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

BEZKONTAKTNÍ MĚŘENÍ ROZMĚRŮ DETERMINAČNÍCH ŠUPIN

NON-CONTACT MEASUREMENT OF THE DIMENSIONS OF DETERMINATION SCALES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Petr Šemora
AUTHOR

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

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Petr Šemora**
Studijní program:	Aplikovaná informatika a řízení
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Pavel Škrabánek, Ph.D.**
Akademický rok:	2021/22

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:

Bezkontaktní měření rozměrů determinačních šupin

Stručná charakteristika problematiky úkolu:

Měření rozměrů objektů s využitím obrazových dat je často řešeným úkolem v řadě technických i netechnických odvětví. Přesnost měření závisí jak na kvalitě a dostupnosti dat, tak i na komplexnosti měřeného objektu. V případě měření biologických objektů je často nutné aplikovat robustní metody, které umožňují realizovat měření variabilních objektů zachycených pomocí běžných komerčních fotoaparátů. Modelovým příkladem takovéto úlohy je měření rozměrů determinační šupiny ještěrky obecné.

Cíle diplomové práce:

Student vypracuje rešerši mapující techniky využívané k měření rozměrů objektů a rešerši mapující techniky využívané k segmentaci obrazu. Student navrhne a implementuje systém pro měření rozměrů determinačních šupin ještěrky obecné na základě obrazových dat, anotuje zadanou datovou sadu, a vyhodnotí efektivitu vytvořeného řešení.

Seznam doporučené literatury:

GONZALEZ, Rafael C. a Richard E. WOODS. Digital image processing. New York, NY: Pearson, [2018]. ISBN 978-0133356724.

LATEEF, Fahad a Yassine RUICHEK. Survey on semantic segmentation using deep learning techniques. Neurocomputing [online]. 2019, 338, 321-348 [cit. 2019-09-03]. DOI: 10.1016/j.neucom.2019.02.003. ISSN 09252312. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S092523121930181X.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Tato práce se zabývá bezkontaktním měřením rozměrů análního štítku ještěrky obecné. V práci jsou nejprve stručně shrnuty techniky využívané k měření rozměrů objektů a techniky využívané k segmentaci obrazu. Následně práce poskytuje základní shrnutí o neuronových sítích a konvolučních neuronových sítích. V praktické části je popsán systém pro měření rozměrů análního štítku ještěrky obecné. Navržené algoritmy jsou implementovány v grafickém uživatelském rozhraní umožňující automatické i ruční měření.

ABSTRACT

This thesis deals with non-contact measuring the dimensions of the sand lizard anal plate. First the thesis briefly summarizes the techniques used to measure object dimensions and the techniques used for image segmentation. Subsequently, the thesis provides a basic overview of neural networks and convolutional neural networks. The practical part describes a system for measuring the dimensions of the sand lizard anal plate. The proposed algorithms are implemented in a graphical user interface enabling automatic and manual measurements.

KLÍČOVÁ SLOVA

umělá inteligence, konvoluční neuronové sítě, detekce objektů, bezkontaktní měření rozměrů, počítačové vidění, segmentace obrazu, Python, TensorFlow

KEYWORDS

artificial intelligence, convolutional neural networks, object detection, non-contact dimensional measurement, computer vision, image segmentation, Python, TensorFlow

ÚSTAV AUTOMATIZACE A INFORMATIKY

2022

BIBLIOGRAFICKÁ CITACE

ŠEMORA, Petr. *Bezkontaktní měření rozměrů determinačních šupin*, Brno. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, 2022. Diplomová práce. Vedoucí diplomové práce: Ing. Pavel Škrabánek, Ph.D.

PODĚKOVÁNÍ

Děkuji vedoucímu práce Ing. Pavlu Škrabánkovi, Ph.D. za jeho ochotu, čas, cenné rady a připomínky při vytváření závěrečné práce. Poděkování patří také Ing. Radovanu Smolinskému, Ph.D. et PhD. za ochotu a čas při konzultacích.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 20. 5. 2022 ……

Petr Šemora

OBSAH

1 ÚVOD

Vznik ještěrů se klade do období prvohorního permu, tedy do doby před zhruba 260 miliony let [1]. I přes to, že jsou tito živočichové podrobováni mnoha výzkumům, stále o nich lidstvo zná velice málo informací. Jedním z nejrozšířenějších druhů ještěrů v České republice je ještěrka obecná, která se projevuje tím, že se v průběhu jejího života mění její zbarvení v závislosti na věku a pohlaví [2]. Kromě toho dochází k asymetrii růstu různých tělesných částí, včetně šupin [3]. Mláďata této ještěrky rostou podobně rychle a v podobných velikostech bez ohledu na její pohlaví. Od určitého roku jejího života se však vývoj změní. Samcům začne mnohem rychleji a více růst hlava, končetiny a ocas. Samice naopak investují více energie do růstu délky těla mezi končetinami, aby zvětšily kapacitu tělesné dutiny kvůli budoucímu počtu vajec [4]. Tato vlastnost se nazývá alometrie růstu. Rozdílný růst jednotlivých částí těla v průběhu jejího života by se měl týkat i šupin. Konkrétně vědci předpokládají, že velikost šupin by se v průběhu života měla měnit, jejich tvar by měl však zůstat stejný. Jako nejvhodnější determinační šupinou pro potvrzení, či vyvrácení tohoto předpokladu byl zvolen anální štítek. Jedná se o velkou a širokou šupinu na ventrální straně, kterou obvykle obklopuje jedna řada preanálních drobnějších šupin [2]. Index análního štítku určuje míru protáhlosti této šupiny v příčném směru a lze jej vypočítat jako poměr šířky a délky. Právě tyto dvě vzdálenosti je důležité co nejpřesněji změřit. Pokud by se předpoklad vědců potvrdil, mohl by anální štítek sloužit jako jednoznačný identifikátor těchto ještěrů, podobně, jako je tomu v případě otisku prstů u lidí.

Úloha měření rozměrů análního štítku ještěrky obecné je modelovým příkladem, ve kterém je nutné aplikovat robustní metody umožňující realizovat měření variabilních objektů zachycených pomocí běžných komerčních fotoaparátů. Přesnost bezkontaktního měření závisí jak na kvalitě a dostupnosti dat, tak i na komplexnosti implementovaného řešení.

Předkládaná diplomová práce nejprve stručně uvádí techniky využívané k měření rozměrů objektů. Poté jsou popsány vybrané metody využívané k segmentaci obrazu. Následuje kapitola o umělých neuronových sítích, na kterou navazuje kapitola o konvolučních neuronových sítích. Poté je rozebrán a popsán systém pro automatické měření rozměrů análního štítku a vyhodnoceny získané výsledky implementovaného řešení.

2 TECHNIKY VYUŽÍVANÉ K MĚŘENÍ ROZMĚRŮ OBJEKTŮ

Měření rozměrů objektů je často řešeným úkolem v řadě technických i netechnických odvětví. Rozměrem je zde myšlena vzdálenost mezi krajními body objektu v určitém směru. V této kapitole jsou popsány některé typy měření rozměrů v rovině a následně vysvětlen a popsán základní princip měření rozměrů v rovině pomocí počítačového zpracování obrazu.

2.1 Typy měření rozměrů v rovině

Mezi základní měřicí systémy patří ruční měřicí zařízení, která jsou velmi variabilní svým měřicím rozsahem i měřenou přesností. Kontaktní měřicí systémy byly ještě donedávna v praxi používané velmi často. S nastupující digitalizací, automatizací a vývojem nových technologií se však do popředí více dostaly bezkontaktní měřicí systémy, které vynikají například kvalitnějšími výstupy či lepší schopností vytvoření hustějších bodů v místech, kde se měřící dotyk těžko dostane. [6]

2.1.1 Ruční měřicí zařízení

Pravítka, metry a pásma

Pravítka se stupnicí patří mezi základní zařízení určená k měření rozměrů. Podle požadavků obsluhy lze využít například pravítka dřevěná, ocelová či plastová. Pro měření delších rozměrů jsou určeny metry. V dnešní době se lze nejčastěji setkat s metry svinovacími a skládacími. Pro měření velkých rozměrů se nejčastěji využívají měřicí pásma, rolmetry (tzv. odvalovací kolečka) nebo různé laserové dálkoměry.

Posuvná měřítka

Posuvné měřítko slouží pro měření různých rozměrů součástek a uplatňuje se především ve strojírenství a průmyslu. Běžné posuvné měřítko umožňuje měřit rozměry do 150 mm. Existují však měřítka umožňující měření i s větším rozsahem. Přesnost analogových posuvných měřítek se pohybuje do 0,02 mm. Digitální posuvná měřítka umožňují měření obvykle s přesností 0,01 mm. [5]

Mikrometry

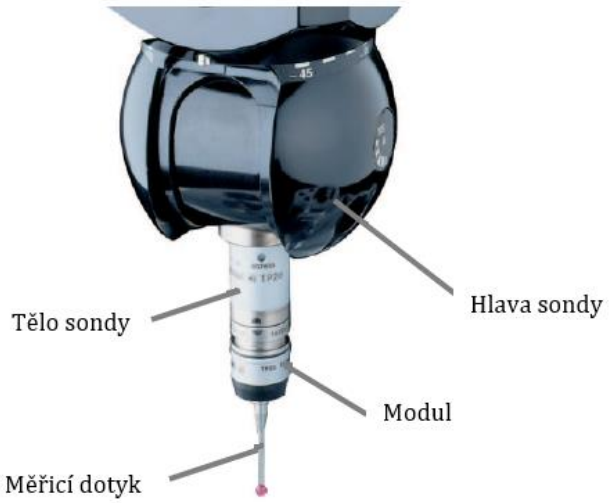
Mikrometry jsou měřicí přístroje, které umožňují změřit malé předměty s vysokou přesností. Tato měřidla disponují mikrometrickým šroubem, jehož vyšroubováním se určuje měřený rozměr. Analogové mikrometry disponují s přesností obvykle 0,01 mm a digitální až 0,001 mm. [5]

2.1.2 Kontaktní měřicí zařízení

Kontaktní měření se vyznačuje zejména tím, že je nutný fyzický kontakt měřicího přístroje s měřenou součástí. Nejsou tedy vhodné pro všechny měřené povrchy, neboť při jejich dotyku s měřeným objektem může dojít k poškození povrchu měřené součásti. Oproti bezkontaktním měřicím systémům disponují i nižší rychlostí měření.

Systém spínacího typu

Jak je vidět na obrázku 1, hlavními částmi spínací sondy je tělo, modul a dotyk. Modul je k tělu obvykle přichycen pomocí magnetu, což umožňuje jeho automatizovanou výměnu a zároveň jej chrání před nárazem. Tělo sondy je zašroubováno do hlavice. [6]



Obr. 1: Dotyková spínací sonda [6]

Elektromechanická sonda obsahuje uvnitř modulu elektrický obvod s rozpínacími kontakty. Tyto kontakty jsou tvořeny kulovými plochami, které jsou vzájemně pootočeny o 120°. V klidovém stavu propojuje všechny kontakty elektricky vodivý prstenec. Odečtení polohy nastane v momentě vychýlení měřicího dotyku ze základní polohy a tím rozpojení jednoho z kontaktů. Při měření dochází k drobné chybě způsobené vychýlením měřicího dotyku a současnému posunu osy stroje oproti okamžiku skutečného kontaktu měřicího dotyku s měřeným povrchem. Této chybě lze předejít použitím **piezoelektrické sondy** využívající schopnosti krystalu generovat elektrické napětí při jeho deformaci. V případě malého vychýlení měřicího dotyku ze základní polohy, dojde uvnitř sondy k deformaci piezokrystalů, což umožní generování elektrických impulsů a nastane odečtení a dočasné uložení souřadnic. Při větším vychýlení dojde k elektromechanickému rozpojení elektrického kontaktu, nastane zastavení stroje a uložení souřadnic. Nevýhodou piezoelektrických sond je jejich vysoká citlivost. Mohou tedy vydat signál i při náhodném zachvění stroje. [6]

Systém snímacího typu

Příkladem systému snímacího typu je tzv. **skenující sonda**. Tato sonda se skládá z těla umístěného v pouzdře snímací hlavy, které je uloženo ve dvojitém nebo trojitém pružném paralelogramu. Při snímání je měřicí hrot v neustálém kontaktu s měřeným objektem. Řídicí systém v okamžiku zaznamenání kontaktu hrotu s měřeným objektem vypne přítlak vyvozující nastavenou měřicí sílu a zapne pohony pro příslušné osy. Skenující sondu lze provozovat ve statickém režimu diskrétního snímání, nebo v dynamickém režimu spojitého snímání. [6]

2.1.3 Bezkontaktní měřicí zařízení

Bezkontaktní měření rozměrů objektů se vyznačuje tím, že nevyžaduje přímý dotyk měřicího přístroje s měřeným objektem. Uplatňuje se zejména při měření velmi malých objektů, které mohou být snímány například pomocí mikroskopu, nebo naopak při měření velmi velkých či vzdálených objektů. Bezkontaktní měření lze použít i v oblastech, ve kterých je dotykové měření nevhodné. Důvodem může být například poškození objektu při dotyku s měřicím přístrojem.

Optické mikrometry jsou založeny na principu analýzy množství dopadajícího světla od vysílače, kterým je obvykle laser nebo LED (*light-emitting diode*) k přijímači, kterým je nejčastěji CMOS (*complementary metal oxide semiconductor*) nebo CCD (*charge-coupled device*) snímač. U jednoduchých provedení se měří jen množství zacloněného světla měřeným předmětem umístěného mezi vysílač a přijímač. U složitějších provedení se na matici snímače vyhodnocuje zastínění jednotlivých pixelů, čímž lze zjistit rozměr měřeného předmětu na jednotky mikrometrů. [7]

Při menších požadavcích na přesnost určení rozměrů se lze spokojit s rozlišením, které je dáno rozměrem jednotlivých pixelů snímače (běžně kolem 14 μm). Pomocí komparátoru (realizovaného např. operačním zesilovačem) potom stačí porovnat každou analogovou hodnotu odpovídající jasu jednotlivých pixelů s nastavenou analogovou rozhodovací úrovní. Výstup z komparátoru představuje pro každý pixel logickou informaci o přítomnosti snímaného objektu. Pro digitalizaci signálu ze snímače lze využít i analogově digitální převodník. Digitalizované hodnoty jasů jednotlivých pixelů lze k určení pozice hrany použít se subpixelovým rozlišením, tedy s rozlišením lepším, než je rozměr jednoho pixelu snímače. [10]

Ultrazvukové systémy využívají pro generování ultrazvukových vln pulsně-akustickou techniku. Tyto vlny se šíří na frekvencích mimo člověkem slyšitelné spektrum [8]. Metody se vyznačují vysokou spolehlivostí a univerzalitou. Zařízení jsou obvykle velmi robustní, takže jsou vhodné i do náročných podmínek průmyslových provozů. [9]

Pneumatické systémy měří rozdíl tlaku vzduchu mezi měřicí hlavou a měřeným objektem. Tento rozdíl je poté převeden na elektrický signál. Tyto systémy disponují poměrně nízkou přesností. Hlavní nevýhodou je možný výskyt tzv. nezměřitelných zón, které se kvůli konečnému průměru senzorové hlavy mohou vyskytnout. [8]

Elektrické systémy využívají principu elektrického odporu, kapacity a vířivých proudů. Pro využití těchto principů však musí být měřený objekt z vodivého materiálu. Pro měření v rozsahu 32–54 mm se přesnost pohybuje do 15 μm. [8]

2.2 Měření rozměrů pomocí počítačového zpracování obrazu

Pro měření rozměrů pomocí počítačového zpracování obrazu je potřeba nejprve získat snímek reálné scény, který je následně třeba převést do digitální formy. Na zdigitalizovaný snímek je obvykle nutné aplikovat různé metody pro zpracování obrazu a upravit jej tak do podoby vhodné pro další manipulaci. Po jeho úpravě lze ze snímku extrahovat důležité příznaky nezbytné pro pochopení jeho obsahu a získání informací pro změření rozměrů a vyhodnocení přesnosti.

2.2.1 Vznik obrazu

Nejběžnějšími zařízeními pro snímání obrazu reálné scény jsou kamery, které lze podle typu snímače rozdělit na kamery s CCD čipem a CMOS čipem.

CCD snímač vznikne uspořádáním fotodiod do matice. Snímač je doplněn o řídicí elektroniku, horizontální posuvný registr a převodník pro převod náboje na napětí. Světlo dopadající na čip vytváří v polovodičích elektrony, které jsou zachytávány do buněk, tzv. „potenciálových jam“. Každá buňka reprezentuje jeden obrazový bod, který se označuje jako pixel. Množství náboje uloženého v jednom obrazovém bodu je dáno délkou expozice, tedy času, po který je čip vystaven světlu. Pro čtení informací z čipu lze náboje přesouvat po ploše čipu. Režimy přenosu náboje lze rozdělit na přenos všech sloupců naráz, nebo přenos po sloupcích. [11]

CMOS snímač je konstrukčně složitější, jelikož obvody pro digitalizaci se nacházejí přímo na čipu. Každá fotodioda obsahuje vlastní převodník náboje na napětí. Ke čtení informací dochází přímo z buněk, není tak třeba žádný posuvný registr. Jejich výhodou je vyšší rychlost čtení, nižší spotřeba energie, nižší výrobní náklady a možnost číst jen část čipu. Naopak nevýhodou je větší temný šum. [12]

2.2.2 Vznik digitálního obrazu

Výstupem ze snímačů je analogový signál, který je třeba pro zpracování na počítači digitalizovat. Obraz projektovaný do obrazové scény lze popsat pomocí obrazové funkce, která je spojitá a obvykle je dána vztahem:

$$f(x,y) = i(x,y)r(x,y), \tag{2.1}$$

kde (x,y) jsou prostorové souřadnice, $i(x,y)$ je množství záření dopadajícího na scénu a $r(x,y)$ je odrazivost objektu.

Původní spojitý obraz je vzorkováním a kvantováním převeden do čtvercového rastru o rozměrech $M \times N$ vzorků, který je funkcí diskrétních souřadnic (x,y) kde $x = 0,1,\ldots,M-1$ a $y = 0,1,\ldots,N-1$. Vzorkovací frekvence se řídí Nyquistovým

teorémem, podle kterého by měla být vzorkovací frekvence f_s minimálně dvojnásobkem nejvyšší frekvence f_{max} nacházející se v obraze, tedy $f_s \geq 2f_{max}$. [13]

Spojitá jasová stupnice je obvykle rozdělena do 2^k stejných intervalů, kde $k = 8$ bitů, tj. 256 jasových úrovní pro monochromatický obraz. Tomuto procesu se říká kvantování, při kterém vždy dochází k nevratné ztrátě části informace. Jeden vzorkovací bod představuje jeden obrazový element zachyceného obrazu. Celkové množství takto získaných vzorků určuje prostorové rozlišení výsledného obrazu.

Pro numerické zpracování obrazu je vhodná reprezentace pomocí 2D numerického pole, které lze popsat vztahem:

$$f(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(0,N-1) \\ \vdots & \ddots & \vdots \\ f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.2}$$

kde element matice $f(i,j)$ odpovídá jasovým hodnotám pixelu na souřadnicích (i,j). Pro spektrální obraz reprezentovaný RGB[1] modelem obsahuje jeden element vektor se třemi prvky. [13]

2.2.3 Základní vztahy mezi pixely

Nechť p, q a z jsou tři body definované v rovině, pak lze funkci D označit vzdáleností, jsou-li splněny podmínky (2.3) až (2.5):

$$D(p,q) \geq 0, \text{přičemž } D(p,q) = 0 \Leftrightarrow p = q, \tag{2.3}$$

$$D(p,q) = D(q,p), \tag{2.4}$$

$$D(p,z) \leq D(p,q) + D(q,z). \tag{2.5}$$

Nejčastěji je pro svoji názornost používaná **Euklidovská vzdálenost**, která je však výpočetně náročná a často vede na neceločíselné hodnoty. Pro Euklidovskou vzdálenost s pixely p a q na souřadnicích (l,m) a (s,t) platí rovnice (2.6). V případě použití **Manhattanské vzdálenosti** jsou povoleny pouze kroky v horizontálním a vertikálním směru a platí pro ni rovnice (2.7). Jsou-li povoleny i kroky v diagonálním směru, pak se jedná o **Čebyševovu vzdálenost**, pro kterou platí rovnice (2.8). [14]

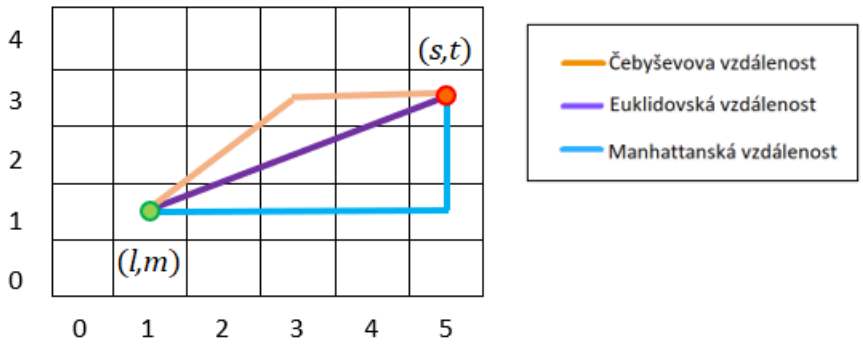
Zmíněné vzdálenosti jsou zobrazeny na obrázku 2 a popsány následujícími rovnicemi:

$$D_E(p,q) = \sqrt{(l-s)^2 + (m-t)^2} \tag{2.6}$$

$$D_4(p,q) = |l-s| + |m-t| \tag{2.7}$$

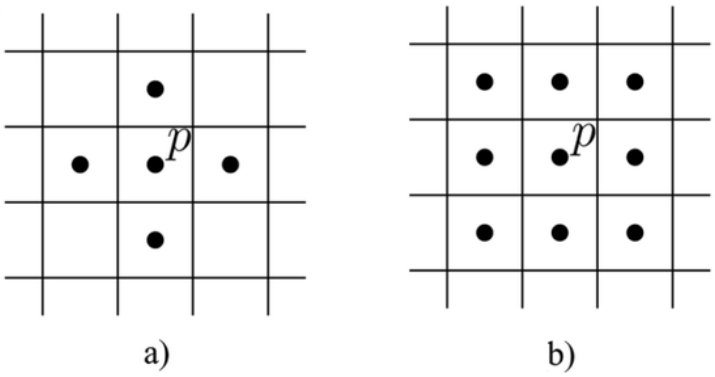
$$D_8(p,q) = \max\{|l-s|, |m-t|\} \tag{2.8}$$

[1] Z anglického *red* (červená), *green* (zelená), *blue* (modrá).



Obr. 2: Znázornění vybraných vzdáleností [15]

Sousedy pixelu p lze označit všechny pixely, které jsou ve vzdálenosti jedna od pixelu p. Při použití Manhattanské vzdálenosti má pixel p celkem čtyři sousedy, jejichž množinu lze označit jako **čtyř-sousedé**, tedy $N_4(p)$. Při použití Čebyševovy vzdálenosti má pixel p celkem osm sousedů, jejichž množinu lze označit jako **osmi-sousedé**, tedy $N_8(p)$. Princip sousedních pixelů zobrazuje obrázek 3.



Obr. 3: a) čtyř-sousedé pixelu p, b) osmi-sousedé pixelu p [16]

Pixely p a q s hodnotami intenzity, které náleží do množiny hodnot intenzity definující sousedství, jsou **čtyř-sousední** (respektive **osmi-sousední**), pokud $q \in N_4(p)$ (respektive $q \in N_8(p)$).

Cestu z pixelu p na souřadnicích (l, m) do pixelu q na souřadnicích (s, t) lze definovat podle vztahu:

$$(x_0, y_0), (x_1, y_1), (x_i, y_i) \ldots, (x_n, y_n), \tag{2.9}$$

kde $(x_0, y_0) = (l, m), (x_n, y_n) = (s, t)$, n je délka cesty z p do q a platí, že $(x_i, y_i) \neq (x_j, y_j)$ pro $\forall i, j \in \langle 1, \ldots, n \rangle$, kde $i \neq j \wedge (x_i, y_i)$ a (x_{i-1}, y_{i-1}) jsou sousední pixely pro $\forall i \in \langle 1, \ldots, n \rangle$.

Pixely p a q jsou souvislé v podmnožině pixelů obrazu $f(x, y)$, jestliže mezi nimi existuje cesta, která se skládá pouze z pixelů z této podmnožiny. Souvislou množinu pixelů lze označit jako **oblast**. Sousední oblasti jsou takové, jejichž sjednocením vznikne souvislá množina. [14]

3 TECHNIKY VYUŽÍVANÉ K SEGMENTACI OBRAZU

Segmentace obrazu je proces dělení obrazu do jednotlivých segmentů podle předem definovaných kritérií. Každému obrazovému elementu je přiřazena hodnota vyjadřující index segmentu. Cílem segmentace je příprava vstupního obrazu a jeho převedení do snadněji analyzovatelného stavu určeného pro jeho pochopení. Výsledkem segmentace je soubor vzájemně se nepřekrývajících oblastí se společnými vlastnostmi. Existuje mnoho možných přístupů pro segmentaci obrazu. Tato kapitola poskytuje výběr pouze základních použitelných technik. [18]

3.1 Prahování

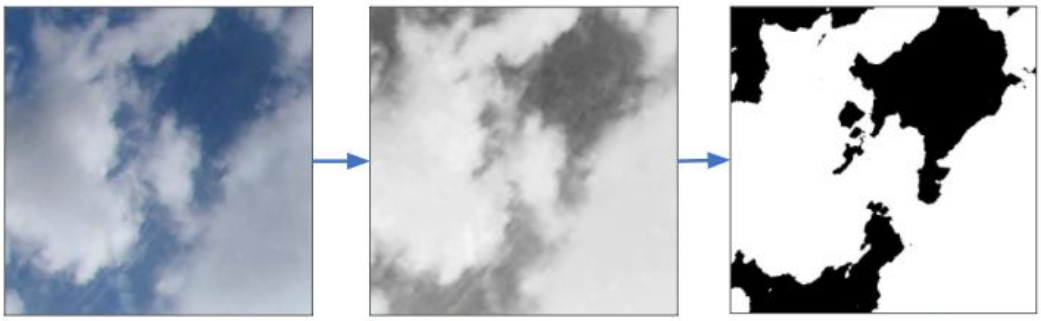
Prahování je rozdělení oblastí podle úrovně intenzity jasu. Hlavní myšlenkou je rozdílná úroveň intenzity jasu objektů v pozadí a v popředí. Jak je vidět na obrázku 4, původní obraz je obvykle nejprve převeden do odstínů šedi, na který je poté aplikováno prahování, které lze popsat vztahem:

$$g(i,j) = \begin{cases} 1 \text{ pro } f(i,j) > T \\ 0 \text{ jinak} \end{cases} \tag{3.1}$$

kde $g(i,j)$ je obrazový element výstupního obrazu, $f(i,j)$ je obrazový element vstupního obrazu a T je hodnota prahu. Obrazový element, který má větší hodnotu než práh, je určen jako pixel popředí. Všechny ostatní pixely jsou naopak považovány za pixely pozadí. Výběr vhodné hodnoty prahu není v některých případech vůbec snadný. Hodnotu této hranice lze často zjistit pomocí histogramu obsahující informace o počtu pixelů v obraze s konkrétní hodnotou šedi. [17]

Jak je uvedeno v [17], prahování lze použít několika způsoby. V případě globálního prahování je určena konstantní hodnota prahu pro celý obraz. Na podobném principu je založeno i procentní prahování, které pracuje s procentním zastoupením obrazových bodů, které mají být větší, menší nebo rovny nějakému vhodnému prahu. V případě poloprahování se změní pouze hodnoty pixelů, které nedosahují hodnoty prahu a ostatním pixelům se jejich hodnota ponechá. Při použití adaptivního prahování je obraz nejdříve rozdělen do několika částí a pro každou část se určí hodnota prahu zvlášť. Hysterezní prahování funguje na principu dvou hodnot prahů – T_1 a T_2. Pixel s hodnotou mezi uvedenými prahy se zahrne do oblasti, když sousední pixel již v oblasti je. Další možností je použití vícestupňového prahování, které umožňuje rozdělit obraz na více než dvě disjunktní množiny a lze jej popsat vztahem:

$$\begin{aligned} g(i,j) &= 1 \text{ pro } f(i,j) \geq T_1 \wedge f(i,j) < T_2 \\ g(i,j) &= 2 \text{ pro } f(i,j) \geq T_2 \wedge f(i,j) < T_3 \\ &\vdots \\ g(i,j) &= n \text{ pro } f(i,j) \geq T_n \end{aligned} \tag{3.2}$$



Obr. 4: Princip globálního prahování [19]

3.2 Metody založené na detekci hran

Jak je vysvětleno v [20], hrana je oblast obrazu, ve které dochází k výrazné změně intenzity obrazové funkce. Tuto změnu obrazové funkce $f(x,y)$ lze zjistit pomocí gradientu, který je kolmý na vektor udávající směr hrany. Gradient je popsaný rovnicí:

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \tag{3.3}$$

kde $\frac{\partial f(x,y)}{\partial x}$ je parciální derivace obrazové funkce podle osy x a $\frac{\partial f(x,y)}{\partial y}$ je parciální derivace obrazové funkce podle osy y. Velikost gradientu je invariantní vůči natočení a je dána rovnicí:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2} \tag{3.4}$$

Úhel ψ, který svírá hrana s osou x, lze spočítat podle vztahu:

$$\psi = \text{arctg}\left(\frac{\frac{\partial f(x,y)}{\partial y}}{\frac{\partial f(x,y)}{\partial x}}\right) \tag{3.5}$$

Hranu je často obtížné určit zcela přesně, protože může nabývat různých podob. Obraz může obsahovat příliš širokou oblast, ve které dochází ke změně intenzity a těžko se pak určuje, kde se hrana nachází. Obsahuje-li hrana šum, můžeme jej eliminovat vhodným

předzpracováním obrazu. V úvahu připadá použití lineárního nízkofrekvenčního filtru nebo použití nelineárního filtru založeném například na mediánu. [20]

Jelikož se derivace pro diskrétní obrazovou funkci počítá těžko, lze ji aproximovat vhodným výpočtem diferenciálu. Obvykle se pro aproximace používají centrální diference, které lze pro digitální obraz zjednodušit na tvar:

$$\Delta_x f(x,y) = \frac{1}{2}\big(f(x+1,y) - f(x-1,y)\big),$$
$$\Delta_y f(x,y) = \frac{1}{2}\,(f(x,y+1)\; -\; f(x,y-1)). \tag{3.6}$$

Pro výpočet jednotlivých složek gradientu lze využít tzv. hranové operátory. Jedná se o konvoluční jádro, se kterým je provedena diskrétní konvoluce obrazové funkce a lze tak jednoduše získat hledané složky gradientu. Princip diskrétní konvoluce je vysvětlen v kapitole 5.1.1. [20]

Při použití Robertsova operátoru jsou k pokrytí obou diagonálních směrů potřeba dvě jádra. Jeho nevýhodou je velká citlivost na šum a výpočet derivací na souřadnicích $\left(x+\frac{1}{2}, y+\frac{1}{2}\right)$. Operátor lze popsat vztahem:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & -0 \end{bmatrix} \tag{3.7}$$

Následující operátory pracují s větším počtem pixelů a jsou obvykle méně citlivé na šum. Operátor Prewittové kombinuje filtraci aritmetického průměru s centrální diferencí a lze jej popsat vztahem:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \tag{3.8}$$

Sobelův operátor vznikne konvolucí centrální diference s aproximací Gaussova nízkofrekvenčního filtru. Tento operátor klade větší důraz na pixely blíže reprezentativního pixelu, je často používán pro detekci vodorovných a svislých hran a lze jej popsat vztahem:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \tag{3.9}$$

Mezi další používané operátory patří například Laplacův, který je založený na druhé derivaci a je invariantní vůči natočení. Mezi jeho nevýhody patří velká citlivost na šum a dvojité odezvy na tenké hrany v obraze. Operátor lze popsat vztahem:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.10}$$

Canny určil, že ideální hranový detektor by měl splňovat následující tři vlastnosti:

- musí být detekovány všechny důležité hrany, ale žádné falešné hrany,
- musí být co nejmenší vzdálenost mezi skutečnou a detekovanou hranou,
- každá hrana musí být detekována pouze jednou.

Vzhledem k těmto podmínkám Canny navrhnul detektor, který pro potlačení vlivu šumu nejprve vytvoří konvoluční jádro Gaussova filtru, následně je ve směru příslušných os obraz rozmazán a jsou vypočteny příslušné první derivace. Z nich je pak spočítána amplituda gradientu. Poté je potřeba najít pixely, jejichž okolí je ve směru a proti směru gradientu nejnižší. Ponecháním lokálních maxim dojde ke ztenčení nalezené hrany. Posledním krokem je aplikace hysterezního prahování popsaného v kapitole 3.1. Porovnání aplikace Cannyho detektoru se Sobelovým operátorem zobrazuje obrázek 5. [20,21]

a) b) c)

Obr. 5: a) Originální snímek, b) Hrany detekované Cannyho detektorem, c) Hrany detekované pomocí Sobelova operátoru [22]

3.3 Regionální metody

Metody orientované na regiony se snaží z hlediska zvoleného parametru rozčlenit obraz do maximálně souvislých homogenních oblastí. Metody jsou vhodné pro obrazy, ve kterých je obtížné detekovat přesné hranice. Obvykle je to z důvodu šumu, který se v těchto obrazech vyskytuje.

Narůstání oblasti

Nejprve jsou v obraze určeny výchozí body (tzv. semínka), které jsou definovány svými vlastnostmi. Ze semínek je segment postupně rozšiřován a jednotlivé pixely z okolí jsou do oblasti začleňovány, pokud splňují podmínku:

$$|p_s - p_j| \leq T, \tag{3.11}$$

kde p_s je referenční parametr počátečního bodu, p_j je parametr porovnávaného pixelu a T je rozhodovací úroveň.

Výskyt jasové nehomogenity v obraze lze potlačit pomocí vhodných dynamických úprav. Kandidátní pixel nemusí být porovnáván s počátečním pixelem, ale například se střední hodnotou z parametrů celé oblasti. [23]

Spojování oblastí

Spojováním malých homogenních oblastí dochází k jejich postupnému narůstání. Počáteční obraz je nejprve rozdělen do velkého počtu malých oblastí (nejlépe o velikosti jeden pixel). Podle zadaného kritéria dochází ke spojování sousedních oblastí, které splňují podmínku:

$$p_i \in \langle p_0 - \Delta p, p_0 + \Delta p \rangle, \tag{3.12}$$

kde p_0 je střední hodnota parametru počáteční oblasti, p_i je střední hodnota parametru porovnávané oblasti a Δp je zvolená tolerance parametru. Pokud již nelze bez porušení kritéria žádné dvě oblasti spojit, algoritmus končí. Podle způsobu určení počátečních oblastí, kritéria spojování, počátku spojování a pořadí spojovaných oblastí může docházet k různým výsledkům. [23]

Dělení a spojování oblastí

Princip této metody spočívá ve spojení dvou segmentačních metod. Obraz je postupně dělen na menší oblasti do stanovené struktury a sousední oblasti jsou spojovány, pokud splňují kritéria homogenity. Nejčastěji používanou strukturou pro dělení je stromová struktura, ve které se oblast dělí na čtyři čtvrtinové oblasti. Kritérium homogenity by mělo brát v potaz druh obrázku a vlastnosti předpokládaných objektů v obraze. [21]

Rozvodí

Metoda rozvodí (*watershed*) je inspirována geografií. Obraz je zde chápán jako terén, ve kterém jas pixelu $f(x)$ určuje výšku terénu, ve kterém je černá barva brána za nejníže položenou oblast a bílá barva za nejvýše položenou oblast. Princip je založen na postupném zaplavování obrazu vodou a vzniku jednotlivých povodí, které vytváří homogenní oblasti. Na hřebeny obklopující tyto oblasti jsou umístěny tzv. hráze, které udávají hranice mezi segmenty. V rámci algoritmu nejprve dojde k uspořádání pixelů obrazu podle jejich intenzity jasu, poté k postupnému výběru pixelů, jejich označení štítkem příslušné oblasti a prohledání jejich osmi-okolí. Každý neoznačený pixel je označen štítkem stejné oblasti, pokud v jeho osmi-okolí je označen alespoň jeden pixel stejné oblasti. Jako hraniční pixel je označen takový, v jehož osmi-okolí se nachází minimálně dva pixely z různých oblastí. Pokud se v osmi-okolí prohledávaného pixelu nevyskytuje žádný označený pixel, tak zůstává i nadále neoznačený. Princip segmentace rozvodím je zobrazen na obrázku 6. [23]

Obr. 6: Princip segmentace rozvodím [24]

Shluková analýza

Jedná se o statistickou metodu analýzy obrazových dat, která spočívá ve vytvoření podmnožin shluků, které se utvářejí na základě podobnosti vlastností porovnávaných parametrů. Každý pixel obsahuje vektor vlastností, kterou může být například střední hodnota či rozptyl intenzity okolních pixelů. Takto získaná data jsou obvykle standardizována, čímž se docílí nulové střední hodnoty a směrodatné odchylky rovné jedné. Následně se mezi jednotlivými páry objektů vypočte matice podobností. Nejčastějším kritériem podobností jsou metriky vzdálenosti, které jsou více popsány v kapitole 2.2.3. Pro shlukování existuje mnoho algoritmů, které lze rozdělit na hierarchické a nehierarchické.

Mezi nejpoužívanější metody nehierarchické shlukové analýzy lze zařadit algoritmus k-středů (*k-means*), jehož úkolem je rozdělit objekty do konečného počtu shluků tak, aby byla co nejmenší suma vzdáleností jednotlivých objektů od středů shluků. Výsledek segmentace obrazu pomocí algoritmu k-středů je zobrazen na obrázku 7. Nejprve je nutné stanovit počet shluků, ke kterým se náhodně stanoví středy. Přiřazení objektů do shluků je výsledkem minimalizace vzdálenosti mezi objekty vůči všem středům, která probíhá podle vztahu:

$$V = \sum_{i=1}^{k} \sum_{x_j \epsilon S_i} (x_j - \mu_i)^2, \tag{3.13}$$

kde k je počet hluků S_i a μ_i je průměr všech hodnot bodů x_j v daném shluku S_i. [23]

a) b)

Obr. 7: a) Originální snímek, b) Snímek segmentovaný pomocí algoritmu k-středů pro k=3 [25]

3.4 Znalostní metody

Do kategorie znalostních metod patří všechny techniky, které využívají již dříve získané znalosti o objektech. Získané znalosti lze chápat jako šablony objektů, které jsou srovnávány s přiloženým obrazem. Srovnávat je možné celé regiony, ale i jednotlivé kontury. Podle vlastností sledovaných objektů lze vytvořit grafy, které jsou pak pomocí grafových algoritmů srovnány. Nikdy nelze očekávat absolutní shodu, jelikož se často vyskytuje šum, různé zkreslení apod. Proto je snaha najít pouze maximum vhodně zvoleného kritéria. [26]

Metoda **AAM** (*active appearance models*) využívá statisticky vytvořený model z manuálně segmentovaných vzorových dat. K vytvoření modelu je nutný rozsáhlý a reprezentativní soubor trénovacích vzorů, které je potřeba ručně anotovat a vyznačit hraniční body. Trénování spočívá v zaznamenání vzájemného vztahu mezi polohou hraničních bodů a intenzitou pixelů v množině vzorů. Model lze vytvořit například pomocí statistické analýzy zpracování dat (*principal component analysis*). Výhodou této metody je rychlé porovnání obrazu a rychlá adaptace modelu na nový obraz. Nevýhodou je požadavek na co nejpřesnější počáteční odhad polohy objektu v prohledávaném obraze a poměrně náročnou anotaci trénovacích vzorů. [26]

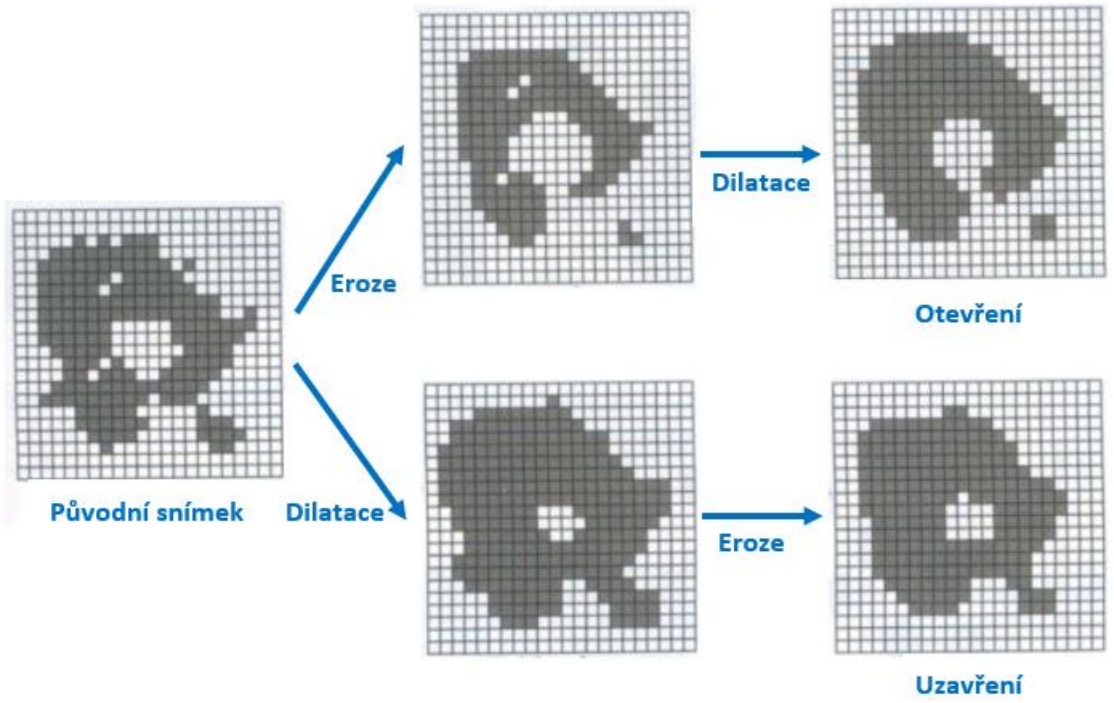
Segmentace využitím atlasů tkáňových pravděpodobnostních map (**TPM**) je vhodná především v sofistikovaných problémech, kterými je ku příkladu segmentace části mozku. Často je potřeba využít poloautomatických metod vyžadující ručně definované orientační body, které umožní vyšší přesnost segmentace. [23]

3.5 Hybridní metody

Mezi hybridní metody lze zařadit takové, které jsou primárně určeny k jiným účelům nebo kombinují různé přístupy metod.

Morfologické operace slouží především pro úpravu a zjednodušení obrazu pro jeho další manipulaci. Operace jsou založeny na nelineárních úpravách obrazu. Mezi

základní operace binární morfologie patří například dilatace, eroze, uzavření či otevření. Dilatace zvětšuje objekty a používá se pro zaplnění malých děr a úzkých zálivů. Eroze eliminuje izolované pixely a ubírá obrysy objektů. Eroze a dilatace jsou vzájemně inverzními operacemi. Otevření přerušuje tenké spoje mezi objekty, zvětšuje v nich mezery a zjednodušuje strukturu objektů. Jedná se tedy o provedení eroze následovanou dilatací. Uzavření spojuje blízké objekty, zaplňuje malé díry a vyhlazuje tvar objektů. Princip zmíněných morfologických operací zobrazuje obrázek 8. [27]



Obr. 8: Morfologické operace eroze, dilatace, otevření a uzavření [28]

Pro segmentaci obrazu lze využít i **umělé neuronové sítě**, které nejsou zaměřeny na konkrétní úlohu, ale poskytují velice flexibilní nástroj pro řešení různorodých problémů. Umělé neuronové sítě umožňují vysokou škálovatelnost pro specifické úlohy. Jejich cílem je zpracovávat vstupní data způsobem typickým lidskému mozku. Fungují na principu učení se z příkladů, zobecňování znalostí a vytvoření modelu z trénovacích dat, které jsou síti postupně předkládány. Více jsou neuronové sítě popsány v následujících kapitolách.

4 UMĚLÉ NEURONOVÉ SÍTĚ

Umělé neuronové sítě (*artifical neural networks*) lze zařadit mezi výpočetní modely umělé inteligence, jejichž struktura je určená pro distribuované paralelní zpracování dat. Neuronové sítě vynikají schopností učit se, zevšeobecňovat, řešit nelineární úlohy a zjišťovat závislosti, které nejsou zcela zřejmé. Aplikací umělých neuronových sítí existuje velké množství a lze je uplatnit takřka v jakémkoli oboru. Mezi příklady jejich použití lze zařadit analýzu textu, rozpoznávání poruch strojů, predikci vývoje cen, automatické ovládání zařízení nebo rozpoznávání vzorů v obraze. [29]

4.1 Historie umělých neuronových sítí

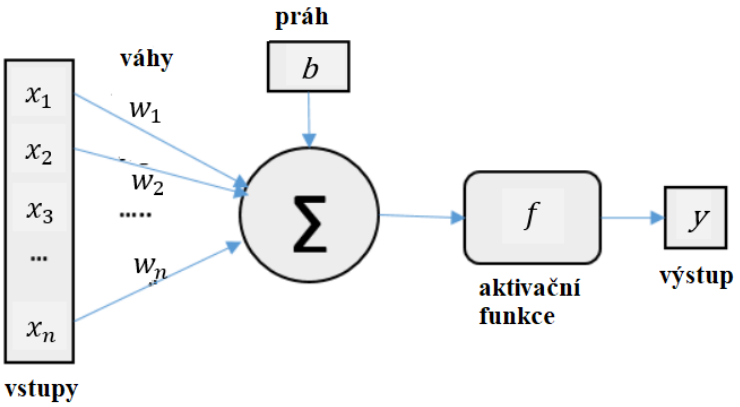
Za vznik oboru zabývající se umělými neuronovými sítěmi je považován rok 1943, kdy McCulloch a Pitts vytvořili jednoduchý matematický model neuronu. V roce 1949 napsal Hebb knihu, v níž navrhl učící pravidlo inspirované myšlenkou, že podmíněné reflexy u živočichů jsou vlastnostmi jednotlivých neuronů. V roce 1951 Minsky zkonstruoval první neuropočítač s názvem Snark, což později inspirovalo mnoho dalších konstruktérů neuropočítačů. Na práci McCullocha a Pittse navázal Rosenblatt, který v roce 1957 vynalezl tzv. perceptron, pro který navrhl učící algoritmus. V roce 1958 sestrojil Rosenblatt ve spolupráci s Wightmanem úspěšný neuropočítač s názvem Mark I Perceptron, který byl navržen pro rozpoznávání znaků. Krátce po objevu perceptronu Widrow se svými studenty vyvinul typ neuronového výpočtu, který nazval ADALINE (*adaptive linear neuron*). V roce 1969 byl pozastaven výzkum neuronových sítí po tvrzení, že jeden perceptron není schopen vyřešit jednoduchou logickou funkci XOR (vylučovací disjunkci). Další velký průlom nastal až v roce 1982, kdy Hopfield publikoval své výsledky z neurovýpočtů. V roce 1986 byly publikovány výsledky popisující učící algoritmus zpětného šíření chyby (*backpropagation*), který je doposud nejpoužívanější učící metodou neuronových sítí. [30]

4.2 Formální neuron

Umělé neuronové sítě jsou inspirovány strukturou lidské nervové soustavy. Základním prvkem biologické i umělé neuronové sítě je neuron, který se v oblasti umělé inteligence často označuje jako perceptron. Jednotlivé neurony jsou podle typu sítě uspořádány do vrstev a mezi vrstvami propojeny vazbami, díky kterým dochází k přenosu signálů mezi neurony. V biologii se těmto vazbám říká synapse. Každý vstup do neuronu je nejprve vynásoben příslušnou váhou, poté jsou vstupy sečteny a výsledek je po přičtení prahové hodnoty transformován přenosovou funkcí a poslán na výstup neuronu. Matematický zápis je dán rovnicí:

$$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right), \tag{4.1}$$

kde x_i jsou vstupy do neuronu, w_i jsou váhy, b je práh, f je aktivační funkce a y je výstup z neuronu. Schéma umělého neuronu zobrazuje obrázek 9. [30]



Obr. 9: Schéma formálního neuronu [31]

Pro řešení jednoduchých úloh lze vystačit s počtem neuronů v řádech jednotek či desítek, které jsou uspořádány do jednotlivých vrstev. První vrstva se označuje jako vstupní, poslední vrstva se označuje jako výstupní a mezi vstupní a výstupní vrstvou se nachází tzv. skryté vrstvy. Pro složité a komplexní úlohy je možné použít sítě složené z mnoha skrytých vrstev. Takové sítě se označují jako hluboké, jejich použití je však velmi výpočetně náročné. [29]

4.3 Metody a funkce využívané v umělých neuronových sítích

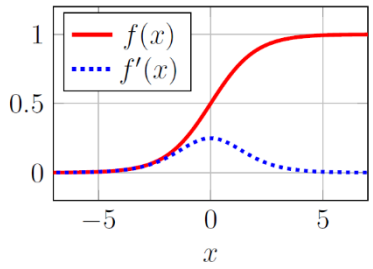
4.3.1 Aktivační funkce

Aktivační funkce (*activation functions*) určují výstup neuronů v závislosti na jeho vstupech. Aby bylo zajištěno, že síť dokáže aproximovat libovolnou spojitou funkci, měly by být aktivační funkce nelineární a spojitě diferencovatelné. Některé aktivační funkce však tyto podmínky nesplňují. Níže jsou uvedeny některé vybrané aktivační funkce.

Sigmoid

Jedná se o tradiční aktivační funkci s oborem hodnot $H(f) \in (0,1)$. Kvůli problému mizejícího gradientu popsaného v kapitole 4.7 není vhodné používat sigmoidální funkci pro aktivaci neuronů ve skrytých vrstvách. Funkce je však často použita pro aktivaci neuronů ve výstupní vrstvě sítě. Průběh funkce je zobrazen na obrázku 10 a její předpis je dán rovnici:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

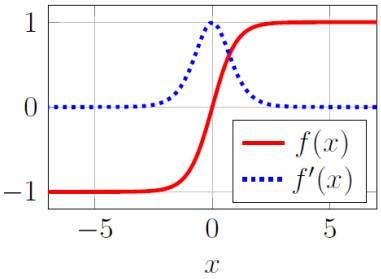


Obr. 10: Aktivační funkce sigmoid [32]

Hyperbolický tangens (tanh)

Tato funkce je podobná sigmoidě, liší se však v oboru hodnot: $H(f) \in (-1,1)$. Pro tuto funkci taktéž platí, že je vhodná zejména pro aktivaci neuronů ve výstupní vrstvě sítě. Průběh funkce je zobrazen na obrázku 11 a její předpis je dán rovnicí:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4.3}$$

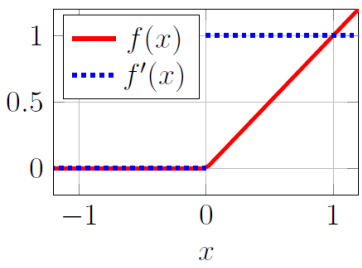


Obr. 11: Aktivační funkce hyperbolický tangens [32]

ReLU:

Usměrněná lineární jednotka (*rectified linear unit*) se stala velice oblíbenou. Oproti předchozím zmíněným funkcím umožňuje rychlejší učení a nižší výpočetní náročnost. Nevýhodou této funkce je absence derivace v bodě $x = 0$, což se řeší předpokladem, že $f'(0) = 0$. Další nevýhodou je nulová hodnota gradientu na intervalu $x \in (-\infty, 0)$, což vytváří tzv. „mrtvé neurony“ neschopné se učit a není tak plně využita kapacita sítě. Obor hodnot této funkce je $H(f) \in (0, \infty)$. Průběh funkce je zobrazen na obrázku 12 a její předpis je dán rovnicí:

$$f(x) = \max(0, x). \tag{4.4}$$

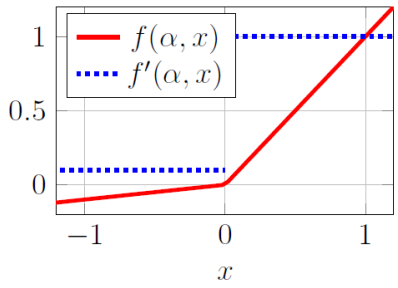


Obr. 12: Aktivační funkce ReLU [32]

Usměrněná lineární jednotka disponuje mnoha variantami. **Leaky ReLU** se liší tím, že pro hodnoty $x < 0$ disponuje mírným sklonem, neobsahuje tak nulovou hodnotu gradientu. Funkce **PReLU** (*parametric ReLU*) je dána předpisem:

$$f(\alpha, x) = \max(\alpha x, x), \tag{4.5}$$

kde α je učitelný parametr sklonu funkce pro hodnoty $x < 0$. Zvýšením počtu učitelných parametrů však dochází k nepatrnému zvýšení výpočetních nároků a pomalejší konvergenci při učení. Průběh funkce je zobrazen na obrázku 13.

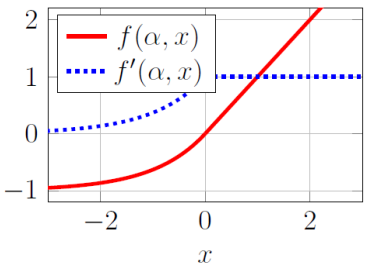


Obr. 13: Aktivační funkce PReLU s hodnotou parametru $\alpha = 0{,}1$ [32]

Podobnou funkcí je **ELU** (*exponential linear unit*) která obsahuje učitelný parametr α. Funkce je definovaná předpisem:

$$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{pro } x \leq 0 \\ x & \text{pro } x > 0 \end{cases} \tag{4.6}$$

Výhodou této funkce je spojitá diferencovatelnost na celém definičním oboru. Průběh funkce je zobrazen na obrázku 14.



Obr. 14: Aktivační funkce ELU s hodnotou parametru $\alpha = 1$ [32]

Softmax

Funkce je vhodná pro použití ve výstupní vrstvě sítě, kde podle vztahu

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{n} e^{x_j}} \tag{4.7}$$

převede vektor reálných čísel $(x_1, x_2, \ldots, x_n)$ na vektor rozdělení pravděpodobností obsahující n hodnot pro $i = 1, 2 \ldots n$. Funkce je často použita při klasifikaci, kde jednotlivé hodnoty určují míru zařazení do tříd.

4.3.2 Ztrátové funkce

Ztrátové funkce (*loss functions*), označované také jako chybové funkce, slouží k vyjádření míry chyby predikce vůči trénovacím datům. Podle typu řešené úlohy lze využít různou ztrátovou funkci, jejíž volba značně ovlivňuje rychlost a kvalitu učení neuronové sítě. Níže jsou uvedeny pouze některé vybrané ztrátové funkce, kde N je počet neuronů výstupní vrstvy, y je očekávaný výstup a $\hat{y}$ je reálný výstup sítě.

Střední kvadratická chyba (*mean squared error*) je často využívána pro regresní úlohy a je dána rovnicí:

$$E = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{4.8}$$

Různými úpravami střední kvadratické chyby lze získat další ztrátové funkce vhodné pro regresní úlohy. Mezi tyto funkce lze zařadit střední absolutní chybu (*mean absolute error*), střední chybu vychýlení (*mean bias error*) nebo střední kvadratickou logaritmickou chybu (*mean squared logarithmic error*).

Křížová entropie pro binární klasifikaci (*binary cross-entropy*) je určena pro klasifikaci do dvou tříd. Předpis funkce je dán rovnicí:

$$E = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i). \tag{4.9}$$

Pro využití této ztrátové funkce je doporučeno pro neurony v poslední vrstvě sítě použití aktivační funkce Sigmoid.

Křížová entropie pro klasifikaci do více tříd (*categorical cross-entropy*) slouží především pro klasifikací do tří a více tříd, přičemž jedna třída odpovídá jednomu výstupnímu neuronu. Funkce je definována vztahem:

$$\mathrm{E} = -\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{y_i} \log(\hat{\mathrm{y}}_\mathrm{i}). \tag{4.10}$$

Pro využití této ztrátové funkce je doporučeno pro neurony v poslední vrstvě sítě použití aktivační funkce Softmax. [39]

4.3.3 Optimalizační metody

Optimalizátor (*optimizer*) je metoda, která slouží pro minimalizaci hodnoty ztrátové funkce. Níže jsou uvedeny některé používané metody, kde w je optimalizovaný parametr, t je krok učení (*step*), α je učící rychlost (*learning rate*) a $\nabla\varepsilon$ je gradient chybové funkce.

SGD

Základní metodou je stochastický gradientní sestup (*stochastic gradient descent*), který využívá gradientu chybové funkce vůči parametrům sítě pro změnu vah jednotlivých neuronů. Metoda je popsána rovnicí:

$$w_{t+1} = w_t - \alpha \nabla \varepsilon \tag{4.11}$$

Některé optimalizační metody využívají hyperparametru, který se nazývá momentum. Tento hyperparametr slouží pro zvýšení stability, rychlejší konvergenci a jeho typická hodnota je 0,9. Metoda gradientního sestupu s momentem je popsána rovnicemi:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \alpha \nabla \varepsilon \\ w_{t+1} &= w_t - v_t \end{aligned} \tag{4.12}$$

kde γ udává hodnotu momenta. Princip metody gradientního sestupu je popsán v kapitole 4.6. [33,34]

Adagrad

Metoda Adagrad (*adaptive gradient descent*) dynamicky upravuje koeficient učení vůči jednotlivým parametrům. Čím více se parametry mění, tím menší je změna učící rychlosti. V hlubokých neuronových sítích s velkým počtem skrytých vrstev je riziko vzniku velmi malé učící rychlosti, která téměř znemožní učení. Metoda je dána rovnicemi:

$$\begin{aligned} v_t &= v_{t-1} + (\nabla \varepsilon)^2 \\ w_{t+1} &= w_t - \frac{\alpha}{\sqrt{v_{t+1}} + \delta} \nabla \varepsilon \end{aligned} \tag{4.13}$$

kde δ je velmi malá hodnota pro zaručení, že nebude děleno nulou. [33]

RMSProp

Metoda RMSProp (*root mean square propagation*) se od metody Adagrad liší tím, že učící rychlost je exponenciálním průměrem gradientů, namísto kumulativního součtu kvadrátů gradientů, jak je tomu v případě metody Adagrad. Nevýhodou metody je pomalá konvergence. Metodu popisují rovnice:

$$\begin{aligned} v_t &= \gamma v_{t-1} + (1-\gamma)(\nabla \varepsilon)^2 \\ w_{t+1} &= w_t - \frac{\alpha}{\sqrt{v_t} + \delta} \nabla \varepsilon \end{aligned} \tag{4.14}$$

ze kterých je patrné, že tato metoda vznikla přidáním hyperparametru momenta do metody Adagrad. [33]

Adadelta

Metoda vychází z metody Adagrad, ale omezuje součty předchozích gradientů na určitou maximální hodnotu. Tím je zajištěno, aby učicí rychlost neklesala příliš rychle.

Adam

Metoda Adam (*adaptive moment estimation*) kombinuje výhody metod RMSProp a Adadelta. Tato metoda vyniká jednoduchou implementací, výpočetní účinností a nízkými paměťovými nároky. Metodu popisují rovnice:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)(\nabla\varepsilon) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla\varepsilon)^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ w_{t+1} &= w_t - \frac{\alpha}{\sqrt{\hat{v}_t}+\delta}\hat{m}_t \end{aligned} \tag{4.15}$$

s doporučenými počátečními hodnotami: $\alpha = 0{,}001$, $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$, $\delta = 10^{-7}$. [33,34]

4.3.4 Metriky

Metriky (*metrics*) nemají vliv na průběh trénování modelu, ale slouží pouze pro určení jeho kvality. Jako metriku lze použít jakoukoli ztrátovou funkci, existují však funkce navržené přímo pro ověření kvality modelu. Nejprve je potřeba si zavést základní pojmy. Jak je uvedeno ve [35], pro obrazová data jsou:

- **TP** (*true positive)* – predikované pixely, které měly být predikovány,
- **FP** (*false positive*) – predikované pixely, které neměly být predikovány,
- **TN** (*true negative*) – nepredikované pixely, které neměly být predikovány,
- **FN** (*false negative*) – nepredikované pixely, které měly být predikovány.

Accuracy udává počet správných predikcí ku celkovému počtu predikcí. V některých případech se jedná o vhodnou metriku, občas však dává falešný pocit vysoké přesnosti, která nereflektuje skutečnou kvalitu modelu. Metrika je dána vztahem:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.16}$$

Preccision říká, kolik vzorků označených jako pozitivních je skutečně pozitivních. Metrika je dána vztahem:

$$Preccision = \frac{TP}{TP + FP} \tag{4.17}$$

Recall říká, kolik pozitivních vzorků bylo označeno jako pozitivní. Metrika je dána vztahem:

$$Recall = \frac{TP}{TP + FN} \tag{4.18}$$

F1 Score udává harmonický průměr mezi Preccision a Recall a lépe tak reflektuje skutečnou kvalitu modelu. Metrika je dána vztahem:

$$F1\ Score = 2 \cdot \frac{1}{\frac{1}{Preccision} + \frac{1}{Recall}} \tag{4.19}$$

IOU (*intersection over union*) udává průnik (*intersection*) očekávaného a reálného výstupu ku sjednocení (*union*) očekávaného a reálného výstupu. Metrika je často použita v úlohách segmentace obrazu a je dána vztahy:

$$IOU = \frac{target \cap real}{target \cup real}$$
$$IOU = \frac{TP}{TP + FP + FN} \tag{4.20}$$

kde *target* je očekávaný výstup a *real* je reálný výstup. [35]

4.4 Typy učení

Učení neuronových sítí spočívá ve stanovení hodnoty vah a prahů s cílem minimalizovat chybu mezi skutečným a požadovaným výstupem. Podle způsobu učení lze neuronové sítě rozdělit do několika kategorií.

V případě **učení s učitelem** (*supervised learning*) je síti postupně předkládána množina vstupních dat a k nim i odpovídající správné výsledky. Cílem je vytvořit předpovědní model založený na vstupních datech i očekávaném výstupu. Vstupní data jsou obvykle dělena na trénovací, validační a testovací. Síti jsou postupně předkládána data z trénovací množiny, na základě kterých se síť učí. Velikost trénovací množiny je zásadní pro správné naučení sítě. Validační množina slouží pro kontrolu průběhu učení. Pokud by byla vysoká úspěšnost predikce pro trénovací data, ale nízká pro testovací, docházelo by k přeučení sítě, které je více vysvětleno v kapitole 4.7. Testovací množina je určena pro vyjádření přesnosti modelu po skončení jeho učení. Učení s učitelem nejčastěji řeší klasifikaci (*classification*) a regresi (*regression*). Klasifikace je zařazení vstupu do tříd a regrese je odhad konkrétní hodnoty na základě znalostí vstupu. [32]

Při použití **učení bez učitele** (*unsupervised learning*) je cílem seskupení a interpretace dat založená pouze na vstupních datech, která nejsou nijak označena ani zařazena do tříd. Typickými úlohami jsou shlukování (*clustering*) a asociace (*association*). Shlukování je zařazení dat do skupin se společnými vlastnostmi. Asociace spočívá v odhalení skrytých vztahů a pravidel. [32]

Algoritmy **zpětnovazebního učení** (*reinforcement learning*) jsou založeny na principu lidského učení typu pokus/omyl. Softwarový agent se učí interakcí s prostředím s cílem dosáhnout maximální hodnoty kumulativní odměny. Časté použití zpětnovazebního učení je plánování trasy mobilního robota. Z důvodu možných komplikací při učení v reálném prostředí, slouží vývojářům různé simulační nástroje, které poskytují velmi věrohodné modelování reálného prostředí.

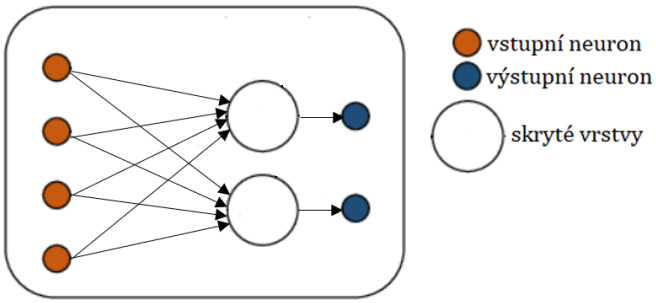
Přenesené učení (*transfer learning*) je metoda učení využívající znalosti získané učením na podobné úloze. Není tak třeba trénovat síť od nuly, ale využít již předtrénovaný model, který se jen dotrénuje na specifické detaily řešeného problému.

4.5 Architektury neuronových sítí

Architektura vícevrstvé neuronové sítě by měla odpovídat složitosti řešeného problému. Síť s malým počtem skrytých vrstev nedokáže řešit komplikovaný problém, protože se učení obvykle zastaví v lokálním minimu. Naopak síť s nepřiměřeně velkým počtem skrytých vrstev špatně generalizuje tréninkové vzory a dochází k problémům, které jsou popsány v kapitole 4.7. [30]

4.5.1 FFN

Dopředné neuronové sítě (*feedforward neural networks*) patří mezi nejjednodušší typy neuronových sítí a často bývají součástí jiných architektur, např. konvolučních neuronových sítí, které jsou popsány v kapitole 5. Výstupy neuronů z jedné vrstvy jsou propojeny se vstupy neuronů z následující vrstvy. Obvykle se jedná o plně propojenou síť (*full connected*), což znamená, že všechny neurony z jedné vrstvy jsou propojeny se všemi neurony z vrstvy následující. Šíření vstupního signálu probíhá pouze jedním směrem, a to od vstupních vrstev přes skryté vrstvy do výstupních vrstev. Schéma dopředné neuronové sítě je zobrazeno na obrázku 15. [32]



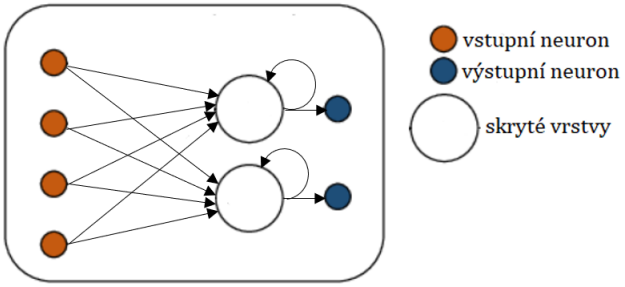
Obr. 15: Dopředná neuronová síť [36]

4.5.2 RNN

Rekurentní neuronové sítě (*recurrent neural networks*) obsahují skupinu neuronů, které jsou spojeny do kruhu a tvoří tak cyklus. Cyklické propojení může být mezi několika neurony, nebo jen v rámci jednoho neuronu. V tomto případě je výstup z neuronu přiveden na jeho vstup. Důvodem vytvoření cyklické struktury je vznik vnitřních stavů, které jsou vhodné zejména pro sekvenční data. Rekurentní neuronové sítě lze využít například pro strojový překlad, generování slov nebo předpovídání časových řad.

Mezi rekurentní neuronové sítě lze zařadit sítě s dlouhou krátkodobou pamětí, které jsou navrženy primárně pro práci se signály a časovými řadami. Hlavní složkou těchto sítí jsou rekurentní LSTM (*long short-term memory*) vrstvy, které jsou schopny

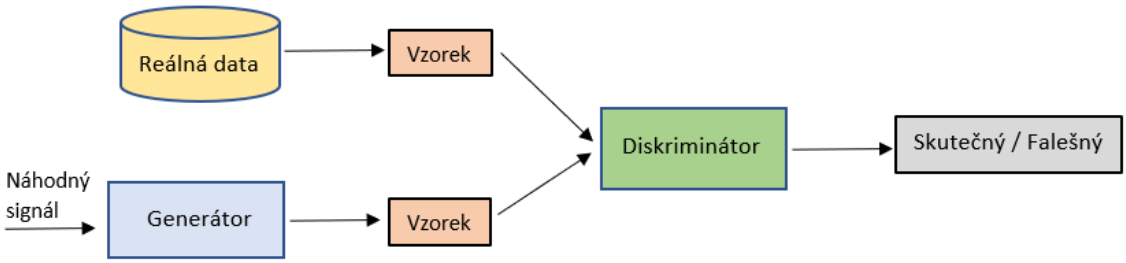
naučit se dlouhodobé závislosti ze vzdálených časových kroků. Schéma rekurentní neuronové sítě je zobrazeno na obrázku 16. [30,32]



Obr. 16: Rekurentní neuronová síť [36]

4.5.3 GAN

Generativní kompetitivní neuronové sítě (*generative adversarial networks*) jsou složeny ze dvou hlavních sítí – generátor a diskriminátor. Obě části soutěží mezi sebou a tím dochází k učení. Úkolem generátoru je vytvářet obsah, který je předkládán diskriminátoru. Diskriminátor má naopak za úkol rozeznat, jestli předložená data pochází z generátoru, nebo z trénovacího datasetu. Snaha generátoru je maximalizovat chybu diskriminátoru tím, že se snaží vytvořit data co nejvíce podobná datům z trénovací množiny. Diskriminátor se naopak snaží svoji chybu minimalizovat lepším rozeznáváním generovaných dat od těch reálných. Generativní kompetitivní neuronové sítě jsou často využívány pro generování obrazů, textů nebo i neexistujících lidských obličejů. Zjednodušený princip generativní kompetitivní sítě je zobrazen na obrázku 17. [32]



Obr. 17: Generativní kompetitivní síť [37]

4.5.4 CNN

Konvoluční neuronová sít (*convolutional neural networks*) je dopředná síť vhodná především pro zpracování obrazu. Architektura neuronové sítě se skládá z různých vrstev, které vykonávají specifické úkoly. Mezi hlavní vrstvy patří konvoluční a sdružovací, po kterých obvykle následuje část plně propojených vrstev. Konvoluční neuronové sítě jsou více popsány v kapitole 5.

4.6 Algoritmus zpětného šíření chyby

Adaptační algoritmus zpětného šíření chyby (*backpropagation*) je podle [30] nejpoužívanějším algoritmem pro učení neuronové sítě. Učení lze chápat jako optimalizační proces, v rámci kterého nejprve dojde k dopřednému šíření vstupního signálu tréninkových vzorů, poté se optimalizátor (viz kapitola 4.3.3) snaží minimalizovat chybovou funkci (viz kapitola 4.3.2), která vznikne rozdílem mezi skutečným a požadovaným výstupem. Následně je spočítán gradient chybové funkce, který je zpětně šířen sítí a dochází k aktualizaci vah a prahů neuronů v jednotlivých vrstvách.

Prvním krokem učení je určení vah a prahů, které jsou inicializovány na náhodné nízké hodnoty. Poté jsou síti předložena vstupní data, ze kterých je vypočítán reálný výstup sítě. Následně je pro konkrétní tréninkový vzor spočítána chybová funkce. Pro všechny dvojice tréninkových vzorů je celková chybová funkce dána vztahem:

$$E(w) = \sum_{k=1}^{n} E_k(w), \tag{4.21}$$

kde k je počet tréninkových vzorů. Pro úpravu vah neuronů v i-té vrstvě platí vztah označovaný jako gradientní sestup (*gradient descent*):

$$w_i^t = w_i^{t-1} - \alpha \nabla \varepsilon, \tag{4.22}$$

kde *t* značí krok učení (*step*), α je učící rychlost (*learning rate*) a $\nabla\varepsilon$ je gradient chybové funkce získaný ze vztahu:

$$\nabla \varepsilon = \frac{\partial E}{\partial w_i} \tag{4.23}$$

Gradient chybové funkce lze vyjádřit jako součin dílčích chybových funkcí, které jsou závislé na jednotlivých neuronech sítě a k výpočtu tak lze využít složené parciální derivace. Hodnotu gradientu pro i-tou váhu lze vypočítat podle vztahu:

$$\frac{\partial E}{\partial w_i} = \sum_{k=1}^{n} \frac{\partial E_k}{\partial w_i} = \sum_{k=1}^{n} \frac{\partial E_k}{\partial y_j} \frac{\partial y_j}{\partial \xi} \frac{\partial \xi_j}{\partial w_i} \tag{4.24}$$

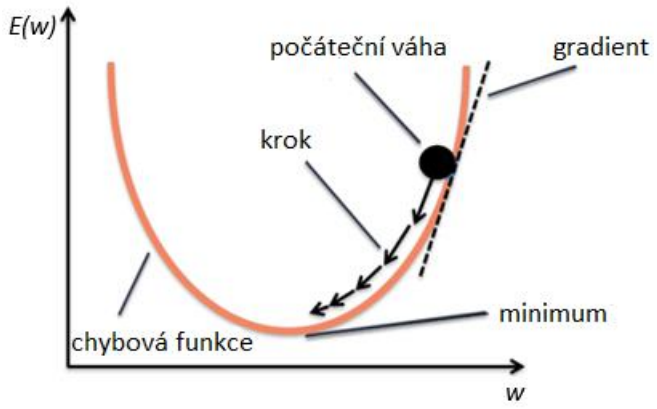
kde $\frac{\partial \xi_j}{\partial w_i}$ je parciální derivace vnitřního potenciálu *j*-tého neuronu podle *i*-té váhy. [38]

Pro lepší pochopení lze gradientní sestup znázornit obrázkem 18, ve kterém je vykreslena chybová funkce $E(w)$, kde osa *x* představuje vektor vah *w* a chyba sítě je znázorněna na ose *y*. Jak je popsáno v úvodu kapitoly, nejprve je váha w^0 inicializovaná na náhodnou hodnotu, očekává se tedy, že chyba sítě bude pravděpodobně velká. V bodě w^0 je ke grafu chybové funkce sestrojen tečný vektor $\nabla\varepsilon$, tedy gradient vyjadřující směr největšího růstu funkce. Jelikož se hledá minimum funkce, posune se vektor dolů proti směru gradientu o hodnotu α, čímž se získá nová hodnota $w^1 = w^0 - \alpha\nabla\varepsilon$, pro kterou je chybová funkce menší, než pro hodnotu původní váhy w^0. Hodnota kroku α je velmi důležitá. V případě velkých hodnot může metoda špatně

konvergovat, či dokonce divergovat. Zvolením nízké hodnoty trvá trénování sítě velmi dlouho. V praxi se často používá dynamického snižování kroku podle specifických pravidel. Proces konstrukce tečného vektoru se opakuje, dokud se vektor nedostane do minima chybové funkce, tedy dokud je splněna podmínka $E(w^t) \geq E(w^{t+1})$. V praxi je metoda obvykle ukončena dosažením požadované hodnoty chyby sítě na validačních datech, nebo dosažením maximálního počtu zadaných iterací. [30]

Gradientní metoda téměř vždy konverguje do minima funkce, může však dojít k uváznutí v lokálním minimu. Předčasné ukončení hrozí i v případě vzniku nulového gradientu. Zjednodušený princip gradientní metody zobrazuje obrázek 18.

Úpravami gradientního sestupu vznikly podobné metody, mezi které lze zařadit stochastický gradientní sestup (*stochastic gradient descent*), dávkový gradientní sestup (*batch gradient descent*) nebo gradientní sestup s menšími dávkami (*minibatch gradient descent*). [39]



Obr. 18: Gradientní sestup [40]

4.7 Problémy v neuronových sítí

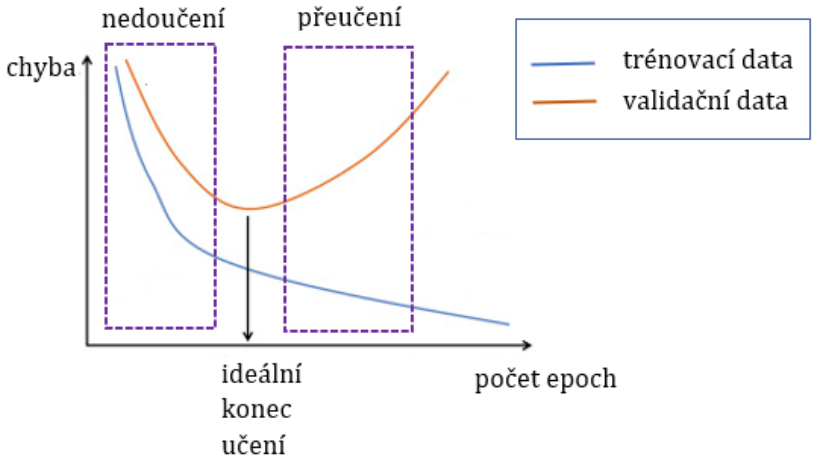
V ideálním případě by při trénování sítě měla chyba trénovací a validační funkce rovnoměrně klesat do bodu stability, ze kterého již pak hodnoty funkce klesají jen mírně. Mezi hodnotou trénovací a validační funkce by v průběhu trénování měl být jen malý rozdíl. [41]

Přeučení (*overfitting*) znamená zapamatování si detailů bez schopnosti jejich zobecnění. Síť se dobře naučí trénovací data, ale pro jiná data již není schopná správně generalizovat. Příčinou je obvykle malá množina trénovacích dat nebo zbytečně velká složitost sítě. Existuje několik možností, kterými lze přeučení předcházet. Jedním z řešení je zvětšit počet trénovacích dat nebo snížit složitost sítě. V případě omezeného počtu vstupních dat si lze pomoci například přivedením náhodného šumu nebo využitím augmentací, což je využití digitálních prostředků pro umělé navýšení množiny vstupních dat. Pro zjednodušení modelu sítě lze náhodně vynechat některé neurony (tzv. *dropout*) nebo omezit hodnoty vah do určitého rozsahu. Možností je i zastavit učení

modelu dříve, než začne docházet k jeho přeučení, jak je zobrazeno na obrázku 19. [32,41]

Nedoučení (*underfitting*) znamená, že se síť z trénovacích dat není schopna naučit ani správně generalizovat. Pokud se křivka chybové funkce v závislosti na jednotlivých epochách nemění, znamená to využití příliš jednoduchého modelu sítě nebo špatně zvolené hyperparametry pro učení. Stále se snižující křivka chybové funkce značí, že učení sítě bylo zastaveno předčasně. Nedoučení lze řešit zvýšením složitosti modelu, aplikací vhodnějších hyperparametrů nebo ponecháním delšího času pro její naučení. [32,41]

V případě řešení složitého problému je nutné využít hluboké neuronové sítě s vysokým počtem vrstev. S rostoucím počtem vrstev však roste i **problém mizejícího gradientu** (*vanishing gradient problem*). Zpětně se šířící gradient postupně slábne, což má za následek, že u neuronů v horních vrstvách sítě dochází jen k malé změně jejich vah. Tomuto problému se lze vyvarovat vhodnou počáteční inicializací vah, normalizací dat v dávce (*batch normalization*) nebo využitím topologií obsahující paralelní propojení vrstev (*residual networks*). [32]



Obr. 19: Učení neuronové sítě [42]

5 KONVOLUČNÍ NEURONOVÉ SÍTĚ

Konvoluční neuronové sítě (*convolutional neural networks*) vychází z klasických neuronových sítí a jsou zaměřeny převážně na zpracování obrazových dat, pro které klasické neuronové sítě nejsou vhodné. Černobílý obraz je reprezentován jako matice hodnot, která je popsána rovnicí (2.2). Barevný obraz je obvykle reprezentován třemi maticemi, přičemž každá matice obsahuje hodnoty pro jednu složku barvy[2]. Jelikož jeden obrazový pixel odpovídá jednomu neuronu, tak takto popsaný obraz obsahuje příliš vysoký počet vstupních a učitelných parametrů, pro který nejsou klasické neuronové sítě navrženy.

5.1 Popis konvolučních neuronových sítí

Oproti klasickým neuronovým sítím, které jsou popsány v kapitole 4, se konvoluční neuronové sítě vyznačují různými vrstvami, které jsou uspořádány do bloků a plní specifické funkce.

Učení konvoluční neuronové sítě je opět založeno na algoritmu zpětného šíření chyby (viz kapitola 4.6), který je upravený pro různé typy vrstev. Informace ze vstupních vrstev jsou zpracovávány jednotlivými vrstvami a postupně šířeny na výstup sítě. Na základě ztrátové funkce je vypočtena chyba sítě, která je zpětně propagována na její vstup a upravovány parametry jednotlivých vrstev. V následujících podkapitolách jsou popsány nejdůležitější vrstvy, které jsou běžně součástí konvolučních neuronových sítí.

5.1.1 Konvoluční vrstva

Konvoluční vrstva (*convolutional layer*) patří mezi základní vrstvy konvolučních neuronových sítí a provádí operaci konvoluce. Na vstup sítě je přiveden obraz, přes který se posouvá konvoluční jádro (často označovaný jako kernel nebo filtr). Jádro se postupně posouvá po celém vstupním obraze a probíhá násobení hodnot, které se překrývají. Poté dojde k sečtení násobků a vypočtená hodnota se uloží na výstup konvoluční vrstvy. Tomuto postupu se říká extrakce příznaků (*feature extraction*). Hlavním úkolem konvolučních vrstev je zachytit co nejvíce příznaků a jednotlivých rysů, kterými je identifikován vstupní obraz. Princip konvoluce je zobrazen na obrázku 20. Matematicky lze 2D konvoluci popsat rovnicí:

$$g(x,y) = f(x,y) * w(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x-i, y-j) \cdot w(i,j), \tag{5.1}$$

kde $g(x,y)$ je výstupní obraz, $f(x,y)$ je vstupní obraz a $w(x,y)$ je konvoluční jádro. [43]

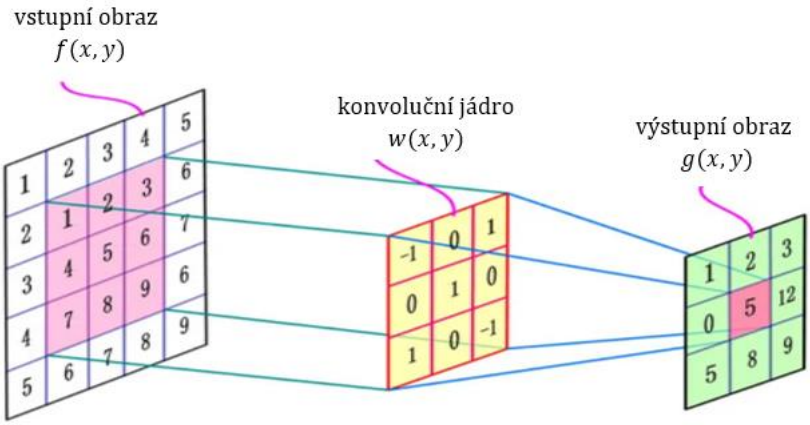
[2] V případě RGB modelu se jedná o barvu červenou, zelenou a modrou.

Počet kanálů konvolučního jádra odpovídá počtu kanálů vstupního obrazu a počet kanálů výstupního obrazu odpovídá počtu konvolučních jader. K výsledku konvoluce je poté přičtena hodnota prahu a aplikována nelinearita.

Mezi hlavní parametry konvoluční vrstvy patří krok (*stride*), odsazení (*padding*) a velikost konvolučního jádra (*kernel size*). Stride udává počet pixelů, o které se posouvá konvoluční jádro. Typickou hodnotou posuvu je jeden pixel. Padding je rozšíření okrajů vstupního obrazu o určitý počet pixelů. Takto přidané pixely nejčastěji disponují hodnotou nula, jedná se tak o tzv. „*zero padding*“. Obvyklý rozměr konvolučního jádra je 3×3, tedy tři pixely na výšku a tři pixely na šířku. Takový rozměr jádra je obvykle efektivnější než jádro o velikosti 5×5. Pro změnu počtu kanálů lze použít jádro o velikosti 1×1. [43]

Učení znamená nastavení hodnot konvolučního jádra a hodnot prahů, k čemuž se využívá algoritmu zpětného šíření chyby. Každé vrstvě výstupu odpovídá jedno konvoluční jádro, přičemž všechny váhy jádra se podílí na každé chybě v této vrstvě. Chyby na výstupních neuronech jsou zpětně šířeny sítí a podle zvolené optimalizační metody upravovány váhy konvolučních jader, které se na konkrétních chybách podílely.

Pro rozšíření kontextu sítě lze použít dilatační konvoluci (*dilated convolution*), která klade důraz na vzdálenější pixely. Využitím dilatace tak lze dosáhnout větší zpracovávané oblasti bez zvyšování výpočetních nároků. V případě špatného nastavení parametru dilatace však hrozí generování nepřesných map příznaků.



Obr. 20: Princip konvoluce s hodnotou kroku 1 a bez odsazení [44]

5.1.2 Sdružovací vrstva

Sdružovací vrstva (*pooling layer*) snižuje velikost vstupů s cílem snížení výpočetní a časové náročnosti. Na každou vrstvu je aplikováno sdružovací okno, uvnitř kterého se provádí vybraná operace tak, že skupina sousedních pixelů (okolí) je převedena pouze na jeden pixel, který danou skupinu reprezentuje.

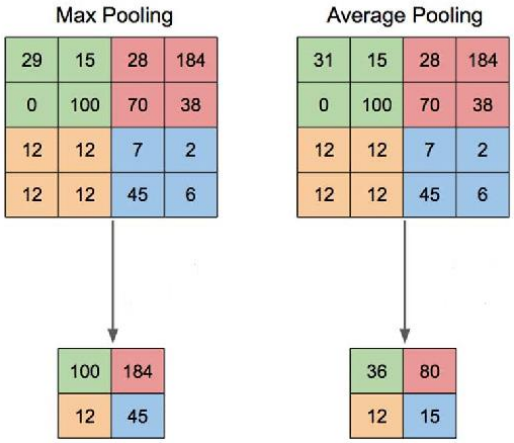
Nejčastější operací je *max pooling*, která z okolí vybere pixel s maximální hodnotou intenzity jasu. Operace je popsána vztahem:

$$g(x,y) = \max f(x,y), \tag{5.2}$$

kde $g(x,y)$ jsou hodnoty pixelů výstupního obrazu a $f(x,y)$ jsou hodnoty pixelů vstupního obrazu. Podobnou operací je *average pooling*, která za reprezentativní pixel vybere průměr intenzity jasů pixelů z okolí. Tato operace je popsána rovnicí:

$$g(x,y) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} f(i,j), \tag{5.3}$$

kde *m* je výška okolí a *n* je šířka okolí. Princip operací je znázorněn na obrázku 21. Parametry této vrstvy jsou velikost okna pro sdružování, velikost posunu okna, typ operace a v některých případech i odsazení. Všechny parametry jsou statické, při procesu učení tak nedochází k jejich změnám. Nejčastější hodnota parametru pro velikost okna je 2×2 px a pro velikost posunu 2 px. [43]



Obr. 21: Princip sdružování pro velikost okna 2×2 px s hodnotou posunu 2 px [45]

5.1.3 Plně propojené vrstvy

Plně propojené vrstvy (*fully connected layers*) jsou zařazovány na konec sítě a slouží pro konečně zpracování informací a jejich převedení do formy vhodné pro konečnou klasifikaci. Všechny neurony z jedné vrstvy jsou propojeny s neurony z následující vrstvy. Počet neuronů ve výstupní vrstvě je dán řešenou úlohou.

5.1.4 Ostatní vrstvy

Aktivační vrstva (*activation layer*) plní stejnou funkci jako aktivační funkce v klasických neuronových sítích. Nelinearita však není aplikována na výstup neuronu, ale na výstup jednotlivých vrstev. Vrstva neobsahuje žádné učící parametry, pouze podle zadané aktivační funkce zpracuje vstupní hodnoty a pošle na výstup.

Nadvzorkovací vrstvy se nachází v konvolučních sítích určených k segmentaci obrazu. Tyto vrstvy slouží pro získání původní velikosti obrazu z map příznaků vytvořených při podvzorkování. K tomu existuje několik metod. Metoda nejbližšího souseda pouze rozkopíruje pixely do okolních oblastí. Metoda bilineární transformace počítá každý

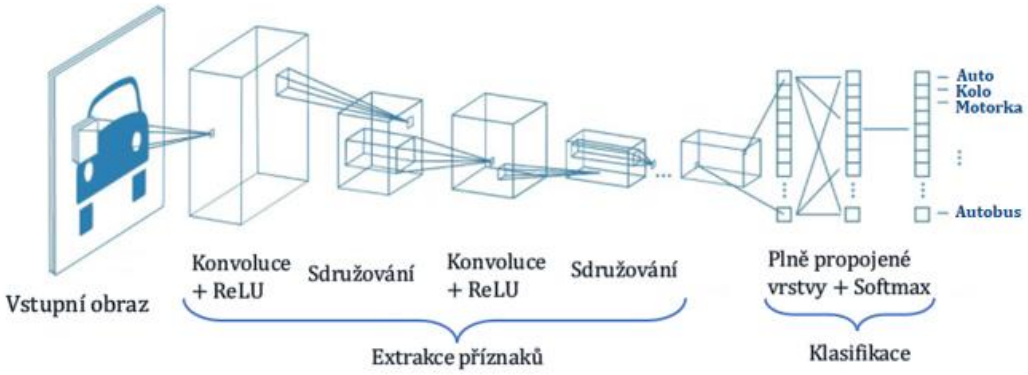
pixel jako váhový průměr z okolních pixelů. Tato metoda umožňuje lepší aproximaci dat, ale má větší výpočetní náročnost. Metoda transponované konvoluce, označovaná také jako dekonvoluce, je nejpoužívanější metoda používaná v nadvzorkovacích vrstvách. Metoda pracuje s učitelnými parametry a umožňuje tak adaptivní zvětšování rozlišení. [46]

Dropout vrstva slouží ke snížení rizika přeučení a lepší generalizaci příznaků. Tato vrstva během dopředného průchodu informací náhodně nuluje některé neurony. Při zpětném průchodu se vynulované neurony nepodílí na propagaci chyby.

5.2 Architektury konvolučních sítí

Obecná architektura konvolučních sítí se skládá z jednotlivých vrstev popsaných v kapitole 5.1, není však podmínkou použití všech vrstev, které jsou v kapitole popsány. Existují různé architektury, které vykazují dobré výsledky na konkrétní typy úloh. Neexistuje však žádná univerzální architektura, která by řešila různorodé problémy. Příklad zjednodušené architektury pro klasifikaci je zobrazen na obrázku 22. Je možné si z jednotlivých vrstev sestavit vlastní architekturu, doporučuje se však využít některou z volně dostupných tzv. „*state of the art*“ architektur, které fungují dobře a často již obsahují předtrénované váhy na velkém počtu různých obrazových dat. Obecně platí, že čím více vrstev architektura obsahuje, tím lépe je schopna řešit zadanou úlohu, roste však riziko výskytu problémů popsaných v kapitole 4.7.

Pro architektury konvolučních sítí existují různá pravidla. První vrstvou je vstupní vrstva, na kterou jsou přiváděna obrazová data. Poslední vrstvou je obvykle plně propojená vrstva, která plní funkci klasifikátoru a slouží tak pro zařazení výstupu do jednotlivých tříd. Mezi vstupní a výstupní vrstvu jsou vloženy konvoluční vrstvy, za kterými se obvykle nachází aktivační vrstvy a sdružovací vrstvy.



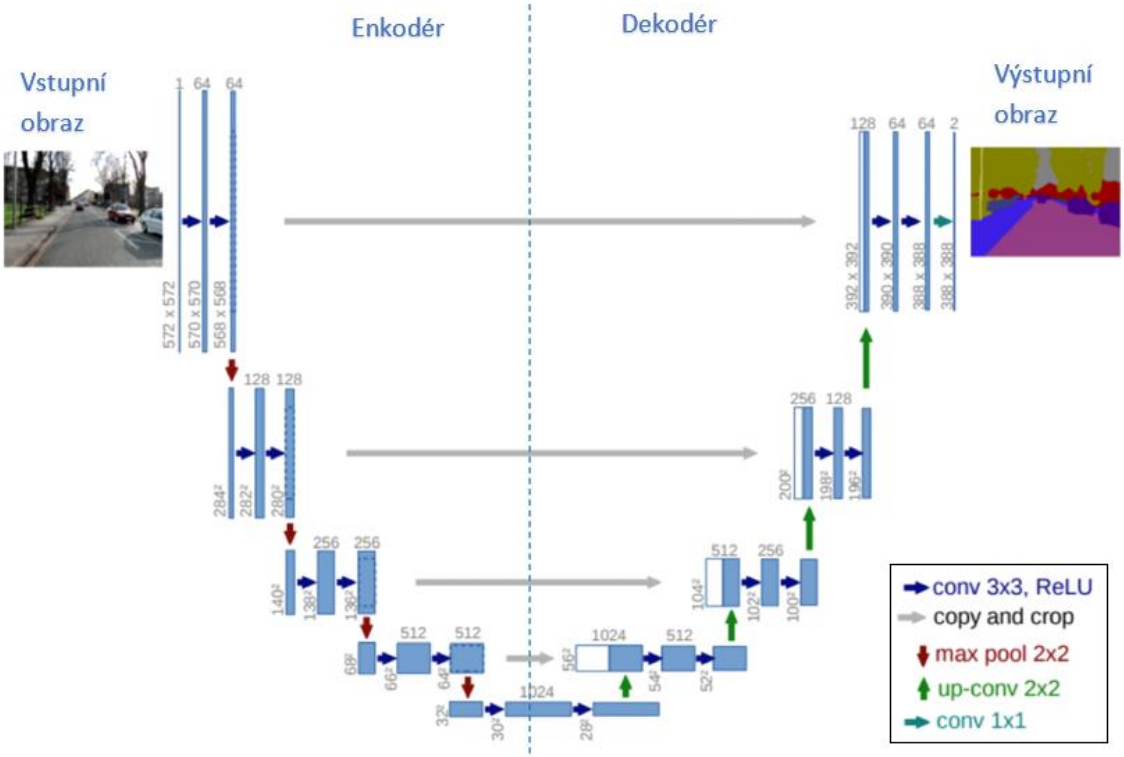
Obr. 22: Zjednodušený příklad architektury konvoluční sítě pro klasifikaci objektů [43]

5.2.1 Unet

Unet je architektura plně konvoluční sítě (*fully convolutional network*) pro segmentaci obrazových dat. Původně se jednalo o model určený pro zpracování biomedicínských

snímků. Jak je vidět na obrázku 23, struktura modelu je inspirována modelem autoenkodéru (*autoencoder*) a skládá se ze dvou hlavních částí. Levá část sítě se nazývá enkodér (*encoder*) a slouží pro podvzorkování (*downsample*) snímků a vytváření map příznaků. Při průchodu dat se snižuje rozlišení vstupního obrazu a zvyšuje počet kanálů. Tato část sítě je někdy označována jako „*feature extractor*“. Pravá část sítě se nazývá dekodér (*decoder*) a slouží pro nadvzorkování (*upsample*) snímků, při kterém je snahou z map příznaků získat obraz s tolika kanály, kolik je predikovaných tříd. Mezi levou a pravou částí se nachází reziduální vazby, které slouží pro spojování filtrů (*concatenate*), což umožní efektivnější propagaci gradientu chybové funkce i do vyšších vrstev. [47]

Původní Unet architektura byla navržena se symetrickou levou i pravou částí. Jelikož je však pro enkodér potřeba větší kapacita sítě a tím i více vrstev, tak je tato myšlenka dnes již překonána. Obvykle se pro levou část sítě využívají robustní architektury vhodné pro klasifikaci, které jsou občas označovány jako „*backbone*“.



Obr. 23: Původní Unet architektura [48]

5.2.2 LinkNet

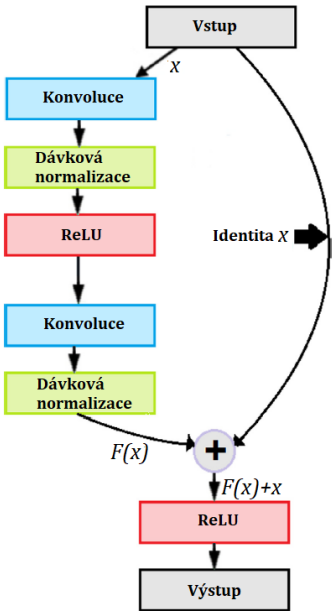
LinkNet architektura slouží pro segmentaci obrazu a vychází z Unet architektury. Hlavní rozdíl oproti architektuře Unet je ve spojeních mezi enkodérem a dekodérem. LinkNet architektura obsahuje propojení, pomocí kterých jsou posílány z kódující do dekódující částí celé příznakové mapy, což snižuje počet učitelných parametrů

i výpočetní náročnost učení. Architektura je tak vhodná pro použití v aplikacích, kde je požadována vysoká rychlost učení a nízká doba predikce. [49]

5.2.3 ResNet

Konvoluční sítě disponují problémem, že čím více vrstev obsahují, tím hůře je zpětně propagován gradient chybové funkce a síť se tak hůře učí. ResNet je architektura konvoluční sítě, která je schopna i s velkým počtem vrstev dosahovat dobrých výsledků. Je to díky velkému počtu reziduálních propojení, jejichž zjednodušený princip je zobrazen na obrázku 24. Data tak nejsou sítí šířena pouze sériově, ale i paralelně, čímž nedochází k velkému snižování gradientu chybové funkce při jeho zpětné propagaci. Podle počtu konvolučních vrstev lze rozlišit různé architektury (např. ResNet34 obsahuje 34 konvolučních vrstev). [51]

Mezi odvozené architektury lze zařadit například ResNeXt, která jednotlivé skupiny vrstev rozdělí do více cest, což vede k lepšímu výsledku, ale i ke zvýšení výpočetní složitosti. Další odvozenou topologií je SE-ResNet, která využívá tzv. „*squeeze and excitation*“ bloky. Blok rozdělí vstupní mapu příznaků na jednotlivé kanály a vytvoří vektor, který jednotlivým kanálům přiřadí učitelné váhy podle jejich důležitosti. [50,51]



Obr. 24: Princip reziduálního propojení vrstev [51]

5.2.4 EfficientNet

EfficientNet je architektura sítě založená na škálovatelnosti tří hyperparametrů. Podle hloubky sítě (počtu vrstev), šířky sítě (počtu filtrů) a vstupního rozlišení lze topologii rozdělit do osmi kategorií, od základní architektury B0 po největší architekturu B7. Součástí architektur jsou „*squeeze and excitation*“ bloky a „*MBConv*“ bloky, což jsou

invertované residuální bloky, které propojují vrstvy na místech s nízkým počtem parametrů (tzv. „*bottleneck*“).
Princip architektury je založen na tzv. složeném škálování, což umožňuje rovnoměrné škálování všech tří zmíněných hyperparametrů v konstantním poměru, čímž se zvýší efektivita učení. Škálování probíhá podle rovnic:

$$\begin{aligned} \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2, \\ d = \alpha^{\Phi}, w = \beta^{\Phi}, r &= \gamma^{\Phi}, \end{aligned} \tag{5.4}$$

za podmínek $\alpha \geq 1$, $\beta \geq 1$, $\gamma \geq 1$, kde α je hloubka sítě d, β je šířka sítě w, γ je vstupní rozlišení r a Φ je koeficient udávající kapacitu výpočetních zdrojů. [52]

5.2.5 YOLO

YOLO (*you only look once*) je architektura konvoluční neuronové sítě zaměřená převážně na detekci objektů v obraze. Její hlavní výhodou je vysoká výpočetní rychlost, díky které je schopna zpracovávat obrazová data v reálném čase. Oproti klasickým konvolučním sítím YOLO zpracovává celý vstupní obraz najednou, což jí umožňuje pracovat i s celkovým kontextem v obraze. Princip spočívá v diskretizaci vstupního obrazu na mřížku, která se skládá z $S \times S$ buněk. Zodpovědnou za detekci je ta buňka, která obsahuje střed nějakého objektu. Pro každou buňku je síť schopna predikovat rámeček ohraničující oblast zájmu (*bounding box*) i skóre určující míru jistoty detekce. Hodnota skóre je založena na pravidle IOU daného rovnicí (4.20). [53]

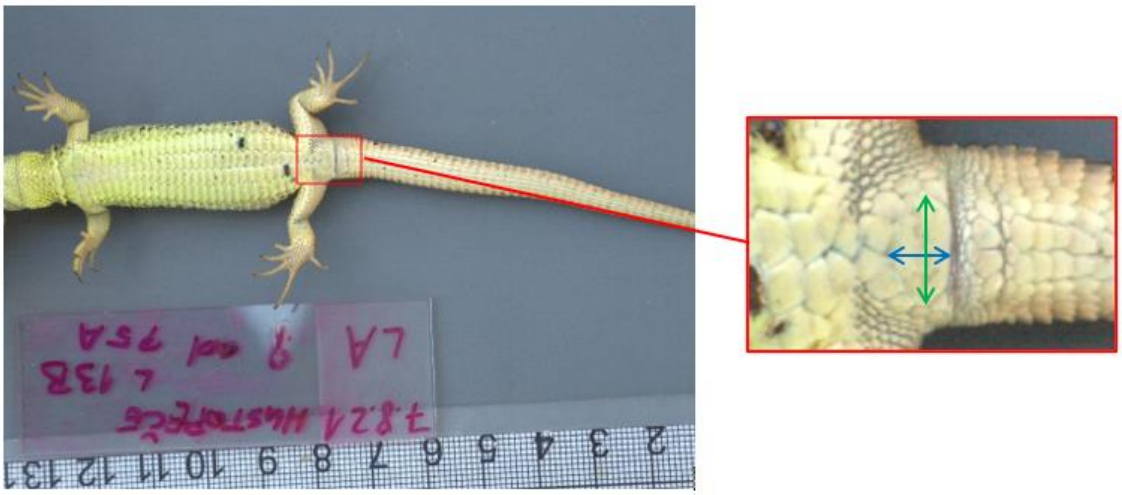
Vylepšením původního modelu vznikla verze YOLOv2, která pro získání příznaků využívá architekturu Darknet19 obsahující 19 konvolučních vrstev. Verze umožňuje klasifikovat data s vyšším rozlišením a umožňuje lepší konvergenci při trénování. Verze YOLOv3 z roku 2018 vznikla vylepšením předchozího modelu. Pro získání příznaků využívá architekturu Darknet53, která obsahuje 53 konvolučních vrstev, což umožňuje přesnější detekci, nižší výpočetní čas, avšak oproti předchozí verzi nižší počet zpracovaných snímku za sekundu (*frames per second*). [53,54]

6 SYSTÉM PRO MĚŘENÍ ROZMĚRŮ ANÁLNÍHO ŠTÍTKU

Úloha měření rozměrů análního štítku ještěrky obecné je modelovým příkladem, ve kterém je nutné aplikovat robustní metody, které umožňují realizovat měření variabilních objektů zachycených pomocí běžných komerčních fotoaparátů. Přesnost bezkontaktního měření závisí jak na kvalitě a dostupnosti dat, tak i na komplexnosti implementovaného řešení.

6.1 Analýza problému

Cílem zadaného problému je co nejpřesněji změřit délku a šířku análního štítku ještěrky obecné. Z těchto údajů lze vypočítat tzv. index análního štítku, který udává míru protáhlosti této šupiny. Rozměry, které je třeba měřit, ukazuje obrázek 25.



Obr. 25: Anální štítek ještěrky obecné s měřenými rozměry, modrá šipka značí délku a zelená šipka šířku šupiny [Foto: R. Smolinský]

Systém pro měření rozměrů análního štítku ještěrky obecné se skládá ze čtyř různých kroků, které jsou podrobně popsány v následujících podkapitolách. Nejprve je potřeba z obrázku detekovat pravítko a anální štítek ještěrky. Na detekované pravítko jsou aplikovány algoritmy počítačového zpracování obrazu, pomocí kterých je zjištěno měřítko obrázku. Detekovaný anální štítek je pomocí natrénovaných modelů konvoluční neuronové sítě segmentován do binární podoby. Takto získaná šupina je následně upravena do vhodné podoby a pomocí předem získaného měřítka jsou změřeny její rozměry.

Pro řešení problému byl využit programovací jazyk **Python**. Jedná se o vysokoúrovňový dynamicky interpretovaný programovací jazyk, který podporuje

různá paradigmata, včetně objektově orientovaného programování. Mezi výhody jazyka patří jeho jednoduchost, ale zároveň i vysoká efektivita.

6.2 Detekce pravítka a análního štítku

Pro automatickou detekci pravítka a análního štítku byly pomocí konvoluční neuronové sítě YOLOv3 natrénovány dva modely umělých neuronových sítí. Jelikož je potřeba detekovat pouze dvě třídy (pravítko a šupinu), bylo pro zachování lepší manipulace rozhodnuto, že jeden model bude odpovídat jedné třídě. Pro více tříd je však obvykle vhodnější vytvořit pouze jeden model, čímž je ušetřen čas při trénování i paměťové nároky pro uložení modelů.

Nejprve bylo potřeba datovou sadu anotovat, tedy označit oblast zájmu určenou pro detekci. K tomu byl použit grafický nástroj **LabelImg**[3], který disponuje jednoduchým ovládáním, ale pro daný účel je velmi efektivní. Pro každou třídu zvlášť bylo z datové sady anotováno 200 obrázků, ke kterým bylo vytvořeno 200 souborů s příponou .txt. Každý z vytvořených souborů obsahuje pětici čísel ve tvaru:

$$<class> <x> <y> <w > <h>,$$

kde *class* definuje danou třídu v seznamu tříd, který se nachází v souboru *classes.txt*, *x* (resp. *y)* udává souřadnice středu zájmové oblasti ve vertikálním (resp. horizontálním) směru a *w* (resp. *h*) udává šířku (resp. výšku) zájmové oblasti vůči celkovým rozměrům obrázku. Souřadnice a rozměry jsou normovány na hodnoty v intervalu $\langle 0, 1 \rangle$.

Pro natrénování modelů konvolučních neuronových sítí bylo použito rozhraní **Google Colaboratory**[4], také známé jako Google Colab. Jedná se o platformu, která využívá programovacího jazyka Python a je vhodná zejména pro umělou inteligenci a strojové učení. Tuto cloudovou službu poskytuje Google Research v základní verzi zdarma. Vývojové prostředí umožňuje uživateli psát zdrojový kód v Jupyter notebooku, který je hostován přes uživatelský účet Google. Velkou výhodou platformy jsou výpočetní prostředky, především výkonné grafické karty, které jsou uživateli poskytovány. Vzhledem k vysokým výpočetním požadavkům při trénování hlubokých neuronových sítí se jedná o klíčový důvod pro využití této platformy. [55]

K učení neuronové sítě byl využit *open-source framework* **Darknet**[5], který byl naklonován do rozhraní Colabu. Před spuštěním trénování bylo potřeba v přednastavených hodnotách konfiguračního souboru *yolov3_training.cfg* provést několik změn. Nejprve bylo potřeba zakomentovat řádky 3 a 4 určené pro testování a odkomentovat řádky 6 a 7 určené pro trénování modelů. Velikost dávky (*batch size*) tak byla přenastavena na 64. Během jedné iterace je tak zpracováno 64 obrázků. Parametr *subdivisions* byl nastaven na hodnotu 16. Dávka je tak během iterace rozdělena do čtyř malých dávek (*mini-batches*), která každá obsahuje 16 obrázků. Konkrétní hodnoty těchto parametrů jsou voleny podle dostupných výpočetních

[3] https://github.com/tzutalin/labelImg
[4] https://colab.research.google.com/
[5] https://github.com/pjreddie/darknet

prostředků. Při použití grafické karty s nízkou pamětí se volí větší hodnota *subdivisions*, čímž je však snížen počet obrázků v *mini-batches* a tím dojde i k delší době trénování. Ve všech třech yolo vrstvách byl změněn počet tříd z 80 na 1 a ve všech konvolučních vrstvách nacházející se v souboru nad yolo vrstvami, tak byl podle vztahu $n = (c + 5) \cdot 3$ změněn počet filtrů na 18, kde *n* je počet filtrů a *c* je počet tříd. Všechny ostatní hodnoty hyperparametrů a nastavení jednotlivých vrstev sítě byly v souboru zachovány. Konfigurační soubor pro trénování s přenastavenými hodnotami je dostupný v příloze (**Příloha 1**).

Dále bylo nutné vytvořit dva nové soubory, které byly pojmenovány *obj.names* a *obj.data*. Do souboru *obj.names* byl uložen seznam trénovaných tříd a do souboru *obj.data* byl na jednotlivé řádky uložen počet trénovaných tříd, cesty k souborům se seznamem trénovacích a validačních obrázků, cesta k seznamu tříd (tedy k souboru *obj.names*) a cesta pro uložení natrénovaného modelu.

Celý dataset byl rozdělen na 180 trénovacích (90 %) a 20 validačních (10 %) obrázků. Pro trénování byla alokována grafická karta Tesla K80 disponující pamětí 11,4 GB. U modelu pro detekci pravítka i u modelu pro detekci análního štítku probíhalo trénování, dokud průměrná hodnota ztrátové funkce (*avg loss*) neklesla pod hodnotu 0,1. U modelu pravítka bylo trénování zastaveno po 1200 iteracích s hodnotou ztrátové funkce 0,097 a přesností mAP (*mean average precision*) 89,93 % s průměrnou hodnotou IOU (*intersects over union*) 63,18 %. U modelu análního štítku bylo trénování zastaveno po 2600 iteracích s hodnotou ztrátové funkce 0,095 a přesností mAP 87,88 % s průměrnou hodnotou IOU 65,35 %. Metrika mAP vychází z metriky IOU, která je popsána v kapitole 4.3.4. Objekt je považován za správně predikovaný, pokud hodnota IOU přesáhne hranici 0,5. Soubor pro trénování modelů v rozhraní Google Colaboratory určených pro detekci je přiložen v příloze (**Příloha 2**).

6.3 Zjištění měřítka obrázku

Zjištění měřítka obrázku probíhá pomocí dvou metod (dále označeny jako metoda A a metoda B). Tyto metody jsou určeny pro zjištění počtu pixelů, které odpovídají reálné vzdálenosti (*pixels per metric*).

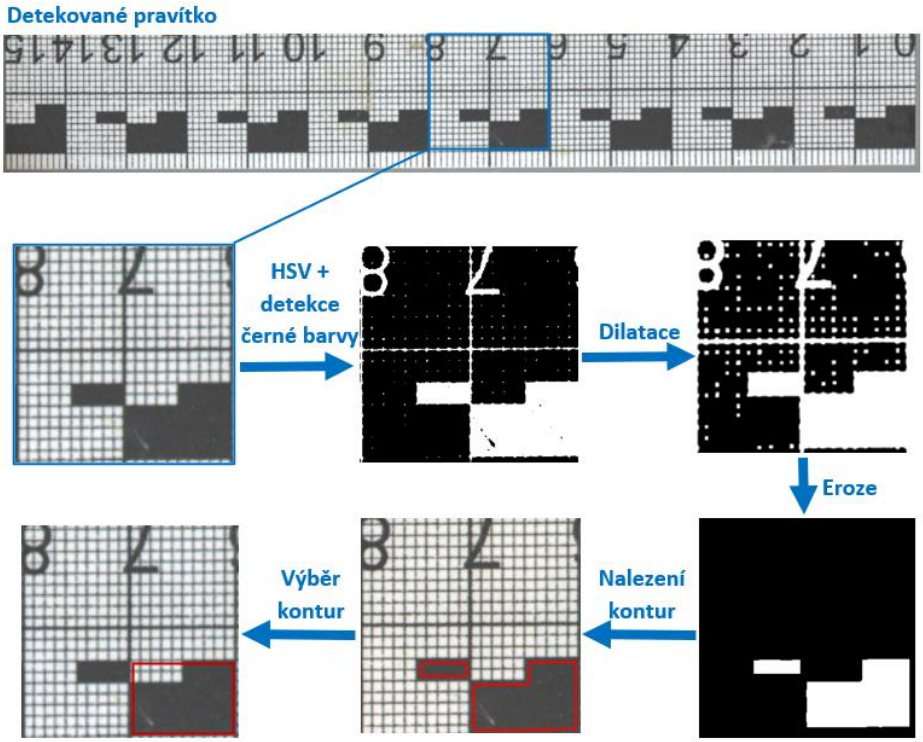
Metoda A

Nejprve je snímek pravítka převeden do barevného modelu HSV[6]. K tomu je využita funkce `cv2.cvtColor()` z knihovny OpenCV. Argumentem této funkce je obrázek pro převedení a typ převodu, který je `cv2.COLOR_RGB2HSV`. Takto převedený snímek umožňuje detekovat objekty černé barvy, což je provedeno funkcí `cv2.inRange()`, jejíž argumenty jsou vstupní obraz, dolní mez barvy a horní mez barvy. Jednotlivé meze jsou dány konkrétní detekovanou barvou a jsou uvedeny ve formátu: [odstín, sytost, jas]. Pro dolní mez černé barvy byly zvoleny hodnoty [0, 0, 0] a pro horní mez černé

[6] Z anglického *hue* (odstín), *saturation* (sytost) a *value* (hodnota jasu).

barvy byly zvoleny hodnoty [180, 150, 100]. Popsanou úpravou obrázku vznikne snímek v binárním formátu, na který je aplikována dilatace. K tomu je využita funkce `cv2.dilate()`. Argumenty této funkce jsou vstupní obraz, jádro a počet iterací. Velikost jádra je 5×5 px a počet iterací má hodnotu jedna. Následně se pro zjednodušení struktury objektů provede pomocí funkce `cv2.erode()` několikanásobná eroze snímku. Velikost jádra je opět 5×5 px a počet iterací byl stanoven celkem na čtyři. Z takto upraveného snímku jsou funkcí `cv2.findContours()` nalezeny všechny obrysy (kontury). Argumenty této funkce jsou vstupní snímek, režim vyhledávání kontur a metoda pro aproximaci kontur. Režim vyhledávání byl určen `cv2.RETR_EXTERNAL` a metoda pro aproximaci kontur byla zvolena `cv2.CHAIN_APPROX_SIMPLE`.

Nalezené kontury jsou statisticky zpracovány a vybrány pouze kontury adekvátní pro nalezení měřítka. Výběr kontur probíhá v následujících krocích. Nalezené kontury jsou nejprve funkcí seřazeny sestupně podle velikosti a do pole *big_cnts* je uloženo pouze osm největších kontur, kterým je funkcí `cv2.minAreaRect()` opsán obdélník s minimálním obsahem. U každého obdélníku jsou spočítány euklidovské vzdálenosti rozměrů hran. Vzdálenosti jsou spočítány funkcí `dist.euclidean()` z knihovny SciPy. Argumenty této funkce jsou souřadnice bodů, vůči kterým probíhá výpočet vzdálenosti. Hledané kontury by měly mít rozměry hran v poměru 7:10, protože obdélníky na pravítku mají 7 mm na výšku a 10 mm na šířku, proto jsou do pole uloženy pouze takové délky hran, které danému poměru přibližně odpovídají. Z tohoto pole je spočítána jedna průměrná hodnota *pixels_per_mm*, která přibližně odpovídá počtu pixelů na jeden milimetr. Bylo zjištěno, že tato hodnota je obvykle o něco málo menší, než skutečný počet pixelů, který odpovídá jednomu milimetru. Z toho důvodu je nalezená hodnota měřítka vynásobena experimentálně zjištěným koeficientem. Pro hodnotu měřítka menší než 15 je koeficient 1,1. Pro hodnotu měřítka větší než 15 a menší než 20 je koeficient 1,08. Pro hodnotu měřítka větší než 20 a menší než 25 je koeficient 1,06 a pro hodnotu měřítka větší než 25 je koeficient 1,04. Princip nalezení měřítka metodou A je znázorněn na obrázku 26.



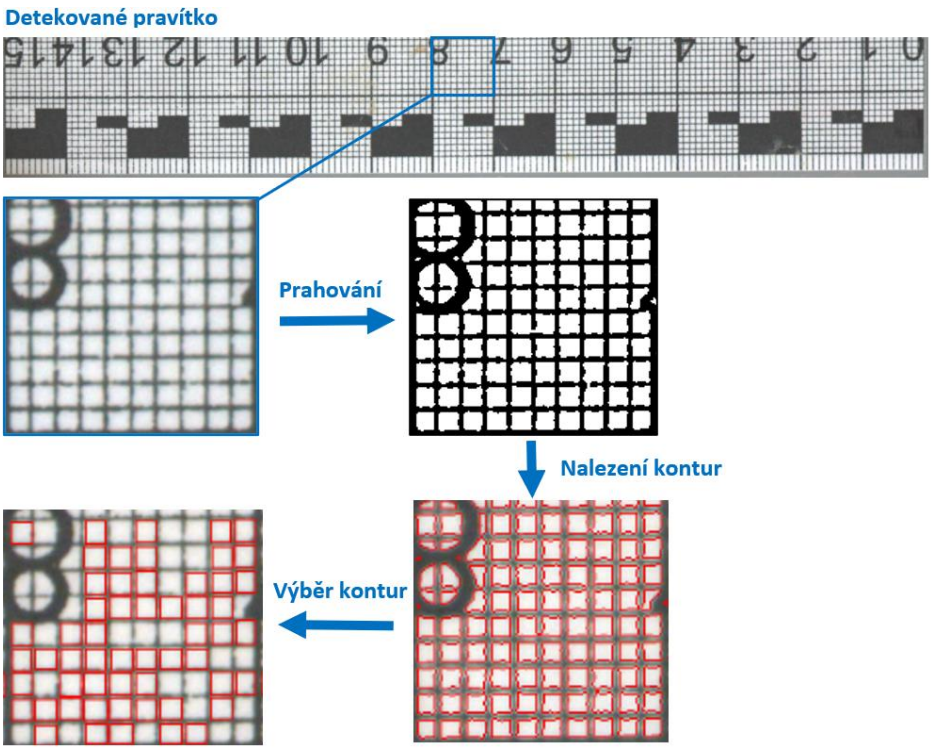
Obr. 26: Princip nalezení měřítka metodou A

Metoda B

Nejprve je snímek pravítka převeden funkcí `cv2.cvtColor()` na stupně šedi. Argumenty funkce jsou vstupní obraz a typ převodu, který je `cv2.COLOR_RGB2GRAY`. Funkce převede snímek pomocí rovnice $Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B$, kde Y je odstín šedi, R je odstín červené, G je odstín zelené a B je odstín modré barvy. Poté je pomocí Gaussova filtru pro vyhlazení šumu aplikováno rozostření (*blur*). Toho je docíleno funkcí `cv2.GaussianBlur()`, jejíž argumenty jsou vstupní snímek a rozměry jádra, které byly zvoleny na hodnoty 5×5 px. Rozostřený snímek je funkcí `cv2.adaptiveThreshold()` prahován do binárních hodnot jasu. Adaptivní prahování bylo zvoleno z důvodu velkých jasových rozdílů na jednotlivých snímcích. Argumenty funkce jsou vstupní snímek, maximální hodnota jasu pixelu, metoda prahování, typ prahování, velikost sousedství pro výpočet hodnoty prahu a konstanta pro korekci hodnoty prahu. Maximálná hodnota jasu pixelu byla stanovena na 255. Metoda prahování `cv2.ADAPTIVE_THRESH_GAUSSIAN_C` počítá hodnotu prahu jako vážený průměr pixelů sousedství, přičemž hodnoty vah jsou dány velikostí Gaussova okna. Typ prahování je `cv2.THRESH_BINARY` a určuje hodnotu jasu pixelu po prahování. Velikost sousedství bylo stanoveno na 11×11 px a konstanta pro korekci na hodnotu 2. Z prahovaného snímku jsou funkcí `cv2.findContours()` nalezeny všechny kontury. Stejně jako v metodě A byly argumenty určeny `cv2.RETR_EXTERNAL` a `cv2.CHAIN_APPROX_SIMPLE`.

Všechny nalezené kontury jsou statisticky zpracovány a vybrány pouze kontury adekvátní pro nalezení měřítka. Výběr kontur probíhá v následujících krocích. Nejprve

je všem nalezeným konturám pomocí funkce `cv2.minAreaRect()` opsán obdélník s minimálním obsahem. U každého obdélníku jsou spočítány euklidovské vzdálenosti rozměrů hran. Hledané kontury by měli odpovídat čtverci, proto jsou z množiny nalezených kontur uloženy do pole *cnts_close* pouze takové kontury, jejichž strany se neliší o více, než je určitá prahová hodnota, která byla experimentálně stanovena na 5 px. Z uložených kontur jsou funkcí `cv2.contourArea()` spočítány obsahy, které jsou uloženy do pole *c_area*, ze kterého jsou smazány extrémně nízké a extrémně vysoké hodnoty obsahů. Z pole kontur *cnts_close* jsou odstraněny kontury, jejichž obsah se od mediánu obsahů pole *c_area* liší více, než je určitá prahová hodnota, která byla experimentálně stanovena na 10 px. Zbylým konturám v poli *cnts_close* jsou pomocí funkce `cv2.boundingRect()` opsány obdélníky. Výšky a šířky těchto obdélníků jsou zprůměrovány a je vypočtena jedna hodnota *pixels_per_mm*, která přibližně odpovídá počtu pixelů na jeden milimetr. Podobně jako v metodě A je nalezená hodnota vynásobena experimentálně zjištěným koeficientem. Pro hodnotu měřítka menší než 15 je koeficient 1,25. Pro hodnotu měřítka větší než 15 a menší než 20 je koeficient 1,2. Pro hodnotu měřítka větší než 20 a menší než 25 je koeficient 1,15 a pro hodnotu měřítka větší než 25 je koeficient 1,1. Princip nalezení měřítka metodou B je znázorněn na obrázku 27.



Obr. 27: Princip nalezení měřítka metodou B

6.4 Segmentace análního štítku

Pro segmentaci análního štítku bylo potřeba natrénovat model konvoluční neuronové sítě. Nejprve však bylo nutné vytvořit dataset, na kterém by se síť učila. K 200 původním obrázkům s obvyklým rozlišením 2602×3906 px byly v programu GIMP[7] vytvořeny odpovídající binární masky ve formátu [0, 255], kde 255 odpovídá bílé barvě a byl tak označen anální štítek a 0 odpovídá černé barvě, kterou bylo označeno všechno ostatní. Z takto získaných obrázků byly kolem análního štítku vytvořeny obdélníkové výřezy s rozlišením 320×480 px a na těchto datech poté byly natrénovány modely konvolučních neuronových sítí pro segmentaci.

K natrénování neuronové sítě byla využita softwarová knihovna **TensorFlow**[8], která byla vytvořena společností Google. Jedná se o knihovnu s otevřeným zdrojovým kódem (*open-source*), která programátorům poskytuje spoustu prostředků pro práci s umělou inteligencí a disponuje vysokými možnostmi použití a podrobně zpracovanou dokumentací. Pro práci s neuronovými sítěmi knihovna poskytuje aplikační rozhraní **Keras**[9], které vyniká relativně snadným používáním, modularitou, kompatibilitou a snadnou rozšiřitelností.

Pro větší kontrolu průběhu trénování neuronové sítě byla využita webová služba **Weights & Biases**[10], která je pro nekomerční využití zdarma. Služba poskytuje přehled o stavu trénování, využití výpočetních prostředků, a to v reálném čase. Umožňuje snadné sdílení výsledků v rámci různých projektů, uložit hotový model i trénovací datasety a poskytuje mnoho dalších funkcí.

Trénování sítí probíhalo v rozhraní Colabu na alokované grafické kartě Tesla K80 disponující pamětí 11,4 GB. Pro trénování neuronové sítě byla využita knihovna Segmentation Models[11], která je založená na aplikačních rozhraních Keras a TensorFlow a je určena pro učení konvolučních sítí pro segmentaci obrazových dat. Celkem bylo naučeno mnoho modelů, ze kterých byly následně vybrány tři a bylo provedeno jejich sloučení (*ensemble*). Mezi fixní parametry sítě pro nastavení správného učení byly zařazeny:

- počet tříd (*classes*) = 2,
- aktivační funkce v poslední vrstvě sítě (*activation*) = sigmoid,
- inicializované váhy sítě: (*encoder weights*) = imagenet,
- vstupní rozlišení (*input shape*) = (320, 480, 3),
- velikost dávky (*batch size*) = 16,
- metrika (*metrics*): iou score.

Počet tříd byl nastaven na hodnotu 2, neboť byl řešen binární problém, ve kterém bylo potřeba segmentovat anální štítek a vše ostatní bylo chápáno jako pozadí. Z toho

[7] https://www.gimp.org
[8] https://www.tensorflow.org/
[9] https://keras.io/api/
[10] https://wandb.ai/site
[11] https://github.com/qubvel/segmentation_models

důvodu byla do poslední vrstvy sítě zařazena aktivační funkce sigmoid. Váhy sítě byly inicializovány na dataset imagenet, který obsahuje přes 14 milionů anotovaných obrázků. Pro správný průchod obrazových dat sítí a vytváření map příznaků bylo vstupní rozlišení obrázků nastaveno na 320×480 px s hodnotou hloubky barevného formátu 3. Velikost dávky (počet obrázků při průchodu sítí v rámci jedné epochy) byla vzhledem k velikosti paměti na grafické kartě zvolena na 16. Metrika pro vyhodnocení průběhu trénování byla zvolena iou score, která je založená na principu *intersects over union* popsaného v kapitole 4.3.4.

S následujícími parametry bylo pro nalezení optimálního modelu různě experimentováno:

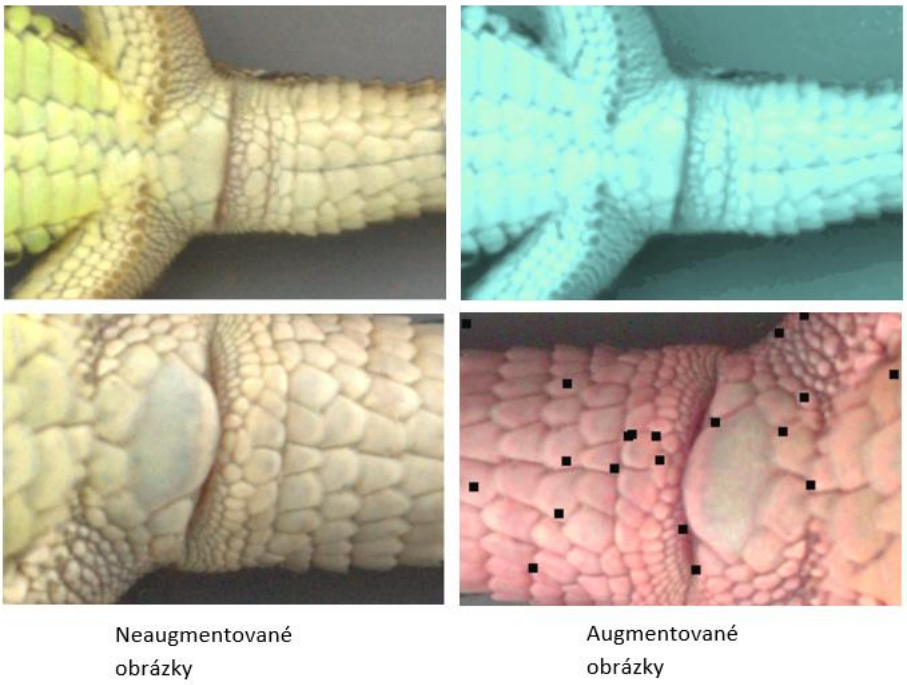
- obecná architektura modelu,
- architektura enkodéru („*backbone*“),
- optimalizační metoda (*optimizer*),
- chybová funkce (*loss function*),
- augmentace dat,
- počet kroků za epochu (*steps per epoch*),
- počet epoch,
- učící rychlost (*learning rate*).

Architektur konvolučních sítí existuje velmi mnoho. Vybrané architektury konvolučních sítí jsou popsány v kapitole 5.2. Mezi obecné architektury trénovaných modelů byl zařazen Unet a Linknet. Architektury použité v enkodéru těchto obecných architektur byly s ohledem na dostupné výpočetní prostředky voleny SE-ResNet18, SE-ResNet34, SE-ResNet50, ResNet50, MobileNetV2 a EfficientNetB0. Tyto architektury se liší zejména svojí robustností a počtem vrstev. S tím souvisí různá velikost modelu a doba trénování či inference. Architektury dekodérů byly ponechány beze změny z obecných použitých architektur. Mezi optimalizační metody byly pro své vlastnosti zvoleny Adam a Adagrad. Obě metody jsou více popsány v kapitole 4.3.3. Mezi zkoušené chybové funkce byly zařazeny binary cross-entropy, JaccardLoss, BinaryCELoss a BinaryFocalLoss. Vybrané chybové funkce jsou více popsány v kapitole 4.3.2.

Z důvodu nízkého počtu originálních trénovacích obrázků bylo potřeba využít poměrně robustní augmentace, pomocí které dojde k umělému rozšíření datové sady. Pro augmentace obrázků byla využita knihovna Albumentations[12], která disponuje jednoduchým používáním, ale velmi efektivními možnostmi dosahujícími takřka 50 různých augmentačních stylů. Mezi tyto styly lze zařadit například rozostření, překlopení, rotace, přidání šumu, posterizace, změna jasu či kontrastu apod. Augmentační styly byly voleny primárně s ohledem na datovou sadu. Obecně platí, že využitím augmentací pro segmentaci obrazových dat je snaha dosáhnout přibližně stejně chybové funkce na trénovacích i validačních datech a vyhnout se tak problémům

[12] https://albumentations.ai/

s přeučením nebo špatným zevšeobecňováním při trénování modelů. Příklad použití augmentací zobrazuje obrázek 28.



Obr. 28: Příklad použití augmentací

Pro trénování finálních modelů byly použity níže popsané augmentace. Pro různorodější generování obrázků pro trénování byly jednotlivé augmentace použity s různou pravděpodobností. Použití knihovny Albumentations s níže popsanými augmentacemi je zobrazen na obrázku 29.

- Rozostření (*Blur*) s generovaným jádrem o velikosti 3 až 6 px.
- Posterizace (*Posterize*) se změnou 0 až 7 bitů pro každý barevný kanál z RGB modelu.
- Sépia filtr (*ToSepia*).
- Přetočení v horizontálním směru (*HorizontalFlip*).
- Výřezy (*RandomResizedCrop*) s výstupním rozlišením 320×480 px, rozsahem pro výřez v poměru mezi 0,7 až 0,8 vůči původnímu obrázku, rozsahem poměru stran mezi 0,75 až 1,3 vůči původnímu obrázku a interpolační metodou `cv2.INTER_NEAREST`.
- Změna parametrů v HSV modelu (*HueSaturationValue*) se změnou odstínu v rozsahu od −20 do 20, sytostí v rozsahu od −30 do 30 a jasem v rozsahu od −20 do 20 z celkového intervalu od −100 do 100.
- Změna kontrastu (*RandomContrast*) s hodnotami změny kontrastu v rozsahu od −0,5 do 0,2 z celkového intervalu od −1 do 1.
- Posun barev v RGB modelu (*RGBShift*) s hodnotami posunu pro každou barvu v rozsahu od −20 do 20 z celkového intervalu od −255 do 255.

```
import albumentations as A

def augment_albu(image, mask):
    t = A.Compose([
        A.Blur(always_apply=False, p=0.5, blur_limit=(3, 6)),
        A.Posterize(always_apply=False, p=0.4, num_bits=[(0, 7), (0, 7), (0, 7)]),
        A.ToSepia(always_apply=False, p=0.4),
        A.HorizontalFlip(always_apply=False, p=0.5),
        A.RandomResizedCrop(always_apply=False, p=0.5, height=320, width=480,
                            scale=(0.7, 0.8), ratio=(0.75, 1.3), interpolation=0),
        A.OneOf([
          A.HueSaturationValue(always_apply=True, hue_shift_limit=(-20, 20),
                               sat_shift_limit=(-30, 30), val_shift_limit=(-20, 20)),
          A.RandomContrast(always_apply=True, limit=(-0.5, 0.2)),
          A.RGBShift(always_apply=True, r_shift_limit=(-20, 20),
                     g_shift_limit=(-20, 20), b_shift_limit=(-20, 20)),
                ],p=1,),
        ])

    aug = t(image=image, mask=mask)

    return aug['image'], aug['mask']
```

Obr. 29: Příklad použití knihovny Albumentaions

Parametr *steps per epoch* udává počet dávek, které jsou předloženy síti během jedné epochy. Obvykle je tato hodnota volena fixně v závislosti na velikosti datasetu a velikosti dávky. Pro řešení byla vytvořena metoda `data_generator()`, která umožňuje generovat obrázky za běhu programu, augmentované obrázky tak nebylo nutné nikam ukládat. Počet epoch udává počet iterací pro trénování sítě. Celkový počet obrázků podílejících se na trénování modelů je tedy dán součinem počtu epoch, počtu kroků za epochu a velikostí dávky. Pro trénování finálních modelů bylo zvoleno 200 kroků za epochu a 10 epoch. Velikost dávky byla zvolena fixně na hodnotu 16 z důvodu popsaného výše v textu. Tento počet se ukázal být dostačující, neboť zvýšením počtu docházelo k přeučení a špatné generalizaci. Význam učící rychlosti je popsán v kapitole 4.6. Její hodnota byla volena dynamicky pomocí funkce `ReduceLROnPlateau()`, která funguje na principu zpětného volání (*callbacks*) a umožňuje po dokončení každé epochy vykonání zadané akce. V rámci této práce byly využity funkce zpětného volání pro dynamickou změnu učící rychlosti, zobrazení predikce pomocí aktuálně natrénovaného modelu, průběžné ukládání dosud nejlepšího modelu a zaznamenávání průběhu učení na webovou službu Weights and Biases. Možné použití funkcí zpětného volání je zobrazeno na obrázku 30. Učící rychlost je ve výchozím stavu obvykle nastavena podle použité optimalizační metody. Pro optimalizační metodu Adam je to hodnota 0,001. Pomocí zpětného volání funkce `ReduceLROnPlateau()` docházelo ke snížení hodnoty učící rychlosti na polovinu, pokud nedošlo ke snížení chybové funkce během dvou po sobě následujících epoch. Vzhledem k nízkému celkovému počtu epoch se toto nastavení ukázalo být nejúčinnější. Pro zobrazení predikce pomocí aktuálně natrénovaného modelu byla využita třída `DisplayCallback()` z knihovny TensorFlow a vytvořená metoda `show_predictions()`. Průběžné ukládání nejlepšího modelu probíhalo pomocí

`ModelCheckpoint()`. Model byl automaticky uložen, pokud byla nižší hodnota chybové funkce na validačních datech trénovaného modelu než již uloženého modelu.

```
from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint

class DisplayCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs=None):
    show_predictions()

reduce_lr = ReduceLROnPlateau(monitor='loss', patience=2, factor=0.5, verbose=1)
checkpoint_filepath = "/content/drive/MyDrive/models/best_model.h5"
model_checkpoint_callback = ModelCheckpoint(filepath=checkpoint_filepath,
                                            save_weights_only=False, save_best_only=True,
                                            monitor='val_loss', verbose=1, mode='min')
```

Obr. 30: Příklad použití funkcí zpětného volání

Pro učení neuronové sítě je nejprve potřeba sestavit model. Ukázka sestavení modelu zobrazuje obrázek 31.

```
import segmentation_models as sm

model = sm.Unet(backbone_name='seresnet34',
                classes=2,
                encoder_weights='imagenet',
                input_shape=(320, 480, 3),
                activation='sigmoid')
```

Obr. 31: Příklad sestavení modelu

Sestavený model je potřeba s vybranými hyperparametry zkompilovat. Příklad kompilace modelu ukazuje obrázek 32.

```
from tensorflow.keras.optimizers import Adam
from segmentation_models.metrics import iou_score
from segmentation_models.losses import binary_crossentropy

model.compile(Adam(),'binary_crossentropy', metrics= [iou_score])
```

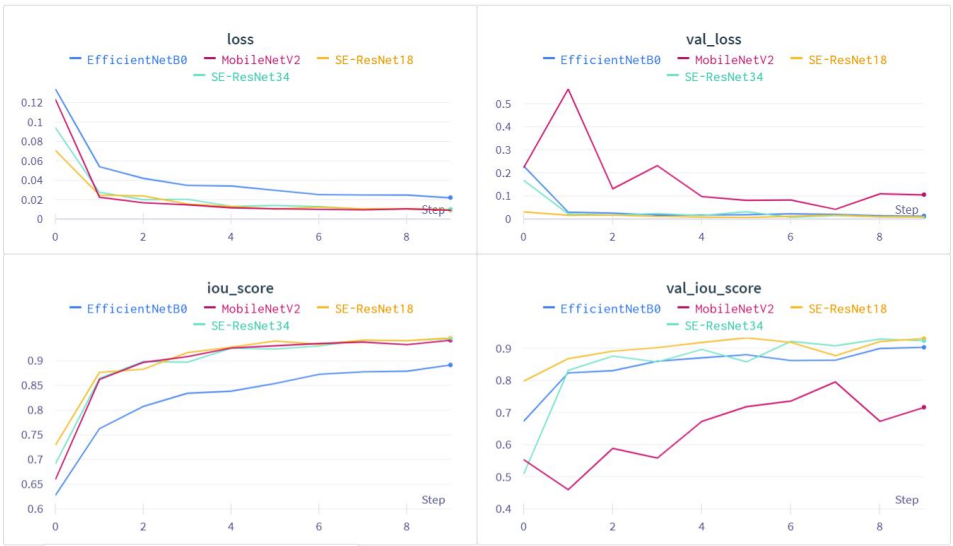
Obr. 32: Příklad kompilace modelu

Po úspěšné kompilaci modelu je možné spustit jeho trénování. Příklad spuštění trénování modelu s dynamickým generováním dat ukazuje obrázek 33. Výběr hyperparametrů a nastavení pro trénování je popsán výše.

```
model_history = model.fit(gen_train(),
                          validation_data=gen_val(),
                          steps_per_epoch=200,
                          validation_steps=100,
                          batch_size=16,
                          epochs=10,
                          callbacks=[reduce_lr, wandb.keras.WandbCallback(),
                                     model_checkpoint_callback, DisplayCallback()])
```

Obr. 33: Spuštění trénování modelu

Průběh trénování pro obecnou architekturu Unet a některé vybrané architektury enkodérů je zobrazen na obrázku 34. Grafy zobrazují hodnotu ztrátové funkce binary cross-entropy a hodnotu metriky iou score pro trénovací a validační data pro 10 epoch při 200 krocích za epochu a optimalizační metodu Adam. Z grafů je patrné, že u některých modelů by bylo dobré pokračovat v trénování, avšak jedná se pouze o ukázku.



Obr. 34: Průběh trénování pro některé architektury enkodéru

Ze všech natrénovaných modelů byly vybrány tři a bylo provedeno jejich sloučení (*ensemble*). Výběr natrénovaných modelů probíhal v závislosti na hodnotě použité metriky. Nevýhodou sloučení více modelů je delší doba pro predikci a potřeba více paměti, které tyto modely zaujímají, což může být problém například u mobilních aplikací. Kombinací modelů však vznikne mnohem robustnější systém, který často umožňuje přesnější predikci a lze tak dosáhnout lepších výsledků. Průběh trénování finálních modelů je zobrazen na obrázku 35. Nastavení vybraných modelů popisuje tabulka 1. Nejvíce se pro řešený problém osvědčila architektura enkodéru SE-ResNet34, proto byla použita ve všech modelech. Obecně se však pro kombinaci modelů doporučuje používat různorodé nastavení. Ztrátové funkce všech modelů byly optimalizovány metodou Adam. V příloze jsou přiloženy obrázky architektur modelu 1 (**Příloha 3**) a modelu 3 (**Příloha 4**).

Tab. 1: Nastavení vybraných modelů

	Obecná architektura modelu	Architektura enkodéru	Ztrátová funkce
Model 1	Unet	SE-ResNet34	Binary crossentropy
Model 2	Unet	SE-ResNet34	JacardLoss
Model 3	LinkNet	SE-ResNet34	Binary crossentropy

Obr. 35: Průběh trénování modelů určených pro sloučení

Zjednodušený princip predikce a následné kombinace modelů zobrazuje obrázek 36. Pro kombinaci modelů je využita funkce `np.tensordot()` z knihovny NumPy, která jednotlivé modely kombinuje pomocí zvolených vah. Pro řešený problém byly váhy jednotlivých modelů zvoleny rovnoměrně. Výsledná predikce tedy závisí na každém modelu stejně. Jelikož trénování modelu probíhalo na dávkách obsahujících několik obrázků, tak je pro predikci potřeba rozšířit vstupní obrázek o jeden rozměr. K tomu je určena funkce `np.expand_dims()`, jejíž argumenty jsou vstupní obraz a osa, podle které dojde k rozšíření. Pro zobrazení predikovaného obrazu je potřeba pro každý pixel určit hodnotu třídy, k čemuž je využita funkce `np.argmax()`, jejíž argumenty jsou vstupní obraz a osa, podle které funkce vrátí indexy maximálních hodnot pole. Soubor pro trénování modelů v rozhraní Google Colaboratory určených pro segmentaci je přiložen v příloze (**Příloha 5**).

```
img_expand = np.expand_dims(detected_img, axis = 0)
pred1 = model1.predict(img_expand)[0]
pred2 = model2.predict(img_expand)[0]
pred3 = model3.predict(img_expand)[0]
preds=np.array([pred1, pred2, pred3])
weights = [1,1,1]
weighted_preds = np.tensordot(preds, weights, axes=((0),(0)))
ensemble_pred = np.argmax(weighted_preds, axis=-1)
```

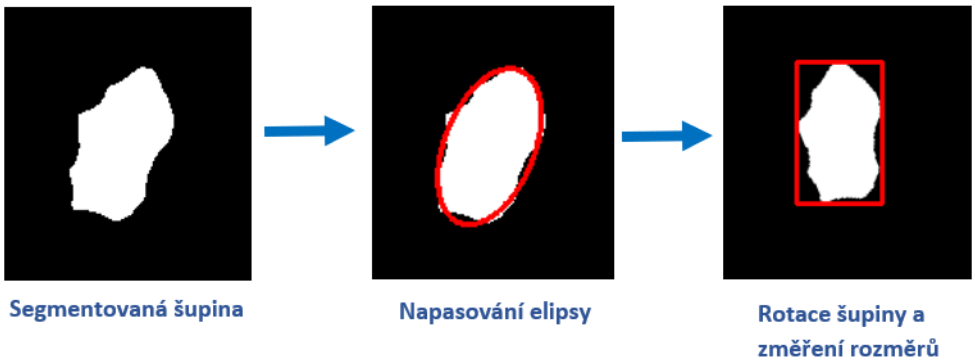
Obr. 36: Predikce a kombinace modelů

6.5 Změření análního štítku

Pro změření rozměrů segmentovaného análního štítku je potřeba v obrázku nejprve provést následné zpracování (*postprocessing*). Po segmentaci šupiny je v obrázku potřeba nalézt největší konturu, u které je předpoklad, že se jedná o hledanou šupinu. Pro nalezení kontury je potřeba obrázek nejprve převést na stupně šedi, toho je docíleno funkcí `cv2.cvtColor()` s typem převodu `cv2.COLOR_RGB2GRAY`. Na takto získaný

snímek je aplikován Cannyho hranový detektor, který je popsán v kapitole 3.2. K tomu je určena funkce `cv2.Canny()`, jejíž argumenty jsou vstupní obraz a hranice pro hysterezní prahování. Dolní hranice byla stanovena na hodnotu 50 a horní hranice na hodnotu 100. Z prahovaného snímku již lze snadno funkcí `cv2.findContours()` nalézt jednotlivé kontury. Režim vyhledávání byl určen `cv2.RETR_TREE` a metoda pro aproximaci kontur byla zvolena `cv2.CHAIN_APPROX_SIMPLE`. Ze všech nalezených kontur je ponechána pouze největší kontura, protože se předpokládá, že se jedná o konturu análního štítku. Všechny ostatní kontury jsou z obrázku vymazány. Takto upravený obrázek je připraven pro změření rozměrů šupiny.

Pro správné změření rozměrů segmentovaného análního štítku je potřeba natočit ji kolmo na osy. Na nalezenou konturu štítku je pomocí příkazu `cv2.fitEllipse()` napasována elipsa, jejíž tvar nejlépe odpovídá nalezené kontuře. Pomocí získané elipsy je nalezen úhel, o který je potřeba konturu štítku otočit. Matice pro natočení obrázku je získána funkcí `cv2.getRotationMatrix2D()`, jejíž argumenty jsou střed rotace, úhel rotace a faktor měřítka pro škálování, který byl ponechán na hodnotě jedna. Rotace je provedena funkcí `cv2.warpAffine()`, jejíž argumenty jsou vstupní obraz, získaná matice natočení a rozměry výstupního obrázku, které byly nastaveny na hodnotu vstupního obrázku. Správně natočené kontuře štítku je pak pomocí funkce `cv2.boundingRect()` opsán obdélník, jehož šířka a výška odpovídá hledaným rozměrům análního štítku. Zjednodušený princip jednotlivých kroků zobrazuje obrázek 37.



Obr. 37: Princip měření šupiny

6.6 Grafické uživatelské rozhraní

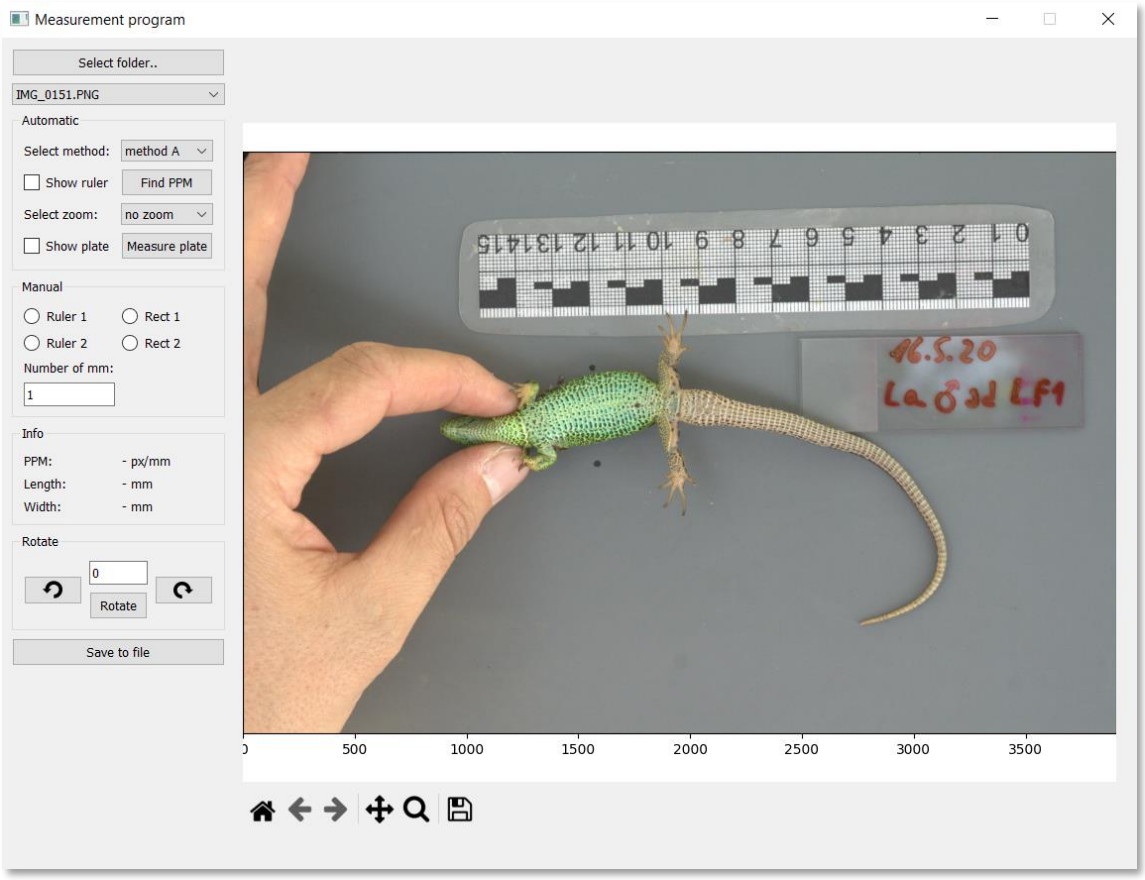
K naprogramovaným algoritmům bylo pomocí knihovny PyQt vytvořeno i jednoduché grafické uživatelské rozhraní zobrazené na obrázku 38, které umožňuje automatické i ruční měření rozměrů análního štítku. Grafická část s ovládacími prvky byla vytvořena v programu QtDesigner a lze ji nalézt v příloze (**Příloha 6**). V příloze je také obsažen soubor s ovládacími prvky převedenými do scriptu v programovacím jazyku Python. (**Příloha 7**).

Po spuštění aplikace se uživateli otevře hlavní rozhraní programu a konzolové okno pro výpis vykonaných akcí. Po stisku na tlačítko „Select folder..“ se otevře průzkumník Windows, ve kterém si uživatel vybere složku s obrázky. Tyto obrázky se uživateli uloží do kombinovaného pole (*combo box*), ze kterého si může jednotlivé obrázky snadno zobrazit a pracovat s nimi.

Pro ruční měření slouží dolní část rozhraní „Manual“. Do textového pole (*text box*) „Number of mm“ uživatel zadá počet milimetrů, které odpovídají reálné vzdálenosti mezi body vloženými do obrázku. Tyto body lze snadno vložit zvolením přepínačů „Ruler1“ a „Ruler2“ a kliknutím do obrázku. Aby byl anální štítek natočen kolmo na osy, lze celý obrázek otáčet pomocí nástrojů označených jako „Rotate“. Tlačítka s šipkami umožňují natočit obrázek o 1° doleva nebo doprava a po kliku na tlačítko „Rotate“ se obrázek otočí o úhel uvedený v textovém poli nad tímto tlačítkem. Pod obrázkem se nachází nástroje pro snadnější manipulaci, které umožňují vrátit obrázek do původního zobrazení, vracet se mezi jednotlivým zobrazením, posun obrázku do stran, přiblížení (*zoom*) vybrané části obrázku a případně i uložení aktuálně zobrazeného obrázku.

Pro automatické měření slouží prvky označené jako „Automatic“. Uživatel si z kombinovaného pole „Select method“ zvolí, jestli chce určit měřítko pomocí metody A, nebo metody B a po kliku na tlačítko „Find PPM“ se určí měřítko obrázku. Pokud je před nalezením měřítka aktivováno zaškrtávací pole (*checkbox*) „Show ruler“, zobrazí se uživateli pro kontrolu dané pravítko s nalezenými konturami, podle kterých se měřítko zjišťuje. Následně si uživatel z kombinovaného boxu „Select zoom“ vybere, jestli se na obrázku nachází ještěrka ve velkém nebo malém přiblížení. Pro trénování modelů určených k segmentaci štítku obsahovala většina datasetu ještěrky ve velkém přiblížení. Z toho důvodu je pro přesnější segmentaci análního štítku u ještěrek s malým přiblížením potřeba detekovaný štítek více přiblížit. Poté již stačí kliknou na tlačítko „Measure plate“, které umožní změření segmentovaného štítku. Pokud je před změřením štítku aktivováno zaškrtávací pole „Show plate“, zobrazí se uživateli pro kontrolu detekovaný štítek s konturou segmentované části, podle které se měřítko zjišťuje. Veškeré relevantní údaje jsou v průběhu používání aplikace průběžně uživateli vypisovány do konzolového okna. V části aplikace „Info“ jsou navíc uživateli vypisovány aktuální údaje o hodnotě měřítka a hodnotách rozměrů análního štítku. Pro uložení naměřených hodnot slouží tlačítko „Save to file“, které uloží získaná data do souboru ve formátu CSV (*comma separated values*) vytvořeného ve stejné složce, ve které se nachází daná aplikace.

Výhodou této aplikace je její modulárnost, kdy si uživatel automaticky zjistí meřítko a rozměry análního štítku, které si nechá zobrazit a v případě nespokojenosti s některou z naměřených hodnot, může tuto hodnotu sám jednoduše přeměřit a změnit. Celá aplikace je ve spustitelném EXE (*executable*) formátu. V příloze je přiložen soubor obsahující nastavení plátna pro zobrazování obrázků (**Příloha 8**) a soubor obsahující všechny algoritmy potřebné pro automatické a ruční měření (**Příloha 9**).



Obr. 38: Grafické uživatelské rozhraní

6.7 Vyhodnocení výsledků implementovaného řešení

Pro vyhodnocení výsledků implementovaného řešení byly Ing. Radovanem Smolinským, Ph.D. et PhD. ručně změřeny rozměry análního štítku na 49 obrázcích. Tyto obrázky sloužily pouze pro testovací účely a vyhodnocení výsledků. Ruční měření bylo provedeno pomocí vytvořeného grafického rozhraní popsaného v předchozí kapitole. Následně bylo na testovacím datasetu provedeno změření rozměrů análních štítků pomocí vytvořeného systému automatického měření s následujícími výsledky. Pravítko bylo správně detekováno na všech 49 obrázcích. Análí štítek byl správně detekován na 47 obrázcích.

Z 49 detekovaných pravítek bylo metodou A zjištěno měřítko u 36 obrázků, neboť některé obrázky disponují pouze částmi pravítek bez obdélníků potřebných pro tuto metodu. Metodou B bylo zjištěno měřítko u 42 obrázků, protože tři obrázky obsahovaly pravítka v příliš nízké kvalitě obrazu a čtyři obrázky obsahovaly nerozpoznatelná pravítka, pro které není systém vytvořen. Ze získaných údajů byly pomocí rovnice:

$$\Delta = |x_0 - x| \tag{6.1}$$

spočítány absolutní chyby Δ, kde x_0 je referenční hodnota a x je automaticky naměřená hodnota vytvořeným systémem. Referenčními hodnotami jsou myšleny ručně změřené rozměry odborníkem. Relativní chyba δ je dána rovnicí:

$$\delta = \frac{\Delta}{|x_0|} \cdot 100 \tag{6.2}$$

Vyhodnocení metod zjišťujících měřítka obrázku je uvedeno v tabulce 2. Výsledky jsou uvedeny pro 36 obrázků, ze kterých bylo možné zjistit měřítko oběma metodami. V tabulce jsou pro jednotlivé metody uvedeny průměrné absolutní chyby a průměrné relativní chyby.

Tab. 2: Vyhodnocení metod zjišťujících měřítko obrázku

	Průměrná absolutní chyba [px/mm]	**Průměrná relativní chyba [%]**
Metoda A	0,442	3,4
Metoda B	0,381	2,3

V tabulce 3 je uvedeno vyhodnocení segmentace análních štítků bez využití měřítka obrázku. Tabulka pro nalezené rozměry šířek a délek 47 segmentovaných šupin uvádí průměrnou absolutní chybu a průměrnou relativní chybu.

Tab. 3: Vyhodnocení segmentace análních štítků

	Průměrná absolutní chyba [px]	**Průměrná relativní chyba [%]**
Šířka	3,376	4,8
Délka	2,403	5,1

V tabulce 4 jsou uvedeny průměrné hodnoty chyb změřených análních štítků s využitím nalezených měřítek obrázků. Pro výpočet byla využita meřítka zjištěná pomocí metody B, protože se jedná o robustnější metodu umožňující nalezení měřítka na více obrázcích. V tabulce jsou uvedeny výsledky pro celkem 40 změřených šupin. Tabulka pro změřené rozměry šířek a délek šupin uvádí průměrnou absolutní chybu, průměrnou relativní chybu a směrodatnou odchylku chyb σ vypočítanou podle rovnice:

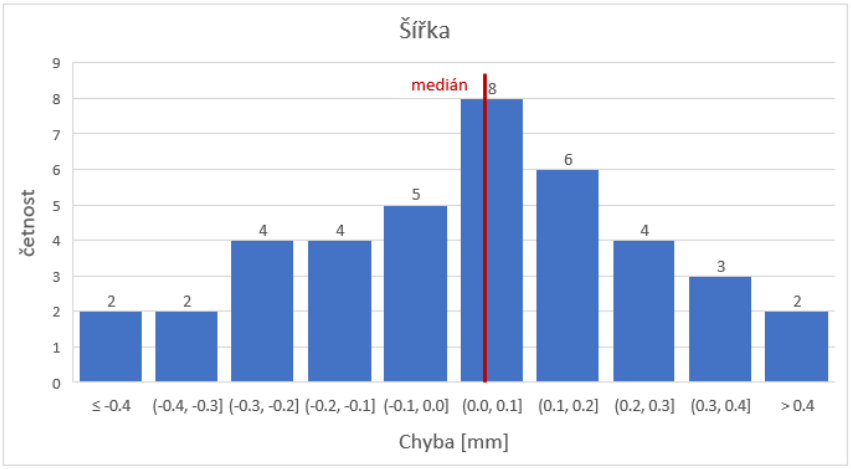
$$\sigma = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{6.3}$$

kde N udává celkový počet změřených šupin, x_i udává jednotlivé chyby spočítané ze vztahu $x_0 - x$ a $\bar{x}$ udává průměr chyb x_i.

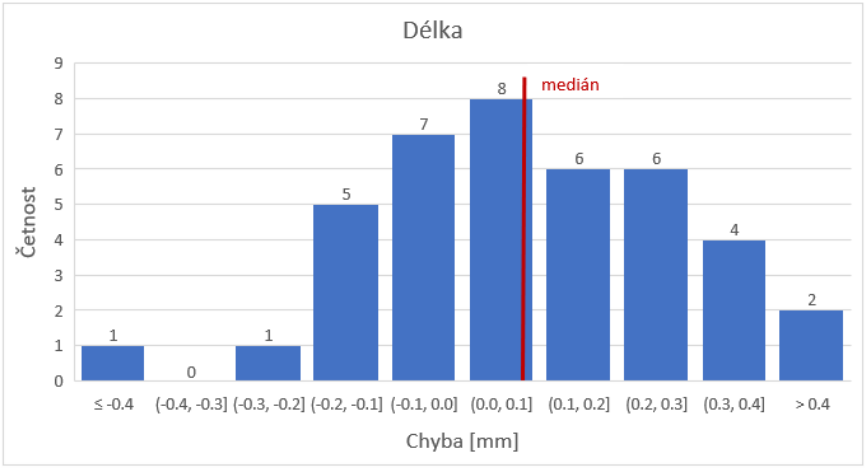
Tab. 4: Vyhodnocení změřených rozměrů

	Průměrná absolutní chyba [mm]	**Průměrná relativní chyba [%]**	**Směrodatná odchylka [mm]**
Šířka	0,261	5,6	0,29
Délka	0,188	6,2	0,23

Pro hodnoty chyb změřených rozměrů análních štítků byl vykreslen histogram neboli graf četností a spočítán medián neboli střední hodnota. Pro chyby šířek medián činí přibližně 0,041 mm a pro chyby délek přibližně 0,089 mm. Obrázek 39 zobrazuje graf četnosti chyb pro šířku a medián těchto chyb. Obrázek 40 zobrazuje graf četnosti chyb pro délku a medián těchto chyb.



Obr. 39: Histogram chyb šířek



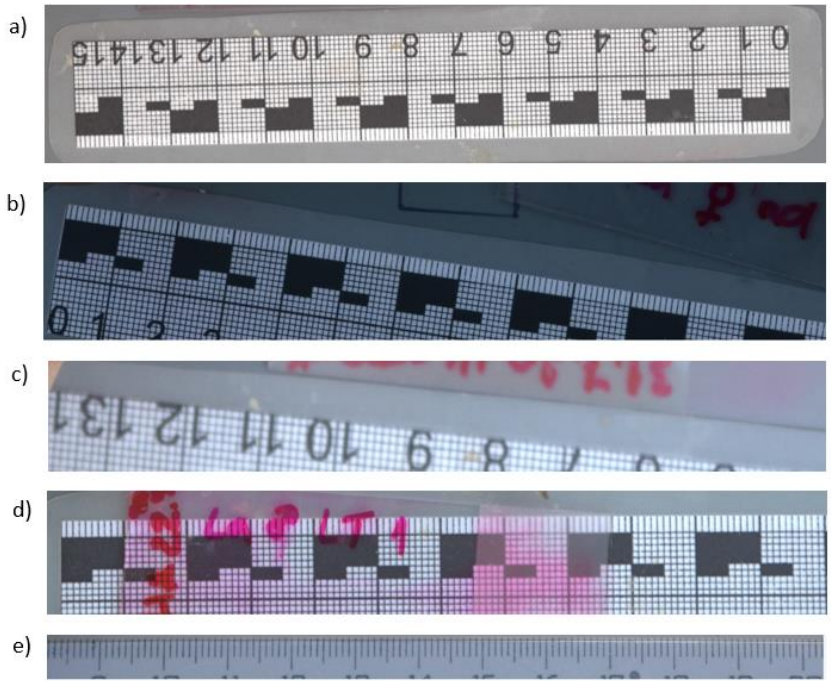
Obr. 40: Histogram chyb délek

7 ZHODNOCENÍ A DISKUZE

Systém pro měření rozměrů análního štítku ještěrky obecné se skládá ze čtyř různých kroků. Nejprve je potřeba z obrázku detekovat pravítko a anální štítek ještěrky. Na detekované pravítko jsou aplikovány algoritmy počítačového zpracování obrazu, pomocí kterých je zjištěno měřítko obrázku. Detekovaný anální štítek je pomocí natrénovaných modelů konvoluční neuronové sítě segmentována do binární podoby. Takto získaná šupina je následně upravena do vhodné podoby a pomocí předem získaného měřítka jsou změřeny její rozměry. Každý ze zmíněných kroků, které jsou pro změření šupiny pomocí vytvořeného systému potřebné, je zatížen určitou chybou.

Jak vyplývá z výsledků popsaných v předchozí kapitole, modely pro detekci pravítka a análního štítku jsou natrénovaný dostatečně, neboť úspěšnost detekce pravítka na testovacím datasetu dosáhla 100 % a úspěšnost detekce análního štítku dosáhla na testovacím datasetu přibližně 96 %. V případě potřeby přesnější detekce nebo potřeby detekovat i odlišná pravítka, tak stačí rozšířit trénovací dataset a příslušný model dotrénovat i na nová data.

Pro nalezení měřítek obrázku byly vytvořeny dvě metody počítačového zpracování obrazu. Výsledky těchto metod jsou zobrazeny v tabulce 1, ze které vyplývá, že metoda B disponuje nižší absolutní i relativní chybou. Tato metoda je i robustnější, protože pro zjištění měřítka obrázku nepotřebuje, aby se na obrázku nacházelo pravítko s velkými černými obdélníky, které jsou pro metodu A nezbytné. Vytvořené metody pro zjištění měřítka obrázku bylo potřeba dělat dostatečně robustní, protože fotografie se nachází v různých formátech a nejednotných obrazových stylech, jejichž příklad zobrazuje obrázek 41. Na obrázku 41 a) je zobrazeno pravítko, pomocí kterého dokáže vytvořený systém získat měřítko metodou A i B. Obrázek 41 b) až d) zobrazuje příklad pravítek, na kterých dokáže vytvořený systém zjistit měřítko, ale s větší hodnotou chyby, protože tyto pravítka obsahují velké množství rušivých elementů, mezi které patří například různé natočení pravítka, různý jas obrázků, překrývající se kontury, nedostatečné zaostření pravítka či špatné osvětlení. Obrázek 41 e) zobrazuje pravítko, pro které není vytvořený systém přizpůsoben a nelze tak pomocí něj získat měřítko obrázku. Pro zajištění větší přesnosti by bylo potřeba zajistit jednotný obrazový styl bez různých rušivých elementů, které se na některých fotografiích nachází. Pro jednodušší a přesnější nalezení měřítka by bylo vhodné využít i různé referenční objekty, které mají fixní rozměry a je snadné je detekovat.



Obr. 41: Příklad nejednotných obrazových stylů s rušivými elementy

Pro segmentaci análního štítku byly natrénovány tři modely konvolučních neuronových sítí, jejichž predikce jsou zkombinovány a vytvořen jeden segmentovaný obraz. Výsledky segmentací análního štítku z testovacího datasetu obsahuje tabulka 3, ze které je patrné, že relativní chyba segmentace je v obou hledaných rozměrech přibližně stejná a pohybuje se kolem 5 % z hledaných rozměrů. Průměrná absolutní chyba je mírně vyšší u rozměrů šířek, protože hodnota šířky šupiny je obvykle větší než hodnota její délky. Pro přesnější natrénování modelů konvoluční neuronové sítě by bylo potřeba rozšířit trénovací dataset o více binárních masek. Tento krok by pomohl pro přesnější segmentaci análních štítků, což by umožnilo přesnější změření jejich rozměrů. Ruční tvorba těchto masek je však velmi pracná a časově náročná. Pro natrénování modelů umožňující přesnější predikci by bylo vhodné využít i robustnější architektury konvolučních neuronových sítí, které z důvodu nedostatečných výpočetních kapacit nebylo možné použít.

Vyhodnocení změřených rozměrů popisuje tabulka 4, ze které vyplývá, že automaticky změřené rozměry vytvořeným systémem se od změřených rozměrů odborníkem liší přibližně o 6 %. Větší průměrná absolutní chyba i směrodatná odchylka u rozměrů šířky je způsobena tím, že rozměr šířky šupiny je obvykle větší než rozměr její délky, jak je popsáno i v předchozím odstavci. Graf 1 zobrazuje četnost chyb pro rozměr šířky a graf 2 zobrazuje četnost chyb pro rozměr délky. Oba grafy podle očekávání přibližně odpovídají normálnímu (Gaussovu) rozdělení. Medián chyb rozměrů šířky je přibližně 0,04 mm a rozměrů délky přibližně 0,09 mm. Obě tyto hodnoty jsou kladné, což značí, že automaticky změřené rozměry jsou obvykle nižší než manuálně změřené rozměry odborníkem, čemuž odpovídají i zobrazené grafy četností. Pro přesnější a průkaznější výsledky by však bylo potřeba více testovacích snímků.

8 ZÁVĚR

Cílem zadaného problému bylo vytvořit systém pro měření šířky a délky análního štítku ještěrky obecné. Z těchto údajů lze vypočítat tzv. index análního štítku, který udává míru protáhlosti této šupiny. Tento index slouží vědcům z Ústavu biologie obratlovců Akademie věd ČR, v. v. i. a Pedagogické fakulty Masarykovy univerzity pro ověření hypotézy, jestli se v průběhu života ještěrky obecné mění pouze rozměry jejích šupin, nebo i tvar.

V rámci práce byla vypracována rešerše mapující techniky využívané k měření rozměrů objektů a rešerše mapující techniky využívané k segmentaci obrazu. Následně jsou v práci popsány neuronové sítě a konvoluční neuronové sítě, které dávají stručný přehled o metodách použitých v implementaci vytvořeného systému pro automatické měření rozměrů análního štítku. Systém je implementován v grafickém uživatelském rozhraní, které uživateli umožňuje manuální i automatické měření. Systém automatického měření se skládá ze čtyř základních kroků – detekce análního štítku a pravítka, nalezení měřítka obrázku, segmentace análního štítku a změření jeho rozměrů.

Nejprve bylo pro natrénování modelů určených pro detekci pravítka a análního štítku anotováno 200 obrázků. Pro trénování modelů byla vybrána architektura YOLOv3 a trénování probíhalo v rozhraní Google Colaboratory pomocí frameworku Darknet. Na detekované pravítko jsou aplikovány algoritmy počítačového zpracování obrazu, pomocí kterých je zjištěno měřítko obrázku, tedy počet pixelů, který odpovídá reálné vzdálenosti jednoho milimetru. Pro natrénování modelů určených pro segmentaci análního štítku bylo vytvořeno 200 binárních masek. Pro segmentaci detekovaného análního štítku byly natrénovány tři modely konvolučních neuronových sítí, které se podílejí na výsledné segmentaci šupiny do binární podoby. Takto získaná šupina je následně upravena do vhodné podoby a pomocí předem získaného měřítka jsou změřeny její rozměry.

Pro automatické měření nebyla stanovena žádná požadovaná hodnota přesnosti. Pro zvýšení přesnosti by bylo potřeba zajistit snímky ve vyšším rozlišení, což by umožnilo získat větší detaily análního štítku. Dále by bylo potřeba zajistit jednotný obrazový styl bez rušivých vlivů, čímž by se výrazně zvýšila přesnost nalezeného měřítka. Větší přesnost měření by zajistila i jednotná ohnisková vzdálenost a jednotná vzdálenost objektivu od snímaného objektu.

9 SEZNAM POUŽITÉ LITERATURY

[1] PALCI, Alessandro. Ancient fossil fills a 75 million-year gap and rewrites lizard and snake history. In: *Theconversation.com* [online]. 31 May 2018 [cit. 2022-02-14]. Dostupné z: https://theconversation.com/ancient-fossil-fills-a-75-million-year-gap-and-rewrites-lizard-and-snake-history-97455

[2] MORAVEC, Jiří a Michal BEREC. *Fauna* ČR. Praha: Academia, 2015. ISBN 978-80-200-2416-9.

[3] GUARINO, Fabio Maria, Ivan Di GIÀ a Roberto SINDACO. Age and growth of the sand lizards (Lacerta agilis) from a high Alpine population of north-western Italy. *Acta Herpetologica* [online]. June 2010, 5(1), 23-29 [cit. 2022-05-19]. Dostupné z: https://www.researchgate.net/publication/207038728_Age_and_growth_of_the_sand_lizards_Lacerta_agilis_from_a_high_Alpine_population_of_north-western_Italy

[4] BORCZYK, Bartosz, Jan KUSZNIERZ, Łukasz PAŚKO a Edyta TURNIAK. Scaling of the sexual size and shape skull dimorphism in the sand lizard (Lacerta agilis L.). Vertebrate Zoology [online]. July 2014, 64(2), 221-227 [cit. 2022-05-19]. Dostupné z: https://www.researchgate.net/publication/265208072_Scaling_of_the_sexual_size_and_shape_skull_dimorphism_in_the_sand_lizard_Lacerta_agilis_L

[5] MICHOVSKÝ TOOLS: *Měřidla a příslušenství Michovský-TOOLS s.r.o* [online]. c2015 [cit. 2022-05-03]. Dostupné z: https://www.meridla.cz/katalog-meridel/

[6] ČEPOVÁ, Lenka a Lenka PETŘKOVSKÁ. Legislativa ve strojírenské metrologii a přesné měření 3D ploch: studijní opora [online]. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2011 [cit. 2022-02-15]. ISBN 978-80-248-2514-4.

[7] VOJÁČEK, Antonín. Automatické bezkontaktní mikrometrové měření předmětů. In: *Automatizace.hw.cz* [online]. 5. únor 2016 [cit. 2022-02-15]. Dostupné z: https://automatizace.hw.cz/komponenty-mereni-a-regulace/automaticke-bezkontaktni-mikrometrove-mereni-predmetu.html

[8] VACHARANUKUL, K. a S. MEKID. In-process dimensional inspection sensors. *Measurement* [online]. The University of Manchester, School of Mechanical, Aerospace and Civil Engineering, M60 1QD Manchester, UK, 2005, **38**(3), 204-218 [cit. 2022-02-15]. ISSN 02632241. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0263224105000795

[9] Ultrazvukové senzory: Bezkontaktní měření vzdálenosti. *Pepperl-fuchs.com* [online]. [cit. 2022-02-15]. Dostupné z: https://www.pepperl-fuchs.com/czech_republic/cs/classid_182.htm

[10] ZELINKA, Ondřej. Bezkontaktní měření rozměrů – optické mikrometry. *Automa: časopis pro automatizační techniku* [online]. 2009, **15**(4) [cit. 2022-03-15]. ISSN 1210-9592. Dostupné z: https://automa.cz/cz/casopis-clanky/bezkontaktni-mereni-rozmeru-opticke-mikrometry-2009_04_38860_5757/

[11] Úvod do techniky CCD čipů. Moravské přístroje: Kamery pro astronomii [online]. 2011 [cit. 2022-02-16]. Dostupné z: https://www.gxccd.com/art?id=303&lang=405

[12] Snímače CMOS. Hardware pro multimédia: *wiki* [online]. 2011 [cit. 2022-02-16]. Dostupné z: http://noel.feld.cvut.cz/vyu/a2b31hpm/index.php/U%C5%BEivatel:Aubreja3

[13] ŠKRABÁNEK, Pavel. Strojové vidění: vznik digitálního obrazu [přednáška]. Brno: VUT v Brně, 2021

[14] ŠKRABÁNEK, Pavel. Strojové vidění: základní vztahy mezi pixely [přednáška]. Brno: VUT v Brně, 2021

[15] LEAGUE, Christopher. Cartesian distance metrics. In: *Liucs.net* [online]. 16 Sep 2018 [cit. 2022-02-17]. Dostupné z: https://liucs.net/cs168f18/cartesian-distance.png

[16] NIETHAMMER, M., W.D. KALIES, K. MISCHAIKOW a A. TANNENBAUM. On the detection of simple points in higher dimensions using cubical homology. *IEEE Transactions on Image Processing* [online]. 2006, 8 August 2006, **15**(8), 2462-2469 [cit. 2022-02-17]. ISSN 1057-7149.Dostupné z: http://ieeexplore.ieee.org/document/1658108/

[17] GONZALEZ, Rafael C. a Richard E. WOODS. Digital image processing. New York, NY: Pearson, [2018]. ISBN 978-0133356724.

[18] ŠPANĚL, Michal a Vítězslav BERAN. Obrazové segmentační techniky: *Přehled existujících metod.* In: *Fit.vutbr.cz* [online]. 2005 [cit. 2022-02-18]. Dostupné z: http://www.fit.vutbr.cz/~spanel/segmentace/

[19] Red-Blue Ratio Thresholding Visualization. In: *Washington University in St. Louis: Cloud Detection and Categorization* [online]. [cit. 2022-02-18]. Dostupné z: https://sites.wustl.edu/clouddetection/cloud-detection/fixed-and-adaptive-thresholding/

[20] ŠKRABÁNEK, Pavel. Strojové vidění: hledání hran [přednáška]. Brno: VUT v Brně, 2021

[21] STRAKA, Stanislav. *Segmentace obrazu* [online]. Brno, 2009 [cit. 2022-02-18]. Dostupné z: https://is.muni.cz/th/tzp80/dp.pdf. Diplomová práce. Masarykova univerzita. Vedoucí práce Radka Pospíšilová.

[22] EL-SAYED, Mohamed A. a Hamida A. M. SENNARI. Convolutional Neural Network for Edge Detection in SAR Grayscale Images. *IOSR journal of VLSI and Signal Processing* [online]. 2014, **4**(2), 75-83 [cit. 2022-02-19]. ISSN 23194197. Dostupné z: https://www.iosrjournals.org/iosr-jvlsi/papers/vol4-issue2/Version-1/L04217583.pdf

[23] KLÁSEK, P. *Segmentace základních částí lidského mozku v MR datech*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 50 s., 1 s. příloh. Vedoucí bakalářské práce Ing. Miloš Malínský.

[24] ZAHRÁDKA Jiří: *Obraz jako výšková mapa*. Brno, 2008, bakalářská práce, FIT VUT v Brně.

[25] AGRAWAL, Shubhang. Image Segmentation using K-Means Clustering. In: *Medium.com* [online]. 17 Jan 2021 [cit. 2022-02-19]. Dostupné z: https://medium.com/swlh/image-segmentation-using-k-means-clustering-46a60488ae71

[26] KALOVÁ, Ilona. Regionální segmentace a shlukování: počítačové vidění [přednáška]. Brno: VUT v Brně [cit. 2022-02-19]. Dostupné z: https://adoc.pub/regionalni-segmentace-a-shlukovani.html

[27] MUDROVÁ, Martina. Matematická morfologie a segmentace obrazu. [přednáška]. [cit. 2022-02-19]. Dostupné z: http://uprt.vscht.cz/mudrova/zob/prednasky/10-MORFOLOGIE/morfologie-tisk.pdf

[28] YAMADA, Yasuharu. FLOOD EXTENT DETECTION IN PADDY AREA AND FUTURE PLAN OF DISASTER INFORMATION SHARING PLATFORM IN RURAL AREAS. *Researchgate.net* [online]. January 2018 [cit. 2022-02-19]. Dostupné z: https://www.researchgate.net/publication/322666366_FLOOD_EXTENT_DETECTION_IN_PADDY_AREA_AND_FUTURE_PLAN_OF_DISASTER_INFORMATION_SHARING_PLATFORM_IN_RURAL_AREAS

[29] DURČÁK, Pavel. Neuronové sítě a princip jejich fungování. In: *Napocitaci.cz* [online]. 8. 9. 2017 [cit. 2022-02-20]. Dostupné z: https://www.napocitaci.cz/33/neuronove-site-a-princip-jejich-fungovani-uniqueidgOkE4NvrWuNY54vrLeM670eFNQh552VdDDulZX7UDBY/

[30] VOLNÁ, Eva. Neuronové sítě 1. Druhé vydání. Ostrava: Ostravská univerzita v Ostravě, 2008.

[31] SABER, Mohammed, Abdessamad El RHARRAS, Rachid SAADANE, Hatim Kharraz AROUSSI a Mohammed WAHBI. Artificial Neural Networks, Support Vector Machine and Energy Detection for Spectrum Sensing based on Real Signals. *Researchgate.net* [online]. April 2019, **11**(1), 52-60[cit. 2022-02-20].Dostupné z: https://www.researchgate.net/publication/332970306_Artificial_Neural_Networks_Support_Vector_Machine_and_Energy_Detection_for_Spectrum_Sensing_based_on_Real_Signals

[32] ŠTOL, Jan. *Strojové učení s využitím metody transfer learning* [online]. Hradec Králové, 2019 [cit. 2022-02-21]. Dostupné z: https://theses.cz/id/idr9ob/. Diplomová práce. Univerzita Hradec Králové, Fakulta informatiky a managementu. Vedoucí práce Ing. Karel Mls, Ph.D.

[33] HANDI, Narmadha. More on Gradient Descent Algorithm and other effective learning Algorithms…. In: *DataDrivenInvestor* [online]. 18 Apr 2019 [cit. 2022-02-21]. Dostupné z: https://medium.datadriveninvestor.com/more-on-gradient-descent-algorithm-and-other-effective-learning-algorithms-a1222a8d6c33

[34] CHANDRA, Akshay L. Learning Parameters, Part 5: AdaGrad, RMSProp, and Adam. In: *Towards Data Science* [online]. [cit. 2022-02-22]. Dostupné z: https://towardsdatascience.com/learning-parameters-part-5-65a2f3583f7d

[35] MISHRA, Aditya. Metrics to Evaluate your Machine Learning Algorithm. In: *Towards Data Science* [online]. 24 Feb 2018 [cit. 2022-02-22]. Dostupné z: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

[36] ALFARRAJ, Motaz a Ghassan ALREGIB. Petrophysical-property estimation from seismic data using recurrent neural networks. SEG Technical Program Expanded Abstracts 2018 [online]. *Society of Exploration Geophysicists*, 2018, 2018-08-27, 2141-2146 [cit. 2022-02-23]. Dostupné z: https://library.seg.org/doi/10.1190/segam2018-2995752.1

[37] DEEPMIND'S. Generative Adversarial Networks (GANs). In: *Medium.com* [online]. 11 Aug 2019 [cit. 2022-02-25].Dostupné z: https://medium.com/@saibharath897/generative-adversarial-networks-gans-560c5c988128

[38] Algoritmus zpětného šíření chyby: (BP algoritmus) [online]. Institut biostatistiky a analýz Lékařské fakulty Masarykovy univerzity [cit. 2022-02-25]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-perceptrony--vicevrstvy-perceptron--algoritmus-zpetneho-sireni-chyby-bp-algoritmus#nsp31

[39] REK, Petr. *Knihovna pro návrh konvolučních neuronových sítí*. Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Lukáš Sekanina, Ph.D.

[40] DAWAR, Harshit. Stochastic Gradient Descent. In: *Medium.com* [online]. 14 May 2020 [cit. 2022-02-26]. Dostupné z: https://medium.com/analytics-vidhya/stochastic-gradient-descent-1ab661fabf89

[41] BROWNLEE, Jason. How to use Learning Curves to Diagnose Machine Learning Model Performance. In: *Deep Learning Performance* [online]. 27 February 2019 [cit. 2022-02-26]. Dostupné z: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

[42] JUNG, Seok-Ki a Tae-Woo KIM. New approach for the diagnosis of extractions with neural network machine learning. *American Journal of Orthodontics and Dentofacial Orthopedics* [online]. 2016, January 2016, **149**(1), 127-133 [cit. 2022-02-27]. ISSN 08895406. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0889540615011683

[43] SAHA, Sumit. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. In: *Towards Data Science* [online]. 15 Dec 2018 [cit. 2022-03-04]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[44] KUMAR ACHARYA, Anuja a Rajalakshmi SATAPATHY. A Deep Learning Based Approach towards the Automatic Diagnosis of Pneumonia from Chest Radio-Graphs. *Biomedical and Pharmacology Journal* [online]. 2020, 25 March 2020, **13**(1), 449-455 [cit. 2022-03-05]. ISSN 09746242. Dostupné z: http://biomedpharmajournal.org/vol13no1/a-deep-learning-based-approach-towards-the-automatic-diagnosis-of-pneumonia-from-chest-radio-graphs/

[45] YANI, Muhamad, S, Si., M.T. BUDHI IRAWAN a S.T., M.T. CASI SETININGSIH. Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. *Journal of Physics: Conference Series* [online]. 2019, 2019, **1201**(1) [cit. 2022-03-05]. ISSN 1742-6588. Dostupné z: https://iopscience.iop.org/article/10.1088/1742-6596/1201/1/012052

[46] PAZDERKA, Radek. *Segmentace obrazových dat pomocí hlubokých neuronových sít*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

[47] RONNEBERGER, Olaf, Philipp FISCHER a Thomas BROX. *U-Net: Convolutional Networks for Biomedical Image Segmentation* [online]. 18 May 2015 [cit. 2022-03-06]. Dostupné z: https://arxiv.org/pdf/1505.04597.pdf

[48] MWITI, Derrick a Katherine (Yi) LI. Image Segmentation in 2021: Architectures, Losses, Datasets, and Frameworks. In: *Neptune.ai* [online]. 21 December 2021 [cit. 2022-03-06]. Dostupné z: https://neptune.ai/blog/image-segmentation

[49] CHAURASIA, Abhishek a Eugenio CULURCIELLO. *LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation* [online]. 14 Jun 2017 [cit. 2022-04-28]. Dostupné z: https://arxiv.org/abs/1707.03718

[50] FENG, Vincent. An Overview of ResNet and its Variants. In: *Towards Data Science* [online]. 15 Jul 2017 [cit. 2022-03-05]. Dostupné z: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

[51] LATEEF, Fahad a Yassine RUICHEK. Survey on semantic segmentation using deep learning techniques. *Neurocomputing* [online]. 2019, **338**, 321-348 [cit. 2022-03-09]. ISSN 09252312. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S092523121930181X

[52] TAN, Mingxing a Quoc V. LE. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* [online]. 28 May 2019 [cit. 2022-03-09]. Dostupné z: https://arxiv.org/pdf/1905.11946.pdf

[53] REDMON, Joseph, Santosh DIVVALA, Ross GIRSHICK a Ali FARHADI. *You Only Look Once: Unified, Real-Time Object Detection* [online]. 8 Jun 2015 [cit. 2022-03-10]. Dostupné z: https://arxiv.org/pdf/1506.02640v5.pdf

[54] KATHURIA, Ayoosh. What's new in YOLO v3?. In*: Towards Data Science* [online]. 23 Apr 2018 [cit. 2022-03-10]. Dostupné z: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

[55] Google Collab nebo Google Colaboratory: co to je. *Hardware zdarma* [online]. [cit. 2022-04-10]. Dostupné z: https://www.hwlibre.com/cs/google-colaboratory/

10 SEZNAM OBRÁZKŮ A TABULEK

Seznam obrázků

Seznam tabulek

11 SEZNAM PŘÍLOH

Příloha 1: yolov3_training.cfg
Příloha 2: Detection.ipynb
Příloha 3: Model1.PNG
Příloha 4: Model3.PNG
Příloha 5: Segmentation.ipynb
Příloha 6: gui.ui
Příloha 5: gui_ui.py
Příloha 7: mplwidget.py
Příloha 8: Measure_program.py