
# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

## ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE


# EFEKTIVNOST HLUBOKÝCH KONVOLUČNÍCH NEURONOVÝCH SÍTÍ NA ELEMENTÁRNÍ KLASIFIKAČNÍ ÚLOZE

EFFICIENCY OF DEEP CONVOLUTIONAL NEURAL NETWORKS ON AN ELEMENTARY CLASSIFICATION TASK

**DIPLOMOVÁ PRÁCE**
MASTER'S THESIS


**AUTOR PRÁCE** — **Bc. Jan Prax**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Pavel Škrabánek, Ph.D.**
SUPERVISOR



**BRNO 2021**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

# Zadání diplomové práce

| | |
|---|---|
| Ústav: | Ústav automatizace a informatiky |
| Student: | **Bc. Jan Prax** |
| Studijní program: | Strojní inženýrství |
| Studijní obor: | Aplikovaná informatika a řízení |
| Vedoucí práce: | **Ing. Pavel Škrabánek, Ph.D.** |
| Akademický rok: | 2020/21 |

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

## Efektivnost hlubokých konvolučních neuronových sítí na elementární klasifikační úloze

**Stručná charakteristika problematiky úkolu:**

Hluboké konvoluční neuronové sítě jsou považovány za nástroj, který umožňuje vytvářet systémy počítačového vidění s vysokou klasifikační přesností. Tyto systémy se osvědčili na řadě úloh s velkou variabilitou v rámci jednotlivých tříd. Neexistují však relativně jednoduché problémy, pro které lze dosáhnout lepších výsledků s využitím člověkem navržených deskriptorů příznaků?

**Cíle diplomové práce:**

Student vypracuje rešerši mapující problematiku klasifikace objektů na základě obrazových dat. Rešerše bude zahrnovat i hluboké konvoluční sítě. Student pro zadanou klasifikační úlohu navrhne a implementuje systémy počítačového vidění s využitím a) člověkem navržených deskriptorů příznaků, b) hlubokých konvolučních sítí. Pro tuto úlohu dále srovná klasifikační přesnost, časovou náročnost a paměťovou náročnost jednotlivých řešení, a získané výsledky dá do kontextu principu, na kterém byly systémy založeny (člověkem navržené deskriptory vs. hluboké konvoluční sítě).

**Seznam doporučené literatury:**

GRAUMAN, Kristen a Bastian LEIBE. Visual Object Recognition. Synthesis Lectures on Artificial Intelligence and Machine Learning [online]. 2011, 5(2), 1-181 [cit. 2020-09-10]. DOI: 10.2200/S00332ED1V01Y201103AIM011. ISSN 1939-4608. Dostupné z: http://www.morganclaypool.com/doi/abs/10.2200/S00332ED1V01Y201103AIM011.

LECUN, Yann, Yoshua BENGIO a Geoffrey HINTON. Deep learning. Nature [online]. 2015, 521(7553), 436-444 [cit. 2020-09-10]. DOI: 10.1038/nature14539. ISSN 0028-0836. Dostupné z: http://www.nature.com/articles/nature14539.

HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, 2016, 2016, s. 770-778 [cit. 2020-09-10]. DOI: 10.1109/CVPR.2016.90. ISBN 978-1-4673-8-51-1. Dostupné z: http://ieeexplore.ieee.org/document/7780459/

HUANG, Gao, Zhuang LIU, Laurens VAN DER MAATEN a Kilian Q. WEINBERGER. Densely Connected Convolutional Networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, 2017, 2017, s. 2261-2269 [cit. 2020-09-10]. DOI: 10.1109/CVPR.2017.243. ISBN 978-1-5386-0457-1. Dostupné z: https://ieeexplore.ieee.org/document/8099726/.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

| | |
|---|---|
| doc. Ing. Radomil Matoušek, Ph.D.<br>ředitel ústavu | doc. Ing. Jaroslav Katolický, Ph.D.<br>děkan fakulty |

# ABSTRAKT

V této diplomové práci jsou srovnány modely hlubokých konvolučních neuronových sítí a modely deskriptorů příznaků. Modely deskriptorů příznaků jsou doplněny o vybraný vhodný klasifikátor. Tyto modely spadají do oblasti strojového učení, a tudíž jsou v práci popsány typy strojového učení. Dále jsou tyto vybrané modely popsány a vysvětleny jejich základy a problémy. Je vypsán hardware a software použitý k uskutečnění testů a jsou vypsány výsledky testů a shrnutí výsledků. Pak je provedeno srovnání uvedených modelů na základě přesnosti validace a časové náročnosti.



# ABSTRACT

In this thesis deep convolutional neural networks models and feature descriptor models are compared. Feature descriptors are paired with suitable chosen classifier. These models are a part of machine learning therefore machine learning types are described in this thesis. Further these chosen models are described, and their basics and problems are explained. Hardware and software used for tests is listed and then test results and results summary is listed. Then comparison based on the validation accuracy and training time of these said models is done.



# KLÍČOVÁ SLOVA

Deskriptor příznaků, neuronové sítě, hluboké neuronové sítě, HOG, SIFT, SVM, ResNet, EfficientNet.



# KEYWORDS

Feature extractor, neural networks, deep neural networks, HOG, SIFT, SVM, ResNet, EfficientNet.

# BIBLIOGRAFICKÁ CITACE

PRAX, Jan. Efektivnost hlubokých konvolučních neuronových sítí na elementární klasifikační úloze, Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, 2021, 65 s. Diplomová práce. Vedoucí práce: Ing. Pavel Škrabánek Ph.D.

# PODĚKOVÁNÍ

Děkuji svojí rodině, která mě během studia velmi podporovala a děkuji vedoucímu práce Ing. Pavlu Škrabánkovi Ph.D., který si toto poděkování zaslouží za svou ochotu a rady, bez kterých by se tato práce neobešla.

# ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Pavla Škrabánka, Ph.D. a s použitím literatury uvedené v seznamu literatury.

V Brně dne 20. 5. 2021 ……

Bc. Jan Prax

# OBSAH

# 1 ÚVOD

V této práci jsou vybráni dva zástupci deskriptorů příznaků používaných pro klasifikaci obrazu a dva zástupci neuronových sítí používaných pro klasifikaci obrazu. K zástupcům deskriptorů příznaků je vybrán i vhodný klasifikátor obrazu. Pro tyto zástupce jsou následně vytvořeny implementace, které jsou následně porovnány na základě jejich výsledné přesnosti na validačních datech a časové náročnosti trénování. Přesnost a časová náročnost je určována na dodaných datových sadách obrazů $n$-úhelníků. Tyto datové sady se dělí na barevnou a černobílou sadu.

Zástupci deskriptorů příznaků a neuronových sítí jsou voleni na základě rešerše. Kritériem výběru pro zástupce deskriptorů příznaků je jeho četnost použití v porovnání s ostatními modely deskriptorů příznaků. Kritériem výběru klasifikátoru obrazu byla nejvyšší přesnost klasifikace v porovnání s ostatními modely, klasifikátorů. Kritériem výběru zástupce neuronových sítí je nejvyšší přesnost zástupce na datové sadě „Imagenet“.

Studovaná problematika této práce je klasifikace obrazu, klasifikace obrazu se dělí na různé klasifikační modely, výše jsou uváděny tyto modely jako zástupci. Trénování klasifikačních modelů spadá do oboru zvaného strojové učení, proto jsou v této práci popsány typy strojového učení. Dále jsou v této práci popsány vybrané modely klasifikace obrazu, které jsou zde děleny do dvou hlavních skupin, což jsou neuronové sítě a deskriptory příznaků s vybraným modelem klasifikátoru pro tyto deskriptory. Poté je uveden hardware a software, který je použit pro testy deskriptorů příznaků a neuronových sítí. Následně jsou vypsány výsledky testů a jejich grafy průběhu přesnosti na čase. Pak jsou výsledky testů shrnuty, zhodnoceny a je pojednáno o jejich problémech.

Cílem porovnání klasifikačních modelů v této práci je určení, zda deskriptory příznaků mohou být pro určité úlohy kompetitivní s neuronovými sítěmi.

# 2 TYPY STROJOVÉHO UČENÍ

Ve strojovém učení, přesněji klasifikaci obrazu strojovým učením je snaha definovat či aproximovat obraz hodnotami, které jej odliší od ostatních. K tomuto postupu slouží různé algoritmy strojového učení, ze kterých se pro klasifikaci obrazu využívají deskriptory a následně se použijí vhodné klasifikátory, nebo v dnešní době populární hluboké neuronové sítě. Obor studie hlubokých neuronových sítí je nazývaný hluboké učení. Tyto algoritmy pro klasifikaci obrazu nejčastěji využívají styl učení s učitelem, kdy srovnávají svoje výsledky s trénovací skupinou dat a postupně, po iteracích, své výsledky zlepšují. Toto zlepšení nepokračuje do nekonečna, ale je zastaveno kombinací mnoha jevů, které nedovolují růst přesnosti klasifikace. Typy strojového učení jsou rozebrány v této kapitole.

## 2.1 Učení s učitelem

Učení s učitelem probíhá s použitím vstupních dat, která jsou rozlišena svým oštítkováním. Algoritmus se snaží vytvořit model, který dosahoval co nejvyšší podobnosti výsledků (predikovaných ze vstupních dat) s jejich štítkem. Tedy se vezme vstupně-výstupní (data-štítek) pár, který se algoritmus snaží určit (aproximovat) určitým modelem. Tento postup se v iteracích projde přes všechna data a algoritmus se snaží určit jejich správnou skupinu [1], [2], [3].

Oštítkování dat jsou dodány člověkem, který je brán jako učitel, jelikož poskytl správné štítky dat a algoritmus se snaží vytvořit model, který tyto data popisuje a tento model je upravován algoritmem, který zde má také funkci učitele. Proto se tento styl učení nazývá učení s učitelem [2], [3].

Učení s učitelem se dělí na dvě kategorie, kterými jsou klasifikační problémy a regresní problémy. Vstupní data pro oba tyto problémy mohou být numerické, či kategorické [3].

Příklady obvyklých algoritmů učení s učitelem:

- neuronové sítě,
- podpůrné vektorové stroje („support vector machines“) ve zkratce SVM,
- Naivní Bayes,
- nejbližší soused,
- rozhodovací stromy,
- lineární regrese.

Pro klasifikaci obrazu, například v SVM a Neuronový sítích, se převážně používá styl učení s učitelem.

## 2.2 Učení bez učitele

V algoritmech učení bez učitele, nejsou dodána označení/kategorie dat (jako u algoritmů učení s učitelem), takže zde není žádný učitel a algoritmy musejí nalézt spojitosti v (dodaných vstupních) datech. Algoritmus se tedy snaží data seskupit, popsat v nich spojitosti nebo najít určitá pravidla v těchto datech. Proto se algoritmy učení bez učitele s výhodou používají, když uživatel potřebuje datům porozumět a tyto algoritmy mu umožňují lepší náhled do těchto dat, a tudíž jednodušší pochopení těchto dat.

Obvyklým příkladem algoritmu učení bez učitele je shlukování. Velmi používaným příkladem algoritmu shlukování je algoritmus K-means [2], [3].

## 2.3 Zpětnovazební učení

Zpětnovazební učení se využívá na problémy, kde algoritmus, kterému se v tomto stylu učení nazývá agent, operuje v neznámém prostředí a jeho řízení závisí pouze na informaci ze zpětné vazby.

Pro toto neznámé prostředí nejsou dostupná trénovací data, ale je dostupná informace cíle, kterého je požadované dosáhnout a také jaké jsou možné akce k dosažení tohoto cíle.

Informace je brána po iteracích z výstupu algoritmu zpětnou vazbou a připojena ke vstupu. Algoritmus probíhá po iteracích a v každé iteraci mění svůj průběh v závislosti na informaci ze zpětné vazby. Tímto postupem se snaží dosáhnout požadovaného výsledku. Většinou je požadovaný výsledek maximalizace, či minimalizace určitého problému.

Algoritmus se tedy učí ze svých informací („zkušeností“) z předchozích stavů do té doby, než prozkoumá všechny možné stavy, kterých může nabývat. Tím získá informaci o všech možných stavech, a tudíž postupech, které může použít [2], [3].

# 3 DESKRIPTORY PŘÍZNAKŮ A KLASIFIKÁTOR SVM

Deskripce příznaků je reprezentace obrazu, která tento obraz zjednodušuje extrahováním užitečných informací, zbylé neužitečné informace jsou zahozeny. Deskriptor příznaků je tedy algoritmus, který konvertuje obraz do vektoru či pole příznaků [4].

Užitečnost informace záleží na klasifikovaném objektu. Důležitou vlastností pro posouzení užitečnosti informace/příznaku je unikátnost určitého příznaku, tedy unikátnější příznak pomáhá lépe popsat objekt v obrazu. Užitečné informace proto často bývají hrany a rohové oblasti klasifikovaného objektu neboli části objektu, kde se výrazně mění informace o tomto objektu.

## 3.1 HOG („histogram of oriented gradients“)

HOG v celém názvu histogram orientovaných gradientů. Tento deskriptor příznaků vypočítá směry gradientů a určí je do histogramů neboli skupin na základě četnosti určitých gradientů. Tyto gradienty v obrazu jsou důležité, jelikož velikost gradientů je největší okolo hran a rohových oblastí v obrazu. Tyto hranové a rohové oblasti udávají velké množství informací o obrazu. Výhodou algoritmu HOG je invariantnost ke geometrickým a fotometrickým transformacím, tedy ke škálování a osvětlení. Nevýhodou však je, že algoritmus není invariantní k rotaci.

### 3.1.1 Postup algoritmu HOG

1) Normalizace obrazu (volitelné)
2) Výpočet gradientů
3) Rozdělení do histogramů
4) Bloková normalizace
5) Zploštění pole histogramů do jednodimenzionálního vektoru příznaků

**Normalizace obrazu**
Zlepšení klasifikace obrazu po normalizaci je minimální [4], [5] a pokusy v této práci, také dokázali, že vylepšení přesnosti klasifikace obrazu v testovaných obrazech bylo minimální, a tudíž tento krok je vynechán.

**Výpočet gradientů**
Výpočet gradientů v složkách $x$ a $y$ (horizontální a vertikální) se provádí filtrováním neboli použitím masky [-1, 0, 1] pro složku $x$ i $y$. Tyto gradienty se vypočítávají pro každý pixel obrazu. Dle [5] tato jednodimenzionální [-1, 0, 1] maska udává nejlepší výsledky.

**Rozdělení do histogramů**
Rozdělení do histogramů je řešeno na základě velikosti gradientu každého pixelu. Může být i funkcí druhé mocniny, či odmocniny gradientu, ale na základě praxe, dělení podle

velikosti gradientu poskytuje nejlepší výsledky [5]. Tyto gradienty každého pixelu jsou děleny do buněk a pro každou buňku je sestaven histogram (rozdělení do kategorií), který ji popisuje. Toto dělení je vhodné z důvodu kompaktnosti řešení bez ztráty informace a z důvodu redukce šumu v obrazu.

**Bloková normalizace**
Bloková normalizace se provádí přes násobek velikosti buňky, ale posun bloku se provádí pouze o jednonásobek velikosti buňky, tudíž se každá buňka normalizuje vícekrát, tím se normalizuje osvětlení obrazu a závislost gradientů na osvětlení je minimalizována. Normalizovaný blok je vždy zapsán jako jednodimenzionální vektor.

**Zploštění pole histogramů**
Zploštění pole histogramů do jednodimenzionálního vektoru příznaků se provádí zřetězením vektorů vzniklých po blokové normalizaci.

**Pravidla velikosti obrazu, bloku, buňky**
Důležité je dodržet, že velikost obrazu je násobek velikosti bloku a blok je násobek velikosti buňky, kde obraz může být obdélník (obdélníkové pole pixelů), ale blok a buňka jsou čtvercové pole [4], [5]. (Například 64x128 pixelů obraz, 16x16 pixelů blok, 8x8 pixelů buňka)

## 3.2 SIFT („scale invariant feature transform“)

SIFT v celém názvu „scale invariant feature transform“ přeložen do českého jazyka: škálovatelně invariantní transformátor příznaků. Příznaky z algoritmu SIFT jsou invariantní ke škálování a rotaci obrazu a částečně invariantní ke změně osvětlení. A jsou dobře určené v prostoru. Dále je dobře určená četnost těchto příznaků v určitém místě v prostoru a tím je snížena citlivost na šum. Požadavky algoritmu na výpočetní zdroje jsou minimalizovány použitím kaskádového filtrovacího procesu, ve kterém jsou výpočetně náročné operace aplikovány pouze na pozice, které splní počáteční podmínku.

### 3.2.1 Postup algoritmu SIFT

1) Detekce extrému pro škálovatelný prostor
2) Lokalizace klíčových bodů
3) Přiřazení orientace
4) Deskriptor klíčových bodů

**Detekce extrému**
První krok neboli detekce extrému pro škálovatelný prostor prohledá celý prostor a určí lokace objektů pro více škál. Toto prohledání využívá funkci diference Gaussiánu. Jakmile jsou nalezeny diference Gaussiánu, je porovnán každý pixel s jeho okolními osmi pixely v aktuální škále a devíti pixely v předchozí a následující škále. Pokud je pixel v tomto porovnání extrémem, tak je potenciálním klíčovým bodem a je tedy nejlépe

popsán v této škále. Tímto postupem se získají potenciální body zájmu neboli potenciální klíčové body, které jsou invariantní ke škálování a orientaci.

**Lokalizace klíčových bodů**

Lokalizace klíčových bodů je provedena s použitím Taylorovy řady a je určen ofset lokálního extrému a pokud tento ofset je v určitém směru větší než 0,5 tak extrém leží blíže jinému bodu v tomto směru, a proto je potenciální klíčový bod změněn na jiný potenciální klíčový bod v tomto směru a je provedena lokalizace v tomto novém bodě.

**Přiřazení orientace**

Orientace je přiřazena, aby se zajistila invariance k rotaci. Přiřazení orientace se vypočítá jako orientovaný histogram gradientů okolí klíčového bodu. Největší vrchol histogramu je přiřazen jako orientace klíčového bodu a vrcholy co dosahují 80 procent velikosti největšího vrcholu, jsou také vzaty jako orientace klíčového bodu. Tímto vzniknou klíčové body o stejné pozici a škále, ale rozdílných orientacích, a tudíž vznikne klíčový bod invariantní k rotaci.

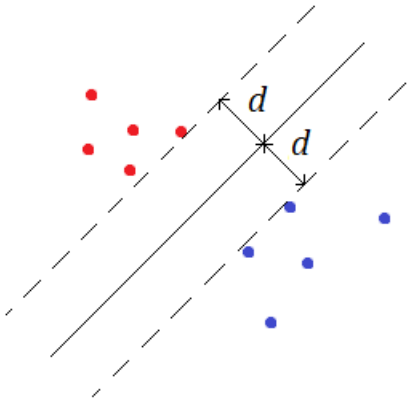
**Deskriptor klíčových bodů**

Deskriptor klíčových bodů je sestaven podobně jako bloková normalizace v algoritmu HOG. Tedy okolí klíčového bodu o velikosti 16x16 pixelů je vzato a rozděleno na 16 menších bloků o velikosti 4x4, pro menší blok je vytvořen histogram, který je rozdělen na 8 kategorií, a tudíž je vytvořen vektor o velikosti 128 hodnot. Tento vzniklý vektor tvoří 128 hodnotový deskriptor příznaků pro určený klíčový bod. Tento vektor je vypočten pro každý klíčový bod [6], [7].

## 3.3 SVM („support vector machines“)

Základní myšlenkou SVM neboli metody podpůrných vektorů je rozdělení n-dimenzionálních dat určitou nadrovinou. Pro lineárně dělitelná data je touto nadrovinou dělící přímka, která má stejnou vzdálenost *d* od podpůrných vektorů. Podpůrné vektory jsou vektory, které když jsou odstraněny, tak změní polohu dělící nadroviny (v Obr. 1 jsou to body ležící na rovnoběžkách s dělící přímkou). Optimální přímka je vedena tak, aby vzdálenost *d* byla maximalizována (Obr. 1), a tudíž dělící přímka dělila tyto dvě skupiny dat v co největší vzdálenosti.

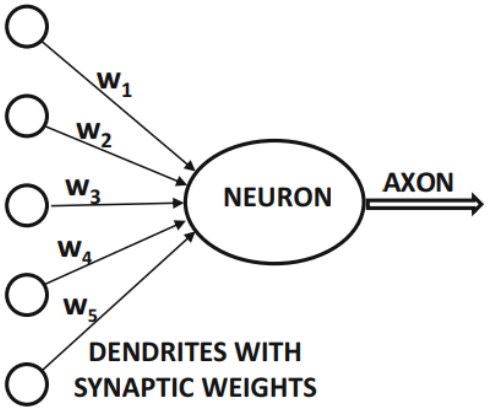
Pokud data nejsou dělitelná lineárně, tak se používá transformace dat. Aby byla nelineární data dělitelná lineárně, tak se data namapují (transformují) do prostoru o vyšší dimenzi. Pro toto mapování se používá kernelová funkce. Velmi používanou kernelovou funkcí pro nelineární data je RBF neboli radiální bázová funkce [8], [9], [10].




Obr. 1: SVM rozdělení lineárních dat, kde červené body jsou data první skupiny, modré body jsou data druhé skupiny, plná čára znázorňuje dělicí přímku, čárkované čáry jsou pomocné přímky pro určení vzdálenosti *d* a jsou rovnoběžné s dělicí přímkou.

# 4 NEURONOVÉ SÍTĚ

Jsou nazvány podle analogie k organickému neuronu, jímž byly inspirovány. Organický neuron mění sílu svých synaptických konexí na základě vnějších stimulů. Tato skutečnost byla analogicky využita v umělých neuronových sítích, kde tyto síly jsou nazvány váhy a tyto váhy jsou určovány různými výpočetními algoritmy. Neuronové sítě jsou cyklické a acyklické, v dnešní době jsou používány acyklické. V této práci je pouze jednáno o acyklických sítích [11], [12].

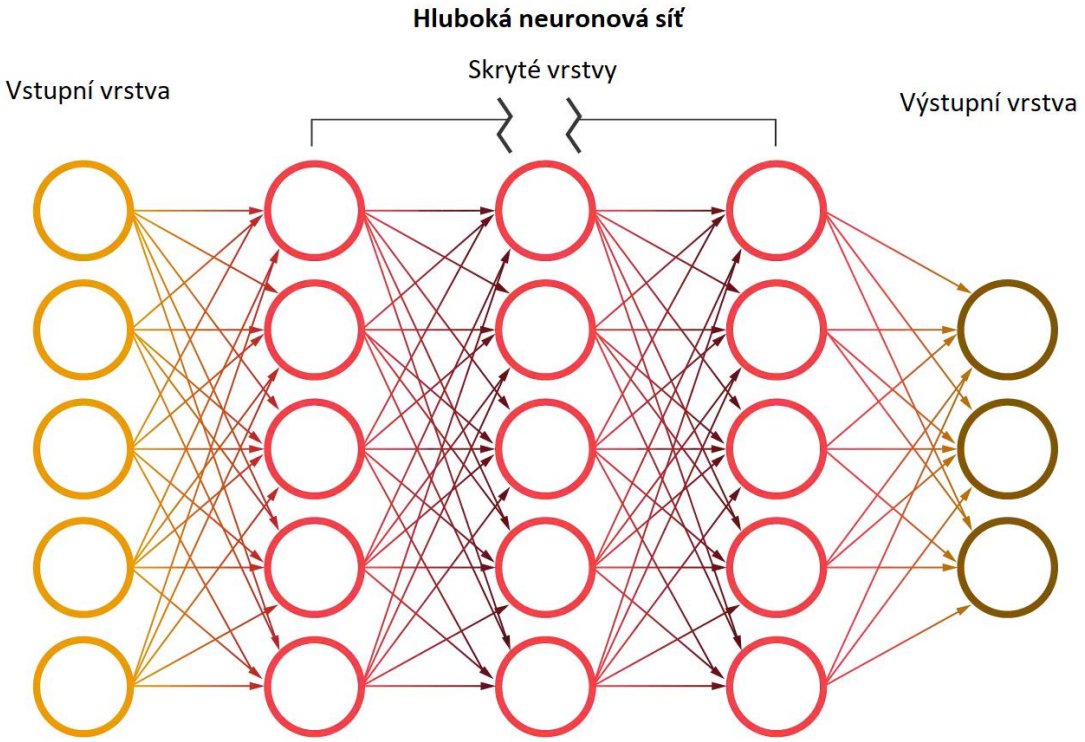



Obr. 2: Ilustrace neuronu [12]

Každý vstup do neuronu je upraven váhou a tato úprava má vliv na výpočet výstupní funkce tohoto neuronu (Obr. 2). Tudíž informace se šíří a zpracovává změnou stavů neuronů mezi vstupními a výstupními neurony a následně jsou na základě stavů neuronů upraveny váhy. Učení tedy nastává změnou vah spojující tyto neurony. Aby se síť mohla učit musíme jí zadat vstupně – výstupní pár dat, například obrazy jako vstup a jména (štítky) těchto obrazů jako výstup. Vstupně – výstupní pár je posléze srovnán s výstupem určeným neuronovou sítí a na základě správnosti tohoto srovnání jsou upraveny váhy. Tento proces projde mnohokrát přes více vstupně – výstupních párů (různých obrazů a jejich štítků), aby se docílilo co nejvyšší přesnosti. Z tohoto důvodu, když je síť nacvičena na rozpoznání určitého obrazu, tak je schopna rozpoznat tento obraz i pozměněný, či v jiném prostředí. Tento proces se nazývá generalizace.

Výpočetní bloky v neuronových sítích bývají složeny ze základních algoritmů využívaných ve strojovém učení, ale síla neuronových sítí spočívá v promyšleném spojení těchto jednotek. Návrh tohoto spojení je důležitou součástí návrhu architektury (celkového složení) neuronové sítě a její konstrukce vyžaduje hluboké znalosti tématu. Výhodou je pak vyšší přesnost správného určení výstupu, a tudíž lépe nastavené váhy. Zmíněné lepší nastavení vah vyplývá z provázanosti výstupů jednotlivých základních algoritmů. Dále z příčiny vhodného provázání jsou tyto sítě schopné řešit komplexnější matematické funkce, které samostatné modely strojového učení nejsou schopné vyřešit.

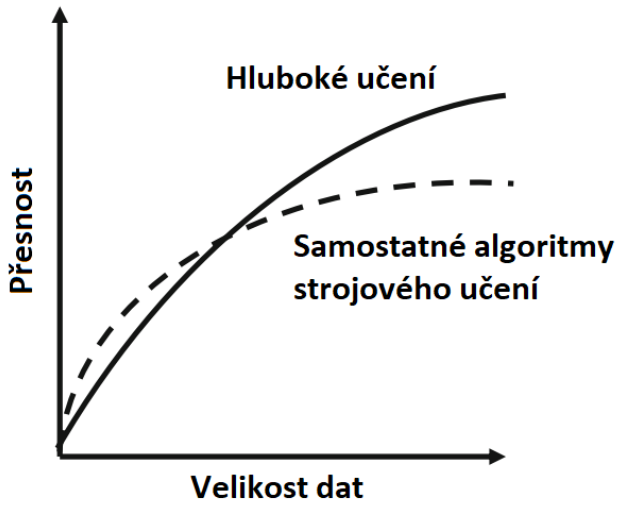
Mnoho neuronových sítí neboli jejich architektur, dosahuje vysoké přesnosti s jejich hloubkou. Hloubkou se rozumí více vrstev výpočetních bloků řazených na sebe. Tyto vícevrstvé architektury se pak nazývají hluboké neuronové sítě. Hlubokými neuronovými sítě jsou označovány pouze sítě, které jsou složeny alespoň ze tří vrstev. Tyto vrstvy se dělí na vstupní, mezilehlé, skryté a výstupní (Obr. 3). Vzájemné propojení vrstev přes neurony vzniká, když výstup jednoho neuronu je vstupem do dalšího, více dalších, či obvykle všech neuronů v jiné vrstvě, toto pravidlo platí pro všechny vícevrstvé neuronové sítě. Pokud neuronová síť má tři a méně vrstev pak je to klasická neuronová síť a neřadí se do hlubokých neuronových sítí neboli hlubokého učení [11], [12].




Obr. 3: Vrstvy hluboké neuronové sítě [13]

## 4.1 Srovnání – hluboké neuronové sítě a strojové učení

Rozvoj výpočetního výkonu dal možnost růstu využití hlubokých neuronových sítí oproti jednotlivým algoritmům strojového učení. Tyto algoritmy bývaly preferovány z důvodu nižší náročnosti na výpočetní výkon. Jelikož na jednodušších úlohách by měly specializované algoritmy poskytovat velmi dobré výsledky a měly by být méně náročné na výpočetní výkon než hluboké neuronové sítě [12].

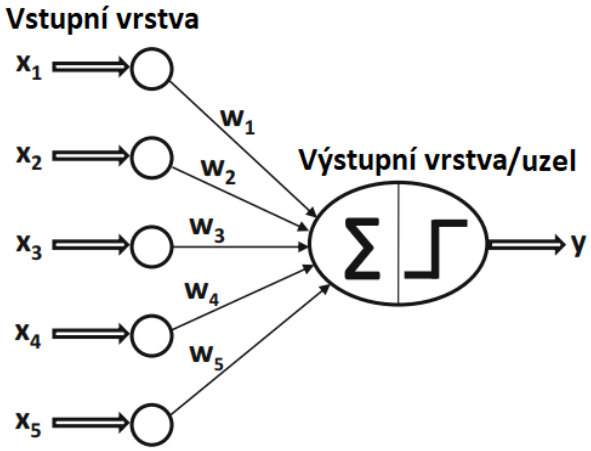



Obr. 4: Ilustrace srovnání hlubokého učení v hlubokých neuronových sítí a ostatních algoritmů strojového učení, závislost přesnosti na velikosti dat [12]

## 4.2 Jednovrstvé neuronové sítě

V jednovrstvých sítích jsou vstupy napojeny do výstupu přes lineární funkci. Toto základní zapojení sítě se nazývá perceptron (Obr. 5). Vstupní vrstva obsahuje proměnné (hodnoty) příznaků $(x_1, \dots, x_n)$ a výstupní vrstva obsahuje určenou hodnotu.

Vstupní vrstva neprovádí výpočet, ten se provádí ve výstupní vrstvě, kde je spočítána suma vstupů vynásobených váhami a výsledek sumy prochází přes aktivační funkci. Perceptron je nazýván jednovrstvou sítí, protože vstupní vrstva neuronových sítí neprovádí výpočty, a tudíž se neuvádí v celkovém počtu vrstev neuronové sítě [11], [12].




Obr. 5: Perceptron schéma [2]

## 4.3 Gradientní sestup

Gradientní sestup je iterativní optimalizace lokálního minima v diferencovatelné funkci. Postupuje se iterativně v záporném směru nejprudšího gradientu [14], [15]. Gradientní sestup je jedním z nejpoužívanějších optimalizačních algoritmů a nejběžněji používaný optimalizační algoritmus pro neuronové sítě, kde je využíván pro úpravu parametrů neboli vah této neuronové sítě [16].

V neuronových sítích je používán stochastický gradientní sestup, jelikož klasický gradientní sestup vypočítává gradient pro ztrátovou funkci celé dávky dat. To je pro hluboké sítě výpočetně velmi náročný proces. Proto se využívá stochastický gradientní sestup, který aktualizuje svoji hodnotu pro každý vstupně-výstupní pár [14], [16].

## 4.4 Aktivační funkce

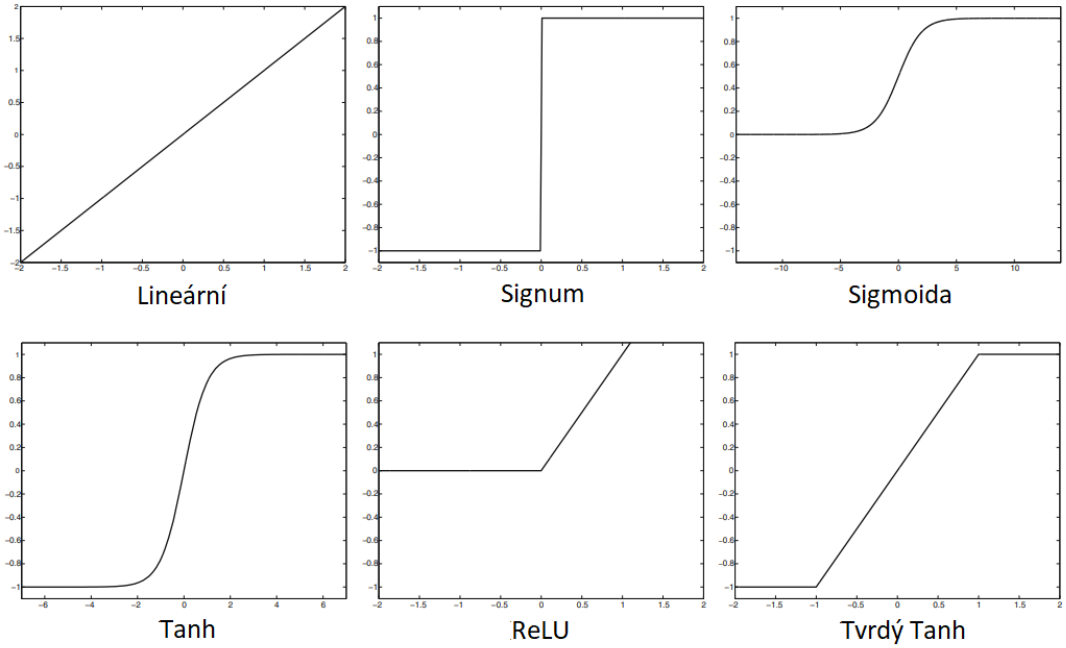
Výběr aktivační funkce je důležitá část návrhu neuronové sítě. V případě perceptronu je výběr aktivační funkce signum (znaménko) podmíněn binárním výstupem, jelikož

perceptron je vhodné používat pouze na data jež je možné lineárně rozdělit. Na datech, které nejsou lineárně dělitelná vykazuje perceptron přílišnou chybovost.

Pokud je sledována pravděpodobnost predikce výsledku, tak se volí funkce sigmoida, jelikož poskytuje výsledky v rozmezí 0 až 1.

Pro vícevrstvé neuronové sítě (architektury) je důležité použít nelineární aktivační funkce, kdyby byla použita lineární aktivační funkce, tak by tyto vícevrstvé sítě měly stejnou chybovost jako jednovrstvý perceptron.

Hodnota před aktivací se nazývá před-aktivační hodnota a hodnota po aktivaci se nazývá po-aktivační hodnota. Výstup z neuronu je vždy po-aktivační hodnota, zatímco před-aktivační hodnota se používá například při výpočtech v algoritmu zpětné propagace, který bude popsán v průběhu této kapitoly.




Obr. 6: Průběhy aktivačních funkcí neuronových sítí [12]

Dříve preferované aktivační funkce sigmoida a hyperbolický tangens (Tanh) jsou v dnešní době nahrazovány. Jednou z často používaných aktivační funkcí v dnešní době je ReLU (usměrněná lineární jednotka „rectified linear unit“), je to po částech definovaná lineární funkce, za zmínku stojí i tvrdý tangens, který je taky po částech definovanou funkcí, ale není využíván tak často jako ReLU. Průběh aktivačních funkcí je ukázán na (Obr. 6).

Velmi důležitou roli pro výběr aktivační funkce na výstupu ze sítě hraje počet uzlů, které ze sítě vystupují (počet uzlů = počet tříd vzorků, ze kterých je prováděna klasifikace). Zatímco ReLU se používá jako aktivační funkce v průběhu výpočtů v síti, nejvíce používanou aktivační funkcí pro výstup ze sítě v klasifikační úloze je funkce softmax. Její průběh je podobný funkci Tanh, ale funkce softmax roste pomaleji. Softmax

bývá zařazen jako výstupní vrstva a převádí výstup předchozí vrstvy na pravděpodobnost. Funkce softmax lze použít a je používána jako aktivační funkce při klasifikaci dat rozdělených do dvou a více tříd vzorků [12].

## 4.5 Ztrátová funkce

Výběr ztrátové funkce je důležitý pro správné definování výstupů. Například funkce softmax je vhodná při klasifikaci do více tříd. Z důvodu, že výstup funkce softmax je pravděpodobnostní tak se využívají tyto ztrátové funkce:

- Logistická regrese pro binární predikce (= dvě třídy)
- Křížová entropie pro kategorické predikce (= více tříd)

Je vhodné poznamenat, že volba typu výstupních uzlů, aktivační funkce a ztrátové funkce záleží na použití. Pro diskrétní hodnoty výstupů se nejčastěji používá aktivační funkce softmax a k tomu křížová entropie jako ztrátová funkce. [12].

## 4.6 Vícevrstvé neuronové sítě

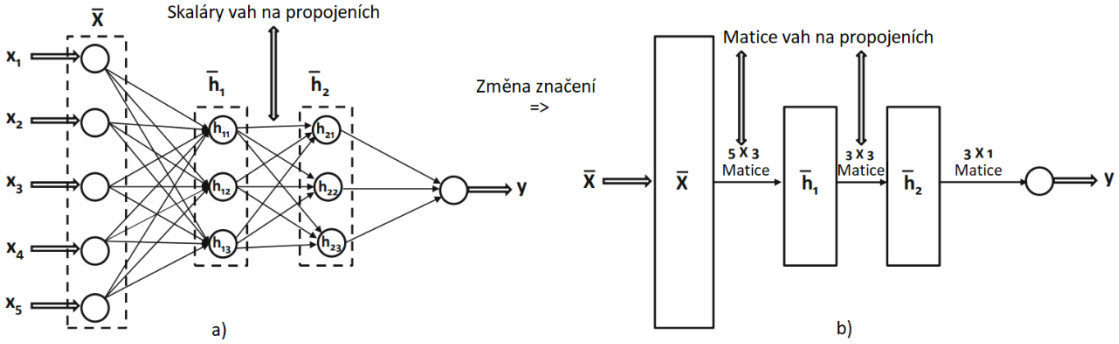
Vícevrstvé neuronové sítě jsou složeny z více jak jedné výpočetní vrstvy. Vstupní vrstva se nepočítá mezi vrstvy sítě, jelikož pouze zpracovává data a posílá je do další vrstvy a neprovádí výpočty. Vrstvy mezi vstupní a výstupní vrstvou se nazývají skryté, jelikož výpočty v těchto vrstvách nejsou pro uživatele viditelné [12].

### 4.6.1 Dopředné neuronové sítě

Převládající architektura vícevrstvých neuronových sítí se skládá z vrstev řazených postupně před sebe, proto se tyto sítě nazývají dopředné neuronové sítě. Základní architektura dopředných sítí je založena na úvaze, že všechny neurony jedné vrstvy jsou propojeny se všemi neurony vrstvy následující a informace proudí jedním směrem, dopředu neboli od vstupní vrstvy přes vrstvy skryté do vrstvy výstupní a nevyskytují se zde zpětné propojení [12][17].

### 4.6.2 Rozměr vrstvy neuronové sítě

Rozměrem, či rozlišením vrstvy neuronové sítě je myšlen počet neuronů (uzlů) v této vrstvě, znázorněno na (Obr. 7 a)) kde $h_{11}$, $h_{12}$ a $h_{13}$ jsou neurony v první vrstvě $\overline{h_1}$, a tudíž rozměr této vrstvy je vektor 3x1. Pro lepší čitelnost architektury sítě se nekreslí jednotlivé neurony, ale celé vrstvy se označují jako obdélník a váhy mezi dvěma následujícími vrstvami se označují maticí jejich rozměrů (Obr. 7 b)). Toto platí pro dvourozměrné sítě, pro trojrozměrné sítě je značení vrstvy kvádr a váhy mezi vrstvami jsou tenzory. Z tohoto spojení lze lépe vidět, že se v jedné vrstvě používá stejná aktivační funkce na všech neuronech, kdyby toto neplatilo tak by z vrstvy vystupovali různá data, která na sebe nemají návaznost. Využití stejné aktivační funkce se nevztahuje pouze na vrstvy, kde je to nutné, ale využívá se v průběhu většiny vrstev sítě [12].




Obr. 7: Značení a rozměr vrstev [12]

## 4.7 Algoritmus zpětné propagace a jeho využití v trénování sítě

Zpětná propagace se používá k výpočtu ztrátové funkce ve vícevrstvých neuronových, jelikož ztrátová funkce musí brát v potaz váhy z předchozích vrstev, a tudíž vzniká komplikovaná složená funkce. Narozdíl od jednovrstvé neuronové sítě, kde se ztrátová funkce vypočítá přímo jako funkce vah, se pro výpočet složené ztrátové funkce ve vícevrstvé síti, musí využít algoritmus zpětné propagace. Algoritmus zpětné propagace je aplikací dynamického programování pro výpočet chybových gradientů, které jsou vypočteny jako sumy lokálních gradientů přes různé cesty z určitého vstupního uzlu do určitého výstupního uzlu. Tento počet cest roste exponenciálně s počtem tříd vzorků (například s počtem tříd obrazů), jelikož je nutné vypočítat gradienty ztrátových funkcí přes všechny cesty a jejich váhy, do všech výstupních uzlů v síti, tak se využívá právě metod dynamického programování, které to jsou schopné provést výpočty s dostatečnou efektivitou. Přesněji se využívá již zmíněné metody zpětné propagace. Algoritmus zpětné propagace se dělí na dvě fáze:

- Dopředná fáze, v této fázi jsou do sítě načteny vstupy a proběhne dopředný výpočet přes všechny vrstvy sítě a použijí se aktuální váhy. Finální určený

(predikovaný) výstup je porovnán se vstupem a je vypočítána derivace ztrátové funkce. Následně je třeba přepočítat derivaci ztrátové funkce s použitím vah, které byly získány ze všech vrstev při průchodu zpětnou fází.

- Zpětná fáze, v této fázi se vypočítává gradient ztrátové funkce, gradient se získá výpočtem složené funkce přes všechny váhy použitím řetízkového pravidla diferenciálního počtu. Tento gradient je použit k nastavení nových vah. Gradient je získán zpětný průchodem sítí, a proto se tomuto procesu nazývá zpětná fáze [12].

## 4.8 Problémy v neuronových sítích / trénování

Neuronové sítě dosáhly své reputace, jelikož jsou schopny univerzálně aproximovat libovolnou funkci. K dosažení této univerzálnosti je nutné tyto sítě pro určitou úlohu natrénovat. Při trénování sítí se vyskytují problémy, které je nutno brát v úvahu, jelikož významně ovlivňují správnost fungování neuronové sítě. Hlavním z těchto problémů je přeučení neboli „Overfitting‘‘.

### 4.8.1 Přeučení (Overfitting)

Problém přeučení znázorňuje fakt, že když je síť naučena na určitém datovém celku, tak nezaručuje, že bude schopná určit správně predikce na validačních datech (validační data pocházejí ze stejných tříd jako data trénovací, ale síť se na nich netrénovala).

Typy přeučení se dělí na dva druhy:

- Velmi komplexní model nebo mnoho trénovacích tříd, ale malý počet trénovacích dat v těchto třídách. Neuronová síť se následně naučí přesně parametry určitých tříd, ale tyto parametry jsou kvůli nedostatku dat nedostatečně obecné, tudíž síť není schopna správně generalizovat a přesnost (správnost) predikce klesá.
- Velké množství trénovacích dat a příliš jednoduchý model, který není schopen dostatečně přesně určit komplexní vztahy mezi parametry a jejich třídou. Proto jsou natrénované vzory příliš generalizované, a tudíž rozdíly mezi třídami jsou nejasné a síť následně určuje špatné výsledky [12].

Jsou různé metody řešení problému přeučení a budou uvedeny u modelů neuronových sítí, které byly vybrány pro provedení testů pro tuto práci.

### 4.8.2 Problém mizejících a explodujících gradientů

Dalším významným problémem hlubokých neuronových sítí jsou mizející gradienty. Tento problém nastává s rostoucí hloubkou sítě, jelikož při používání řetízkového pravidla v algoritmu zpětné propagace se při použití velkého množství vrstev v síti narušuje stabilita přírůstků. Pak nastává jev, kdy gradienty postupně „mizí“ (když jsou přírůstky předchozích vrstev velmi malé), nebo gradienty explodují (když jsou přírůstky

předchozích vrstev velké). Toto je způsobeno výpočtem v algoritmu zpětné propagace, který exponenciálně roste, či exponenciálně klesá.

Vyjádřit tento růst či pokles lze na příkladu vícevrstvé sítě, která je složena v každé vrstvě pouze z jednoho neuronu. Tudíž každá lokální derivace v průchodu sítí je výsledkem výpočtu váhy a derivace aktivační funkce. A celková derivace vypočtena z algoritmu zpětné propagace, je tedy výsledkem těchto lokálních derivací. Z toho vyplývá, že pokud tyto lokální derivace jsou hodnoty menší než jedna, tak hodnota celkového výsledku exponenciálně klesá a gradienty „mizí“, nebo pokud lokální derivace jsou hodnoty větší než jedna, tak hodnota gradientů exponenciálně roste a gradienty „explodují“. Toto jsou pouze extrémy tohoto chování gradientů a nestabilita se vyskytuje i při hodnotě lokálních derivací rovné jedné. Z toho vyplývá, že problém mizejících a explodujících gradientů je pro hluboké neuronové sítě přirozený, a tudíž trénovací proces je zatím vždy částečně nestabilní.

Řešení tohoto problému je v mnoha architekturách různé. Vhodným řešením je například aktivační funkce ReLU, jelikož její derivace pro kladné hodnoty je vždy jedna. Další příklady vhodných řešení jsou residuální sítě (například ResNet) a dávková normalizace. Tyto řešení jsou velmi často kombinována [12].

## 4.9 Konvoluční neuronové sítě

Jsou inspirovány kočičím zrakem, kde bylo vypozorováno, že určité části zorného pole aktivují určité neurony. V konvolučních neuronových sítí je toto chování produkováno tak, že vrstvy v těchto sítích mají tři dimenze. Tyto dimenze jsou závislé na počtu příznaků. Ve vstupní vrstvě tyto příznaky jsou kanály barev a ve skrytých vrstvách jsou to informace, které obsahují informace o různých tvarech v obrazu.

Architektury založené na konvolučních sítích, obsahují vždy konvoluční vrstvy a „pooling“ vrstvy. Tyto dva typ vrstev nejsou jediné typy vrstev v architektuře sítě, ale jsou základním stavebním prvkem konvolučních sítí.

**Konvoluční vrstvy**

V konvolučních vrstvách je definována konvoluční operace, která používá filtr vah (kterému se také nazývá kernel). Tento filtr vah má tři dimenze a jeho hloubka je závislá na hloubce předchozí vrstvy a jeho prostorový rozsah je menší než prostorový rozsah vrstvy. Tento filtr je pak přesouván postupně po vrstvě a v každém posunutí je vždy vypočítán nový tensor v závislosti na typu filtru. Každý nový tensor má stejný rozměr jako filtr, který byl pro jeho výpočet použit. Na každý tento nový tensor je použita aktivační funkce. Tyto nové tensory jsou hodnoty pro další vrstvu.

**„Pooling“ vrstvy**

„Pooling“ vrstva se podobá konvoluční vrstvě v postupu výpočtu. Podobá se tím, že se použije tensor stejné hloubky jako předchozí vrstva, ale o menším prostorovém rozsahu a například pro „max pooling“ se vezme maximum z tohoto tensoru a do následující

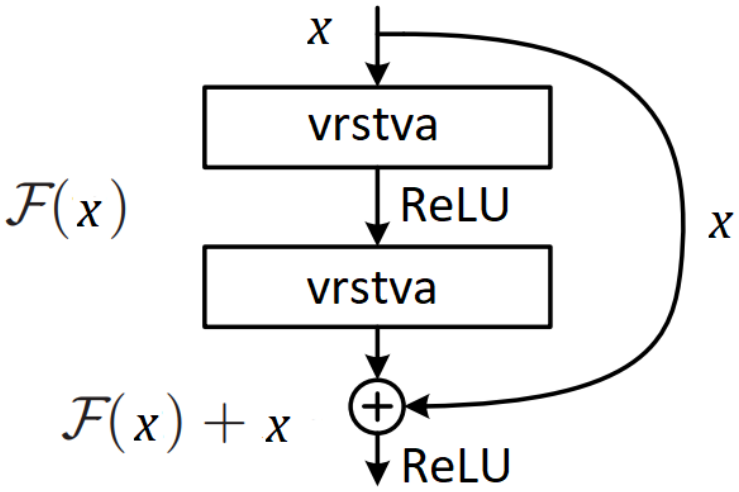
vrstvy se použije jen tato hodnota. Tento tensor se opět vypočítá pro každou hodnotu, a tudíž tensorů je více a definují následující vrstvu.

Tyto vrstvy se používají pro redukci rozměru, extrahování dominantních dat a „max pooling“ slouží i k redukci šumu [12], [18].

## 4.10 ResNet

ResNet je představitelem residuálních sítí, tyto sítě řeší problém, který nastává v určitém bodě trénování, kdy se přesnost sítě saturuje a následně rapidně degraduje (problém mizejících gradientů) a přidání vrstev vede k vyšší chybě trénování. Problém mizejících gradientů je řešen normalizací, ale toto platí pouze pro sítě v řádu desítek vrstev a pro větší sítě se opět začíná vyskytovat problém mizejících gradientů. Toto je řešeno residuálními propojeními kdy se provede propojení identitou přes více vrstev sítě.

Residuální propojení (Obr. 8) jsou založeny na myšlence, že když je neuronová síť schopna aproximovat komplikované funkce, tak je schopna aproximovat residuální funkce. Tudíž se místo aproximace funkce $H(x)$, která je výslednou funkcí předchozích vrstev, aproximuje funkce $F(x)$. Funkce $F(x)$ je tedy $H(x) - x$ a originální funkce je tedy $F(x) + x$. Tímto nahrazením funkce a připojením po určitém množství vrstev se zlepšuje přesnost s rostoucí hloubkou sítě a sítě jsou lépe optimalizovatelné, než sítě bez tohoto propojení [19].




Obr. 8: Residuální propojení (křivka propojené hodnoty $x$), residuální stavební blok sítě [19]

## 4.11 EfficientNet

Je to rodina modelů značena od B0 (základní model) až po B7 (nejpřesnější a největší model). Pro sestavení této nové architektury konvolučních neuronových sítí použito automatizované hledání neurální architektury. Tuto architekturu nazvali EfficientNet.

EfficientNet je v dnešní době architektura dosahující nejvyšší přesnosti na datasetu ImageNet a dalších [20].

### 4.11.1 EfficientNet architektura

Základní architektura sítě EfficientNet je EfficientNet-B0 (Obr 9.), skládá se z různých stavebních bloků, kde hlavním stavebním blokem je blok MBConv a doplněný o „squeeze and excitation block“. Pro dosažení největšího modelu architektury EfficientNet-B7 je poté základní architektura škálována složeným škálováním. Vhodné koeficienty složeného škálování pro síť EfficientNet-B0 jsou dle autorů [20] $\alpha = 1{,}2$, $\beta = 1{,}1$, $\gamma = 1{,}15$.

| Stav $i$ | Operátor $\hat{\mathcal{F}}_i$ | Rozlišení $\hat{H}_i \times \hat{W}_i$ | Kanály $\hat{C}_i$ | Vrstvy $\hat{L}_i$ |
|---|---|---|---|---|
| 1 | Conv3x3 | $224 \times 224$ | 32 | 1 |
| 2 | MBConv1, k3x3 | $112 \times 112$ | 16 | 1 |
| 3 | MBConv6, k3x3 | $112 \times 112$ | 24 | 2 |
| 4 | MBConv6, k5x5 | $56 \times 56$ | 40 | 2 |
| 5 | MBConv6, k3x3 | $28 \times 28$ | 80 | 3 |
| 6 | MBConv6, k5x5 | $14 \times 14$ | 112 | 3 |
| 7 | MBConv6, k5x5 | $14 \times 14$ | 192 | 4 |
| 8 | MBConv6, k3x3 | $7 \times 7$ | 320 | 1 |
| 9 | Conv1x1 & Pooling & FC | $7 \times 7$ | 1280 | 1 |

Obr. 9: EfficientNet-B0 architektura, kde každý řádek popisuje: stav $i$ o určitém počtu vrstev $\hat{L}_i$, operátor značící typ stavebního bloku každé vrstvy pro stav $i$, rozlišení značící rozlišení vstupních dat do vrstev pro stav $i$ a kanály $\hat{C}_i$ značící počet kanálů ve vrstvách pro stav $i$. FC značí „fully connected“ plně propojenou vrstvu [20]

**MBConv**

Blok MBConv je invertovaný residuální blok, invertovaný je ve smyslu, že klasický residuální blok propojuje vrstvy o mnoha parametrech, zatímco invertovaný residuální blok propojuje „bottleneck“ vrstvy neboli zúžené vrstvy, které mají menší množství parametrů než okolní vrstvy [21], [22].

**„Squeeze and Excitation“ blok**

Tento blok provede na začátku globální zprůměrování („global average pooling“) dat, tudíž provede „squeeze“ (zmáčknutí) dat.

A pak přepočítá tyto data přes plně propojenou vrstvu následovanou aktivační funkcí ReLU, další plně propojenou vrstvou a aktivační funkcí sigmoida. A převede zpětným škálováním na původní plný rozměr dat. Tudíž provede excitaci („excitation“) dat. Tento postup zaručí novou kalibraci příznaků a zlepšuje výkon sítě [23].

### 4.11.2 Složené škálování

Architektura EfficientNet dosahuje své přesnosti z důvodu implementace složeného škálování. Složené škálování staví na myšlence škálování hloubky sítě, šířky sítě a rozlišení sítě. Kde hloubka sítě je počet vrstev sítě, šířka sítě je počet uzlů ve vrstvě a rozlišení sítě je rozlišení vstupních dat (například vstupního obrazu). Jelikož škálování v jedné dimenzi sítě se potýká s problémy, které způsobují snížení přesnosti, tak je vhodné škálovat ve více dimenzích. Škálování do šířky a hloubky, ale vyžaduje náročné manuální nastavení a zvýšení přesnosti je suboptimální. Ale škálování do hloubky, šířky a rozlišení v konstantním poměru, přináší zvýšení přesnosti a náročnost škálování je nižší.

Složené škálování se skládá z parametrů $\alpha$, $\beta$, $\gamma$, které jsou škálovány koeficientem $\Phi$. Koeficienty $\alpha, \beta, \gamma$ vyjadřují hloubku sítě $d$, šířku vrstvy $w$ a rozlišení $r$. Koeficient $\Phi$ je volen v závislosti na výkonu výpočetních zdrojů. Tyto koeficienty jsou vyjádřeny těmito rovnicemi:

$$\begin{aligned} &\text{hloubka:} && d = \alpha^{\Phi} \\ &\text{šířka:} && w = \beta^{\Phi} \\ &\text{rozlišení:} && r = \gamma^{\Phi} \end{aligned} \tag{1}$$

$$\text{pro takové } \alpha, \beta, \gamma, \text{ kde platí:} \quad \alpha \,.\, \beta^2 . \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

# 5 TESTOVACÍ HARDWARE A SOFTWARE

Testování probíhalo na stroji Lenovo Legion 5 15ARH05H, CPU (centrální procesorová jednotka): AMD Ryzen 5 4600H, GPU (grafická procesorová jednotka): NVIDIA GeForce RTX 2060 6GB, RAM: 16GB DDR4-3200, Úložiště: Samsung mzvlb512hbjq (512GB SSD M.2 PCIe NVMe)

## 5.1 Výběr a implementace deskriptorů a klasifikátoru příznaků

### 5.1.1 Výběr

Pro tuto práci byly vybrány deskriptory příznaků SIFT a HOG, které byly využity s klasifikátorem SVM. Výběr byl založen na četnosti použití deskriptoru příznaku v různých projektech a na základě jejich přesnosti klasifikace obrazu.

Výběr HOG byl volen na základě těchto prací: [5], [24], [25].
V práci [24] algoritmus HOG dosahuje velmi přesných výsledků v klasifikaci obrazu.

Výběr SIFT byl volen na základě [25], kde je popsán jako algoritmus, který v porovnání s ostatními deskriptory dosahuje vyšší přesnosti.

Výběr SVM byl volen na základě [25], kde je tvrzeno, že SVM je často používaný klasifikátor pro klasifikaci obrazu. SVM dále dosahuje vyšší přesnosti klasifikace než ostatní klasifikátory příznaků [25], jiné než neuronové sítě [26].

### 5.1.2 Implementace

Pro implementaci těchto algoritmů byl využit programovací jazyk Python verze 3.8, knihovny OpenCV a Sklearn [27], [28].

V implementacích algoritmů HOG a SIFT byl vstupní obraz převeden na stupně šedi.

**HOG**
Při implementaci byly čerpány informace z oficiálních zdrojů knihovny OpenCV a [29].

**SIFT**
Při implementaci byly čerpány informace ze zdrojů [7], [30], [31], [32].

Při implementaci algoritmu SIFT, byly jeho deskriptory převedeny metodou BOVW („bag of visual words“) na histogramy deskriptorů, aby bylo možné tyto deskriptory využít v klasifikátoru SVM. K převedení na histogram byla využita metoda K-means, kterou byly sjednoceny jednotlivé deskriptory do klastrů a následně těmto klastrům byly přiřazeny označení, tyto označení jsou nutné pro algoritmus SVM (jelikož je to algoritmus učení s učitelem).

**SVM**
Při implementaci byly čerpány informace ze zdrojů [10], [28], [29]. Byla využita implementace algoritmu SVM, založena na knihovně Sklearn (Scikit-learn) a její třídě

„sklearn.svm.SVC“, která je schopna zpracovávat nelineární data, ale její časová náročnost roste exponenciálně s počtem příznaků obrazů (tedy většinou s počtem obrazů).

## 5.2 Výběr a implementace architektury neuronových sítí

### 5.2.1 Výběr

Pro tuto práci byly vybrány architektury hlubokých neuronových sítí ResNet a EfficientNet. Tento výběr byl zvolen na základě prací [19] pro ResNet a [20] pro EfficientNet. Dále bylo voleno na základě porovnání na stránce [33]. Také bylo voleno na základě autorova výběru, kde bylo bráno v potaz, že:

- EfficientNet je v dnešní době architekturou, která v různých optimalizacích dosahuje nejvyšší přesnosti na datové sadě ImageNet.
- ResNet je sice starší architektura, ale na určitých datových sadách dosahuje v různých optimalizacích nejvyšší přesnost, dále dosahuje v různých optimalizacích velmi vysoké přesnosti na datové sadě ImageNet a v roce 2015 to byla přelomová architektura, která výrazně vylepšila rozvoj neuronových sítí. Dále ResNet využívá jiné bloky než EfficientNet a využití odlišné sítě je vhodné, aby byla zajištěna větší objektivita práce.

### 5.2.2 Implementace

Pro implementaci EfficientNet a ResNet byl využit programovací jazyk Python verze 3.8, knihovny Tensorflow a Keras. Dále byl využit software Anaconda pro jednodušší instalaci knihovny Tensorflow a její GPU verze. Software Anaconda byl využit, jelikož zajišťuje kompatibilitu instalace správných verzí podpory CUDA („compute unified device architecture“) v grafických kartách NVIDIA a správné verze Tensorflow GPU balíčku.

Při implementaci architektur EfficientNet a ResNet byly čerpány informace z oficiálních zdrojů knihoven Tensorflow a Keras [34].

# 6 TESTY

Pro hodnocení přesnosti všech algoritmů byla použita kategorická křížová entropie pro výpočet ztrátové funkce a jako metrika byla použita přesnost určení.

Pro určení využití výpočetních zdrojů (systémových prostředků) byl využit program sledování prostředků a program správce úloh. Pro určení doby běhu a přesnosti predikce byl využit kód v implementacích algoritmů, které jsou součástí této práce.

Předzpracování obrazu nebylo použito, kromě výjimky testování implementace HOG a SVM, kde byly obrazy zmenšeny na rozměr 64x128 pixelů. Toto zmenšení je vhodné pro použití algoritmu HOG v základním nastavení. V ostatních implementacích obraz není předem zpracováván, jelikož normalizace by poškodila kontrast na hranách a úprava rozměru není potřeba, jelikož datová sada obsahuje obrazy shodných rozměrů (224x224 pixelů), vhodných pro použité architektury neuronových sítí i algoritmus SIFT.

**Testování s použitím neuronových sítí**

Toto testování probíhalo s použitím velikosti dávky 16 obrazů. Pro klasifikaci neuronovými sítěmi byl použit postup „transfer learning“, kdy bylo trénování na již před-trénovaných vahách zastaveno, a tudíž je trénování nepřepsalo. Dále byla přidána vrstva pro dávkovou normalizaci, vrstva provádějící globální normalizaci a plně propojená vrstva. Tyto vrstvy byly přidány na konec sítě ve vypsaném pořadí. Pro proces trénování bylo zvoleno 25 epoch, na základě [34].

Pro testování jsou použity modely EfficientNet-B0 a ResNet-50, jelikož to jsou nejmenší modely architektur poskytované knihovnou Keras. Dále EfficientNet-B0 je základní model architektury EfficientNet. A ResNet-50 je často používaný model architektury ResNet, takže je to vhodný zástupce hlubokých neuronových sítí pro porovnávání s deskriptory příznaků. Pro výpočty při testování s použitím neuronových sítí byla využita GPU.

**Testování s použitím deskriptorů příznaků a klasifikátoru SVM**

Pro toto testování nebyly použity před-trénované váhy pro algoritmus SVM a algoritmus se trénoval od začátku pouze na dodaných datových sadách.

Pro výpočty prováděné algoritmy deskriptorů příznaků a klasifikátoru SVM byla využita CPU.

## 6.1 Datové sady

Pro testování byly využity dodané datové sady obrazů $n$-úhelníků dělených do $n$ kategorií, kde $n$ nabývá velikostí od $n = 3$ po $n = 20$. Kde každá kategorie $n$ obsahuje pouze $n$-úhelníky o stejném $n$, tedy například v kategorii $n = 3$ jsou pouze $n = 3$ úhelníky (trojúhelníky). Každá tato kategorie obsahuje 1000 obrazů v rozlišení 224x224 pixelů, v náhodném škálování a náhodně rotované.

Datové sady jsou děleny na barevnou a černobílou. Barevná sada obsahuje v každé kategorii náhodné rozdělení barev, ale toto rozdělení barev se shoduje napříč kategoriemi (například obraz 1 v kategorii $n = 3$ je barevně shodný s obrazem 1 v kategorii $n = 4$ ). Černobílá sada obsahuje bílé $n$-úhelníky s černým pozadím. Obrazy $n$-úhelníků jsou štítkovány od „0001.png“ do „1000.png“ pro každou kategorii. Datové sady byly děleny v poměru 80 % pro trénování a 20 % pro validaci.

Pro implementaci SIFT, BOVW, SVM byly dodány dvě validační sady, barevná a černobílá o stejných parametrech jako předchozí sady, ale každá kategorie obsahuje pouze 250 obrazů, aby bylo dodrženo rozdělení 80 % trénovací sada a 20 % validační sada. Toto přidání validační sady sice způsobí časový nárůst trénování a testování (validace) dat, ale je částečně vyrovnáno odstraněním nutnosti konstrukce rozdělení dat při běhu implementace, jak je to provedeno v implementacích ResNet-50, EfficieNet-B0 a implementace HOG a SVM. Toto přidání validační datové sady pro implementaci SIFT, BOVW, SVM je pouze z implementačních důvodů.

## 6.2 Testování na celé datové sadě, barevná sada

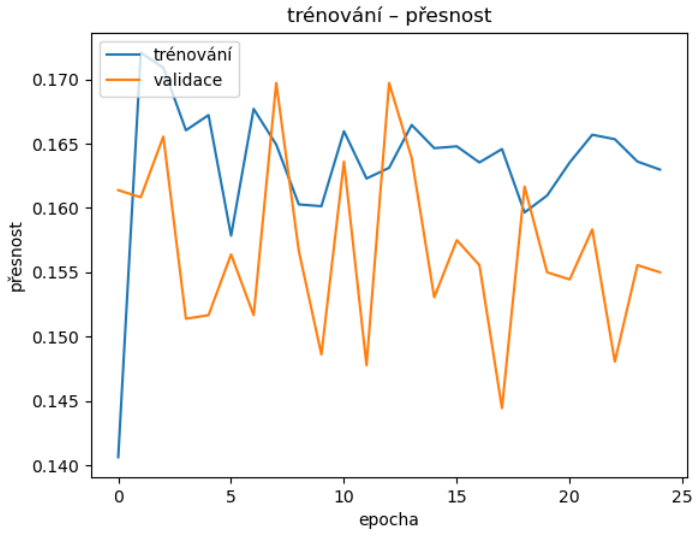
V této podkapitole jsou zobrazeny výsledky testování na celé datové sadě 18 000 barevných obrazů.

### 6.2.1 ResNet-50

Průměrné využití výpočetních zdrojů:

- CPU: 17 %,
- RAM: 2,0 GB,
- SSD: minimální (0–1 %),
- CUDA („compute unified device architecture“): 88 %,
- paměť GPU: 5,3 GB,
- Čas trénování: 1427 s (sekunda).

Celková přesnost validace: 15 %, průběh přesnosti validace v čase znázorněn na (Obr. 10).



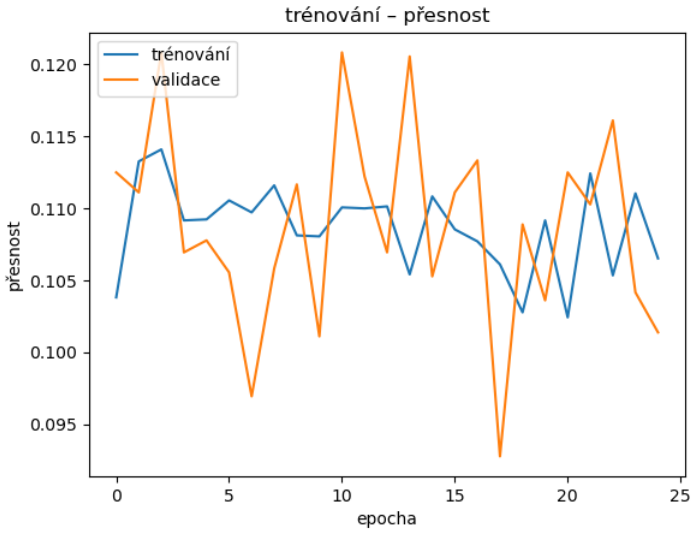

Obr. 10: ResNet-50 průběh trénování, závislost přesnosti na epoše, barevná sada

### 6.2.2 EfficientNet-B0

Průměrné využití výpočetních zdrojů:

- CPU: 18 %,
- RAM: 2,1 GB,
- SSD: minimální (0–1 %),
- CUDA: 84 %,
- paměť GPU: 5,3 GB,
- Čas trénování: 1031 s,

Celková přesnost validace: 11 %, průběh přesnosti validace v čase znázorněn na (Obr. 11).




Obr. 11: EfficientNet-B0 průběh trénování, závislost přesnosti na epoše, barevná sada

### 6.2.3 HOG a SVM

Průměrné využití výpočetních zdrojů:

- CPU: 8 %,
- RAM: 1,0 GB,
- SSD: minimální (0–1 %),
- CUDA: nebyla využívána,
- paměť GPU: nebyla využívána,
- Čas trénování: 786 s,

Celková přesnost validace: 13 %.

### 6.2.4 SIFT, BOVW, SVM

Průměrné využití výpočetních zdrojů:

- CPU: 30 %,
- RAM: 0,1 GB,
- SSD: minimální (0–1 %),
- CUDA: nebyla využívána,
- paměť GPU: nebyla využívána,
- Čas trénování: 368 s,

Celková přesnost validace: 5 %.

Celková přesnost validace této implementace na této barevné sadě má velmi nízkou vypovídací hodnotu, jelikož tato implementace má přes všechny kategorie velmi nekonstantní přesnost, která se pohybuje v rozmezí 0–100 % s průměrnou přesností validace 5,5 %. To je z důvodu, že schopnost algoritmu SIFT určit příznaky na barevné sadě je minimální, a tudíž některé kategorie $n$-úhelníků mají 0 % přesnost predikce. Tato velmi nízká schopnost určit příznaky na barvené sadě je zapříčiněna, pro algoritmus SIFT, nedostatečným kontrastem pozadí a popředí (n-úhelníku) obrazu. Testované metody na vylepšení kontrastu dostupné v knihovně OpenCV tento problém nevyřešily.

## 6.3 Testování na celé datové sadě, černobílá sada

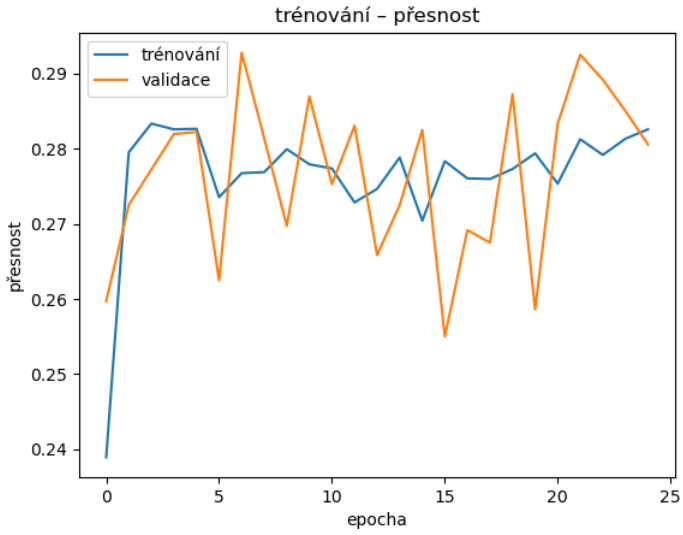
V této podkapitole jsou zobrazeny výsledky testování na celé datové sadě 18 000 černobílých obrazů.

### 6.3.1 ResNet-50

Průměrné využití výpočetních zdrojů:

- CPU: 17 %,
- RAM: 2,0 GB,
- SSD: minimální (0–1 %),
- CUDA: 88 %,
- paměť GPU: 5,3 GB,
- Čas trénování: 1430 s (sekunda),

Celková přesnost validace: 28 %, průběh přesnosti validace v čase znázorněn na (Obr. 12).



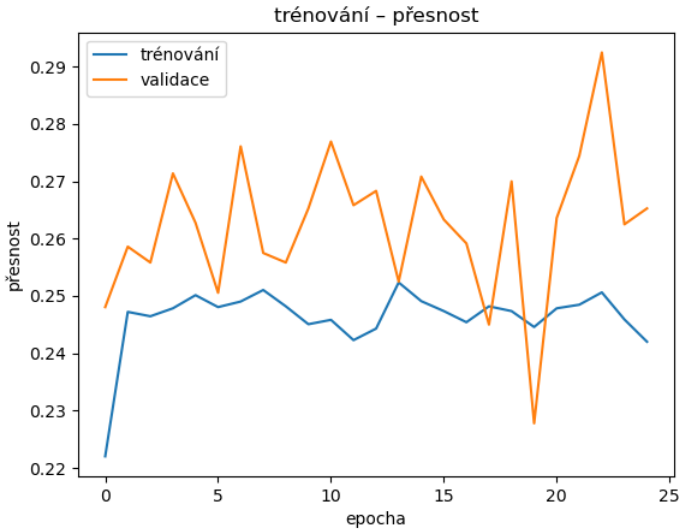

Obr. 12: ResNet-50 průběh trénování, závislost přesnosti na epoše, černobílá sada

### 6.3.2 EfficientNet-B0

Průměrné využití výpočetních zdrojů:

- CPU: 18 %,
- RAM: 2,1 GB,
- SSD: minimální (0–1 %),
- CUDA: 83 %,
- paměť GPU: 5,3 GB,
- Čas trénování: 1020 s,

Celková přesnost validace: 26 %, průběh přesnosti validace v čase znázorněn na (Obr. 13).




Obr. 13: EfficientNet-B0 průběh trénování, závislost přesnosti na epoše, černobílá sada

### 6.3.3 HOG a SVM

Průměrné využití výpočetních zdrojů:

- CPU: 8 %,
- RAM: 1,0 GB,
- SSD: minimální (0–1 %),
- CUDA: nebyla využívána,
- paměť GPU: nebyla využívána,
- Čas trénování: 767 s,

Celková přesnost validace: 13 %.

### 6.3.4 SIFT, BOVW, SVM

Průměrné využití výpočetních zdrojů:

- CPU: 35 %,
- RAM: 0,2 GB,
- SSD: minimální (0–1 %),
- CUDA: nebyla využívána,
- paměť GPU: nebyla využívána,
- Čas trénování: 367 s,

Celková přesnost validace: 6 %.

Celková přesnost validace této implementace na černobílé sadě má vyšší vypovídací hodnotu oproti sadě barevné, jelikož tato implementace má přes všechny kategorie konstantní přesnost, která se pohybuje v rozmezí 3–11 % s průměrnou přesností validace 5,6 %. To je z důvodu, že algoritmus SIFT je schopný lépe určit příznaky na černobílé sadě, díky vysokému kontrastu pozadí a popředí (n-úhelníku) obrazu.

## 6.4 Testování na částečné datové sadě, barevná sada

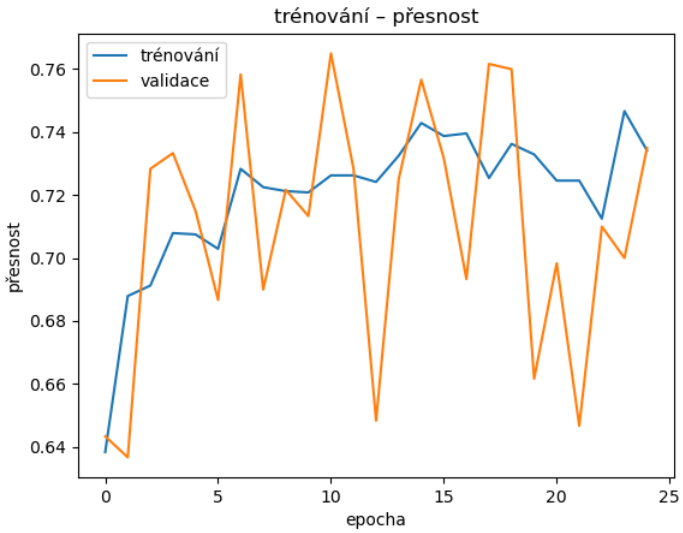
V této podkapitole jsou zobrazeny výsledky testování na částečné datové sadě, která se skládá z kategorií vybraných z celé datové sady. Tyto kategorie jsou, kategorie $n$-úhelníků kde $n = \{3, 4, 8\}$. Tudíž částečná sada se skládá z 3 000 barevných obrazů. Tento výběr byl zvolen na základě, jednoduchosti a rozdílnosti obrazů v jedné kategorii oproti obrazům v ostatních kategoriích. Ve validační sadě pro implementaci SIFT, BOVW, SVM byly vybrány stejné kategorie.

### 6.4.1 ResNet-50

Průměrné využití výpočetních zdrojů:

- CPU: 17 %,
- RAM: 2,0 GB,
- SSD: minimální (0–1 %),
- CUDA: 88 %,
- paměť GPU: 5,3 GB,
- Čas trénování: 258 s,

Celková přesnost validace: 72 %, průběh přesnosti validace v čase znázorněn na (Obr. 14).



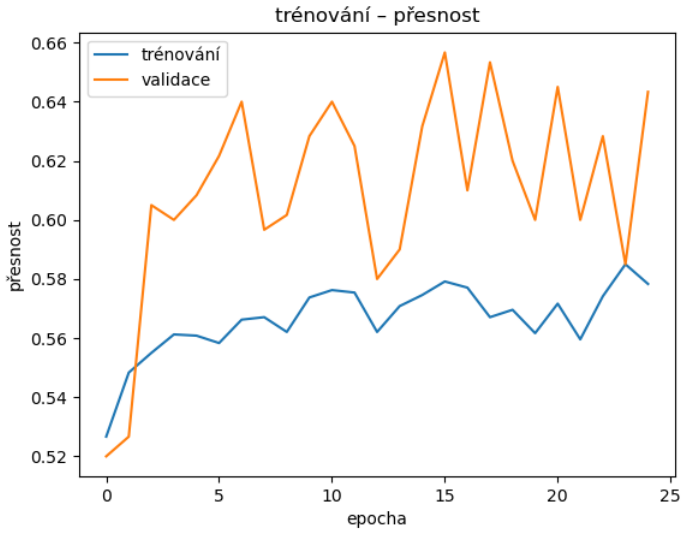

Obr. 14: ResNet-50 průběh trénování, závislost přesnosti na epoše, částečná barevná sada

### 6.4.2 EfficientNet-B0

Průměrné využití výpočetních zdrojů:

- CPU: 17 %,
- RAM: 2,1 GB,
- SSD: minimální (0–1 %),
- CUDA: 83 %,
- paměť GPU: 5,3 GB,
- Čas trénování: 186 s,

Celková přesnost validace: 62 %, průběh přesnosti validace v čase znázorněn na (Obr. 15).




Obr. 15: EfficientNet-B0 průběh trénování, závislost přesnosti na epoše, částečná barevná sada

### 6.4.3 HOG a SVM

Průměrné využití výpočetních zdrojů:

- CPU: 8 %,
- RAM: 0,2 GB,
- SSD: minimální (0–1 %),
- CUDA: nebyla využívána,
- paměť GPU: nebyla využívána,
- Čas trénování: 15 s,

Celková přesnost validace: 69 %.

### 6.4.4 SIFT, BOVW, SVM

Průměrné využití výpočetních zdrojů:

- CPU: 30 %,
- RAM: 0,1 GB,
- SSD: minimální (0–1 %),
- CUDA: nebyla využívána,
- paměť GPU: nebyla využívána,
- Čas trénování: 51 s,

Celková přesnost validace: 33 %.

Stejně jako na celé barevné datové sadě, celková přesnost validace pro tuto implementaci a částečnou barevnou sadu má velmi nízkou vypovídací hodnotu.

## 6.5 Testování na částečné datové sadě, černobílá sada

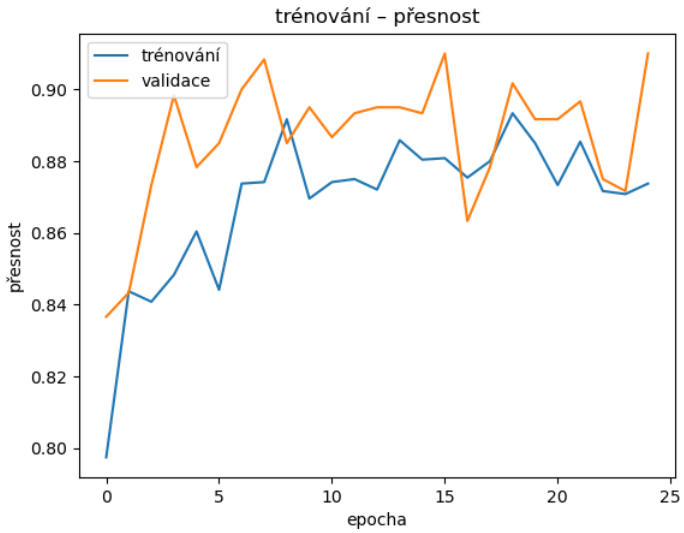
V této podkapitole jsou zobrazeny výsledky testování na částečné datové sadě, která se skládá z kategorií vybraných z celé datové sady. Tyto kategorie jsou, kategorie $n$-úhelníků kde $n = \{3, 4, 8\}$. Tudíž částečná sada se skládá z 3 000 černobílých obrazů. Tento výběr byl zvolen na základě, jednoduchosti a rozdílnosti obrazů v jedné kategorii oproti obrazům v ostatních kategoriích. Ve validační sadě pro implementaci SIFT, BOVW, SVM byly vybrány stejné kategorie.

### 6.5.1 ResNet-50

Průměrné využití výpočetních zdrojů:

- CPU: 17 %,
- RAM: 2,0 GB,
- SSD: minimální (0–1 %),
- CUDA: 88 %,
- paměť GPU: 5,3 GB,
- Čas trénování: 264 s,

Celková přesnost validace: 90 %, průběh přesnosti validace v čase znázorněn na (Obr. 16).



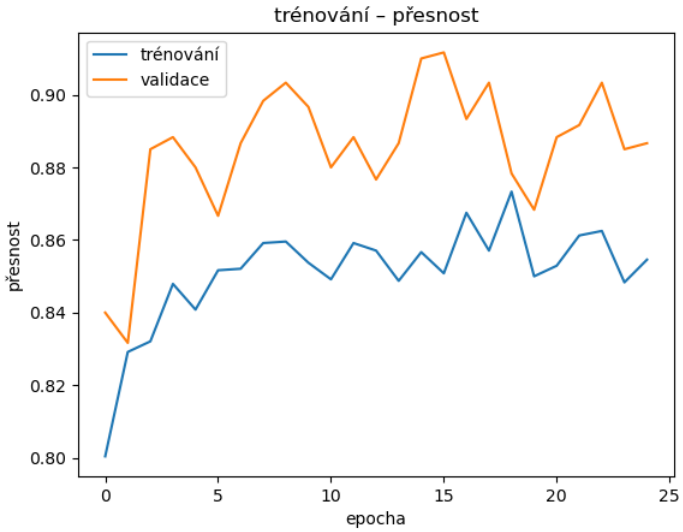

Obr. 16: ResNet-50 průběh trénování, závislost přesnosti na epoše, částečná černobílá sada

### 6.5.2 EfficientNet-B0

Průměrné využití výpočetních zdrojů:

- CPU: 17 %,
- RAM: 2,1 GB,
- SSD: minimální (0–1 %),
- CUDA: 83 %,
- paměť GPU: 5,3 GB,
- Čas trénování: 188 s,

Celková přesnost validace: 88 %, průběh přesnosti validace v čase znázorněn na (Obr. 17).




Obr. 17: EfficientNet-B0 průběh trénování, závislost přesnosti na epoše, částečná černobílá sada

### 6.5.3 HOG a SVM

Průměrné využití výpočetních zdrojů:

- CPU: 8 %,
- RAM: 0,2 GB,
- SSD: minimální (0–1 %),
- CUDA: nebyla využívána,
- paměť GPU: nebyla využívána,
- Čas trénování: 16 s,

Celková přesnost validace: 66 %.

### 6.5.4 SIFT, BOVW, SVM

Průměrné využití výpočetních zdrojů:

- CPU: 33 %
- RAM: 0,1 GB,
- SSD: minimální (0–1 %),
- CUDA: nebyla využívána,
- paměť GPU: nebyla využívána,
- Čas trénování: 57 s,

Celková přesnost validace: 31 %.

Celková přesnost validace pro tuto implementaci a částečnou černobílou sadu má dobrou vypovídací hodnotu.

# 7 SHRNUTÍ, ZHODNOCENÍ A DISKUZE

Pro shrnutí byla vytvořena tabulka (Tab. 1), která ukazuje přesnost a časovou náročnost jednotlivých implementací, na každé testované datové sadě. V porovnávání časové náročnosti je nutné brát v potaz, že výpočty v implementacích ResNet-50 a EfficientNet-B0 probíhaly na GPU a výpočty implementací „HOG a SVM“ a „SIFT, BOVW, SVM“ probíhaly na CPU.

Tab. 1: Shrnutí výsledků přesnosti implementací neuronových sítí a deskriptorů příznaků.

| | Celá datová sada | | Částečná datová sada | |
|---|---|---|---|---|
| | Barevná | Černobílá | Barevná | Černobílá |
| | přesnost [%] / čas [s] | | | |
| ResNet-50 | 15/1427 | 28/1430 | 72/258 | 90/264 |
| EfficientNet-B0 | 11/1031 | 26/1020 | 62/186 | 88/188 |
| HOG a SVM | 13/786 | 13/767 | 69/15 | 66/16 |
| SIFT, BOVW, SVM | 5/368 | 6/367 | 33/51 | 31/57 |

**Přesnost**

V průběhu testů lze vidět, že nejlepší výsledky v přesnosti dosahuje na dodaných datových sadách architektura ResNet, následovaná architekturou EfficientNet a „HOG a SVM“ implementací. Implementace „SIFT, BOVW, SVM“ dosahuje mnohem horších výsledků než ostatní implementace.

Dále je ukázáno, že přesnost implementací ResNet-50, EfficientNet-B0 je na barevné sadě znatelně nižší než na sadě černobílé a je vidět, že implementace „HOG a SVM“ na barevné sadě dosahuje vyšší přesnosti než implementace EfficientNet-B0. Přesnost implementace „HOG a SVM“ je pro barevnou i černobílou sadu téměř totožná a potvrzuje invariantnost algoritmu HOG k osvětlení neboli změně kontrastu. Porovnání přesnosti implementace „SIFT, BOVW, SVM“ mezi barevnou a černobílou datovou sadou není vhodné, jelikož tato implementace má na barevné datové sadě téměř nulovou vypovídací hodnotu, protože algoritmus SIFT v některých kategoriích barevné sady určil žádné, či velmi malé množství příznaků.

**Časová náročnost**

V tabulce shrnutí (Tab. 1) je vidět, že časová náročnost implementace „HOG a SVM“ klesá exponenciálně s klesajícím množstvím obrazů v datové sadě. A časová náročnost ostatních implementací klesá lineárně s klesajícím množstvím obrazů v datové sadě. Pro vysvětlení, celá datová sada obsahuje 18 000 obrazů a částečná datová sada obsahuje 3000 obrazů a v (Tab. 1) je vidět, že časová náročnost u ResNet-50, EfficientNet-B0 a „SIFT, BOVW, SVM“ při testování na částečné datové sadě klesla přibližně na $\frac{1}{6}$ oproti testování na celé datové sadě. Exponenciální časový pokles je způsoben časovou náročností trénování algoritmu SVM jak tvrdí [28], která se mění exponenciálně

v závislosti na počtu příznaků obrazů. Tento exponenciální pokles by měl platit i pro použití SVM v implementaci „SIFT, BOVW, SVM“, ale zde je tento rys algoritmu SVM potlačen nízkou schopností algoritmu SIFT určit příznaky v datových sadách použitých v této práci.

Nízká schopnost algoritmu SIFT určit příznaky dodané datové sady způsobila rychlý výpočet příznaků a rychlé trénovaní algoritmu SVM, a tudíž časovou náročnost implementace „SIFT, BOVW, SVM“ (Tab. 1) ovlivnila pouze konstrukce histogramů metodou BOVW, kde časová náročnost metody BOVW klesala lineárně.

**Problémy v hlubokých neuronových sítích**

Oscilace přesnosti v grafech implementací ResNet-50 a EfficientNet-B0 (Obr. 10-17), je způsobena volbou malé dávky (velikost dávky byla volena 16 obrazů). Při volbě menší dávky, síť konverguje rychleji a většinou dosahuje lepších hodnot [35], než síť s větší dávkou, ale finální výsledek je více ovlivněn šumem.

Vyšší hodnoty validační přesnosti (oranžová křivka) v porovnání s trénovací přesností (modrá křivka) v grafech (Obr. 11, 13, 15, 17) bývají způsobeny mnoha faktory. Například jak zmiňuje [36], při trénování sítí se zavádí náhodné vypuštění dat („dropout“), jako prevence před přeučením („overfitting“) a toto vypuštění dat způsobuje v síti narušení a může způsobit menší trénovací přesnost. Další problém může být nedostatečný počet vrstev v architektuře EfficientNet-B0 pro naučení vzorů v testovaných datových sadách.

Doladění („fine-tuning“) implementací neuronových sítí nebylo prováděno, jelikož doladění implementace neuronové sítě, vždy záleží na architektuře neuronové sítě a datové sadě. Dále smyslem této práce je porovnání hlubokých neuronových sítí a deskriptorů příznaků. Proto dosažení doladěné maximální přesnosti hlubokých neuronových sítí není kritické pro tuto práci.

**Problém v barevnosti**

Rozdílné výsledky hlubokých neuronových sítí na barevné a černobílé datové sadě, jsou pravděpodobně způsobeny faktem, že obrazy jsou obarveny náhodně a barva o nich nic nevyjadřuje, jelikož jsou klasifikovány tvary $n$-úhelníků v obrazech. A tyto tvary $n$-úhelníků nemají s barvou obrazu a $n$-úhelníku žádnou spojitost. Hluboké neuronové sítě tedy modelovaly i spojitost tvaru na barvě, což se projevilo jako velmi rušivý prvek.

Tento problém se v implementaci „HOG a SVM“ nevyskytuje, jelikož algoritmus HOG bere jako vstup pouze obraz v stupních šedi (tudíž byl vstupní obraz převeden do stupňů šedi) a vypočítává gradienty změny kontrastu (a ty nejsou ovlivněny barvou).

Dále mohl být problém i v nízkém kontrastu pozadí a popředí barevného obrazu, kde popředím obrazu je myšlen klasifikovaný $n$-úhelník. Tento nižší kontrast mohl způsobit nižší hodnotu vah hranových oblastí, které mají vysokou informační hodnotu o tvaru objektu. A tudíž snížit přesnost klasifikace.

**Problémy přesnosti a datové sady**

Bylo dosaženo celkově nízké přesnosti všech klasifikačních modelů, tato nízká přesnost je pravděpodobně způsobena datovými sadami, jelikož obrazy v těchto datových sadách

jsou náhodně škálovány a rotovány. Dále se v obrazech vyskytuje problém, že při škálování dosahovala minimální velikost $n$-úhelníku v obrazu i jednoho pixelu, či velmi malých hodnot pixelů. Z toho je zřejmé, že v tomto rozlišení $n$-úhelník nelze klasifikovat.

**Hluboké neuronové sítě a deskriptory příznaků srovnání**

Z (Tab. 1) jde vidět, že implementace hlubokých neuronových sítí dosahují téměř konstantně vyšší přesnosti a mají mnohem větší potenciál dosažení vyšší přesnosti. Ale implementace „HOG a SVM“ na barevné datové sadě dosahovala téměř totožné přesnosti jako implementace hlubokých neuronových sítí, přesněji dosahovala vyšší přesnosti než implementace EfficientNet-B0.

Vysoká přesnost hlubokých neuronových sítí je vykoupena, časovou náročností jejich trénování. Ale lze vidět, že časová náročnost algoritmu SVM roste s počtem obrazů exponenciálně, zatímco časová náročnost neuronových sítí rostla lineárně s počtem obrazů, a tudíž se s rostoucím množstvím dat časová náročnost mění ve prospěch hlubokých neuronových sítí. Tedy při použití velmi velké datové sady, či datové sady s obrazy o velkém rozlišení, se hluboké neuronové sítě budou trénovat rychleji než algoritmus SVM.

Algoritmus HOG dosahoval dobrých výsledků přesnosti i přes fakt, že obrazy byly rotované a algoritmus HOG není invariantní k rotaci, tudíž jeho přesnost tímto faktem byla snížena. Hlavní výhodou algoritmu HOG a tedy implementace „HOG a SVM“ bylo použití na malé datové sadě, kde byla implementace „HOG a SVM“ mnohem rychlejší než ostatní implementace.

Algoritmus SIFT dosahoval velmi nízkých výsledků přesnosti, hlavně na barevné datové sadě, kde se potvrdila jeho nedostatečná invariantnost k osvětlení neboli kontrastu.

# 8 ZÁVĚR

Bylo provedeno srovnání implementací ResNet-50, EfficientNet-B0, „HOG a SVM“, „SIFT, BOVW, SVM“ na datových sadách obrazů $n$-úhelníků. Tyto sady jsou děleny na barevnou a černobílou. Jednotlivé metody klasifikace obrazu a jejich postupy jsou popsány v průběhu práce.

Je ukázáno, že na barevné sadě dosahují implementace ResNet-50, EfficientNet-B0 a „HOG a SVM“ téměř totožné přesnosti. I přes téměř totožné výsledky přesnosti těchto tří implementací dosahovala implementace ResNet-50 nejvyšší přesnosti, druhé nejvyšší přesnosti dosahovala implementace „HOG a SVM“ a implementace EfficientNet-B0 dosahovala třetí nejvyšší přesnosti. Ve srovnání, implementace „SIFT, BOVW, SVM“ dosahovala mnohem nižší přesnosti než ostatní implementace.

Na černobílé sadě dosahovali znatelně nejvyšších výsledků hluboké neuronové sítě, přesněji implementace ResNet-50 a EfficientNet-B0. Implementace „HOG a SVM“ dosahovala nižších výsledků, ale tyto výsledky jsou pořád uspokojivé. Implementace „SIFT, BOVW, SVM“ opět dosahovala mnohem nižších výsledků přesnosti.

Znatelně nižší přesnost hlubokých neuronových sítí na barevné sadě v porovnání s černobílou sadou je s největší pravděpodobností způsobena faktem, že je klasifikován tvar $n$-úhelníků v obrazu a tento tvar nemá žádnou spojitost s barvou obrazu. Tudíž neuronové sítě modelovaly spojitost tvaru a barvy v obraze což způsobilo snížení přesnosti. Dále toto snížení přesnosti mohlo být způsobeno i nižším kontrastem pozadí a popředí barevných obrazů, který mohl působit horší určení hranových oblastí, které mají vysokou informační hodnotu o tvaru objektu.

Ve výsledku je vidět, že deskriptory příznaků s vhodně voleným klasifikátorem mohou být pro specifické využití pořád kompetitivní s hlubokými neuronovými sítěmi.

# 9 SEZNAM POUŽITÉ LITERATURY


[1] QIONG, Liu a Ying WU. Supervised Learning [online]. 2012 [cit. 2021-5-21]. Dostupné z: doi:https://doi.org/10.1007/978-1-4419-1428-6

[2] FUMO, David. Types of Machine Learning Algorithms You Should Know [online]. 15. 1. 2017 [cit. 2021-5-12]. Dostupné z: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

[3] BROWNLEE, Jason. 14 Different Types of Learning in Machine Learning [online]. 11. 11. 2019 [cit. 2021-5-12]. Dostupné z: https://machinelearningmastery.com/types-of-learning-in-machine-learning/

[4] MALLICK, Satya. Histogram of Oriented Gradients explained using OpenCV [online]. 6. 12. 2016 [cit. 2021-5-12]. Dostupné z: https://learnopencv.com/histogram-of-oriented-gradients/

[5] DALAL, Navneet a Bill TRIGGS. Histograms of Oriented Gradients for Human Detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), [online]. IEEE, 2005, 886-893 [cit. 2021-5-21]. Dostupné z: doi:10.1109/CVPR.2005.177

[6] LOWE, David G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60 [online]. 2004 [cit. 2021-5-21]. Dostupné z: doi:https://doi.org/10.1023/B:VISI.0000029664.99615.94

[7] Introduction to SIFT (Scale-Invariant Feature Transform) [online]. [cit. 2021-5-12]. Dostupné z: https://docs.opencv.org/3.4/da/df5/tutorial_py_sift_intro.html

[8] CORTES, Corinna a Vladimir VAPNIK. Support-vector networks. Machine Learning volume [online]. 1995, (20), 273–297 [cit. 2021-5-21]. Dostupné z: doi:https://doi.org/10.1007/BF00994018

[9] BERWICK, Robert. An Idiot's guide to Support vector machines (SVMs) [online]. [cit. 2021-5-12]. Dostupné z: http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

[10] RASCHKA, Sebastian. How to Select Support Vector Machine Kernels [online]. 2016 [cit. 2021-5-12]. Dostupné z: https://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html

[11] VOLNÁ, Eva. *Neuronové sítě a genetické algoritmy*. Ostrava: Ostravská univerzita, 1998. ISBN 80-7042-762-0.

[12] AGGARWAL, Charu C. Neural Networks and Deep Learning: A Textbook [online]. 1. Cham: Springer, 2018 [cit. 2021-4-18]. 1. ISBN ISBN 978-3-319-94463-0. Dostupné z: doi:https://doi.org/10.1007/978-3-319-94463-0

[13] Neural Networks [online]. [cit. 2021-04-20]. Dostupné z: https://www.ibm.com/cloud/learn/neural-networks

[14] Gradient descent. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-22]. Dostupné z: https://en.wikipedia.org/wiki/Gradient_descent#cite_note-1

[15] Gradient descent [online]. © Copyright 2017 Revision db6ab9d9 [cit. 2021-04-22]. Dostupné z: https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

[16] RUDER, Sebastian. *An overview of gradient descent optimization algorithms* [online]. 15. 9. 2016 [cit. 2021-04-22]. Dostupné z: https://arxiv.org/abs/1609.04747v2

[17] KUMAR, Niranjan. Deep Learning: Feedforward Neural Networks Explained [online]. 1. 4. 2019 [cit. 2021-4-25]. Dostupné z: https://medium.com/hackernoon/deep-learning-feedforward-neural-networks-explained-c34ae3f084f1

[18] SAHA, Sumit. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 wayv [online]. 15. 12. 2018 [cit. 2021-5-14]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53¨

[19] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. Deep Residual Learning for Image Recognition. Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, 2016, 770-778 [cit. 2021-5-21]. Dostupné z: doi:10.1109/CVPR.2016.90

[20] TAN, Mingxing a Quoc V. LE. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [online]. 11. 9. 2020 [cit. 2021-5-21]. Dostupné z: https://arxiv.org/pdf/1905.11946.pdf

[21] SANDLER, Mark, Andrew HOWARD, Menglong ZHU, Andrey ZHMOGINOV a Liang-Chieh CHEN. MobileNetV2: Inverted Residuals and Linear Bottlenecks [online]. 21. 3. 2019 [cit. 2021-5-21]. Dostupné z: https://arxiv.org/abs/1801.04381

[22] TAN, Mingxing, Bo CHEN, Ruoming PANG, Vijay VASUDEVAN, Mark SANDLER, Andrew HOWARD a Quoc V. LE. MnasNet: Platform-Aware Neural Architecture Search for Mobile [online]. 29. 5. 2019 [cit. 2021-5-16]. Dostupné z: https://arxiv.org/abs/1807.11626

[23] HU, Jie, Li SHEN, Samuel ALBANIE, Gang SUN a Enhua WU. Squeeze-and-Excitation Networks [online]. 16. 5. 2019 [cit. 2021-5-16]. Dostupné z: https://arxiv.org/abs/1709.01507v4

[24] ÖZTÜRK, Saban a Akdemir BAYRAM. Comparison of HOG, MSER, SIFT, FAST, LBP and CANNY features for cell detection in histopathological images [online]. 2018 [cit. 2021-5-21]. Dostupné z: doi:http://dx.doi.org/10.29042/2018-3321-3325

[25] BANSAL, Monika, Munish KUMAR a Manish KUMAR. 2D Object Recognition Techniques: State-of-the-Art Work. Archives of Computational Methods in Engineering [online]. 2021, (28), 1147–1161 [cit. 2021-5-21]. Dostupné z: doi: https://doi.org/10.1007/s11831-020-09409-1

[26] RACZKO, Edwin a Bogdan ZAGAJEWSKI. Comparison of support vector machine, random forest and neural network classifiers for tree species classification on airborne hyperspectral APEX images. European Journal of Remote Sensing [online]. 2017, (50:1), 144-154 [cit. 2021-5-21]. Dostupné z: doi:10.1080/22797254.2017.1299557

[27] PEDREGOSA, Fabian, Gaël VAROQUAUX, Alexandre GRAMFORT, Vincent MICHEL, Bertrand THIRION, Olivier GRISEL, Mathieu BLONDEL, Andreas MÜLLER, Joel NOTHMAN, Gilles LOUPPE, Peter PRETTENHOFER, Ron WEISS, Vincent DUBOURG, Jake VANDERPLAS, Alexandre PASSOS, David COURNAPEAU, Matthieu BRUCHER, Matthieu PERROT, Édouard DUCHESNAY (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[28] Sklearn.svm.SVC [online]. [cit. 2021-5-2]. Dostupné z: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[29] POERIO, Tony. Code for a Support Vector Machine Entry with Scikit-Learn ~ LB 1.6 (Benchmark) [online]. [cit. 2021-5-17]. Dostupné z: https://www.kaggle.com/tonypoe/svm-implementation-for-nature-conservatory/log

[30] Bag-of-words model with SIFT descriptors. Kaggle [online]. [cit. 2021-5-17]. Dostupné z: https://www.kaggle.com/pierre54/bag-of-words-model-with-sift-descriptors

[31] YALÇINER, Aybüke. Bag of Visual Words(BoVW) [online]. 12. 7. 2019 [cit. 2021-5-17]. Dostupné z: https://medium.com/@aybukeyalcinerr/bag-of-visual-words-bovw-db9500331b2f

[32] Bag of Visual Words Model for Image Classification and Recognition: Implementing Bag of Visual words for Object Recognition [online]. 13. 7. 2016 [cit. 2021-5-17]. Dostupné z: https://kushalvyas.github.io/BOV.html

[33] Image Classification on ImageNet [online]. [cit. 2021-5-17]. Dostupné z: https://paperswithcode.com/sota/image-classification-on-imagenet

[34] FU, Yixing. Image classification. Keras [online]. 30. 6. 2020 [cit. 2021-5-17]. Dostupné z: https://www.tensorflow.org/tutorials/images/classification

[35] MASTERS, Dominic a Carlo LUSCHI. REVISITING SMALL BATCH TRAINING FOR DEEP NEURAL NETWORKS [online]. 20. 4. 2018 [cit. 2021-5-20]. Dostupné z: https://arxiv.org/pdf/1804.07612.pdf

[36] D-K (https://stats.stackexchange.com/users/111209/d-k). Validation Error less than training error?: Cross Validated [online]. 2016-04-06 [cit. 2021-5-20]. Dostupné z: https://stats.stackexchange.com/q/205831

# 10 SEZNAM ZKRATEK, SYMBOLŮ, OBRAZŮ A TABULEK

**Seznam obrázků:**



**Seznam tabulek:**



**Seznam zkratek**

SIFT: scale invariant feature transform (škálovatelně invariantní transformátor příznaků)

HOG: histogram of oriented gradients (histogram orientovaných gradientů)

SVM: support vector machines (podpůrné vektorové stroje)

BOVW: bag of visual words

ReLU: rectified linear unit (usměrněná lineární jednotka)

Tanh: hyperbolický tangens

Sklearn: scikit-learn

CPU: central processing unit (centrální procesorová jednotka)

GPU: graphics processing unit (grafická procesorová jednotka)

CUDA: compute unified device architecture

RAM: random acces memory (paměť s náhodným přístupem)

SSD: solid state drive

# 11 SEZNAM PŘÍLOH

1) test_dataset_barevny
2) test_dataset_barevny_castecny
3) test_dataset_cernobily
4) test_dataset_cernobily_castecny
5) train_dataset_barevny
6) train_dataset_barevny_castecny
7) train_dataset_cernobily
8) train_dataset_cernobily_castecny
9) Implementace_HOGaSVM
10) Implementace_ResNet50_EfficientNetB0
11) Implementace_SIFT_BOVW_SVM

# PŘÍLOHY

K této práci jsou přiloženy dodané datové sady („dataset“), kde „test_dataset“ je validační datová sada a „train_dataset“ je celková datová sada. Sady „barevny_castecny“ a „cernobily_castecny“, jsou částečné datové sady.

1) test_dataset_barevny
2) test_dataset_barevny_castecny
3) test_dataset_cernobily
4) test_dataset_cernobily_castecny
5) train_dataset_barevny
6) train_dataset_barevny_castecny
7) train_dataset_cernobily
8) train_dataset_cernobily_castecny
9) Implementace_HOGaSVM
10) Implementace_ResNet50_EfficientNetB0
11) Implementace_SIFT_BOVW_SVM