učení skončí s chybou. Nakonec jsou dávky předběžně načteny do paměti pro rychlejší učení pomocí funkce *prefetch()*.

```
def prepare_dataset(dataset, cache=True, shuffle=True, shuffle_buffer_size=65000,
                    batch_size=32, autotune=1):
    if cache:
        if isinstance(cache, str):
            dataset = dataset.cache(cache)
        else:
            dataset = dataset.cache()
    if shuffle:
        dataset = dataset.shuffle(buffer_size=shuffle_buffer_size)
    dataset = dataset.repeat()
    dataset = dataset.batch(batch_size)
    dataset = dataset.prefetch(buffer_size=autotune)
    return dataset


train_dataset = prepare_dataset(train_dataset,
                                shuffle_buffer_size=TRAIN_IMAGE_COUNT,
                                batch_size=batch_size, autotune=AUTOTUNE)
val_ds = prepare_dataset(val_dataset, shuffle=False, batch_size=batch_size,
                         autotune=AUTOTUNE)
```

Zdrojový kód 5

Augmentace dat – Následující úpravy jsou pro každou datovou sadu jiné. Augmentace probíhá pouze u učící datové sady. Obrázky v datové sadě byly částečně předzpracovány. Otázkou je, zda to bylo vhodné. Všechny objekty leží v centru obrázku a konvoluční filtry se tak naučí jejich pozici a pro použití na reálných datech by model byl poté nepoužitelný. Pro dobrou generalizaci je třeba vstupní data upravit. Na každém

obrázku je provedeno několik náhodně vybraných operací zaručující rozmanitost vstupních dat.

Těmi jsou:

- Rotace o 15 stupňů,
- posunutí do šířky a výšky o 20 %,
- horizontální převrácení,
- přiblížení o 10 %,
- zakřivení o 5 %,
- úprava jasu v rozmezí 80–120 %.

Dalším krokem je inicializovat generátor. Následně jsou do generátoru inicializována data. Posunuté, otočené obrázky byly doplněny bílou barvou do pozadí.

```
train_ImageDataGenerator = ImageDataGenerator(rescale=1. / 255, rotation_range=15,
                                              width_shift_range=0.2,
                                              height_shift_range=0.2,
                                              horizontal_flip=True,
                                              zoom_range=0.1,
                                              shear_range=0.1,
                                              fill_mode="constant",
                                              cval=255,
                                              brightness_range=(0.8, 1.2), )

train_generator = train_ImageDataGenerator.flow(test_images, test_labels,
                                                batch_size=batch_size,
                                                #save_to_dir=AUGMENTED_IMAGES_DIR,
                                                # save_prefix='',
                                                # save_format='png',
                                                )
```

Zdrojový kód 6

V rámci testování rychlosti učení bylo dalším krokem převod dat do tenzoru, ve kterém byla dodávána do modelu, pomocí funkce *tf.data.Dataset.from_generator()*. Posledním krokem přípravy vstupních dat je předběžné načtení dávek do paměti *prefetch()*.

```
train_dataset = tf.data.Dataset.from_generator(lambda: train_generator,
                                               output_types=(
                                               tf.float32, tf.float32),
                                               output_shapes=(
                                                   [None, IMAGE_WIDTH,
                                                    IMAGE_HEIGHT,
                                                    IMAGE_CHANNELS],
                                                   [None, CLASS_COUNT]))

train_dataset = train_dataset.prefetch(AUTOTUNE)
```

Zdrojový kód 7

5.2.3 Inicializace modelu

Model je inicializován se všemi svými vrstvami a parametry. K vytvoření modelu je definována funkce *create_model()*. Následně pomocí *complie()* je model kompilován s optimizérem, ztrátovou funkcí a sledovanými (základními) metrikami. Modely jsou nastavovány ručně, proto je zde ukázka pouze jednoho modelu.

```
model_best = keras.Sequential([
    keras.layers.Conv2D(128, kernel_size=(5),
                        input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS),
                        name="conv1", padding="same",
                        kernel_regularizer=keras.regularizers.l2(0.01),
                        bias_regularizer=keras.regularizers.l2(0.01),
                        kernel_initializer='he_uniform',
                        bias_initializer='zeros', ),
    keras.layers.PReLU(shared_axes=[1, 2]),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(strides=2),
    keras.layers.Conv2D(128, kernel_size=(3), name="conv2", padding="same",
                        kernel_regularizer=keras.regularizers.l2(0.01),
                        bias_regularizer=keras.regularizers.l2(0.01),
                        kernel_initializer='he_uniform',
                        bias_initializer='zeros', ),
    keras.layers.PReLU(shared_axes=[1, 2]),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(strides=3),
    keras.layers.Flatten(),
    keras.layers.Dense(512, activation="relu"),
    keras.layers.Dropout(0.25),
    keras.layers.Dense(CLASS_COUNT, activation="softmax")
])


Adam = keras.optimizers.Adam(learning_rate=LEARNING_RATE, beta_1=MOMENTUM,
                             beta_2=0.999,
                             amsgrad=False)

def create_model(model):
    model.compile(optimizer=Adam,
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model

model = create_model(model_best)
```

Zdrojový kód 8

V modelech byly používány konvoluční vrstvy *Conv2D*. Za zmínku stojí parametr *padding*, kterým se uvádí, zda se má zachovat původní prostorové dimenze vstupní vrstvy, či ji zmenšit podle filtru (vrstva by se ořízla o zbytek po celočíselném podílu velikosti vrstvy a filtru). Při zachování stejných dimenzí se doplní na okraje nuly tak, aby zmíněný podíl byl celé číslo. *Flatten* vrstva serializuje 3D výstup do vektoru pro FC vrstvu.

5.2.4 Učící a testovací smyčka

Před samotným učením bylo potřeba alokovat paměť grafické karty ručně. Pokud se tak nestalo, občas proces učení skončil chybou „nedostatek paměti na grafické kartě“. Alokace byla zajištěna pomocí skriptu z oficiálních stránek [2] na začátku vlastního skriptu za importy.

Nejprve je spočten počet kroků za epochu pro každou datovou sadu. Dále jsou definovány *callback* funkce, které pracují přímo s modelem v průběhu učení. Byly použity funkce pro předčasné zastavení modelu, pokud se nezlepšuje validační přesnost nebo ztrátová funkce po dobu 25 epoch, uložení vah nejlepšího modelu podle ztrátové funkce i přesnosti na validační sadě. Funkce *fit()* spustí učení modelu s učící a validační sadou. Validace probíhá na konci každé epochy. Výsledky učení jsou ukládány do proměnné pro vizualizaci průběhu procesu.

```
training_epoch_steps = tf.math.ceil(TRAIN_IMAGE_COUNT / BATCH_SIZE).numpy()

es_acc = keras.callbacks.EarlyStopping(monitor='val_accuracy', mode='max',
                                       verbose=1,
                                       patience=25)
es_loss = keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1,
                                        patience=25)

mc_best_loss = keras.callbacks.ModelCheckpoint('runs\\best_model_loss.h5',
monitor='val_loss', mode=min, verbose=1, save_best_only=True)

mc_best_acc = keras.callbacks.ModelCheckpoint('runs\\best_model_acc.h5',
monitor='val_accuracy', mode=max, verbose=1, save_best_only=True)

history = model.fit(train_dataset, epochs=EPOCHS,
                    steps_per_epoch=training_epoch_steps,
                    validation_data=val_dataset,
                    validation_steps=val_epoch_steps,
                    callbacks=[mc_best_loss, mc_best_acc,
                               es_loss, es_acc])
```

Zdrojový kód 9

Naučený model lze vyhodnotit metodou *evaluate()* s testovací sadou dat. Výsledkem je hodnota ztrátové funkce a přesnost. Metoda *predict()* poté vrací softmax hodnotu poslední vrstvy sítě s pravděpodobností náležitosti do určité kategorie. Číslo kategorie lze získat funkcí *np.argmax()*. Obě funkce pro vyhodnocení modelu vrátí shodné výsledky na stejné datové sadě.

```
predicted_labels = model.predict(test_dataset, steps=test_epoch_steps)
predicted_labels = np.argmax(predicted_labels, axis=1)
predicted_labels = predicted_labels[:len(test_labels_original)]
test_labels_original = np.argmax(test_labels_original, axis=1)

score = model.evaluate(test_dataset, steps=test_epoch_steps)
```

Zdrojový kód 10

5.2.5 Ladění hyperparametrů

Ladění hyperparametrů probíhalo v cyklu s předem nastavenými parametry. Výsledky byly ukládány a poté vyhodnoceny. Modely byly nastaveny předem a poté probíhal cyklus učení a vyhodnocení s každým modelem po sobě. Původní odhad modelu byl vytvořen pomocí algoritmu Hyperband z knihovny *keras-tuner*. Definoval se model stejně jako dříve s tím rozdílem, že se zadal rozsah parametrů. Konvoluční blok byl definován následovně.

```
for i in range(hp.Int('num_layers', 1, 4)):
    model.add(Conv2D(filters=hp.Int('filters' + str(i),
                                    min_value=16,
                                    max_value=512,
                                    step=32,
                                    default=64),
                     kernel_size=(5), name="conv" + str(i + 1),
                     padding="same",
                     kernel_regularizer=keras.regularizers.l2(0.01),
                     bias_regularizer=keras.regularizers.l2(0.01),
                     kernel_initializer='he_uniform',
                     bias_initializer='zeros'))
    model.add(PReLU(shared_axes=[1, 2]))
    model.add(MaxPooling2D())
```

Zdrojový kód 11

Nejprve je definován rozsah vrstev, tedy 1 až 4, poté filtry konvoluční vrstvy, 16 - 512 s rozestupy 32. Vrstvy i filtry parametrů musí mít unikátní jméno. K dispozici jsou

i jiné parametry než jen celé číslo, to jsou výběr ze zadaných hodnot či desetinné číslo.

Před zahájením učení musí být inicializován *tuner* s maximálním počtem epoch a optimalizační úlohou. V tomto případě maximalizace validační přesnosti. Metoda *search()* poté na základě dat spustí učení. V důsledku pouští metodu *fit()*.

```
tuner = Hyperband(
    create_model,
    max_epochs=40,
    objective='val_loss',
    executions_per_trial=3,
    directory='hyperband',
    project_name='360Fruits'
)

tuner.search_space_summary()

tuner.search(train_dataset, steps_per_epoch=training_epoch_steps,
             validation_data=val_dataset, validation_steps=val_epoch_steps)

tuner.results_summary()
```

Zdrojový kód 12

5.2.6 Metriky a vizualizace výsledků

Výsledky byly zpracovávány z metody *predict()* vysvětlené výše. Jako první byly nahrány do modelu nejlepší váhy. Dále byly převedeny získané výsledky z formátu pravděpodobnosti na čísla kategorie s nejvyšší pravděpodobností funkcí *np.argmax()*. Pravdivé hodnoty byly uloženy již při nahrání modelu. Je třeba sjednotit počet hodnot tak, že se zkrátí predikovaná data na počet původních testovacích. Predikováno bylo více hodnot z důvodu dodržení velikosti poslední dávky, pokud byla menší než nastavená hodnota, byla doplněna o data z učící sady.

```
predicted_labels = np.argmax(predicted_labels, axis=1)
predicted_labels = predicted_labels[:len(test_labels_original)]
test_labels_original = np.argmax(test_labels_original, axis=1)
```

Zdrojový kód 13

Následuje vykreslení matice změn pro všechny kategorie, nejprve absolutně a poté jsou data normalizovaná a uložena. Dále je spočtena řada metrik a výstup také uložen. Pro spočtení metrik je používán balíček *sklearn.metrics*.

```
cm = confusion_matrix(test_labels_original, predicted_labels)
cm_norm = confusion_matrix(test_labels_original, predicted_labels,
normalize="true")
figure = print_confusion_matrix(cm, target_labels,
                                (CLASS_COUNT, CLASS_COUNT))
plt.savefig("CM-" + str(key) + ".png")
figure = print_confusion_matrix(cm_norm, target_labels,
                                (CLASS_COUNT, CLASS_COUNT),
                                normalized=True)
plt.savefig("CM_norm-" + str(key) + ".png")

f1 = metrics.f1_score(test_labels_original, predicted_labels, average="weighted")
classification_report = metrics.classification_report(test_labels_original,
                                                      predicted_labels,
                                                      target_names=target_labels,
                                                      output_dict=True)
precision_score = metrics.precision_score(test_labels_original, predicted_labels,
                                          average="weighted")
recall_score = metrics.recall_score(test_labels_original, predicted_labels,
                                    average="weighted")
predicted_labels = tf.keras.utils.to_categorical(predicted_labels,
                                                 CLASS_COUNT)
test_labels_original = tf.keras.utils.to_categorical(test_labels_original,
CLASS_COUNT)
roc_auc_score = metrics.roc_auc_score(test_labels_original, predicted_labels,
                                      average="weighted",
                                      multi_class="ovo")

metrics_array = [f1, classification_report,
                 precision_score,
                 recall_score,
                 roc_auc_score]
```

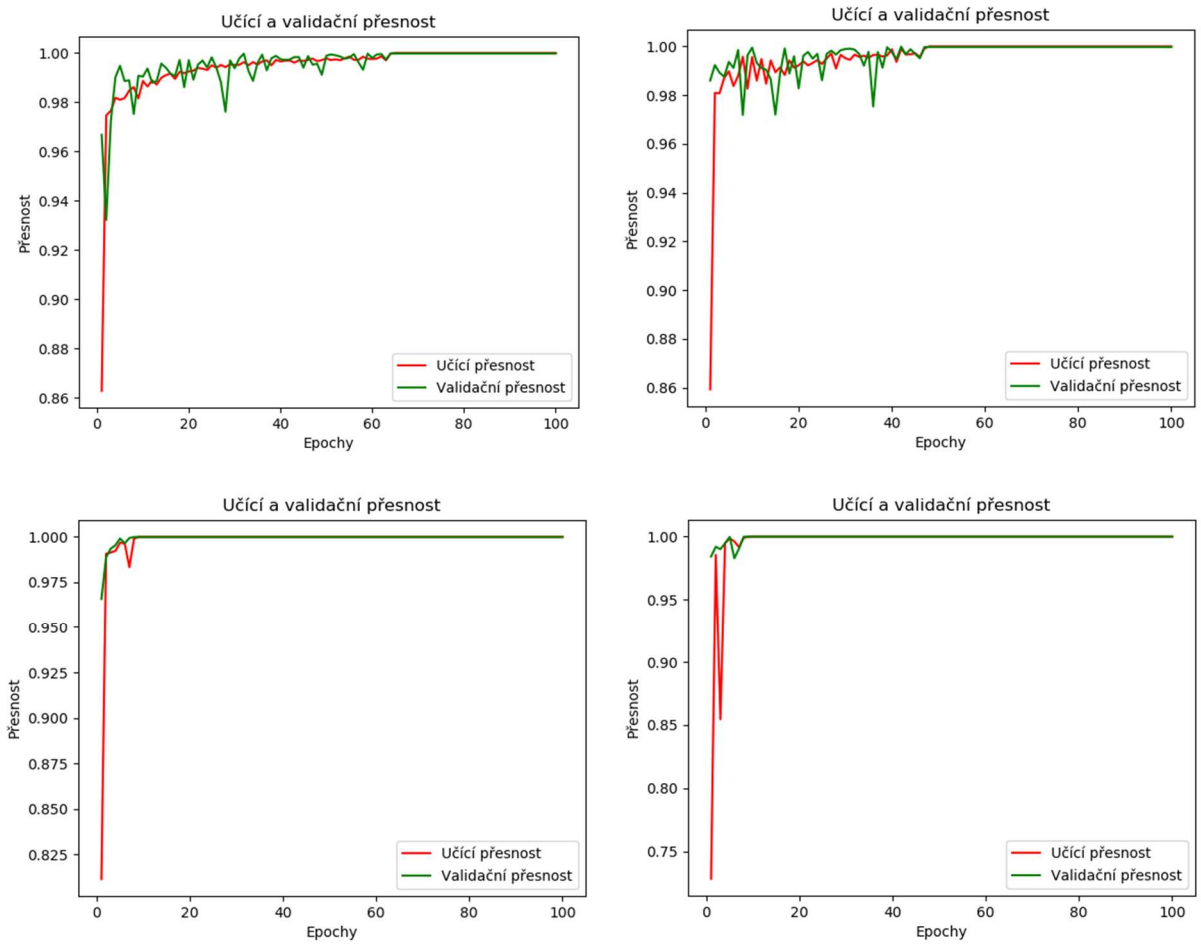
Zdrojový kód 14

5.3 Vyhodnocení

V této kapitole byly vyhodnoceny výsledky získané během experimentů. Byl popsán nejlepší model, zjištěn vliv daných parametrů na datovou sadu Fruits-360 a určena nejvýhodnější kombinace parametrů modelu. Na závěr je práce porovnána s podobnými pracemi v této oblasti výzkumu.

5.3.1 Velikost dávky

Byly testovány velikosti dávky 16, 32, 64, 128, 160, 256, 320 a 512. V souladu s teorií byla pozorována plynulejší a rychlejší konvergence ztrátové funkce pro větší dávky, jak lze vidět na grafech níže. Jsou zobrazeny grafy pro dávky 32, 64, 256 a 512.



Graf 2 Průběh učení podle dávek (horní řada 32 a 64, spodní řada 256 a 521)

Zdroj: Vlastní zpracování

S dávkou 16 se několikrát nepodařilo dojít konvergence. Bylo dosahováno vyšší úspěšnosti na testovací sadě s dávkou 32 oproti 512 až o 2 %. Konstantní část grafu však naznačuje rychlé přetrénování. Proto byl experiment proveden znovu s aplikací normalizačních technik – dropout a dávkové normalizace (později Model CD). Výsledná

úspěšnost dávky 32 zůstala lepší než u vyšších dávek s delší dobou učení, proto jsou dále v práci experimenty prováděny s dávkou 32.

5.3.2 Architektura sítě

Jako první byly testovány 4 zmíněné architektury modelů. Trénování bylo nastaveno na 100 epoch s předčasným ukončením, pokud se nebude snižovat validační ztráta, nebo zvyšovat validační přesnost po více než 25 epoch. Tento předpoklad vychází z prvotního spuštění modelu na 250 epoch, kdy se průměrně po 75 epochách již stav neměnil. Parametry jsou stejné jako v kapitole 5.1, míra učení byla nastavena na 0,001 a momentum na 0,9.

Tabulka 5 obsahuje výsledky experimentu. Jak lze vidět, nejlepších výsledků dosáhl model 4, pouze o 0,17 % nad modelem 1. Ostatní modely jsou minimálně o 2 % horší. Odpovídají tomu i ostatní sledované metriky. Z tohoto důvodu se dále v experimentech pokračuje s modelem 1 a 4. Zajímavé je, že podobných výsledků dosáhly rozdílné počty konvolučních bloků. Tato skutečnost je dále zkoumána v dalších částech této práce. Graf přesnosti a ztrátové funkce lze najít níže.

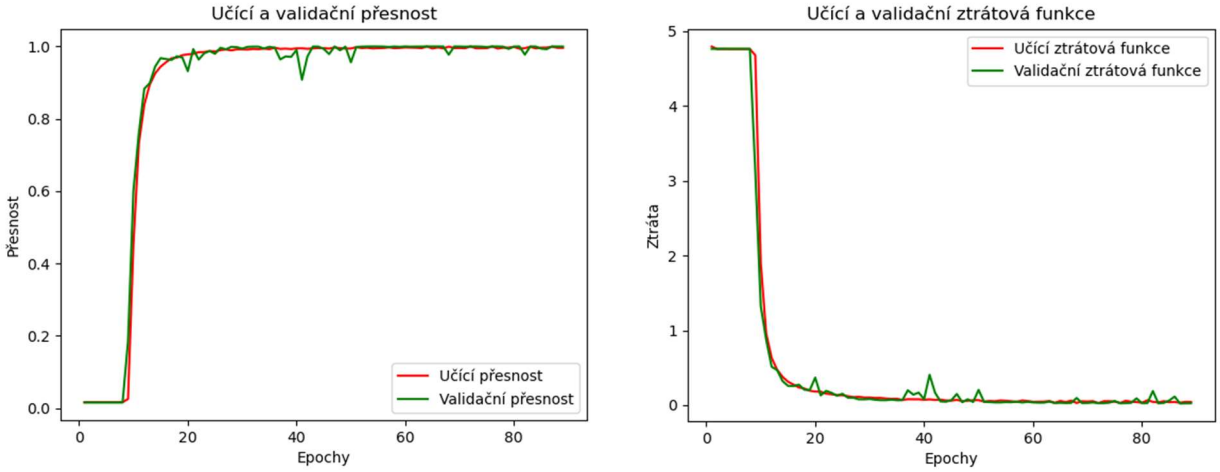
	Model 1	Model 2	Model 3	Model 4
Přesnost	94,51 %	90,99 %	92,66 %	**94,68 %**
F1-skóre	0,9437	0,9087	0,9254	**0,9457**
Preciznost	0,9520	0,9173	0,9349	**0,9532**
Citlivost	0,9451	0,9098	0,9264	**0,9467**

Tabulka 6 Výsledky experimentu s architekturou modelu

Zdroj: Vlastní zpracování

Model 4 měl největší problém rozeznat třídu sladkých brambor, kdy ji určil správně v méně než 50 % případů. Zaměňoval ji za cibuli, červenou bramboru, květák a zázvor. Model 1 nejhůře (též pod 50 %) rozpoznával borůvky, které měnil za granátové

jablko. Model 1 měl na rozdíl od modelu 4 problém rozeznat mezi sebou i jednotlivé druhy jablek (60 %), jak bylo naznačeno v původní práci k datové sadě [32].



Graf 3 Průběh učení modelu 4

Zdroj: Vlastní zpracování

5.3.3 Vliv velikosti konvolučních jader

Dále byla testována velikost jádra. U modelu 1 i 4 nejprve u všech vrstev z 5 na 3 (varianta 1) a poté z 5 na 3 v druhé polovině modelu (varianta 2). Ostatní parametry modelu zůstaly stejné. Výsledky byly zaneseny do tabulky níže. Vzhledem ke snižování dimenze obrazu během modelu, je předpoklad zvýšení úspěšnosti modelu s variantou 2.

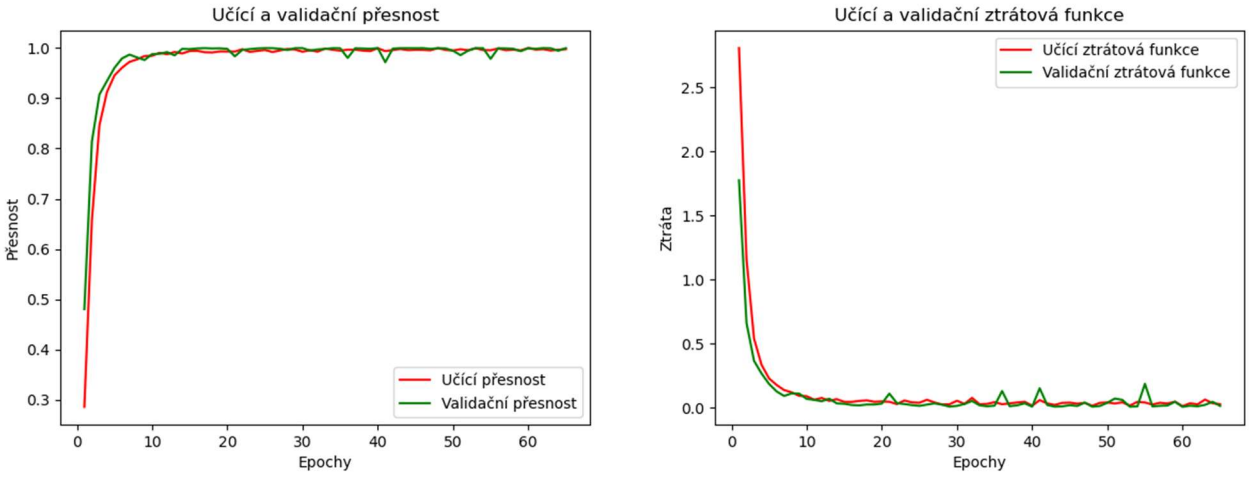
	Model 1			Model 4		
Velikost jader	5	5/3	3	5	5/3	3
Přesnost	94,51 %	**95,81 %**	95,77 %	94,68 %	91,26 %	**95,05 %**
F1-skóre	0,9437	**0,9575**	0,9562	0,9457	0,9102	**0,9494**
Preciznost	0,9520	**0,9628**	0,9612	0,9532	0,9176	**0,9548**
Citlivost	0,9451	**0,9580**	0,9576	0,9467	0,9125	**0,9504**

Tabulka 7 Výsledky změny velikosti konvolučních jader v modelu 1 a 4

Zdroj: Vlastní zpracování

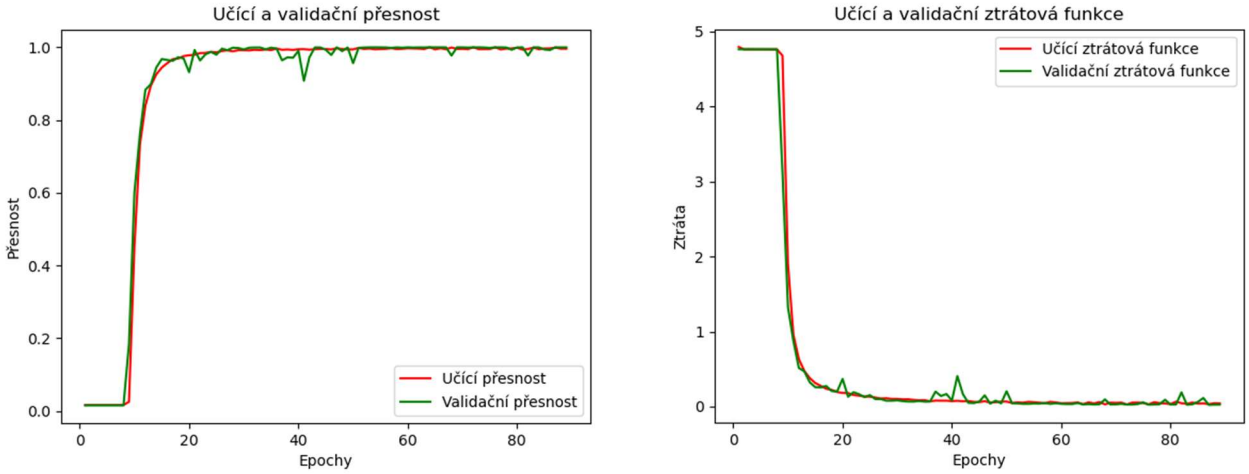
V rámci modelu 1 dopadla srovnatelně varianta 1 i 2 s mizivým rozdílem 0,04 %. Obě varianty byly úspěšnější než původní testovaná síť s velikostí jádra 5. Jinak tomu bylo u modelu 4, kdy kombinovaný počet jader byl o 3 % méně úspěšný než původní velikost. Testovaná varianta 2 měla úspěšnost vyšší o 0,37 %. Kombinace velikosti jader ve variantě 1 u modelu 1 je pravděpodobně více citlivá právě kvůli menšímu počtu

konvolučních vrstev. Model 1 byl v nejúspěšnější variantě o 0,76 % lepší než model 4. Dále v experimentech je používán model 1 ve variantě 1. Pro občasné porovnání směru vývoje experimentů jsou použity i model 1 ve variantě 2 a model 4 ve variantě 2. Na grafu níže lze vidět vývoj učení modelu 1 varianty 1 ve srovnání s modelem 4 varianty 2, na kterých je vidět optimální křivka učení bez větších extrémů nebo výkyvů.



Graf 5 Průběh učení modelu 1 s velikostí konvolučních jader 5 a 3

Zdroj: Vlastní zpracování



Graf 4 Průběh učení modelu 4 s velikostní konvolučních jader 3

Zdroj: Vlastní zpracování

5.3.4 Vliv konvoluční vrstvy 1x1

Tento experiment je označován se sufixem A. Jedná se o přidání konvoluční vrstvy o stejném počtu filtrů jako předchozí konvoluční vrstva s jádrem 1. Tato struktura se objevuje při práci s RGB barevným schématem [8]. Parametry modelu zůstávají stejné. Testovány byly dva nejúspěšnější modely z předešlého experimentu – model 1 A varianta

1, dále označován jen jako model 1 A, a model 4 A varianta 2, dále jako model 4 A. Vzhledem k výsledkům v tabulce níže byl testován i model 1 s velikostí jádra 3.

	Model 1 A		Model 4 A
Velikost jader	5/3	3	3
Přesnost	**92,57 %**	75,46 %	86,06 %
F1-skóre	**0,9248**	0,7462	0,8564
Preciznost	**0,9310**	0,7883	0,8718
Citlivost	**0,9257**	0,7546	0,8605

Tabulka 8 Výsledky přidání konvoluční vrstvy s jádrem 1

Zdroj: Vlastní zpracování

U všech testovaných modelů se objevuje stejný trend poklesu úspěšnosti nezávisle na počtu jader, a to v rozmezí 3 až 20 %. V matici změn není pozorována zásadní změna chybně klasifikovaných tříd u modelu 1 varianty 1. Model 4 A si oproti své původní variantě velmi pohoršil v rozpoznávání druhů jablek mezi sebou. Z těchto důvodů nebude varianta A použita dále v experimentech.

5.3.5 Vliv nahrazení sdružovacích vrstev

Podle N. Muhammad a spol. [26], je testována výměna sdružovacích vrstev za konvoluční vrstvu s jádrem 1 a posunem o velikosti jádra sdružovací vrstvy. Redukce dimenze rozměrů zůstane stejná a to by mělo mít pozitivní vliv na přesnost modelu či alespoň by se model neměl zhoršit. Ostatní parametry modelu zůstávají stejné. Model 1 byl testován s posunem druhé sdružovací vrstvy 2 i 3 v důsledku pouze dvou sdružovacích vrstev a prostoru pro větší redukci dimenze na poslední konvoluční vrstvě. Jedná se o variantu modelu B. Vzhledem k výsledkům v tabulce níže, které neodpovídávají předpokladu alespoň o zachování stejné úspěšnosti, byl testován i model 1 s velikostí jádra 3.

	Model 1 B		Model 4 B
Velikost jader	5;3	3	3
Přesnost	92,07 %	91,84 %	87,45 %
F1-skóre	0,9190	0,9171	0,8762
Preciznost	0,9285	0,9294	0,8864
Citlivost	0,9207	0,9183	0,8790

Tabulka 9 Výsledky nahrazení sdružovací vrstvy konvoluční vrstvou (varianta B)

Zdroj: Vlastní zpracování

Jak bylo naznačeno, výsledky neodpovídají původnímu předpokladu o nahrazení sdružovacích vrstev konvolučními bez snížení úspěšnosti ani u jednoho z testovaných modelů. Úspěšnost se snížila o 3 až 9 %. V matici změn je viditelná změna oproti předchozímu modelu 4, kdy vzniká problém při rozpoznávání tříd jablek a jen s pravděpodobností 0,3 rozezná nektarinku. Proto nebude tato varianta používána dále v experimentech.

5.3.6 Vliv normalizace

V rámci normalizace bylo experimentováno s variantou C (dávková normalizace), D (dropout) a jejich kombinací. Vstupními modely jsou – model 1 s oběma variantami (jádra 5,3 i 3), jelikož doposud měl nejlepší výsledky, a model 4 s velikostí jádra 3. Ostatní parametry zůstávají nezměněné. Jako první byl proveden experiment C, přidání dávkové normalizace pro každou vrstvu sítě. Podle X. Li [55] je důležité pořadí ve kterém bude normalizace aplikována. Proto je umístěna za aktivační funkci. Pořadí vrstev je tedy konvoluční – aktivační (PReLU) – normalizační – sdružovací.

	Model 1 C		Model 4 C
Velikost jader	5;3	3	3
Přesnost	95,82 %	96, 43 %	**96,82 %**
F1-skóre	0,9572	0,9638	**0,9674**
Preciznost	0,9638	0,9692	**0,9721**
Citlivost	0,9581	0,9642	**0,9681**

Tabulka 10 Výsledky dávkové normalizace (varianta C)

Zdroj: Vlastní zpracování

Dle výsledků v tabulce výše je zlepšení téměř nepatrné u modelu 1 C varianty 1. U varianty 2 se jedná o zlepšení o 0,7 % a u modelu 4 C je zlepšení 1,8 %. Na matici změn lze pozorovat změnu trendu u modelu 4 C, kde se objevil problém s rozpoznáváním tříd jablek mezi sebou, záměnnou borůvek za modré hroznové víno. Záměna brambor zůstala stejná. Ostatní kategorie si ve skóre polepšily.

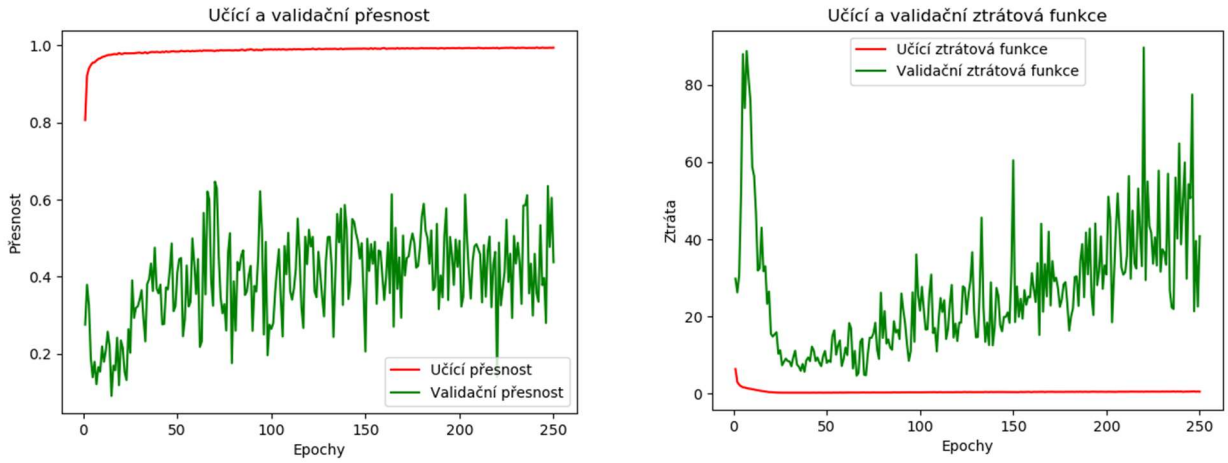
Dále byl proveden experiment s dropout vrstvou přidanou za každou konvoluční vrstvu podle A. Krizhevsky [54]. Pořadí vrstev je doporučeno konvoluční – aktivační (PReLU)– sdružovací– dropout. Dropout vrstva byla přidána i mezi FC vrstvy. Parametr míra zahození byl nastaven na 0,2 na vstupní vrstvě, 0,25 na následujících a 0,5 na všech plně propojených. Výsledek lze vidět v tabulce níže.

	Model 1 D		Model 4 D
Velikost jader	5;3	3	3
Přesnost	**97,22 %**	94,65 %	95,51 %
F1-skóre	**0,9704**	0,9435	0,9544
Preciznost	**0,9754**	0,9520	0,9607
Citlivost	**0,9698**	0,9436	0,9551

Tabulka 11 Výsledky normalizační techniky zahazování (varianta D)

Zdroj: Vlastní zpracování

Dropout vrstva zlepšila přesnost modelu 1 D ve variantě 1 o 1,4 %, na 97,22 %. Varianta 2 se o 1,1 % zhoršila. To je překvapující výsledek, protože obě varianty modelu 1 se chovaly podobně. Pokus byl opakován několikrát vždy se stejným výsledkem. Model 4 D se zlepšil o 0,5 % na 95,51 %. Dále budou testovány všechny modely kvůli rozdílným výsledkům při variantě C a D. Při experimentu s dávkovou normalizací a zároveň dropout vrstvou po každé konvoluční vrstvě v pořadí konvoluční – aktivační (PreLU) – normalizační – sdružovací - dropout, jak uvádí X. Li [55], se schopnost generalizace rapidně zhoršila na kolísavých 50 % přesnosti oproti použití pouze dávkové normalizace. Průběhy jsou vidět na grafech níže. Pro tento specifický případ byl při experimentu nastaven počet epoch na 250.



Graf 6 Průběh učení modelu při aplikaci normalizační techniky zahazování v kombinaci s dávkovou normalizací v celé síti

Zdroj: Vlastní zpracování

Tento nesoulad dvou normalizačních technik se děje z toho důvodu, že dropout posunul rozptyl konkrétní neurální jednotky, když byl přenášen stav tohoto neuronu z učícího datového setu na testovací. Dávková normalizace by však ve své testovací fázi zachovala svůj statistický rozptyl, který byl akumulován během celého postupu učení. Nekonzistence rozptylu (variační posun) způsobuje nestabilní numerické chování inferencí, které nakonec vede k chybným předpovědím při aplikaci Dropout před BN [55].

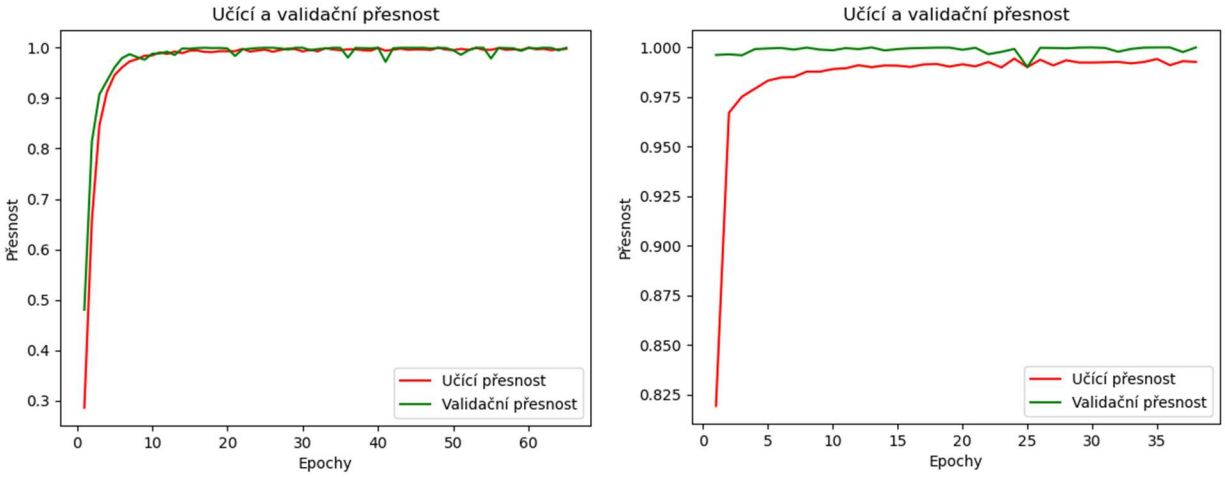
Při aplikaci doporučení ze stejné studie, že sloučení těchto dvou normalizačních technik je možné bez ztráty úspěšnosti pouze při aplikování dropout vrstvy až za všechny vrstvy dávkové normalizace, bylo dosaženo výsledků prezentovaných v tabulce 12. Byly testovány dvě dropout vrstvy, jedna za posledním konvolučním blokem a druhá za FC vrstvou s mírou zahazování 0,5 a 0,5, později 0,25 a 0,5. Ani jedna varianta nepřinesla zlepšení. Byla tedy testována pouze jedna dropout vrstva za FC vrstvou. Míra zahazování byla nastavena na 0,25. V rámci ladění tohoto parametru byly modely testovány i s hodnotou 0,15; 0,35; 0,5. Výsledky byly u všech modelů o 0,5 % až 1 % nižší než prezentované výsledky.

	Model 1 CD		Model 4 CD
Velikost jader	5;3	3	3
Přesnost	**98,09 %**	95,11 %	95,78 %
F1-skóre	**0,9808**	0,9503	0,9577
Preciznost	**0,9826**	0,9565	0,9649
Citlivost	**0,9809**	0,9510	0,9574

Tabulka 12 Výsledky kombinace normalizačních technik zahazování a dávkové normalizace (varianta C+D)

Zdroj: Vlastní zpracování

Kombinace varianty C a D přinesla doposud nejvyšší úspěšnost, a to u modelu 1 CD varianty 1, 98,09 %. U zbylých dvou modelů tato varianta nepřinesla zásadní zlepšení úspěšnosti oproti samotné variantě C i D. Pro další experimenty byla použita tato varianta modelů. Jak lze vidět na grafu 7, normalizace přinesla lepší generalizaci, kdy úspěšnost na validační sadě je vyšší než na trénovací.



Graf 7 Průběh učení před (levo) a po (pravo) aplikaci normalizačních technik zahazování a dávkové normalizace (varianty C+D

Zdroj: Vlastní zpracování)

5.3.7 Vliv aktivační funkce

Jedním z posledních experimentů byla editace aktivační funkce. Podle doporučení od S.-H. Wang a Y. Chen [42] byly použity PReLU aktivační funkce. PReLU se sdílenými parametry měla o 0,3 až 0,5 % lepší přesnost než PReLU s nesdílenými parametry u všech testovaných. Proto je dále hovořeno o PReLU se sdílenými parametry. Byla testována standardní aktivační funkce ReLU, a dle H. Mureşana [32] i ELU, která by měla mít pozitivní efekt až od 5 konvolučních vrstev a více. Tyto předpoklady se potvrdily dle výsledků v tabulce níže.

	Model 1 CD		Model 4 CD
Velikost jader	5;3	3	3
ELU	90,39 %	94,18 %	**95,94 %**
PReLU	**98,09 %**	**95,11 %**	95,78 %
ReLU	92,34 %	93,32 %	93,41 %

Tabulka 13 Výsledky změny aktivační funkce

Zdroj: Vlastní zpracování

Ve všech variantách modelu 1 měla nejlepší úspěšnost PReLU aktivační funkce. Potvrdil se předpoklad zlepšení při 5 a více konvolučních vrstvách za použití ELU aktivační funkce. Dále je pracováno s nejlepšími variantami. FC vrstva má aktivační funkci ReLU, testována byla i PReLU a lineární aktivační funkce, kde byl propad v úspěšnosti o několik procent ve všech modelech a variantách.

5.3.8 Vliv augmentace dat

Posledním experimentem byla augmentace dat, kdy byly použity doposud nejlepší získané modely. Augmentace obsahuje rotaci, posun, úpravu jasu, zakřivení, horizontální rotaci a přiblížení. Vždy náhodně vybrané z předem stanoveného rozmezí. Původní data jsou těmito nahrazena. Počet dat tedy zůstává stejný. Na obrázku lze vidět ukázku augmentovaných dat.

Obrázek 17 Ukázka augmentované datové sady [32]

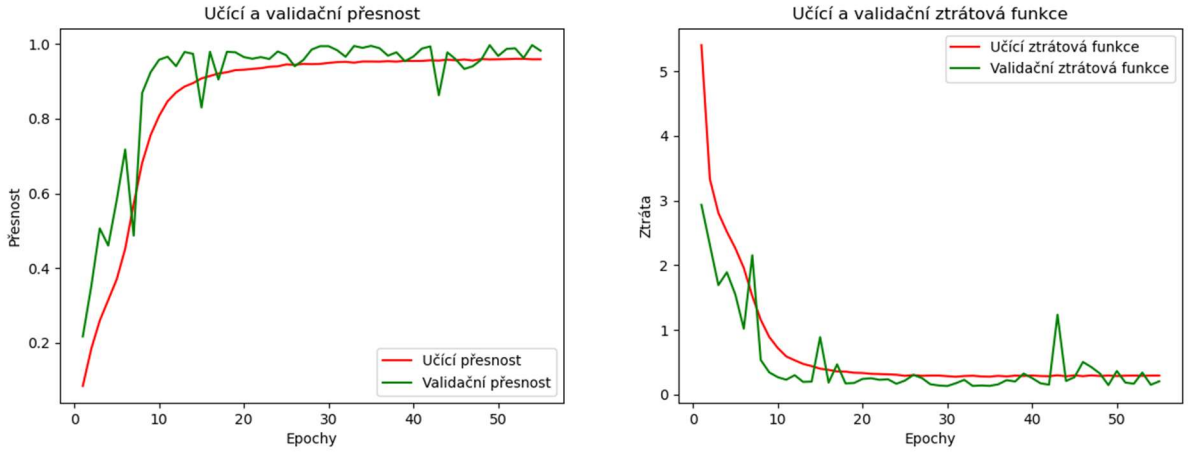
Zdroj: Vlastní zpracování

	Model 1 CD		Model 4 CD
Velikost jader	5;3	3	3
Přesnost	96,27 %	96,02 %	92,89 %
F1-skóre	0,9622	0,9600	0,9226
Preciznost	0,9705	0,9663	0,9528
Citlivost	0,9627	0,9602	0,9289

Tabulka 14 Výsledky augmentace dat

Zdroj: Vlastní zpracování

Dle výsledků má tato augmentace negativní vliv na všechny modely, a to o 1 až 4 %. Dle matice změn augmentace prohloubila rozdíly u několika již předem problematických tříd a zlepšila spoustu ostatních. Model 1 například lépe rozezná několik druhů jablek mezi sebou, nicméně stejně tak jeden druh plně mění za jiný. Stejně tak zmíněné brambory, kde míra jejich rozpoznání klesla na 0,13. Na grafech lze vidět, že i augmentace dat přinesla lepší generalizaci, kdy validační hodnoty jsou lepší než trénovací.



Graf 8 Průběh učení na augmentovaných datech modelu 1 CD varianta 1

Zdroj: Vlastní zpracování

6 Výsledky

Výsledky této diplomové práce jsou jednak teoretické, ale i praktické. Byla provedena rešerše odborné literatury zabývající se rozpoznáváním ovoce a následně na základě zjištěných informací navržen a realizován ukázkový model.

6.1 Teoretické výsledky

Byla provedena obsáhlá rešerše literatury, celkem bylo využito 51 zdrojů, které se věnují strojovému učení, 23 z nich pak přímo rozpoznávání ovoce, viz tabulka na str. 43. Podařilo se zjistit, jaké jsou nejčastější používané techniky, jaké datové sady je možné využít, s jakými obtížemi se doména ovoce (a konkrétní aplikace) potkává, jaké řešení autoři navrhují, jak lze dané řešení implementovat a jaké jsou výsledky. Klasické metody strojového učení používalo 8 prací. Nejlepší přesnost měl model s klasifikací pomocí SVM na laboratorních datech, téměř 100 %. Zbylých 15 používalo konvoluční modely. Ty dosahovaly v průměru vyšších přesností. Nejlepší model měl přesnost 99,99 % na obrázcích z reálného prostředí.

To vedlo k rozhodnutí realizovat model strojového učení jako konvoluční neuronovou síť na obsáhlé, veřejně dostupné datové sadě, která byla použita i třemi zkoumanými modely. V některých případech nebyly uvedeny všechny informace o datové sadě, a bylo tak pracováno se vším, co bylo dostupné. Na základě informací zjištěných v teoretické části byla navržena architektura modelů, postup experimentů, ladění parametrů a metriky pro vyhodnocení úspěšnosti.

6.2 Praktické výsledky

K vytvoření ukázkové aplikace bylo navrženo několik architektur konvolučních neuronových sítí na základě velikosti vstupních dat a předchozích prací v oblasti rozpoznávání ovoce. Rozdíly byly v počtu konvolučních bloků, velikosti konvolučních jader a velikosti plně propojené vrstvy.

Modelem s nejlepšími výsledky byl Model 1 CD varianta 1 s 98,09% přesností na testovací datové sadě. Model má dva konvoluční bloky a jednu plně propojenou vrstvu. Konvoluční bloky obsahují dávkovou normalizaci za aktivační funkcí PReLU. První s konvolučním jádrem 5 a sdružovací vrstvou o velikosti 2, druhý s konvolučním jádrem 3 a sdružovací vrstvou o velikosti 3. Plně propojená vrstva má aktivační funkci ReLU. Následuje ji dropout vrstva a poté výstup v podobě softmax vrstvy. Druhým

nejlepším modelem byl Model 4 C s 96,82% přesností, třetím pak Model 1 ve variantě C (96,43 %), stejně jako Model 4.

Jako první bylo testováno několik variant velikosti dávky, aby se uzpůsobilo testování všech následujících modelů podle nejlepších dosažených výsledků. Ty měla dávka 32, v souladu s teorií. Větší dávky měly menší míru generalizace a podléhaly rychlejší konvergenci. S menší dávkou se nepodařilo dojít konvergence.

Druhý experiment se zabýval velikostí konvolučního jádra. Ta měla zásadní vliv na další vývoj modelů. Prvně bylo viditelné zlepšení u modelů v průměru pouze o procento, během normalizace však velikost jádra měla velký vliv. Především u modelu 1, kde kombinace velikosti jader 5 a 3 měla vyšší přesnost o 3 % než velikosti 3 a 3. Před normalizací byl tento rozdíl pouze 0,04 %.

Další experiment se věnoval přidání konvoluční vrstvy o velikosti jádra 1 za již používanou konvoluční vrstvu, a to z důvodu většího pochopení barevného spektra. Tento předpoklad se v experimentech ukázal jako mylný. Modely s touto vrstvou navíc měly menší přesnost o několik procent.

Dalším experimentem byla výměna sdružovacích vrstev za vrstvy konvoluční s jádrem 1 a posunem stejným jako u nahrazovaných vrstev. Předpokládalo se zlepšení, nebo alespoň zachování dosažené přesnosti se sdružovacími vrstvami. I tento předpoklad byl mylný. Pokles přesnosti byl v jednotkách procent.

Poslední úpravou architektury byla normalizace. Jednalo se o přidání dávkové normalizace a dropout vrstvy. Nejprve jedno po druhém a poté jejich kombinace. Model 1 C s velikostí jader 3 a Model 4 dosáhly s dávkovou normalizací doposud nejlepších výsledků, 96,43 % a 96,82 %. Poté bylo testováno přidání pouze dropout vrstvy za každou konvoluční i plně propojenou vrstvu. Krom Modelu 1 s velikostí jader 5 a 3, který měl přesnost 97,22 %, modely nedosáhly úspěchů jako s dávkovou normalizací. Následovalo přidání dropout vrstvy spolu s dávkovou normalizací. To vyvolalo ztrátu přesnosti až na kolísavých 50 %. Tento problém je zdokumentován v odborné literatuře a byl vyřešen přidáním dropout vrstvy až za všechny dávkové normalizace. Byly tedy přidány dvě vrstvy před poslední dvě plně propojené vrstvy. Výsledky nebyly uspokojivé, v průměru o 1 % nižší než zatím nejlepší dosažené výsledky. Proto bylo experimentováno s parametrem zahazování a přidáním pouze jedné dropout vrstvy před výstupní vrstvu. Poslední zmíněná úprava spolu s laděním míry zahazování způsobila navýšení přesnosti

o 0,7 % u Modelu 1, tedy na 98,09% přesnost. Ostatní modely na tom byly lépe ve variantě C, tedy pouze s dávkovou normalizací.

S nejlepšími třemi modely proběhl experiment změny aktivační funkce z PReLU na ReLU a ELU. Model 4 benefitoval z ELU díky 5 konvolučním vrstvám. V teorii se uvádí, že ELU je vhodnější používat pro hluboké sítě, tj. 5 a více skrytých vrstev. PReLU měla vyšší přesnost než ReLU o několik procent za cenu učící vrstvy navíc. Jako poslední byl zkoumán faktor augmentace dat, který nezlepšil přesnost modelu na dané testovací datové sadě, ale zlepšil generalizaci modelu.

Model 1 CD varianta 1, v porovnání s původním modelem uvedeným k datové sadě, měl lepší přesnost o 2 %. Oproti původnímu modelu, kde byly referovány potíže s rozpoznáváním jednotlivých druhů jablek mezi sebou, má největší problém s rozpoznáním druhů brambor, které zaměňuje mezi sebou, a nektarinek, které zaměňuje za broskve. Nižší míra rozpoznání než 0,99 je pouze u 3 druhů jablek z 16.

7 Závěry a doporučení

V této práci byly popsány základní přístupy ke strojovému učení, zvláště pak dosavadní metody strojového vidění a rozpoznávání obrazu. Rozebrány byly state-of-the-art modely a techniky pro rozpoznávání ovoce Doména strojového vidění je velmi komplexní, stejně tak i její podmnožina rozpoznávání ovoce. Má různé aplikace použití, kde se každá vyznačuje mírně odlišnými požadavky, funkcionalitou, provedením a omezeními. Používá se jak klasifikace, tak segmentace či detekce a sčítání objektů. Hlavním cílem bylo zhodnocení možností využití metod a technik strojového učení při identifikaci jednotlivých druhů ovoce z fotografií, dále pak navrhnout a vytvořit ukázkový model za použití technik a metod s nejlepšími výsledky.

Doposud se v praxi používají dva hlavní přístupy pro identifikaci ovoce z fotografie. Z průzkumu odborné literatury vyplývá, že použití klasických metod strojového učení začíná ustupovat a místo nich se používají konvoluční neuronové sítě, které mají lepší výsledky. Je to zapříčiněno výkonnějším hardwarem a obsáhlejšími datovými sadami, jejich náročnost na ruční extrakci charakteristik prudce roste při každé změně. Vstupní data by měla být stále normalizována, není však potřebné extrahovat charakteristiky ručně díky skrytým vrstvám, které tyto charakteristiky extrahují a vyhodnocují autonomně.

Model navržený v této práci vychází z těchto zjištění a staví na konvoluční neuronové síti. Jako vstupní data byla použita již vytvořená datová sada obsahující 81 tisíc obrázků (převážně) ovoce ve 120 kategoriích. Tato datová sada obsahuje z části normalizované fotografie (se zredukovanou fixní velikostí) a odstraněné pozadí. Bylo experimentováno s architekturou konvoluční neuronové sítě dle informací zjištěných z průzkumu literatury. Síť byla učena celá, od začátku. Nebylo použito předučených modelů. V experimentech bylo pokračováno s modelem s nejvyšší přesností. U této architektury byly laděny hyperparametry a kontrolovány s dalšími dvěma modely. Experimenty s některou kombinací parametrů nevedly ke zlepšení modelu, nýbrž k zhoršení jeho přesnosti. Jiná kombinace pro změnu fungovala lépe na jiném modelu než na doposud nejlepším. Proto byly použity při experimentech 3 varianty. Výsledný model této práce měl v rozmezí jednoho až dvou procent srovnatelnou přesnost se state-of-the-art algoritmy trénovanými na použité datové sadě. Měl lepší výsledky (98,09 %) než síť vyvinutá autory datové sady (96,13 %). Model tedy splnil předpoklad o vysoké přesnosti identifikace ovoce.

Jako první doporučení autora této práce výzkumníkům v oblasti rozpoznávání ovoce je poskytnutí souborů pro datovou sadu v originálním stavu (s nezměněnou velikostí a pozadím), aby se data stala univerzálnějšími pro další použití. V oblasti rozpoznávání ovoce se jedná o doposud nejobsáhlejší datovou sadu a toto je její značné omezení.

Dalším doporučením je zajištění dostatečně výkonného stroje pro učení, nejlépe několika strojů pro učení více variant modelů najednou. Obdobou by mohlo být distribuované učení za použití více grafických karet nebo stanic pro zrychlení procesu učení.

Výzkum může pokračovat v několika směrech. Zaprvé v prozkoumávání dalších architektur modelu, například by šlo použít VGG, ResNet či Inception architekturu ve dvou variantách. První by bylo použití předučených modelů, a tedy vstupních dat převedených do potřebného formátu (rozměru). Jako druhá varianta by bylo možné použití klíčových prvků architektur a jejich aplikování na zmenšený model. Druhý směr pokračování výzkumu je v práci s odlišnými barevnými schématy či rozšířit datovou sadu.

8 Seznam literatury

[1] F. Chollet, “Keras,” 2019. [Online]. Available: https://keras.io/.

[2] Google Inc, “Tensorflow,” 2020.

[3] J. D. Kelleher, B. Mac Namee, and A. D’Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. The MIT Press, 2015.

[4] S. Raschka and V. Mirjalili, *Python Machine Learning, 3rd Edition*, 1st ed. Packt Publishing, 2019.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[6] P. Honzík, “Strojové učení,” Brno, 2006.

[7] A. C. Müller and S. Guido, “Introduction to machine learning with Python : a guide for data scientists.” 2017.

[8] L. Fei-Fei, “CS231n: Convolutional Neural Networks for Visual Recognition,” *Stanford University*, 2019. [Online]. Available: http://cs231n.github.io/convolutional-networks/.

[9] Chih-Wei Hsu and Chih-Jen Lin, “A comparison of methods for multiclass support vector machines,” *IEEE Trans. Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[10] N. Buduma and N. Locascio, *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*, 1st ed. O’Reilly Media, Inc., 2017.

[11] S. Liu and M. Whitty, “Automatic grape bunch detection in vineyards with an SVM classifier,” *J. Appl. Log.*, vol. 13, no. 4, Part 3, pp. 643–653, 2015.

[12] A. Kamilaris and F. X. Prenafeta-Boldú, “Deep learning in agriculture: A survey,” *Comput. Electron. Agric.*, vol. 147, pp. 70–90, 2018.

[13] J. Davis and M. Goadrich, “The Relationship between Precision-Recall and ROC Curves,” in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 233–240.

[14] G. Cheng and J. Han, “A survey on object detection in optical remote sensing images,” *ISPRS J. Photogramm. Remote Sens.*, vol. 117, pp. 11–28, 2016.

[15] D. Sugumar, V. Harshavarthan, S. Kavisri, M. S. Aezhisai Vallavi, and P. T. Vanathi, “Citrus classification and grading using machine learning algorithms,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 10, pp. 2616–2621, 2019.

[16] M. G. Selvaraj *et al.*, “AI-powered banana diseases and pest detection,” *Plant Methods*, vol. 15, no. 1, 2019.

[17] K. Collins-Thompson, "Multi-Class Evaluation," *Coursera*, 2019. [Online]. Available: https://www.coursera.org/lecture/python-machine-learning/multi-class-evaluation-1ugJR. [Accessed: 23-Feb-2020].

[18] R. Katarzyna and M. Paweł, "A vision-based method utilizing deep convolutional neural networks for fruit variety classification in uncertainty conditions of retail sales," *Appl. Sci.*, vol. 9, no. 19, 2019.

[19] A. Koirala, K. B. Walsh, Z. Wang, and C. McCarthy, "Deep learning – Method overview and review of use for fruit detection and yield estimation," *Comput. Electron. Agric.*, vol. 162, pp. 219–234, 2019.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[21] K. Hameed, D. Chai, and A. Rassau, "A comprehensive review of fruit and vegetable classification techniques," *Image Vis. Comput.*, vol. 80, pp. 24–44, 2018.

[22] L. Zhou, C. Zhang, F. Liu, Z. Qiu, and Y. He, "Application of Deep Learning in Food: A Review," *Compr. Rev. Food Sci. Food Saf.*, vol. 18, no. 6, pp. 1793–1811, 2019.

[23] Y. Lecun, "Generalization and network design strategies," in *Connectionism in perspective*, R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels, Eds. Elsevier, 1989.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[26] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for Simplicity: The All Convolutional Net," *CoRR*, vol. abs/1412.6, 2014.

[27] N. Muhammad, A. Nasir, Z. Ibrahim, and N. Sabri, "Evaluation of CNN, Alexnet and GoogleNet for Fruit Recognition," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 12, pp. 468–475, 2018.

[28] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, "Deepfruits: A fruit detection system using deep neural networks," *Sensors (Switzerland)*, vol. 16, no. 8, 2016.

[29] J. O. Adigun, F. M. Okikiola, E. E. Aigbokhan, and M. M. Rufai, "Automated system for grading apples using convolutional neural network," *Int. J. Innov. Technol. Explor. Eng.*, vol. 9, no. 1, pp. 1458–1464, 2019.

[30] S. Ashraf, I. Kadery, A. A. Chowdhury, T. Z. Mahbub, and R. M. Rahman, "Fruit Image

Classification Using Convolutional Neural Networks," *Int. J. Softw. Innov.*, vol. 7, no. 4, pp. 51–70, 2019.

[31] B. Jiang *et al.*, "Fusion of machine vision technology and AlexNet-CNNs deep learning network for the detection of postharvest apple pesticide residues," *Artif. Intell. Agric.*, vol. 1, pp. 1–8, 2019.

[32] H. Mureşan and M. Oltean, "Fruit recognition from images using deep learning," *Acta Univ. Sapientiae, Inform.*, vol. 10, pp. 26–42, 2018.

[33] C. Zhao, X. Li, Y. Zhao, and X. Shi, "Object Detection under Natural Illumination Conditions using Superpixels and Local Binary Pattern Feature," in *Journal of Physics: Conference Series*, 2019, vol. 1237, no. 3.

[34] Y.-D. Zhang *et al.*, "Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation," *Multimed. Tools Appl.*, vol. 78, no. 3, pp. 3613–3632, 2017.

[35] M. R. Mia, M. J. Mia, A. Majumder, S. Supriya, and M. T. Habib, "Computer vision based local fruit recognition," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 1, pp. 2810–2820, 2019.

[36] J. Redmon and A. Farhadi, "{YOLO9000:} Better, Faster, Stronger," *CoRR*, vol. abs/1612.0, 2016.

[37] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective Search for Object Recognition," *Int. J. Comput. Vis.*, 2013.

[38] R. B. Girshick, "Fast {R-CNN}," *CoRR*, vol. abs/1504.0, 2015.

[39] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster {R-CNN:} Towards Real-Time Object Detection with Region Proposal Networks," *CoRR*, vol. abs/1506.0, 2015.

[40] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *CoRR*, vol. abs/1506.0, 2015.

[41] W. Astuti, S. Dewanto, K. Soebandrija, and S. Tan, "Automatic fruit classification using support vector machines: a comparison with artificial neural network," *IOP Conf. Ser. Earth Environ. Sci.*, vol. 195, p. 12047, 2018.

[42] S.-H. Wang and Y. Chen, "Fruit category classification via an eight-layer convolutional neural network with parametric rectified linear unit and dropout technique," *Multimed. Tools Appl.*, 2018.

[43] S. Sakib, Z. Ashrafi, and M. Siddique, "Implementation of Fruits Recognition Classifier using Convolutional Neural Network Algorithm for Observation of

Accuracies for Various Hidden Layers." 2019.

[44] M. S. Hossain, M. Al-Hammadi, and G. Muhammad, "Automatic Fruit Classification Using Deep Learning for Industrial Applications," *IEEE Trans. Ind. Informatics*, vol. 15, no. 2, pp. 1027–1034, Feb. 2019.

[45] D. T. P. Chung and D. Van Tai, "A fruits recognition system based on a modern deep learning technique," in *Journal of Physics: Conference Series*, 2019, vol. 1327, no. 1.

[46] Yogesh, A. K. Dubey, and R. Ratan, "Development of feature based classification of fruit using deep learning," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 12, pp. 3285–3290, 2019.

[47] Y. Gurubelli, M. Ramanathan, and P. Ponnusamy, "Fractional fuzzy 2DLDA approach for pomegranate fruit grade classification," *Comput. Electron. Agric.*, vol. 162, pp. 95–105, 2019.

[48] J. B. Ivars and S. C. García, "Image database: Apple 'Golden,'" 2018. [Online]. Available: http://www.cofilab.com/portfolio/goldendb/.

[49] G. Wu, Q. Zhu, M. Huang, Y. Guo, and J. Qin, "Automatic recognition of juicy peaches on trees based on 3D contour features and colour data," *Biosyst. Eng.*, vol. 188, pp. 1–13, 2019.

[50] S. Sun, M. Jiang, D. He, Y. Long, and H. Song, "Recognition of green apples in an orchard environment by combining the GrabCut model and Ncut algorithm," *Biosyst. Eng.*, vol. 187, pp. 201–213, 2019.

[51] D. Liu, J. Shen, H. Yang, Q. Niu, and Q. Guo, "Recognition and localization of actinidia arguta based on image recognition," *Eurasip J. Image Video Process.*, vol. 2019, no. 1, 2019.

[52] S. Xing, M. Lee, and K.-K. Lee, "Citrus pests and diseases recognition model using weakly dense connected convolution network," *Sensors (Switzerland)*, vol. 19, no. 14, 2019.

[53] J. O. Pinzón-Arenas, R. Jiménez-Moreno, and C. G. Pachón-Suescún, "ResSeg: Residual encoder-decoder convolutional neural network for food segmentation," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 1, pp. 1017–1026, 2020.

[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[55] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the Disharmony between Dropout

and Batch Normalization by Variance Shift," 2018.

[56] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: Bandit-based configuration evaluation for hyperparameter optimization," *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, pp. 1–15, 2019.

[57] P. Murugan, "Hyperparameters Optimization in Deep Convolutional Neural Network / Bayesian Approach with Gaussian Process Prior," 2017.

[58] N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere, and M. Smelyanskiy, "On large-batch training for deep learning: Generalization gap and sharp minima," *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, pp. 1–16, 2019.

[59] H. Mureşan and M. Oltean, "Kaggle 360 Fruit," 2019. [Online]. Available: https://www.kaggle.com/moltean/fruits.

[60] INRIA, "User guide scikit learn," 2019. [Online]. Available: https://scikit-learn.org/stable/user_guide.html.

[61] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine Learning for Fluid Mechanics," *Annu. Rev. Fluid Mech.*, vol. 52, no. 1, pp. 477–508, 2020.

9 Přílohy

1) Tabulka četností kategorií datové sady
2) Architektura modelu 1 CD varianta 1 (98,09 %)
3) Zadání diplomové práce
4) Archiv se zdrojovými kódy (Zdrojove_soubory_Patrik_Dite.zip) Obsah archivu:
    a. Dataset_save.py
    b. Definition.py
    c. Dataset_visualisation.py
    d. Functions.py
    e. Train.py
    f. Aug_train.py
    g. Hyperband.py
    h. Models.py
5) Matice změn modelu 1 CD varianty 1 (CM_model_1_CD_varianta1_9806.png)
6) Soubor s vahami modelu 1 CD varianty 1 (CM_model_1_CD_varianta1_9806.h5)

1) Tabulka četností kategorií datové sady

Kategorie	Učící sada	Testovací sada	Celkem
Apple Braeburn	492	164	656
Apple Crimson Snow	444	148	592
Apple Golden 1	492	164	656
Apple Golden 2	492	164	656
Apple Golden 3	481	161	642
Apple Granny Smith	492	164	656
Apple Pink Lady	456	152	608
Apple Red 1	492	164	656
Apple Red 2	492	164	656
Apple Red 3	429	144	573
Apple Red Delicious	490	166	656
Apple Red Yellow 1	492	164	656
Apple Red Yellow 2	672	219	891
Apricot	492	164	656
Avocado	427	143	570
Avocado ripe	491	166	657
Banana	490	166	656
Banana Lady Finger	450	152	602
Banana Red	490	166	656
Beetroot	450	150	600
Blueberry	462	154	616
Cactus fruit	490	166	656
Cantaloupe 1	492	164	656
Cantaloupe 2	492	164	656
Carambula	490	166	656
Cauliflower	702	234	936
Cherry 1	492	164	656
Cherry 2	738	246	984
Cherry Rainier	738	246	984
Cherry Wax Black	492	164	656
Cherry Wax Red	492	164	656
Cherry Wax Yellow	492	164	656
Chestnut	450	153	603
Clementine	490	166	656
Cocos	490	166	656
Dates	490	166	656
Eggplant	468	156	624

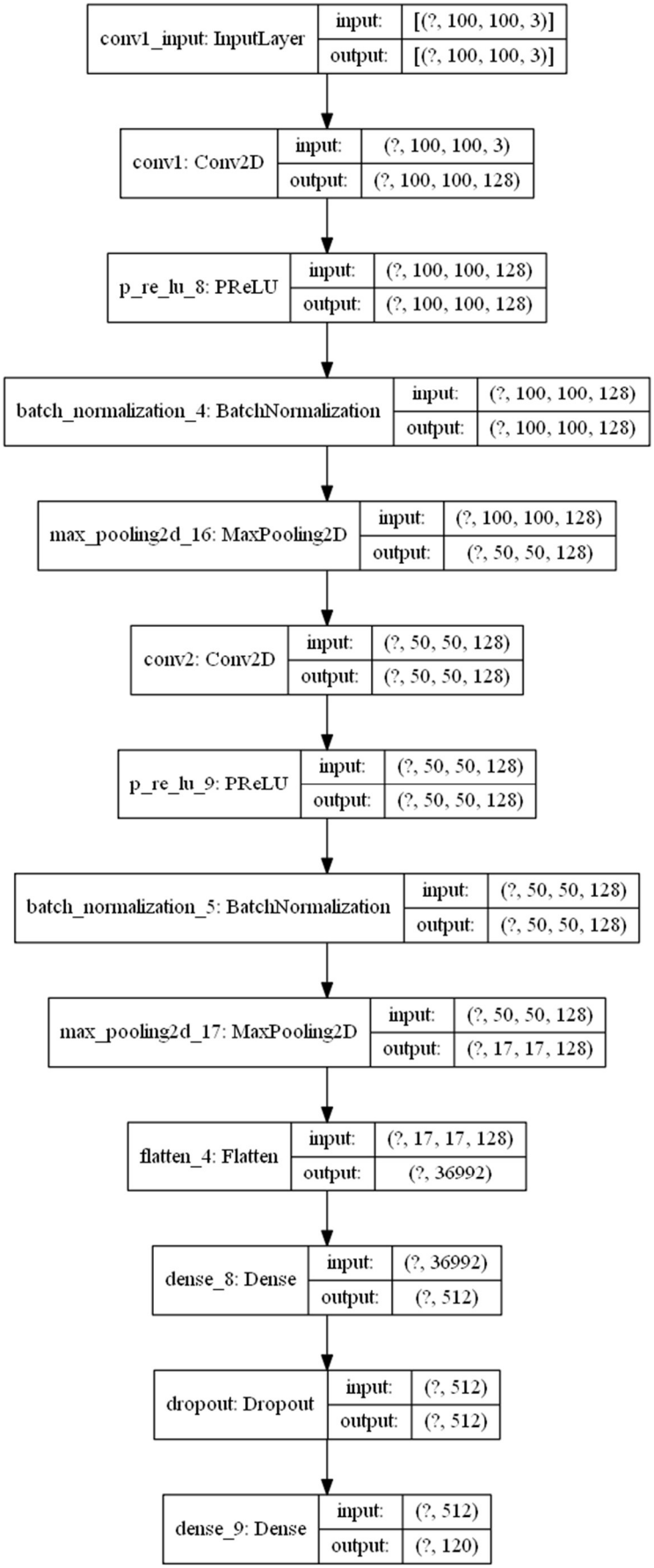
Ginger Root	99	99	198
Granadilla	490	166	656
Grape Blue	984	328	1312
Grape Pink	492	164	656
Grape White	490	166	656
Grape White 2	490	166	656
Grape White 3	492	164	656
Grape White 4	471	158	629
Grapefruit Pink	490	166	656
Grapefruit White	492	164	656
Guava	490	166	656
Hazelnut	464	157	621
Huckleberry	490	166	656
Kaki	490	166	656
Kiwi	466	156	622
Kohlrabi	471	157	628
Kumquats	490	166	656
Lemon	492	164	656
Lemon Meyer	490	166	656
Limes	490	166	656
Lychee	490	166	656
Mandarine	490	166	656
Mango	490	166	656
Mango Red	426	142	568
Mangostan	300	102	402
Maracuja	490	166	656
Melon Piel de Sapo	738	246	984
Mulberry	492	164	656
Nectarine	492	164	656
Nectarine Flat	480	160	640
Nut Forest	218	218	436
Nut Pecan	178	178	356
Onion Red	450	150	600
Onion Red Peeled	445	155	600
Onion White	438	146	584
Orange	479	160	639
Papaya	492	164	656
Passion Fruit	490	166	656
Peach	492	164	656

Peach 2	738	246	984
Peach Flat	492	164	656
Pear	492	164	656
Pear Abate	490	166	656
Pear Forelle	702	234	936
Pear Kaiser	300	102	402
Pear Monster	490	166	656
Pear Red	666	222	888
Pear Williams	490	166	656
Pepino	490	166	656
Pepper Green	444	148	592
Pepper Red	666	222	888
Pepper Yellow	666	222	888
Physalis	492	164	656
Physalis with Husk	492	164	656
Pineapple	490	166	656
Pineapple Mini	493	163	656
Pitahaya Red	490	166	656
Plum	447	151	598
Plum 2	420	142	562
Plum 3	900	304	1204
Pomegranate	492	164	656
Pomelo Sweetie	450	153	603
Potato Red	450	150	600
Potato Red Washed	453	151	604
Potato Sweet	450	150	600
Potato White	450	150	600
Quince	490	166	656
Rambutan	492	164	656
Raspberry	490	166	656
Redcurrant	492	164	656
Salak	490	162	652
Strawberry	492	164	656
Strawberry Wedge	738	246	984
Tamarillo	490	166	656
Tangelo	490	166	656
Tomato 1	738	246	984
Tomato 2	672	225	897
Tomato 3	738	246	984

Tomato 4	479	160	639
Tomato Cherry Red	492	164	656
Tomato Maroon	367	127	494
Tomato Yellow	459	153	612
Walnut	735	249	984

Zdroj: Vlastní zpracování

2) Architektura modelu 1 CD varianta 1 (98,09 %)



3) Zadání diplomové práce

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. Patrik Dítě**
Studium:	I1800341
Studijní program:	N6209 Systémové inženýrství a informatika
Studijní obor:	Informační management
Název diplomové práce:	**Identifikace ovoce metodami strojového učení**
Název diplomové práce AJ:	Identification of Fruit Types Using Machine Learning

Cíl, metody, literatura, předpoklady:

Cílem práce je zhodnotit možnosti využití metod strojového učení při identifikaci druhů ovoce, navrhnout a realizovat ukázkovou aplikaci.

Goodfellow, I., Bengio, J., Courville, A. (2016) Deep Learning. MIT Press Kelleher, J.D. et al. (2015) Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms. Worked Examples, and Case Studies. MIT Press. Buduma, N. (2017) Fundamentals of Deep Learning: Designing Next-Generation Machine Learning Algorithms. OReilly. Muller, A.C., Guido, S. (2016) Introduction to Machine Learning with Python: A Guide for Data Scientists. OReilly. Raschka, S., Mirjalili, V. (2017) Python Machine Learning. Packt. Garreta R. et al. (2017) Scikit-learn: Machine Learning Simplified Learning Path. Packt

Garantující pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	doc. RNDr. Kamila Štekerová, Ph.D.
Oponent:	Ing. Karel Mls, Ph.D.
Datum zadání závěrečné práce:	21.10.2014