SOUHRN

SUMMARY

PODĚKOVÁNÍ

OBSAH

1 ÚVOD

2 LITERÁRNÍ ČÁST

2.1 Detekce objektu pomocí umělé inteligence

Detekce objektu v obrazu je stejně jako sledování objektů, segmentace nebo klasifikace obrazu součásti interdisciplinárního oboru počítačového vidění jež se rozkládá na pomezí informatiky a umělé inteligence. Hlavním cílem detekce objektu je identifikace určitého objektu a zároveň určení jeho přesné polohy v obraze. Tato technologie má rozsáhlé využití v oborech autonomního řízení vozidel či rozpoznávání tváří, využívá se také v bezpečnostních systémech a mnoha dalších oblastech [1]. Další využití nacházejí metody detekce objektu v obrazu v lékařské diagnostice [1]. Tímto odvětvím, konkrétně detekcí hlasivek ve snímcích z laryngoskopických vyšetření se zabývá i tato práce.

Pro nalezení a identifikaci objektů lze použit dvě základní metody. Původní konvenční přístup pracuje ve třech krocích. V první fázi je vybrán region výskytu objektu algoritmem posuvného okna, následně dochází k extrakci vlastností určeného místa a klasifikaci konkrétního objektu. Tento přístup naráží na velkou výpočetní náročnost a nízkou přizpůsobivost [2], proto od něj bylo v posledních letech upuštěno na úkor metod hlubokého učení.

2.1.1 Detekce objektu s využitím hlubokého učení

Metody pro detekci objektů pomocí hlubokého učení využívají technologie konvolučních neuronových sítí (CNN) k rozeznání jednotlivých vlastností obrazu a následné detekci všech v něm nacházejících se objektů. Tento přístup je charakteristický rozdělením na tři hlavní sekce algoritmu [2].

- Extrakce vlastností (feature extraction), kdy je vstupní obraz zpracován pomocí CNN, která detekuje klíčové rysy obrazu jako jsou hrany, tvary či složitější textury a sestaví mapu vlastností obrazu.
- Lokalizace objektů (object localization) z mapy vlastností určí místo pravděpodobných výskytů objektů.
- Klasifikace přidá lokalizovanému objektu třídu, která udává, o jaký typ objektu se jedná.

Metody využívající neuronové sítě můžeme dále rozdělit podle typu algoritmu do dvou základních skupin na two-stage a one-stage detektory.

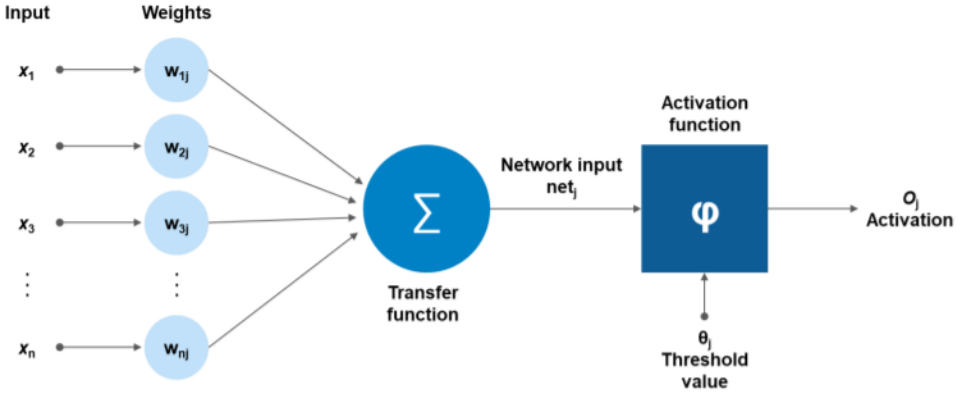
2.2 Umělé neuronové sítě

Princip umělých neuronových sítí (ANN) je inspirován funkcí neuronového systému v mozku člověka. Základní jednotkou algoritmu ANN je stejně jako v lidském mozku neuron, který v případě umělé sítě provádí matematické operace (viz. **Obr. 2.1**). Vstupem do neuronu je vektor $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$, na který je aplikován skalární součin s maticí vah $\boldsymbol{w} = (w_{11}, w_{21}, \dots, w_{n1})$, následně je přičten bias b_1. Hodnota z_1 je vypočítána podle rovnice (2.1) [3].

$$z_1 = \sum(\boldsymbol{x} \cdot \boldsymbol{w}^T) + b_1 \tag{2.1}$$

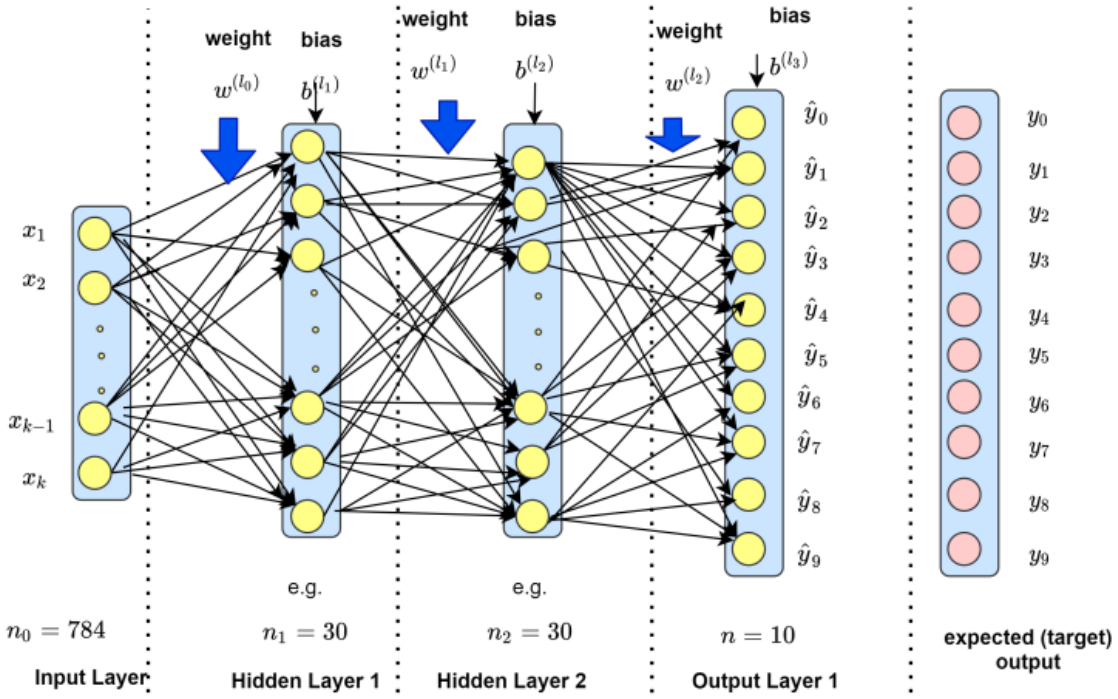
Pomocí aktivační funkce je vypočten konečný výstup (2.2), který následně může být použitý jako vstup do dalších umělých neuronů [3].

$$y_1 = f(z_1) \tag{2.2}$$



Obr. 2.1: Schéma neuronu ANN (převzato z [4])

Trénink ANN se skládá ze tří částí. První částí je dopředný průchod sítí (feedforward), skládající se z vrstev, kde každá vrstva obsahuje určitý počet neuronů. V případě plně propojených vrstev je každý neuron dané vrstvy svázán se všemi neurony z vrstvy předchozí. Na začátku tréninku jsou náhodně či pomocí specifických metod inicializovány biasy a vstupní váhy neuronů. Pro ukázku je na **Obr. 2.2** zobrazen průchod plně propojenou ANN se dvěma skrytými vrstvami, s počtem neuronů $n = 30$ a jednou výstupní vrstvou s $n = 10$. Pro obraz velikosti 28 × 28 je vytvořen vstupní vektor 784 × 1, vstupní vrstva má tudíž 784 hodnot. Ty vstupují do vrstvy, kde je rovnicí (2.1) vypočítána hodnota z_n pro každý neuron a následně se pomocí aktivační funkce (2.2) získá výstupní hodnota y_n. Každá tato hodnota je použita jako vstup do všech neuronů následující vrstvy (**Obr. 2.2**) [3].



Obr. 2.2: Schéma umělé neuronové sítě (převzato a upraveno z [3])

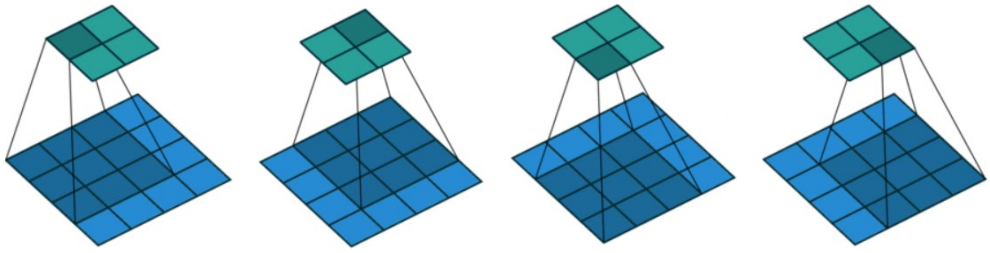
Ve druhém kroku zvaném zpětná propagace (backpropagation) je použita ztrátová funkce, která vypočítá chybu výstupní vrstvy oproti předpokládanému správnému výstupu. Následně je zpětně dopočítáno, jak se chyba šíří při průchodu neuronovou sítí a pro každý bias a váhu neuronů všech vrstev je navržena změna [3].

V posledním kroku jsou aktualizovány biasy a váhy neuronů tak, aby hodnota ztrátové funkce byla co nejnižší, tím se dosáhne nejoptimálnější funkce modelu [3].

2.3 Konvoluční neuronové sítě

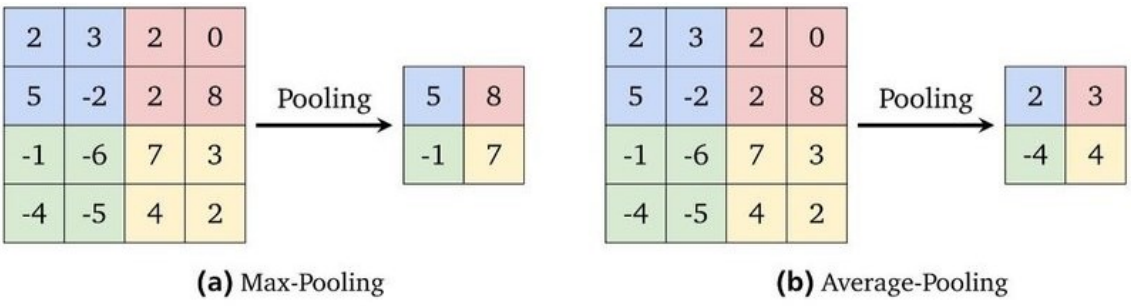
Nedostatkem ANN je její plochý vstup ve formě vektoru, kvůli němuž neuronová síť není schopna zohlednit prostorové upořádání vstupního obrazu. Problémem ANN algoritmu je i její vysoká propojenost, kvůli které při použití vstupů s větším rozlišením vznikají obrovské matice s hodnotami vah. To řeší přístup konvoluční neuronové sítě (CNN), který vstupy ve formě vektorů převádí do dvou či více dimenzí nazývaných mapy vlastností. Na rozdíl od předchozího neurony v CNN nejsou plně propojeny, ale mají vazbu pouze s několika prostorově blízkými hodnotami v předchozí vrstvě. Stejně jako v ANN i zde jsou vstupní hodnoty skalárně násobeny maticemi hodnot, které se v tomto případě nazývají filtry. Ty mají čtvercový tvar s obvyklými velikostmi stran mezi 1 a 11 [3].

Při tvorbě hodnot následující vrstvy se použije mapa vlastností předchozí vrstvy, na níž je použit posouvající se filtr. Každá hodnota následující vrstvy (tyrkysová na **Obr. 2.3**) je vytvořena konvolucí několika hodnot mapy vlastností z přechozí vrstvy (modrá na **Obr. 2.3**), které jsou váženy a upravovány pomocí zvolených vah a biasů filtru, nastavených v průběhu tréninku modelu. Takových filtrů může být použito více na jednu vrstvu, čímž lze dosáhnout většího počtu map vlastností v jedné vrstvě [3].



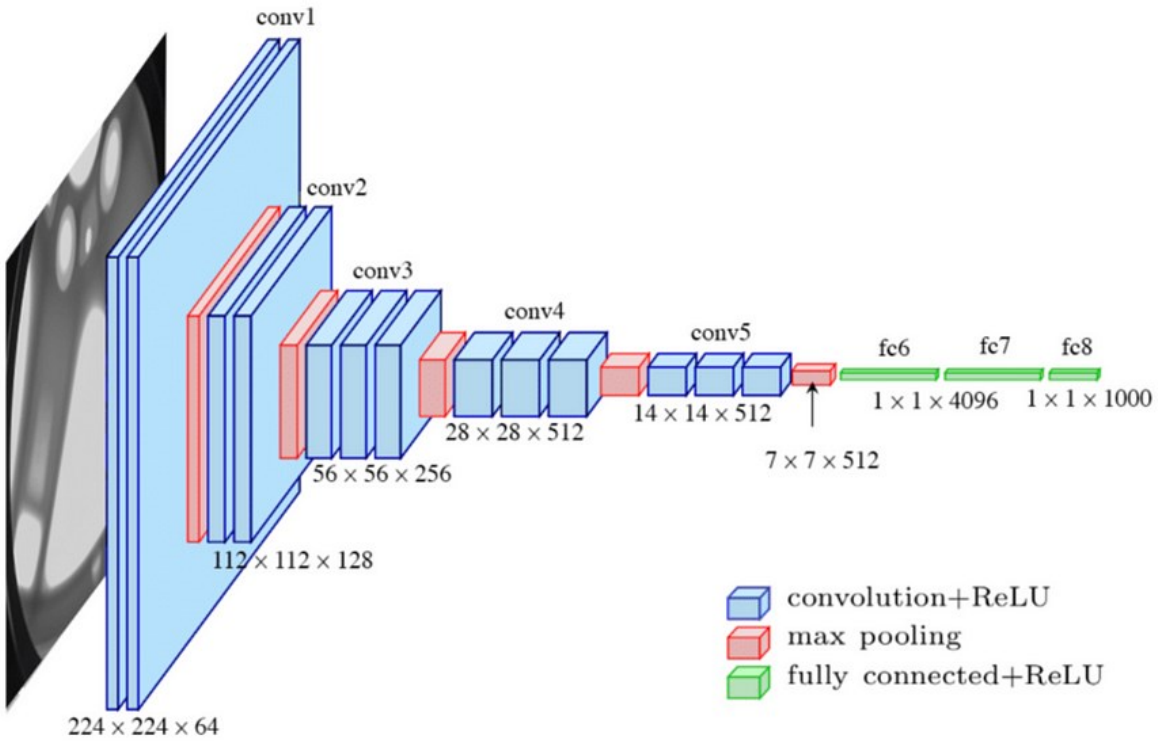
Obr. 2.3: Schéma použití filtru na mapu vlastností (upraveno a převzato z [5])

Kromě konvolučních filtrů se v CNN používají filtry také pro pooling, což je technika sloužící k zmenšení velikosti mapy vlastností. Pooling filtr daného rozměru vypočítá z obsažené oblasti hodnotu následující vrstvy a posune se o určitý počet kroků. V případě ukázky (**Obr. 2.4**) se filtr velikosti 2 x 2 posouvá vždy o 2 pole, tedy s krokem 2. Nejčastěji používané pooling algoritmy jsou max-pooling (**Obr. 2.4**a), kdy je vybrána nejvyšší ze zkoumaných hodnot a average pooling (**Obr. 2.4**b), jež počítá průměr sledované oblasti. Zmenšení je vhodné pro snížení paměťové stopy mapy vlastností a zároveň umožňuje algoritmu sledování větších textur a útvarů ve vstupním obrazu [3].



Obr. 2.4: Schéma max-pooling a average pooling (upraveno a převzato z [23])

Tyto dva postupy jsou kombinovány a opakovaně používány v různých architekturách CNN. Pro ukázku byl vybrán algoritmus neuronové sítě VGG-16 (**Obr. 2.5**), která je využita v řade one-stage (např. Fast-RCNN a Faster-RCNN) i two-stage metod pro detekci objektu (původní verze YOLO či SSD) [3].



Obr. 2.5: Schéma architektury CNN VGG-16 (převzato z [6])

Vstupující 3 kanálový obraz má rozlišení 224 × 244 pixelů. V tabulce (**Tabulka 2.1**) jsou uvedeny parametry všech po sobě jdoucích vrstev této CNN. V závěru algoritmu je po poslední pooling operaci použit flattening, který rozvine výstup konvolučních vrstev do 1D vektoru, a následují 3 plně propojené vrstvy.

Tabulka 2.1: Struktura architektury CNN VGG-16 (data z [7, 8])

	Vrstva	Velikost filtru	Krok	Velikost	Počet map vlastností
Vstup	Obraz	-	-	224 × 224	3
1	2 × konvoluce	3 × 3	1	224 × 224	64
3	max pooling	2 × 2	2	112 × 112	64
4	2 × konvoluce	3 × 3	1	112 × 112	128
6	max pooling	2 × 2	2	56 × 56	128
7	3 × konvoluce	3 × 3	1	56 × 56	256
10	max pooling	2 × 2	2	28 × 28	256
11	3 × konvoluce	3 × 3	1	28 × 28	512
14	max pooling	2 × 2	2	14 × 14	512
15	3 × konvoluce	3 × 3	1	14 × 14	512
18	max pooling	2 × 2	2	7 ×7	512

	Flattening			Převede na 1D vektor	
19	Plně propojená vrstva	-	-	4096	-
20	Plně propojená vrstva	-	-	4096	-
21	Plně propojená vrstva	-	-	1000	-

2.4 Two-stage detektory

Tato skupina detekčních algoritmu, také nazývaná jako region-based detectors dělí proces detekce na lokalizaci objektu a jeho následnou klasifikaci. V prvním kroku algoritmus navrhne několik oblastí zájmu (RoI), které označí jedním z přednastavených referenčních bouding boxů. Následně jsou navržené zóny přiřazeny k odpovídající třídě objektu a ohraničující boxy jsou upraveny na optimální rozměry [9]. Výhodou této metody je vysoká přesnost detekce, která je ovšem vykoupena velkou časovou náročností.

2.4.1 Region-based convolutional neural network (RCNN)

Průkopníkem v oblasti two-stage detektorů je algoritmus Region-based convolutional neural network (RCNN), který používá CNN k extrakci vlastností z každého regionu zájmu. Metoda používá neuronové sítě jako AlexNet či VGG16, které jsou předtrénovány na velkých datasetech (např. ImageNet) a v průběhu tréninku modelu pouze upravují svoje váhy za nízké hodnoty rychlosti učení [10]. Architektura RCNN sestává ze čtyř kroku:

Nejprve algoritmus typu selective search identifikuje oblasti zájmu v obraze na základě různých tvarů, textur nebo barevných vzorů. Těchto oblastí je vybráno přibližně 2000. V druhém kroku jsou všechny oblasti zájmu přeškálovány na stejnou velikost tak, aby odpovídaly požadovanému rozlišení obrazu vstupujícího do neuronové sítě. Pomocí průchodu přes CNN jsou extrahovány vlastnosti vstupujících oblastí. Získaný vektor vlastností pak prochází přes algoritmus support vector machine (SVM), který na jeho základě přiřadí lokalizovanému objektu odpovídající třídu, případně navrženou oblast zavrhne. Po klasifikaci všech nalezených objektů dojde k upřesnění rozměrů jejich ohraničujících bouding boxů pomocí modelu lineární regrese [2].

Ačkoliv se jedná o průlomovou metodu, která značně přispěla k rychlosti a přesnosti detekce objektů, trpí tento algoritmus mnoha limitacemi zejména v rychlosti detekce [2]. To je způsobeno použitím vícestupňového algoritmu či testováním velkého množství oblastí zájmu. I přes to, že metoda RCNN používá CNN k extrakci vlastností, klasifikace a regresní kroky pro

lokalizaci bouding boxů jsou zprostředkovány prostřednictvím SVM případně jinými algoritmy, proto se nejedná o plně neuronový model. Metoda SVM zároveň při klasifikaci objektů kontroluje oblasti zájmu pro každou třídu jednotlivě, což má také významný dopad na rychlost detekce [11]. Nutno dodat, že při vzniku architektury RCNN (2014) použití plně neuronového modelu zdaleka nebylo standardem.

2.4.2 Fast-RCNN

Fast-RCNN přichází s vylepšením z hlediska rychlosti i přesnosti. Na rozdíl od předchozího, tato architektura spojuje tři části: extrakci vlastností, klasifikaci objektu a závěrečnou úpravu bouding boxu do jednoho. Zároveň optimalizuje práci s oblastmi zájmu, kdy do neuronové sítě vstupuje celý obraz, který je zpracován jedním průchodem přes CNN a výstupem je společná mapa vlastností. Z této mapy se vyberou selective search metodou sledované oblasti zájmu. Na získaných RoI je aplikován algoritmus RoI pooling, který zapříčiní vytvoření fixních délek vektorů vlastností. Ty jsou poslány do plně propojené CNN, která současně klasifikuje třídu objektu a zároveň provádí přesnou lokalizaci využitím softmax vrstvy a lineární regrese [2].

Ačkoliv se jedná o výrazný pokrok oproti architektuře RCNN, stále se jedná o časově náročný proces zejména kvůli využití konvenčních metod pro vyhledávání oblastí zájmu, jako je algoritmus selective search [2].

2.4.3 Faster-RCNN

Architektura Faster-RCNN navazuje a vylepšuje předchozí Fast-RCNN. Nahrazuje tradiční časově náročné přístupy pro vyhledávání oblastí zájmu jako selective search nebo MCG (Multiscale combinatorial grouping) pomocí CNN zvané Regional Proposal Network (RPN), která je schopna se v průběhu tréninku učit [2]. Zavedení RPN zrychluje dobu zpracování obrazu z několika sekund na milisekundy [12]. Díky propojení vrstev RPN s neuronovou sítí detekční sekce se zásadně zvyšuje i přesnost detekce a celková efektivita algoritmu.

2.5 One-stage detektory

One-stage frameworky pro detekci objektu konají celý proces současně. Lokalizace i identifikace objektu probíhají zároveň za použití hluboké konvoluční neuronové sítě. Pomocí tohoto přístupu lze dosahovat zpracování obrazu za mnohem kratší dobu, jelikož dochází pouze k jednomu průchodu vstupu neuronovou sítí, při kterém jsou lokalizovány všechny bounding boxy zároveň. Součástí stejného průchodu algoritmem CNN je také přiřazení hodnoty

pravděpodobnosti příslušnosti boxu k určité třídě. Do této skupiny se řadí např. architektury DetectorNet, OverFeat, SSD nebo YOLO [2].

2.5.1 SSD

Single Shot Multibox Detector (SSD) umožňuje detekci více typů objektů zároveň. Proces lokalizace objektu je inspirován architekturou Faster-RCNN, ze které je převzat mechanismus kotev. Díky tomu může SSD extrahovat vlastnosti objektů různých velikostí s podobnou přesností jako Faster-RCNN [2]. Metoda stojí na neuronové sítí VGG-16 [13], která lokalizuje objekty pomocí bounding boxů a zároveň každému boxu přiřazuje pravděpodobnosti příslušností k jednotlivým třídám objektů (nikoliv pravděpodobnost, že se jedná o jakýkoliv objekt). Je tedy třeba za třídu objektu považovat i pozadí jako negativní detekci, aby bylo možné ohraničujícímu rámečku nepřiřadit žádnou konkrétní třídu. Zavrhnutí nevhodně přiřazených objektů zajišťuje metoda non-maximum suppression (NMS). Zejména díky použití RPN dosahuje architektura SSD velmi vysokých detekčních rychlostí při udržení vysokého standardu přesnosti detekce. Pro řadu úloh, z porovnání s algoritmy YOLOv3, Faster R-CNN a dalšími v článku [14], se ale zdá být použití metody SSD nevyhovujícím řešením z důvodu nepřesné detekce malých objektů [13].

2.6 You Only Look Once (YOLO)

Algoritmus You Only Look Once je dalším algoritmem spadajícím do kategorie one-stage detektorů. Základní myšlenka všech verzí této metody stojí na rozdělení vstupního obrazu na mřížku, kde každá buňka zodpovídá za detekci objektů spadajících svým středem na její území. YOLO používá konvoluční neuronovou síť k predikci všech bouding boxů najednou. K detekci každého objektu tedy jsou využity všechny vlastnosti vstupujícího obrazu [15].

Podle [1] se jedná o často využívanou architekturu pro detekci objektů v obraze napříč všemi detekčními přístupy, jelikož disponuje řadou konkurenčních výhod. Mezi nejdůležitější patří vysoká rychlost detekce způsobená použitím one-stage přístupu a řadou dalších optimalizačních opatření. To umožňuje YOLO algoritmu rychle detekovat objekty a cíle v živých video přenosech s vysokým rozlišením v reálném čase, což z něj dělá nejvyužívanější detekční metodu v oblastech autonomního řízení či dohlížecích a bezpečnostních kamerových systémů. Modely umělé inteligence pro detekci objektu v obraze lze pomocí YOLO architektury vytrénovat na velmi vysokou přesnost, přičemž rychlost procesu je zachována. Díky tomu lze algoritmus aplikovat v dalších oblastech jako jsou bezpečnostní kontrola přístupu či inteligentní brány, kde může být totožnost osob ověřena například pomocí

rozpoznání obličeje nebo SPZ. YOLO se také využívá k ovládání robotů jako nástroj vidění robota, který je poté schopen se bezpečně přemísťovat díky detekování blížících se překážek.

2.6.1 Algoritmus YOLO

YOLO architektura podléhá nepřetržitému vývoji a zlepšování detekčních schopností. Aktuálně je dostupných 11 verzí algoritmu od původní varianty YOLO až po nejnovější YOLOv11. Následující obecný mechanismus algoritmu je společný pro všechny verze.

V prvním kroku je vstupní obraz proveden přes CNN, pomocí které jsou extrahovány jeho vlastnosti [16]. YOLO využívá v jednotlivých verzích různé CNN od backbone zvané Darknet v prvotních verzích [13], přes ELAN v YOLOv7 [17] a mnoho dalších.

Získaná mapa vlastností je rozdělena na mřížku. Po průchodu mapy přes plně propojenou CNN každá buňka této mřížky detekuje všechny objekty jejichž středy spadají do oblasti této buňky. Výstupem každé buňky pak jsou nalezené bouding boxy ohraničující objekty a k nim náležící pravděpodobnosti. Každou takovou hodnotou model vyjadřuje pravděpodobnost, že se jedná o nějaký objekt a zároveň jistotu přesnosti určení polohy objektu [16].

Díky mřížkové metodě vznikne řada redundantních bounding boxů způsobených mnohočetnou detekcí jednoho objektu různými buňkami či falešných detekcí objektů s nízkou pravděpodobností. Z tohoto důvodu přichází na řadu algoritmus NMS, který vyřazuje bounding boxy s nízkou šancí na přítomnost objektu. Zároveň na základě metriky Intersection over union (IoU) porovnává, zda se jedná o vícečetnou detekci, či rozdílné objekty. V případě mnohonásobné detekce ponechá bounding box s nejvyšším confidence score [16].

Intersection over Union (IoU)

IoU je metrika založená na porovnání překryvu ploch 2 útvarů. V algoritmu YOLO se využívá v průběhu procesu detekce při použití NMS k porovnání obsahů bounding boxů pocházejících z různých detekcí nebo při statistickém vyhodnocení spolehlivosti modelu pomocí metrik mean Average Precision (mAP). V tomto případě se porovnává míra plochy překrytí referenčního ohraničujícího boxu z datasetu s plochou bounding boxu detekovaného objektu. IoU je vypočteno z rovnice (2.3), kde S_I je obsah plochy průniku dvou oblastí a S_U značí obsah útvaru sjednocujícího tyto dvě plochy [18].

$$IoU = \frac{S_I}{S_U} \tag{2.3}$$

Z rovnice (2.3) je zřejmé, že IoU musí náležet $0 < IoU < 1$. Hodnota je pak obvykle porovnána s prahovou hodnotou IoU_{tresh} a tento výsledek rozhoduje o úspěšnosti detekce.

2.6.2 Vývoj YOLO

Účelem nových verzí YOLO je zvýšení výkonosti detekčních schopností algoritmu oproti verzi předešlé. Hlavním rozdílem mezi verzemi je použití rozdílné architektury neuronové sítě, která se liší téměř v každé variantě. V průběhu vývoje YOLO dochází k častým změnám v pojetí ztrátových funkcí, které ovlivňují průběh tréninku modelu. Každá verze disponuje ztrátovou funkcí vytvořenou na často míru pro dosažení co nejlepších výsledků.

Ztrátová funkce (loss function)

Ztrátová funkce je využívána v průběhu trénování modelu. Měří rozdíl mezi aktuální predikcí modelu a správnými detekcemi zprostředkovanými pomocí informací z datasetu [14]. Ztrátová funkce se skládá z 3 hlavních částí. Ztráta lokalizace měří rozdíl mezi predikovanými bounding boxy a referenčními boxy z databáze. Ztráta důvěryhodnosti bere v potaz rozdíl mezi předpokládaným a skutečným confidence score detekovaných objektů. Ztráta klasifikace udává rozdíl mezi klasifikací modelu a správnými třídami objektů. Tyto funkce jsou sečteny a vyváženy příslušnými koeficienty. Na základě vypočtené hodnoty je rozhodováno o dalším průběhu tréninku modelu [15] [16].

$$L = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2$$

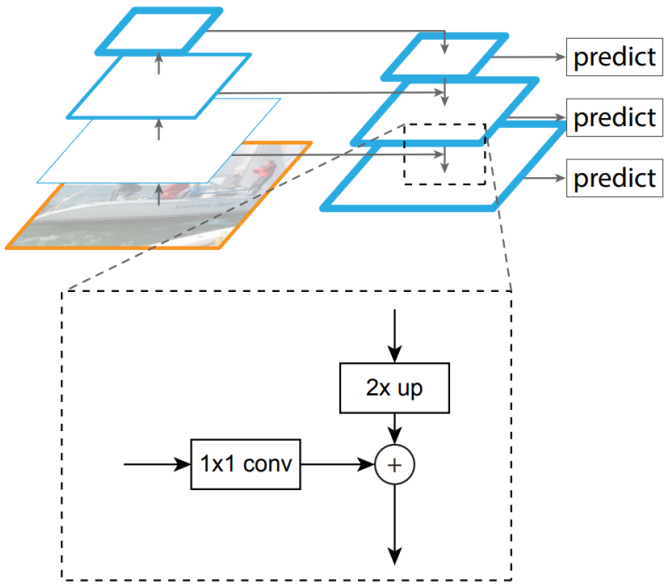
$$+ \sum_{i=0}^{S^2} \sum_{c \in classes} \mathbb{1}_{i}^{obj} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

$$+ \sum_{i=0}^{S^2} \sum_{c \in classes} \mathbb{1}_{i}^{obj} \left(p_i(c) - \hat{p}_i(c)\right)^2 \quad (2.4)$$

Od YOLOv2 je přidána metoda kotvových boxů. Model při detekci objekt ohraničí jedním z přednastavených kotvových boxů o pevné velikosti i poměru stran a poté predikuje posuny boxu k určení přesné polohy objektu [19]. Od YOLOv5 je použita metoda dynamických kotvových boxů, kdy si model vytváří rozměry předdefinovaných boxů v průběhu tréninku jako nejpravděpodobnější tvary objektů vyskytujících se v datasetu [20]. Důležitou změnou byla také implementace konceptu FPN (feature pyramid networks) ve verzi YOLOv3 [19].

Feature pyramid networks (FPN)

FPN slouží jako metoda pro detekci objektů různých rozlišení skládající se ze dvou částí bottom-up a top-down cesty. Bottom-up cesta je standardní CNN, která vytvoří vrstvy různého rozlišení (např. C2, C3, C4, C5), kde se zvyšující se vrstvou klesá prostorové rozlišení, ale zvyšuje se sémantická hodnota. Konvoluční vrstvy jsou použity k vytvoření odpovídajících map vlastností (P2, P3, P4, P5). Nejvyšší vrstva C5 je převedena konvolucí na mapu vlastností P5. Ta je díky vysoké sémantické hodnotě schopna detekovat největší objekty. Následně je použita konvoluce na vrstvu C4 a k vzniklé mapě vlastností je přičtena P5 upsamplovaná na odpovídající velikost. Tím vzniká mapa vlastností P4. Obdobně dochází ke vzniku ostatních map vlastností. Každá mapa je pak schopna detekovat objekty jiných velikostí (viz **Obr. 2.6**) [21].



Obr. 2.6: Schéma algoritmu feature pyramid network (převzato z [21])

2.6.3 Velikosti YOLO

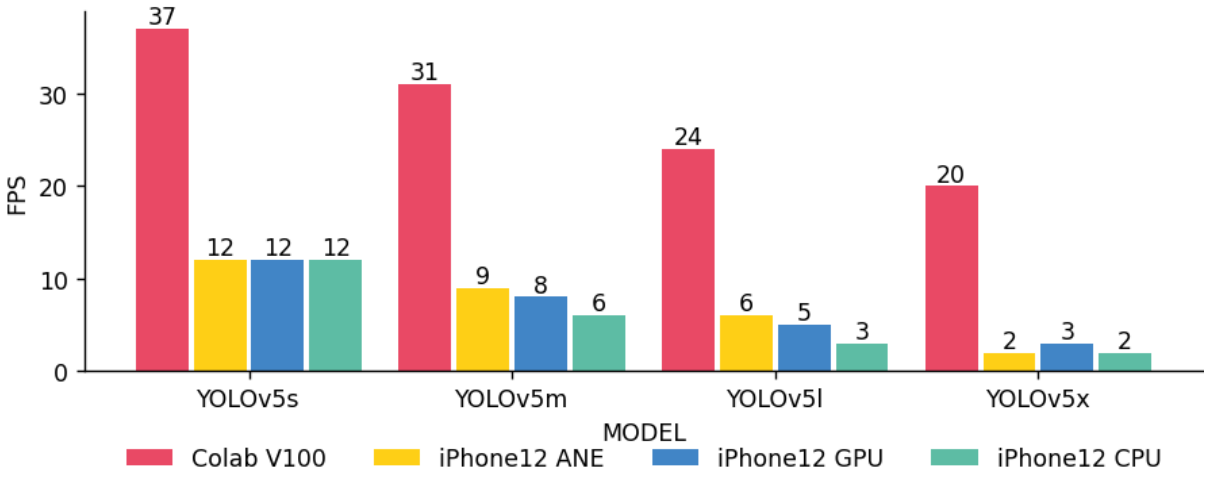
Ultralytics nabízí v každé verzi několik velikostí modelu YOLO (v nejnovějších verzích obvykle n – nano, s – small, m – medium, l – large a x – extra large). S velikostí modelu se zvyšuje přesnost detekce objektů, zároveň ale výrazně stoupá časová i výpočetní náročnost jak při tréninku modelu, tak při samotném detekčním procesu. Zároveň platí, že pro menší velikosti datasetů není třeba využívat velké modely. Kvůli nedostatečnému množství trénovacích dat není využit potenciál složitější architektury neuronové sítě a výsledný model pak dosahuje obdobných, ne-li horších detekčních schopností za mnohem vyšších hardwarových i časových nároků.

V článku [22] byly testovány rozdíly velikostí modelů YOLOv5 natrénovaných za stejných podmínek na datasetu o 10 000 položkách pocházejících z COCO datasetu s rozložením 80/20, kde 80 % dat patřilo trénovací části, ostatní materiál byl umístěn do validační sekce. V **Tabulka 2.2** jsou zobrazeny výsledky tohoto měření. Trénink i detekce objektu probíhali v prostředí Google Colab s dostatečným výpočetním výkonem.

Tabulka 2.2: Porovnání výkonosti velikostí modelů YOLOv5 (převzato a upraveno z [22])

Model	Dataset	mAP@50	mAP@[50:95]	Doba detekce [ms]	GFLOPS
YOLOv5s	výběr z datasetu COCO	38,3	23,6	27	17,0
YOLOv5m		43,7	28,7	32	51,3
YOLOv5l		46,8	31,5	41	115,4
YOLOv5x		48,5	32,8	49	218,8

Dále bylo srovnáno použití modelů pro detekci videa na Google Colab a několika variantách zařízení IPhone 12. Z grafu (**Obr. 2.7**) závislosti počtu snímků za sekundu (fps) na velikosti modelu a použitém zařízení je zřejmé, že při použití hardwarově slabšího zařízení je třeba zvolit jednoduší detekční algoritmus pro udržení dostatečné rychlosti pro detekci objektů v reálném čase i za cenu snížení přesnosti detekce.



Obr. 2.7: Závislost rychlosti detekce v fps na použité velikosti modelu a zařízení (data z [22])

2.6.4 Výstup YOLO algoritmu

Výstupem po zpracování algoritmem je pro každý objekt nalezený v obrazu rámeček ohraničující nalezený objekt pomocí opsaného obdélníku. Ke každé lokalizaci náleží také třída, která udává, o jaký typ objektu se jedná. Dále je ke každému objektu přiřazeno confidence score, které vypovídá jak o pravděpodobnosti detekce objektu správné třídy, tak o jistotě správného určení polohy objektu.

2.7 Srovnání One-stage vs. Two-stage

3 EXPERIMENTÁLNÍ ČÁST

Cílem této práce je vytvoření modelu umělé inteligence schopného detekovat objekty ve snímcích z laryngoskopického vyšetření. Laryngoskopie je lékařská metoda sloužící k diagnostice oblasti hrtanu. Používá se v případě problému s dýcháním, chronickým kašlem, problémy s hlasem nebo při přítomnosti zánětů a nádorů v okolí hlasivek. Postiženému pacientovi je do hrtanu zavedena kamera, která pořídí videozáznam této okolí hlasivek. Následně dochází k posouzení snímané oblasti posuzujícím lékařem. Toto vyhodnocení je velmi subjektivní, závisí na znalostech a schopnostech lékaře a zejména v hraničních situacích se může v podobných případech lišit.

Detekce hlasivek využívající metod zpracování obrazu pomocí umělé inteligence má za cíl posloužit jako první krok k objektivizaci laryngoskopického vyšetření. Na základě vytvořeného modelu umělé inteligence je v budoucnu možné posuzovat vady (např. nedomykavost chlopní hlasivek) počítačově. V návaznosti na tuto práci lze v budoucnu vytvořit další model detekující nádory a záněty vyskytující se na hlasivkách. Softwarovým vyhodnocením v průběhu laryngoskopie by došlo k objektivizaci a značnému urychlení průběhu vyšetření. Vyhodnocení by mohlo probíhat softwarově v reálném čase přímo v průběhu vyšetření.

3.1 Dataset

Jako vstupní dataset byly použity několika sekundové záznamy z laryngoskopických vyšetření, kde každé video pochází z vyšetření jiného pacienta. Do datasetu byli zahrnuti jak zdraví pacienti, tak pacienti s jedním z možných postižení hlasivek. Videa byla pořízena ze dvou různých kamer, což zvyšuje variabilitu datasetu a poskytuje záznamy ve více variantách rozlišení. Tato videa byla rozdělena na snímky a v nich následně byly anotovány (vyznačeny) jednotlivé části hlasivek.

Celý dataset se skládá z … snímků. V každém snímku byla vyznačena levá hlasivka (červená barva), pravá hlasivka (modrá barva) a glotická štěrbina (žlutá barva). Zároveň byly do datasetu zahrnuty obrázky, na nichž je zobrazena jenom část hlasivky, nebo snímky, na kterých není hlasivka viditelná vůbec. Díky tomu model nutně nevyhledává části hlasivek v každém snímku a získává schopnost správně zpracovávat obrazy bez přítomnosti hlasivky. Jedná se vítanou vlastnost, jelikož během kamerového záznamu z laryngoskopického vyšetření není v každém momentu viditelné celé hlasivkové ústrojí.

3.1.1 Manuální anotace

Tvorba datasetu se skládala ze dvou částí. V první části byly snímky anotovány (označovány) manuálně pomocí webové aplikace LabelStudio. Pro označování částí hlasivek byly použity polygonální anotace, kdy každý hledaný objekt byl obtažen uzavřenou lomenou čárou ve formě nepravidelného n-úhelníku. Z hlediska použití datasetu k účelu detekce objektů pomocí algoritmu YOLOv11 nemají polygonální anotace oproti použití obdélníkových boxů žádnou výhodu, poslouží ale k reprodukovatelnosti datasetu při jeho použití k segmentačním účelům. Manuálně bylo označeno přibližně … snímků.

3.1.2 Semi-automatické anotace

Druhá část byla anotována (označována) semi-automatickým způsobem pomocí modelu trénovaného na manuálně vytvořené části datasetu.

3.1.3 Rozložení datasetu

4 VÝSLEDKY A DISKUSE

5 ZÁVĚR

LITERATURA

(1) Cong, X.; Li, S.; Chen, F.; Liu, C.; Meng, Y. A Review of YOLO Object Detection Algorithms based on Deep Learning. *Frontiers in Computing and Intelligent Systems* **2023**, *4* (2), 17-20. DOI: 10.54097/fcis.v4i2.9730.
(2) Ravpreet, K.; Sarbjeet, S. A comprehensive review of object detection with deep learning. *Digital Signal Processing* **2023**, *132*.
(3) Tesema, S. N. *Deep Convolutional Neural Network Based Object Detection Inference Acceleration Using FPGA*; Université Bourgogne Franche-Comté, 2022.
(4) Buettgenbach, M. H. *Explain like I'm five: Artificial neurons*. 2021. https://towardsdatascience.com/explain-like-im-five-artificial-neurons-b7c475b56189 (accessed 2025 25. 1.).
(5) vdumoulin. *conv_arithmetic*. 2016. https://github.com/vdumoulin/conv_arithmetic?tab=readme-ov-file (accessed 2025 27.1.).
(6) Le, K. *An overview of VGG16 and NiN models*. 2021. https://lekhuyen.medium.com/an-overview-of-vgg16-and-nin-models-96e4bf398484 (accessed.
(7) Adams, J.; Qiu, Y.; Posadas, L.; Eskridge, K.; Graef, G. Phenotypic trait extraction of soybean plants using deep convolutional neural networks with transfer learning. *Big Data and Information Analytics* **2021**, *6*, 26-40. DOI: 10.3934/bdia.2021003.
(8) Yu, J.; Li, J.; Sun, B.; Chen, J.; Li, C. Multiclass Radio Frequency Interference Detection and Suppression for SAR Based on the Single Shot MultiBox Detector. *Sensors* **2018**, *18* (11). DOI: 10.3390/s18114034.
(9) Carranza-García, M.; Torres-Mateo, J.; Lara-Benítez, P.; García-Gutiérrez, J. On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data. *Remote Sensing* **2021**, *13* (1), 89.
(10) Yao, J.; Huang, X.; Wei, M.; Han, W.; Xu, X.; Wang, R.; Chen, J.; Sun, L. High-Efficiency Classification of White Blood Cells Based on Object Detection. *Journal of Healthcare Engineering* **2021**, (23), 1-11. DOI: 10.1155/2021/1615192.
(11) Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, Columbus, OH, USA; 2014.
(12) Ahmed, K.; Ghareh Mohammadi, F.; Matus, M.; Shenavarmasouleh, F.; Pereira, L.; Ioannis, Z.; Amini, M. H. Towards Real-time House Detection in Aerial Imagery Using Faster Region-based Convolutional Neural Network. *IPSI Transactions on Internet Research* **2023**, *19* (2), 46-54. DOI: 10.58245/ipsi.tir.2302.06.
(13) Aziz, L.; Haji Salam, M. S. B.; Sheikh, U. U.; Ayub, S. Exploring Deep Learning-Based Architecture, Strategies, Applications and Current Trends in Generic Object Detection: A Comprehensive Review. *IEEE Access* **2020**, *8*, 170461-170495. DOI: 10.1109/ACCESS.2020.3021508.
(14) Liu, Y.; Sun, P.; Wergeles, N.; Shang, Y. A survey and performance evaluation of deep learning methods for small object detection. *Expert Systems with Applications* **2021**, *172*. DOI: //doi.org/10.1016/j.eswa.2021.114602.
(15) Lavanya, G.; Pande, S. Enhancing Real-time Object Detection with YOLO Algorithm. *EAI Endorsed Transactions on Internet of Things* **2023**, *10*. DOI: 10.4108/eetiot.4541.
(16) Badgujar, C. M.; Poulose, A.; Gan, H. Agricultural object detection with You Only Look Once (YOLO) Algorithm: A bibliometric and systematic literature review. *Computers and Electronics in Agriculture* **2024**, *223*. DOI: https://doi.org/10.1016/j.compag.2024.109090.
(17) Yanyun, S.; Liu 刘迪, D.; Chen, J.; Wang, Z.; Wang, Z.; Zhang, Q. On-Board Multi-Class Geospatial Object Detection Based on Convolutional Neural Network for High Resolution Remote Sensing Images. *Remote Sensing* **2023**, *15* (16). DOI: 10.3390/rs15163963.

(18) Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. In 32nd IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA; Paper 8953982, 2019.
(19) Kaur, S.; Kaur, L.; Lal, M. A Review: YOLO and Its Advancements. In 6th International Conference on Recent Innovations in Computing, ICRIC 2023, Jammu, Indie; 2024.
(20) Tai, W.; Wang, Z.; Li, W.; Cheng, J.; Hong, X. DAAM-YOLOV5: A Helmet Detection Algorithm Combined with Dynamic Anchor Box and Attention Mechanism. *Electronics* **2023**, *12* (9). DOI: 10.3390/electronics12092094.
(21) Lin, T.-Y.; Ramanauskaitė, S.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In 30TH IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR 2017), Honolulu, HI, USA; 2017.
(22) Dlužnevskij, D.; Stefanovič, P.; Ramanauskaitė, S. Investigation of YOLOv5 Efficiency in iPhone Supported Systems. *Baltic Journal of Modern Computing* **2021**, *9* (3). DOI: 10.22364/bjmc.2021.9.3.07.

SEZNAM POUŽITÝCH ZKRATEK

SEZNAM SYMBOLŮ

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

PŘÍLOHY