SOUHRN

SUMMARY

PODĚKOVÁNÍ

OBSAH

Naformátováno: Obsah 3

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování: Standardní písmo odstavce, Kontrolovat pravopis a gramatiku

nastavil formátování ...

nastavil formátování ...

nastavil formátování ...

nastavil formátování ...

nastavil formátování ...

nastavil formátování ...

nastavil formátování ...

nastavil formátování ...

nastavil formátování ...

nastavil formátování ...

nastavil formátování ...

nastavil formátování ...

1 ÚVOD

V posledních letech zažívá oblast umělé inteligence (AI) výrazný rozmach napříč mnoha vědními obory. Jednou z oblastí, kde dochází k její dynamické aplikaci, je zdravotnictví. Automatizace rutinních činností, podpora lékařského rozhodování či analýza obrazových a signálových dat představují nové možnosti, jak zefektivnit a zpřesnit diagnostické procesy. Tato práce se zabývá aplikací metod umělé inteligence ve zpracování obrazových dat z lékařského vyšetření zvaného laryngoskopie.

Laryngoskopie je lékařská metoda sloužící k přímému pozorování hrtanu, zejména oblasti hlasivek. Využívá se k diagnostice v případě obtíží 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 okolí hlasivek, následně dochází k posouzení snímané oblasti, které je v současné době plně v rukou lékaře. Tento proces je do značné míry subjektivní, závisí na znalostech a schopnostech lékaře a zejména v hraničních případech se může v závislosti na posuzující osobě diagnóza hlasivek lišit.

Cílem práce je vytvořit model umělé inteligence schopný detekovat specifické anatomické struktury ve snímcích z laryngoskopického vyšetření. Detekce hlasivek využívající metod zpracování obrazu pomocí AI může sloužit jako první krok k objektivizaci vyšetření. Na základě vytvořeného modelu bude v budoucnu možné posuzovat vady (např. nedomykavost chlopní hlasivek) počítačově. V návaznosti na tuto práci mohou být vytvořeny další AI modely detekující nádory a záněty vyskytující se v oblasti hlasivek. Nasazení vyhodnocovacího nástroje v klinické praxi by přispělo nejen k objektivizaci, ale také ke značnému urychlení průběhu vyšetření. Systém by mohl vyhodnocovat stav hlasivek v reálném čase přímo v průběhu laryngoskopie a upozorňovat na podezřelé oblasti, čímž by podpořit lékaře v rozhodování a zvýšil by kvalitu péče o pacienty.

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 AI. 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. 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 použít dva základní přístupy: konvenční metody a metody hlubokého učení využívající konvoluční neuronové sítě. Konvenční přístup pracuje ve třech krocích. V první fázi algoritmus pomocí posuvného okna prohledává obraz a vybírá oblasti, kde by se mohl nacházet objekt. Následně z těchto oblastí získává popisné znaky (např. texturu, barvu či tvar), které slouží k rozpoznání a zařazení 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 modely založené na konvolučních neuronových sítí (CNN) k rozeznání jednotlivých příznaků obrazu a následné detekci všech v něm se nacházejících objektů. Tento přístup je charakteristický rozdělením na tři hlavní sekce algoritmu [2].

- Extrakce příznaků, kdy je vstupní obraz zpracován pomocí CNN, která detekuje klíčové rysy obrazu, jako jsou hrany, tvary či složitější textury a sestaví mapu příznaků obrazu.
- Lokalizace objektů z mapy příznaků 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.

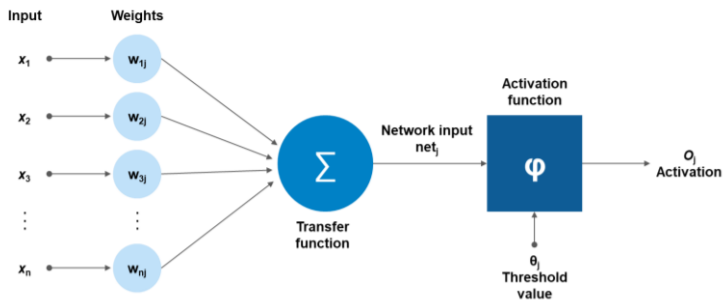
2.2 Umělé neuronové sítě

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 vektorem vah $\boldsymbol{w_j} = (w_{1j}, w_{2j}, \dots, w_{nj})$, následně je přičten bias b_j. Hodnota z_j je vypočítána podle rovnice (2.1) [3].

$$z_j = \sum \boldsymbol{x} \cdot \boldsymbol{w_j^T} + b_j \left(\boldsymbol{x} \cdot \boldsymbol{w_j^T}\right) + b_j \tag{2.1}$$

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

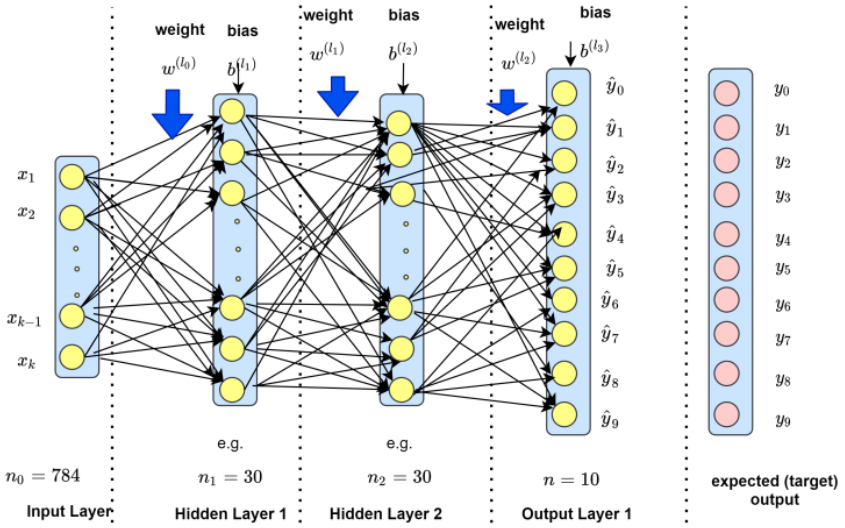
$$y_j = f(z_j) \tag{2.2}$$



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

Trénink ANN se skládá ze tří částí. Na začátku tréninku jsou náhodně či pomocí specifických metod inicializovány biasy a vstupní váhy neuronů. 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, které obsahují ~~s počtem neuronů $n = 30$~~ 30 neuronů a jednou výstupní vrstvou s 10 neurony~~$n = 10$~~. 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~~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~~ optimální 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), která vstupy ve formě vektorů převádí do dvou či více dimenzí nazývaných mapy příznaků. Na rozdíl od předchozího, neurony v CNN nejsou plně propojeny, ale mají vazbu pouze s několika prostorově blízkými hodnotami v předchozí vrstvě. Namísto skalárního součinu s vektorem hodnot v ANN (rovnice 2.1) je zde provedena konvoluce ~~Obdobně jako v ANN je zde provedena konvoluce~~ vstupního

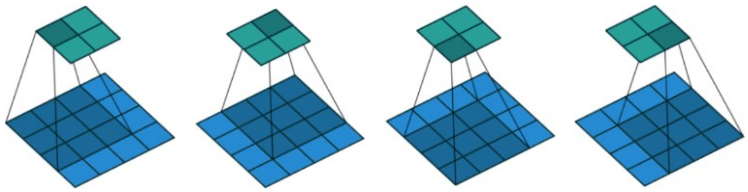
Okomentoval(a): [J1]: Oproti ANN je zde provedena..

vektoru s maticemi hodnot, které se v tomto případě nazývají filtry či jádra. Ty mají obvykle čtvercový tvar s obvyklými velikostmi stran mezi 1 a 11 [3]. Výstup z konvoluční vrstvy se počítá pomocí diskrétní konvoluce, která se ve dvou dimenzích formálně zapisuje jako (rovnice 2.3)

$$(f * h)(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x - i, y - j) \cdot h(i, j), \tag{2.3}$$

kde f je vstupní obraz, h značí konvoluční jádro a $(f * h)$ je výsledná mapa příznaků.

Při výpočtu hodnot, které jsou vstupem do následující vrstvy se použije mapa příznaků předchozí vrstvy a provede se výpočet konvoluce. Každá hodnota následující vrstvy (tyrkysová na **Obr. 2.3**) je vytvořena konvolucí několika hodnot mapy příznaků z předchozí 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 příznaků v jedné vrstvě [3].

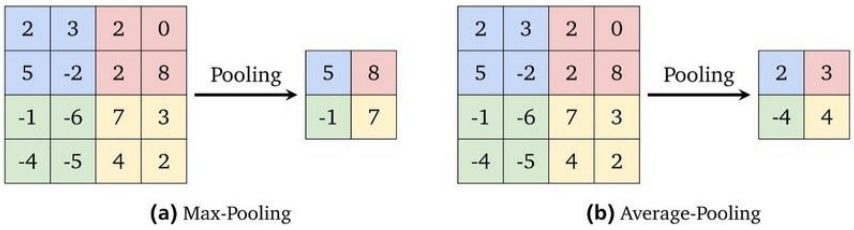


Obr. 2.3~~2.3~~: Schéma použití filtru na mapu příznaků (upraveno a převzato z [5])

Kromě konvolučních filtrů se v CNN používají filtry také pro pooling, což je technika sloužící k zmenšení velikosti mapy příznaků. Pooling okno 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. 2.4a**), kdy je vybrána nejvyšší ze zkoumaných hodnot a average pooling (**Obr. 2.4b**), jež počítá průměr vybrané oblasti. Pooling vybere nejreprezentativnější část oblasti, čímž umožňuje algoritmu sledování větších textur a útvarů ve vstupním obrazu a zároveň dochází ke snížení paměťové stopy mapy příznaků [6].

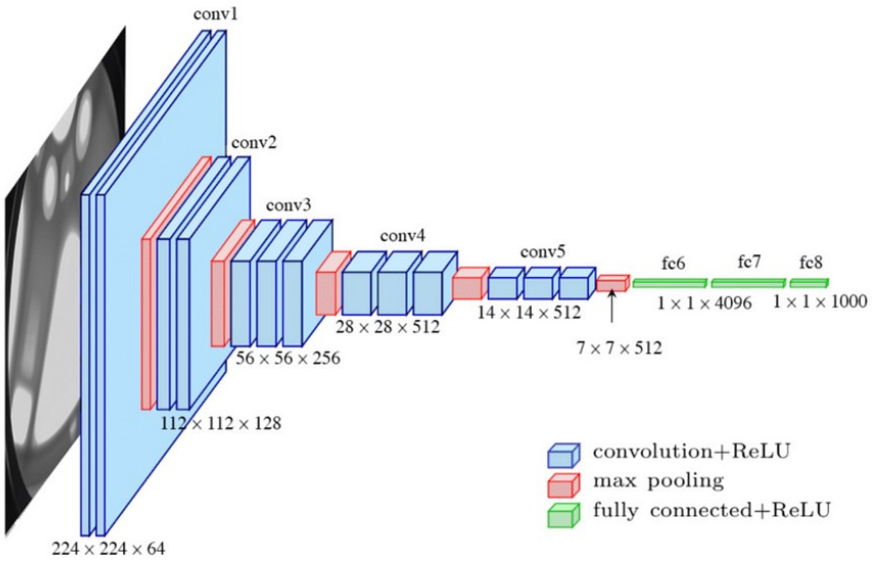
nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné



Obr. 2.4~~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 řadě one-stage (např. Fast R-CNN a Faster R-CNN) i two-stage (některé varianty Single-shot Detector (SSD)) metod pro detekci objektu [3].



Obr. 2.5~~2.5~~: Schéma architektury CNN VGG-16 (převzato z [7])

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

Tabulka 2.1~~2.1~~: Struktura architektury CNN VGG-16 (data z [8, 9])

Vrstva		Velikost filtru	Krok	Velikost	Počet map příznaků
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 detektory 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 bounding boxů. Následně jsou navržené oblasti přiřazeny k odpovídající třídě objektu a ohraničující boxy jsou upraveny na optimální rozměry [10]. Výhodou této metody je vysoká přesnost detekce, která je ovšem vykoupena velkou časovou náročností.

2.4.1 Region-based convolutional neural network (R-CNN)

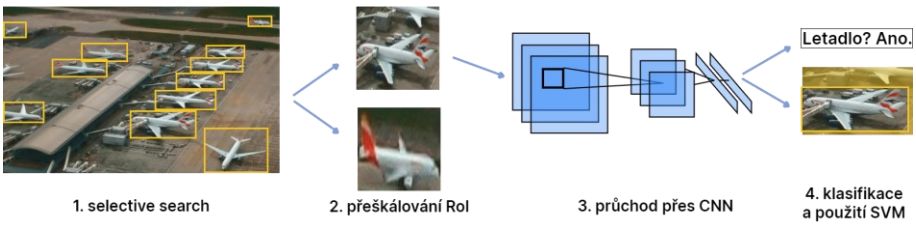
Průkopníkem v oblasti two-stage detektorů je algoritmus Region-based convolutional neural network (R-CNN), který používá CNN k extrakci příznaků z každého regionu zájmu. Metoda používá neuronové sítě jako AlexNet či VGG16, které jsou předtrénovány na velkých datasetech (např. ImageNet [11]) a během trénování modelu se jejich váhy pouze jemně dolaďují pomocí nízké hodnoty rychlosti učení [12].

Okomentoval(a): [JI2]: Mám znova přepisovat tu zkratku celým názvem, když jsem to udělal už v nadpisu?

Modely R-CNN pracují ve čtyřech krocích (**Obr. 2.6**). Nejprve algoritmus typu selective search (popsán níže) 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 příznaky vstupujících oblastí. Získaný vektor příznaků pak prochází přes algoritmus podpůrných vektorů (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].

Okomentoval(a): [JT3]: Asi by se vyplatilo někde krátce vysvětlit

Okomentoval(a): [JI4R3]: Mám přidat i obrázek?



Obr. 2.6~~2.6~~: Schéma jednotlivých fází algoritmu R-CNN

Ačkoliv se jednalo o průlomovou metodu, která významně přispěla ke zlepšení přesnosti detekce objektů, z dnešního pohledu trpí tento algoritmus mnoha limitacemi zejména v rychlostí detekce [2]. To je způsobeno použitím vícestupňového algoritmu či testováním velkého množství oblastí zájmu. I přes to, že metoda R-CNN používá CNN k extrakci příznaků, klasifikace a regresní kroky pro lokalizaci ~~boudingbounding~~ 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 [13]. Nutno dodat, že při vzniku architektury R-CNN (v roce 2014) nebylo použití plně neuronového modelu zdaleka běžné.

Selective search

Při vytváření RoI pomocí algoritmu selective search se vstupní obraz nejprve rozdělí na superpixely – malé oblasti s podobnými barvami nebo texturou. Následně je provedeno slučování těchto oblastí, které se vyhodnocuje na základě společných rysů regionů, tím vzniknou nové oblasti zájmu. Zmíněný krok slučování se iterativně opakuje do té doby, něž se všechny RoI postupně opět spojí do jedné oblasti obsahující celý vstupní obraz. Všechny

vzniklé návrhy RoI v průběhu selective search (často nižší tisíce oblastí) jsou následně předány do další fáze detekčního algoritmu (např. R-CNN) [14].

2.4.2 Fast R-CNN

Fast R-CNN přichází s vylepšením z hlediska rychlosti i přesnosti. Na rozdíl od předchozího přístupu, tato architektura spojuje tři části: extrakci příznaků, klasifikaci objektu a závěrečnou úpravu bouding boxu do jedné. 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 příznaků. Z této mapy se vyberou selective search metodou sledované oblasti zájmu. Na získaných RoI je aplikován pooling, který zajistí vytvoření fixních délek vektorů příznaků. Ty jsou poslány do plně propojené CNN, která současně klasifikuje třídu objektu a zároveň provádí přesnou lokalizaci využitím softmax vrstvy a lineární regrese [2].

Okomentoval(a): [IJ5]: Z RoI pooling -> pooling...abych to nemusel vysvětlovat (pooling je uvedený výše), je to ok?

Ačkoliv se jedná o výrazný pokrok oproti architektuře R-CNN, 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 R-CNN

Architektura Faster R-CNN navazuje a vylepšuje předchozí Fast R-CNN. 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 [15]. Integrací RPN do detekční sítě architektury došlo také k výraznému zlepšení i v přesnosti detekce objektů.

2.5 One-stage detektory

One-stage ~~frameworky~~ algoritmy pro detekci objektu provádí lokalizaci a identifikaci objektu zároveň, 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~~ Single Shot Detector nebo You Only Look Once (YOLO) [2].

2.5.1 Single Shot Multibox Detector (SSD)

Okomentoval(a): [JI6]: Mám ta být i to Multibox?

Single Shot Multibox Detector (SSD) umožňuje detekci více typů objektů zároveň. Proces lokalizace objektu je inspirován architekturou Faster R-CNN, ze které je převzat mechanismus kotev. Kotvové boxy jsou předdefinované obdélníky různých poměrů stran a velikostí. Při lokalizaci nemusí CNN polohu objektu předpovídat od nuly, pouze vybere jeden z těchto útvarů a poté predikuje posuny boxu k určení přesné polohy objektu [16]. Díky tomu může SSD extrahovat příznaky objektů různých velikostí s podobnou přesností jako Faster R-CNN [2]. Algoritmus využívá neuronovou síť VGG-16 [17], 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, v porovnání s algoritmy YOLOv3, Faster R-CNN a dalšími v článku [18], se ale zdá být použití metody SSD nevyhovujícím řešením z důvodu nepřesné detekce malých objektů [17].

2.6 You Only Look Once (YOLO)

You Only Look Once je dalším algoritmem spadajícím do kategorie one-stage detektorů. Základní myšlenka všech verzí YOLO stojí na rozdělení vstupního obrazu na mřížku, kde každá buňka zodpovídá za detekci objektů spadajících svým středem do~~na~~ její ~~území~~oblasti. YOLO používá konvoluční neuronovou síť k predikci všech bouding boxů najednou. K detekci každého objektu tedy jsou využity všechny příznaky vstupujícího obrazu [19].

Okomentoval(a): [JT7]: Pokud jinde tento typ citace nepoužíváš, není ho vhodné použít ani tady

Okomentoval(a): [JI8R7]: Je lepší to dát na konec, když to cituje celý odstavec?

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 AI pro detekci objektu v obraze pomocí YOLO architektury lze na~~vy~~trénovat 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~~v robotice jako nástroj vidění robota, který je schopen se bezpečně přemísťovat díky detekování blížících se překážek. V posledních letech je tato oblast zastoupena zejména využitím YOLO algoritmu v oblasti vývoje autonomního řízení vozidel.

Okomentoval(a): [JI9]: Přidal jsem jen autonomní vozidla. O Tesle jsem našel, že mají to YOLO nějak vylepšené, tak už bych tam radši o YOLO nepsal.

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 zpracován CNN, pomocí které jsou extrahovány jeho příznaky [20]. YOLO využívá v jednotlivých verzích různé CNN od backbone (část neuronové sítě zodpovědná za extrakci relevantních rysů z různých úrovní rozlišení vstupního obrazu) zvané Darknet v prvotních verzích [17], přes ELAN v YOLOv7 [21] a mnoho dalších.

Získaná mapa příznaků 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 [20].

Naformátováno: Normální

~~**2.6.1**~~ 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 – viz 2.8.1) porovnává, zda se jedná o vícečetnou detekci, či rozdílné objekty. V případě mnohonásobné detekce ponechá bounding box s nejvyšším confidence score [20].

Okomentoval(a): [IJ10]: Tady bylo „Přidal bych sem i Loss který YOLO používá," ale tohle je o vyhodnocovení obrázků, ne o trénování, loss function je popsaná níž

~~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~~zpracován ~~CNN, pomocí které jsou extrahovány jeho příznaky [20]. YOLO využívá v jednotlivých verzích různé CNN od backbone (část neuronové sítě zodpovědná za extrakci relevantních rysů z různých úrovní rozlišení vstupního obrazu) zvané Darknet v prvotních verzích [17], přes ELAN v YOLOv7 [21] a mnoho dalších.~~

~~Získaná mapa příznaků 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 [20].~~

~~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 – viz 2.8.1) porovnává, zda se jedná o vícečetnou detekci, či rozdílné objekty. V případě mnohonásobné detekce ponechá bounding box s nejvyšším confidence score [20].~~

Okomentoval(a): [IJ11]: Tady bylo „Přidal bych sem i Loss který YOLO používá," ale tohle je o vyhodnocovení obrázků, ne o trénování, loss function je popsaná níž

~~2.6.6~~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 často na 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. Ztrátová funkce se skládá z 3 hlavních částí. Ztráta lokalizací 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 (viz rovnice 2.4). Na základě vypočtené hodnoty je rozhodováno o dalším průběhu tréninku modelu [22, 23].

Ztráta lokalizace

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

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

Ztráta důvěryhodnosti

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

Ztráta klasifikace

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

Od YOLOv2 je přidána metoda kotvových boxů. 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 [24]. Důležitou změnou byla také implementace konceptu Feature Pyramid Networks (FPN) ve verzi YOLOv3 [16].

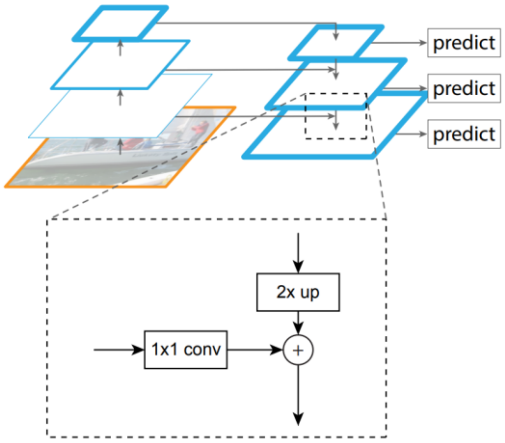
Feature pyramid networks (FPN)

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 příznaků (P2, P3, P4, P5). Nejvyšší vrstva C5 je převedena konvolucí na mapu příznaků P5. Ta je díky vysoké sémantické hodnotě schopna detekovat největší objekty. Následně je použita konvoluce na vrstvu C4 a k vzniklé mapě příznaků je přičtena P5 nadvzorkovaná na odpovídající velikost. Tím vzniká mapa příznaků P4. Obdobně dochází ke vzniku ostatních map příznaků. Každá mapa je pak schopna detekovat objekty jiných velikostí (viz **Obr. 2.7**) [25].

Okomentoval(a): [JT12]: sémantická hodnota je terminus technicus? Jestli ne tak bych to nějak popsal jinak

Okomentoval(a): [JI13R12]: „semantic value“ by měl být termín

Okomentoval(a): [JI14]: Přeloženo „upsampling“, ale slovník mi nenašel, že by slovo nadvzorkování existovalo



Obr. 2.7~~2.7~~: Schéma algoritmu feature pyramid network (převzato z [25])

~~2.6.7~~2.6.3 Velikosti YOLO

Ultralytics nabízí 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 schopnost rozlišovat složitější struktury v obraze, zároveň ale výrazně stoupá časová i výpočetní náročnost jak při tréninku modelu, tak při samotném detekčním procesu. Současně 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 při mnohem vyšších hardwarových i časových nárocích.

Okomentoval(a): [JT15]: , firma vydávající YOLO,

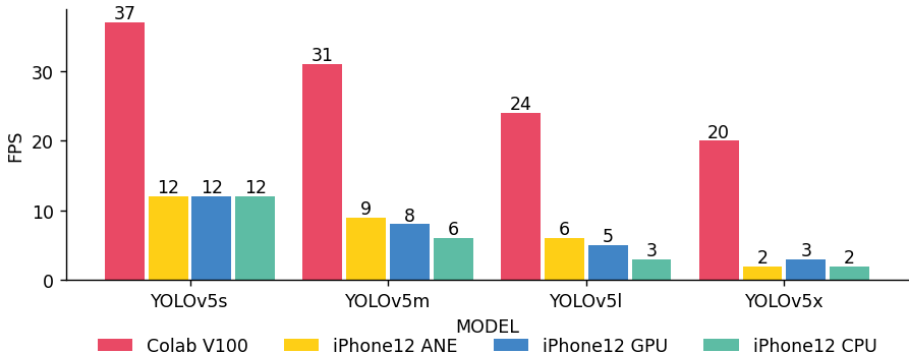
V článku [26] byly testovány rozdíly velikostí modelů YOLOv5 natrénovaných za stejných podmínek na datasetu o 10 000 položkách pocházejících z COCO datasetu [27] s rozložením 80/20, kde 80 % dat patřilo trénovací části, ostatní materiál byl umístěn do validační sekce. V **Tabul~~cka~~ 2.2** jsou zobrazeny výsledky tohoto měření. Trénink i detekce objektu probíhali v prostředí Google Colab s odpovídacím výpočetním výkonem.

Tabulka 2.2~~2.2~~: Porovnání výkonosti velikostí modelů YOLOv5 (převzato a upraveno z [26])

Model	Dataset	mAP@50	mAP@[50:95]	Doba detekce [ms]	GFLOPS
YOLOv5s		38,3	23,6	27	17,0
YOLOv5m		43,7	28,7	32	51,3

YOLOv5l	výběr z datasetu COCO	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. 2.8**) 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. 2.8~~2.8~~: Závislost rychlosti detekce v fps na použité velikosti modelu a zařízení (data z [26])

~~2.6.8~~2.6.4 Výstup YOLO algoritmu

Výstupem po zpracování algoritmem je pro každý objekt nalezený v obrazu bounding box 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 algortimů

Porovnání výše uvedených přístupů není snadnou záležitostí. Spolehlivost každého algoritmu lze měnit různým nastavením jeho hyperparamertů, zároveň žádný model není nejefektivnějším řešením pro všechny úlohy. Každý z uvedených modelů má své využití v jeho aplikační oblasti, různým úlohám tedy vyhovují různé detekční algoritmy. V článku [28] byla srovnána přesnost a rychlost rozdílných one-stage a two-stage přístupů, z nichž vybrané modely byly uvedeny v **Tabulka 2.3**. Je třeba mít na paměti, že výsledky mohou být do určité míry ovlivněny diskutovanými faktory.

Tabulka 2.3~~2.3~~: Srovnání one-stage vs. two-stage algoritmů (převzato a upraveno z [28])

	Model	mAP@50	FPS
two-stage	R-CNN	-	0,03
	Fast R-CNN	39,9	0,5
	Faster R-CNN	42,7	7
one-stage	SSD512	46,5	19
	YOLOv4 608x608	65,7	23

2.8 Evaluační metriky

K vyhodnocování detekční přesnosti modelů se využívá celá řada veličin a metrik. Dále jsou uvedeny ty, která se používají pro vyhodnocování modelů v této práci.

2.8.1 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 [29].

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

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

2.8.2 Precision

Veličina precision udává procento správně identifikovaných objektů vůči všem predikcím. Výpočet $Precision$ se provádí následovně (rovnice 2.5):

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

kde TP (true positive) je počet predikcí modelu, které jsou skutečně správnými a FP (False positive) udává počet nesprávně nalezených predikcí.

Okomentoval(a): [JS16]: Bylo by vhodné rozvést, jak se vlastně určují TP, FP, TN a FN.

~~kde TP (true positive) je počet predikcí modelu, které jsou skutečně správnými a FP (False positive) udává počet nesprávně nalezených predikcí.~~

~~2.8.4~~2.8.3 Recall

Recall značí procentuální počet správně nalezených objektů modelem ze všech ground truth boxů. Veličina $Recall$ se vypočte podle rovnice 2.6

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

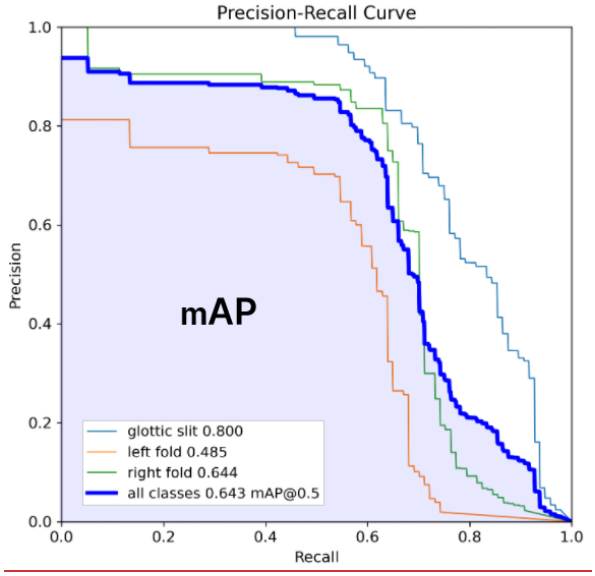
kde TP má identický význam jako v rovnici 2.5 a FN (false negative) značí počet modelem nepředpovězených objektů, které se ve skutečnosti v obraze nacházejí.

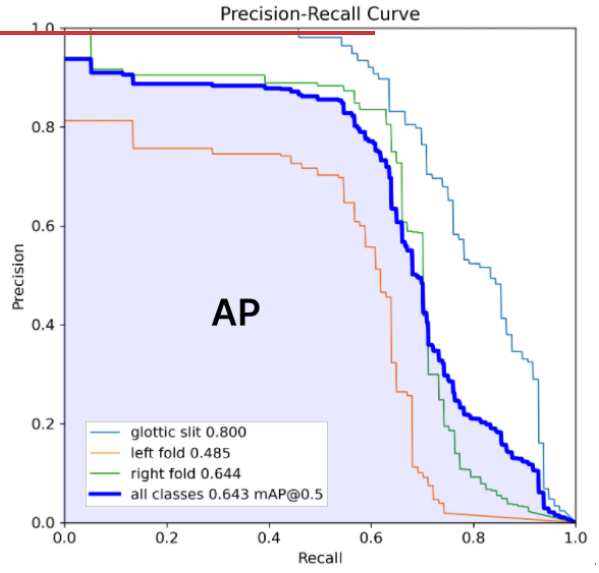
~~2.8.5~~2.8.4 Average Precision (AP)

Metrikou mean Average Precision (mAP) je měřena výkonost natrénovaného modelu. Model je vyzkoušen na testovací části datasetu a ze vzniklých bounding boxů a ground truth boxů z databáze jsou vypočteny IoU. Úspěšnost detekce je obvykle vyhodnocena na mnoha hladinách IoU_{tresh}. Následující postup pak probíhá pro každé IoU_{tresh} zvlášť. Nejprve je na základě výsledků úspěšných a neúspěšných detekcí sestavena závislost zvaná precision-recall ~~zvaná precision-recall~~ curve (**Chyba! Nenalezen zdroj odkazů.**). Plocha pod touto křivkou je definována jako average precision AP a je vypočtena stejně jako precision-recall curve pro každou třídu objektu zvlášť [30].

Okomentoval(a): [IJ17]: Přeložit znamená „precision-recall křivka“ nebo „křivka přesnosti a úplnosti“? Stejně jsou v odstavci i jiné názvy anglicky, tak nevím, jestli to překládato

Okomentoval(a): [JS18R17]: Tyto metriky nemají dobré překlady, obzvlášť přesnost je slovo, kterým se překládá pojem accuracy, což je ale zase jiná metrika.
Pokud budeš mít ostatní pojmy přeložené do češtiny, byl bych za to při první zmínce uvést originální anglický název do závorek za český překlad.





Obr. 2.9~~2.9~~: Ukázka Precision-Recall curve a plochy AP

Okomentoval(a): **[IJ19]:** „Zdůraznit že to už jsou výsledky tvého modelu“ -> to sice jsou moje výsledky, ale je to výsledek nějakého random špatně fungujícího modelu, protože na těch dobrých jsou všechny čáry přes sebe a nic tam nejde vidět. Tenhle je tam jen z ukázkových důvodů.

Zprůměrováním všech AP vzniká výsledná hodnota mAP zvaná mean Average Precision.

mAP@50

Běžné používanou metrikou pro vyhodnocování modelů AI je $mAP@50$, kde označení @50 udává, že za úspěšnou detekci se považuje taková predikce objektu, jejíž IoU překryvu s ground truth boxem je alespoň 50 %.

mAP@[50:95]

Další obvyklou metrikou je $mAP@[50:95]$, která postupně vypočítá hodnoty mAP na deseti prazích úspěšnosti $0{,}50; 0{,}55; 0{,}60; \ldots; 0{,}95$ a z průměrů dílčích hodnot vypočítá výslednou $mAP@[50:95]$. Jedná se tedy o přísnější metriku oproti $mAP@50$, kvůli vyšším nárokům na predikce modelu.

3 EXPERIMENTÁLNÍ ČÁST

Při laryngoskopickém vyšetření je pořízen videozáznam oblasti hrtanu, pomocí něhož je následně vyhodnocován stav hlasivkového ústrojí pacienta. Tato práce si klade za cíl vytvořit model AI, který bude schopen identifikovat a správně detekovat několik částí hlasivek zachycených na záznamu, a to jak v případě zdravého stavu, tak v případě přítomnosti vizuálně patrných patologických změn, například zánětů či nádorových útvarů.

K vytvoření modelu AI pro detekci objektů v obrazech byl použit algoritmus YOLO, který patří do kategorie učení s učitelem, kdy se model trénuje na datech, která obsahují, jak vstupní obraz, tak i odpovídající anotace objektů. Algoritmus YOLO byl zvolen zejména pro svou vysokou rychlost a efektivitu při detekci objektů v obraze. Na rozdíl od tradičních detekčních metod, které analyzují jednotlivé části obrazu postupně, YOLO zpracovává celý obraz v rámci jednoho průchodu neuronovou sítí. Tento přístup nejen významně zrychluje detekci, ale zároveň umožňuje dosažení velmi dobré přesnosti v reálném čase. Díky této vlastnosti je YOLO ideálním kandidátem pro medicínské aplikace, kde je klíčová kombinace rychlosti a spolehlivosti, jako je právě automatizovaná analýza videí z laryngoskopických vyšetření.

Nedílnou součástí vytvoření modelu AI pro detekci částí hlasivek je příprava vstupních dat, anotace sledovaných objektů a volba správného rozložení datasetu pro trénink tak, aby data poskytnutá modelu co nejlépe vystihovala reálnou situaci a dobře model připravila na jeho využití v praxi. V další fázi je YOLO model trénován na vytvořeném vstupním datasetu a souběžně je prováděna optimalizace jeho hyperparametrů pro dosažení co nejpřesnější detekce.

3.1 Dataset

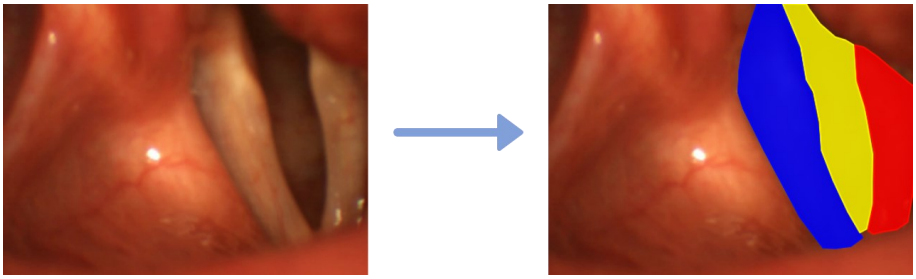
Jako vstupní dataset byly použity několika sekundové záznamy z laryngoskopických vyšetření, kde každé video pochází z vyšetření jiného pacienta. Dataset byl ~~získán~~ vytvořen ve spolupráci s otorinolaryngologickou klinikou Fakultní nemocnic~~í~~ Královské Vinohrady, která ~~poskytla~~ pořídila potřebné obrazové materiály pro analýzu. Do datasetu byli zahrnuti jak zdraví pacienti, tak pacienti s určitým postižením hlasivek. Videa byla pořízena ze dvou různých kamer, což zvyšuje variabilitu datasetu a poskytuje záznamy v různých kvalitách rozlišení. Tato videa byla rozdělena na snímky, ve kterých byly vyznačeny jednotlivé části hlasivek.

Dataset se skládá laryngoskopických videozáznamů oblasti hrtanu různých délek (do 60 sekund) pocházejících od 20 pacientů. Tato videa jsou~~byla~~ pořízena s frekvencí 30 snímků za sekundu a po jejich rozdělení na jednotlivé snímky obsahuj~~oval~~ dataset celkem 12 991 vzorků.

V každém byla oanotována levá hlasivka (červená barva), pravá hlasivka (modrá barva) a glotická štěrbina (žlutá barva). Zároveň byly do datasetu zahrnuty snímky, 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 o 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 manuálně pomocí webové aplikace LabelStudio [31]. Pro označování částí hlasivek byly použity polygonální anotace, kdy každý hledaný objekt byl obtažen uzavřenou lomenou čárou ve formě nepravidelného n-úhelníku (**Obr. 3.1**). 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 trénování segmentačních modelů. Manuálně bylo označeno přibližně 3 500 snímků.



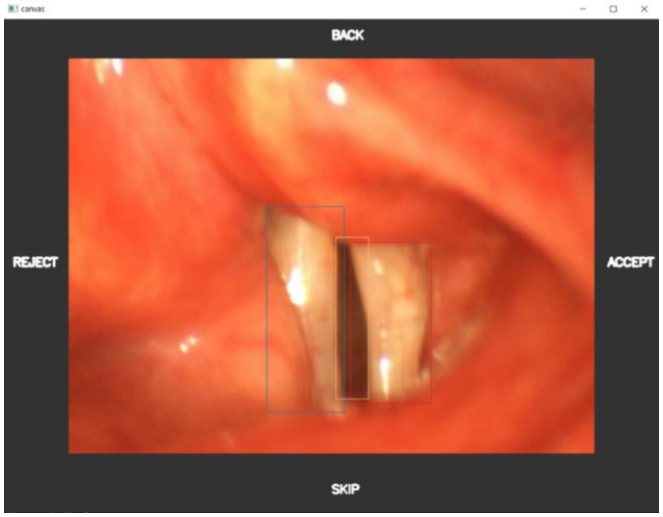
Obr. 3.1~~3.1~~: Schéma postupu při manuální anotaci snímků hlasivek

3.1.2 Semi-automatické anotace

Druhá část byla anotována semi-automatickým způsobem pomocí modelu trénovaného na manuálně vytvořené části datasetu. V programovacím jazyce Python byl vytvořen skript pro automatickou anotaci laryngoskopických snímků skládající se ze tří částí. V první fázi jsou snímky z videozáznamu jednoho pacienta označeny dříve vytvořeným YOLO modelem natrénovaným na manuálně oanotované části datasetu. Zde jsou části hlasivek označeny pomocí obdélníkových bounding boxů vystupujících z algoritmu YOLO (**Obr. 3.2**). V další části jsou tyto nově anotované snímky manuálně zkontrolovány a rozřazeny na úspěšné anotace, které lze přidat do datasetu a neúspěšné anotace. Neúspěšné anotace se nahrají do webové aplikace

Okomentoval(a): [IJ20]: Stačí toto vysvětlení, jak byly zkontrolovány?

LabelStudio a provede se jejich manuální anotace. Využití semi-automatických anotací podstatně zefektivňuje a usnadňuje tvorbu datasetu.



Obr. 3.2~~3.2~~: Interface skriptu pro semi-automatickou anotaci snímků hlasivek

3.1.3 Rozložení datasetu

Pro vytvoření finálního datasetu jsou všechny snímky převedeny do varianty obdélníkových anotací a dataset je rozdělen do 3 skupin.

Trénovací data

Trénovací data obsahují největší část datasetu (cca 80 % všech obrázků). Na těchto datech se model v průběhu tréninku učí a zlepšuje pomocí optimalizace vah a biasů.

Validační data

Validační data (cca 10 % snímků) jsou částí datasetu určenou ke sledování výkonosti modelu během tréninku. Model tyto data nevidí přímo v průběhu učení, tj. neoptimalizuje na nich parametry, ale po ukončení každé epochy na nich jako na nezávislých datech vyhodnotí přesnost modelu. Hlavním důvodem přítomnosti validační části v datasetu je částečné předejití přeučení modelu (viz 3.1.3.1). Pokud se model v průběhu epoch zlepšuje na trénovacích datech, ale predikce validačních dat se zhoršují, trénink modelu je ukončen.

Testovací data

Trénovací data (asi 10 % datasetu) jsou část datasetu, která se během trénování vůbec nepoužívá a slouží až k finálnímu vyhodnocení modelu. Výhodou těchto dat je jejich úplná nezávislost na trénovacím procesu, což umožňuje objektivnější a realističtější hodnocení modelu.

Při tvorbě datasetu se i přes rozdělení videí zachovává příslušnost každého snímku k jeho pacientovi. Následně se dataset dělí do zmíněných 3 skupin přibližným poměrem 80 % – 10 % – 10 % a všechny snímky každého pacienta se vyskytují vždy jen v jedné části datasetu. Toto rozložení zajišťuje, že si model nezapamatuje konkrétní rysy hlasivek určitých pacientů, jejichž snímky by se vyskytli jak v trénovací, tak ve vyhodnocovací části datasetu, ale učí se obecné rysy společné pro každé hlasivkové ústrojí. Zároveň trénování a vyhodnocování modelu probíhá na zcela odlišných pacientech a přináší realističtější výsledky, které lépe simulují reálné použití modelu v praxi na neznámé hlasivky nového pacienta. Toto rozdělení respektující jednotlivé pacienty poskytuje realističtější generalizaci modelu, snižuje možnost přeučení na jednotlivých pacientech a vytváří nenadhodnocené modely, kvůli sdílení informací mezi částmi datasetu, díky čemuž jsou metriky modelu robustnější a důvěryhodnější.

Okomentoval(a): [IJ21]: Přepsáno ze „zvané subject-wise“

3.1.3.1 Přeučení

Přeučení je jev, při kterém dochází k nadměrnému přizpůsobení k trénovacím datům, kdy model vykazuje velmi dobré výsledky na vlastním trénovacím datasetu. Při jeho použití na neznámá data se ale výkonnost modelu razantně zhoršuje. K přeučení dochází především tehdy, pokud architektura modelu obsahuje více parametrů, než kolik je možné vzhledem k poskytnutým datům smysluplně odůvodnit. [32].

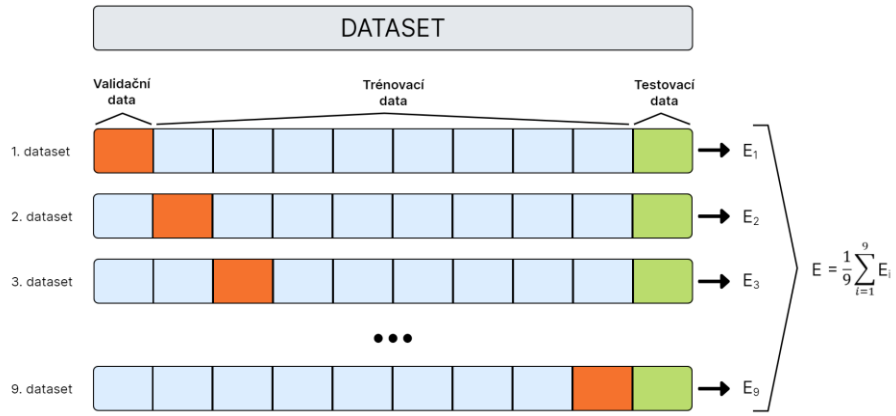
Okomentoval(a): [JI22]: Upraveno

3.1.4 Křížová validace

Při použití rozložení datasetu po pacientech závisí vyhodnocovací metriky (zejména při malém datasetu) na výběru konkrétních pacientů do části validačních a testovacích dat. Pokud by do vyhodnocovací množiny připadly snímky pacientů náročnějších na predikci, výsledné metriky by mohly podhodnotit skutečnou schopnost modelu generalizovat. Naopak při zařazení jednodušších případů by mohly metriky modelu nadhodnocovat výsledky. Pro zajištění objektivního zhodnocení byla výsledná výkonost modelu získaná metodou křížové validace. Při ní je vybrána a oddělena skupina testovacích dat, zbylá část datasetu se rozdělí v tomto případě na 9 podobně objemných částí. Poté se vytvoří devět verzí datasetu, které použijí postupně každou část jako validační data a do trénovacích dat zahrnou zbylých 8 částí (**Obr.**

Okomentoval(a): [IJ23]: Tady bylo „subject-wise rozložení“

3.3). Vznikne tedy 9 obdobných datasetů se stejnými testovacími daty, ale jiným rozložením snímků mezi validační a trénovací částí. Po natrénovaní modelu na každém z těchto datasetů se jako finální výkonost modelu použije metrika spočítaná zprůměrováním jednotlivých výsledků.



Obr. 3.3~~3.3~~: Schéma trénování modelu AI pomocí křížové validace

3.2 Trénování modelu

K vytvoření modelu pro detekci částí hlasivek v obrazech z laryngoskopického vyšetření byl použit algoritmus YOLO, konktrétně různé varianty YOLOv11. Model byl trénován na velikostech YOLOv11n, YOLOv11s, YOLOv11m a YOLOv11l. Pro správné natrénování modelu je třeba vhodně nastavit řadu hyperparametrů, jejichž hodnoty mohou značně ovlivnit výslednou přesnost detekce modelu. Dále jsou ~~v~~ pro ukázku vysvětleny některé ze základních hyperparametrů.

3.2.1 Počet epoch

Jedna epocha představuje kompletní průchod dat a zpětnou propagaci chyb skrze neuronovou síť. Počet epoch tedy udává, kolikrát model projde celým trénovacím datasetem. V průběhu epoch dochází k optimalizaci vah a biasů modelu, který se postupně optimalizuje na míru předloženým datům. Při nastavení nedostatečného počtu epoch model nemá dostatek času naučit se správně reprezentovat data, při nastavení příliš vysokého počtu epoch může docházet k přeučení na trénovací data. V této práci byla zjištěna jako optimální hodnota počtu epoch 300 s přidáním doplňujícího hyperparametrů patience (3.2.2).

3.2.2 Patience

Hyperparametr patience je použit v kombinaci s počtem epoch. Slouží k ukončení tréninku v případě, že se model po určitou dobu nezlepšuje na validačních datech. Parametr pomáhá zabránit přeučení modelu a zároveň šetří výpočetní čas. Pro model hlasivek byla hodnota patience nastavena na 50, trénink tedy byl v případě stagnace výkonosti během posledních 50 epoch ukončen.

3.2.3 Optimalizační algoritmus

Optimalizační algoritmus (též optimalizátor) se stará o průběžnou aktualizaci váhových koeficientů a biasů při zpětné propagaci chyb na základě vypočtené chyby. Výběr vhodného optimalizátoru má zásadní vliv na rychlost konvergence modelu ke správnému řešení i na jeho stabilitu, pro každou úlohu je tedy třeba vybrat co nejvhodnější optimalizační metodu. YOLO umožňuje použití optimalizačních algoritmů, mezi něž patří např. Stochastic~~Scholastic~~ gradient descent (SGD), Root Mean Square Propagation (RMSProp), Adaptive Moment Estimation (Adam), AdamW, což je modifikovaná verze optimalizátoru Adam, který efektivněji pracuje s regularizačním hyperparametrem weight decay.

3.2.4 Rychlost učení

Při každé aktualizaci vektoru vah a biasů je vypočtená změna násobena velikostí hyperparametru rychlosti učení. Tento parametr určuje, jak velkým krokem se mají parametry modelu měnit ve směru vypočítaného gradientu. Nastavení příliš nízké rychlosti učení způsobuje pomalou konvergenci modelu k optimálnímu řešení, což vede k prodloužení tréninkového času, případně ideálního výsledku není dosaženo vůbec. V některých případech se model může ustálit v lokálním minimu ztrátové funkce, které není nejlepším řešením. Naopak příliš vysoká rychlost učení způsobuje velké změny v parametrech modelu, což se projevuje zvýšenou nestabilitou tréninku a model nemusí konvergovat k ideálnímu řešení problému. Hodnota rychlosti učení je během tréninku modelu zakomponována, v závislosti na zvoleném optimalizačním algoritmu, v rovnici pro výpočet aktualizace vektoru vah. V případě této práce se vyskytuje jako konstanta η ve výpočtu vektoru vah pomocí optimalizátoru AdamW (rovnice 3.5).

3.2.5 Batch size

Batch size určuje, kolik vzorků z datasetu se zpracuje najednou při jednom dopředném průchodu neuronovou sítí, než dojde k aktualizaci parametrů modelu. Zpracování celého

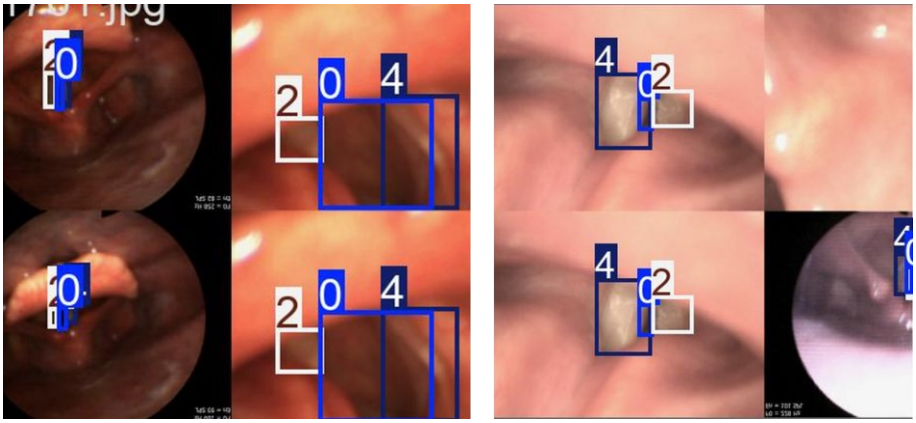
datasetu zároveň by bylo příliš paměťově náročné, zatímco při procházení snímků jednotlivě by se zvyšovala výpočetní náročnost a nestabilita tréninku kvůli častým aktualizacím vah. Batch size představuje kompromis mezi těmito dvěma extrémy, optimalizuje výkon modelu i dobu tréninku v závislosti na dostupném výpočetním hardwaru.

3.2.6 Augmentace

Okomentoval(a): [IJ24]: Dal bych augmentace do teorie

Augmentace dat představuje soubor technik, které slouží k rozšíření a obohacení datasetu aplikací různých transformací na snímky. Mezi obvyklé techniky augmentace patří např. změna měřítka, rotace, horizontální a vertikální zrcadlení, změna jasu, či přidání šumu. Cílem augmentace je zvýšit schopnost modelu generalizovat, tedy naučení modelu správně rozpoznávat objekty i v situacích, které nejsou přesně zastoupeny v datasetu. Tyto úpravy obrázků simulují různorodé reálné podmínky a pomáhají se adaptovat na variabilitu dat v praxi, čímž se také snižuje riziko přeučení modelu.

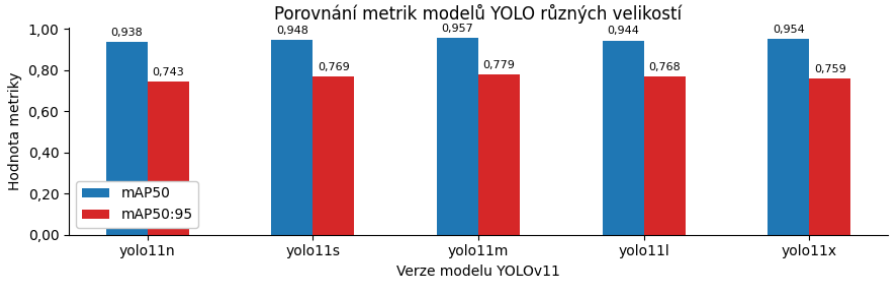
Specifickou formou augmentace dat pro algoritmy YOLO je tzv. mozaiková augmentace. Tato technika náhodně vybírá 4 snímky z trénovacích dat, na které aplikuje běžné augmentační transformace jako oříznutí, změna měřítka atd. a následně je spojí do jednoho obrázku ve formě mřížky o rozměrech 2×2 (**Obr. 3.4**). Mozaiková augmentace zvyšuje robustnost modelu, protože umožňuje trénovat model na nezvyklých kompozicích či při částečném zakrytí objektů, zároveň zvyšuje rozmanitost pozadí. Umožňuje také efektivní trénink i při nutnosti nastavení nižší hodnoty batch size.



Obr. 3.4~~3.4~~: Oanotované snímky modelem YOLO vytvořené pomocí mozaikové augmentace

3.2.7 Testování velikostí YOLO modelů

K vytvoření dobře fungujícího modelu je třeba zvolit vhodnou velikost YOLO modelu, proto bylo vytvořeno 5 modelů všech dostupných velikostí (n, s, m, l a x) YOLO11 a jejich výsledné metriky $@mAP50$ a $@mAP50:95$ byly porovnány na Obr. 3.5~~Obr. 3.5~~. Trénování všech modelů probíhalo na stejném datasetu (viz 3.1) za identického nastavení hyperparametrů. Z grafu je patrné, že výsledné metriky různých velikostí modelu dosáhly velmi podobné úrovně výkonnosti. Všechny testované varianty modelů tedy vykazují schopnost relativně přesně a konzistentně reprezentovat vlastnosti trénovacích dat. Nejlepších hodnot metrik $@mAP50$ a $@mAP50:95$ však dosáhl model YOLOv11m, což odpovídá očekávání, neboť tato velikost modelu představuje rovnováhu mezi složitostí architektury a množstvím dostupných trénovacích dat (přibližně 13 000 snímků). Tento výsledek podporuje hypotézu, že volba modelu přiměřené velikosti vzhledem ~~k~~ ke složitosti problému a datovému objemu je klíčová pro dosažení optimálního výkonu.



Obr. 3.5: Porovnání metrik různých velikostí modelů YOLOv11 spočítaných na testovacích datech~~Obr. 3.5: Porovnání metrik různých velikostí modelů YOLOv11 spočítaných na testovacích datech~~

3.2.8 Volba optimalizačního algoritmu

Při výběru nejvhodnějšího optimalizátoru byly testování optimalizační algoritmy SGD, Adam, AdamW a RMSProp. Jako optimalizační algoritmus pro tuto úlohu byl zvolen AdamW, který poskytoval modely s nejvyšší přesností detekce. Zároveň, při použití toho algoritmu, model konvergoval k optimálnímu řešení nejrychleji v porovnání s jinými testovanými optimalizátory. AdamW optimalizuje váhy modelu následovně. Nejprve je vypočítán gradient ztrátové funkce g_t, z něj je získán první moment gradientu ztrátové funkce m_t v čase t, který je počítán jako průměr aktuálního g_t a předchozího prvního momentu vážený hyperparametrem β_1, který udává, jakou váhu přikládáme minulým gradientům ztrátové funkce (rovnice 3.1).

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné

Naformátováno: Titulek

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.1}$$

Dále je obdobně vypočítán druhý moment ztrátové funkce (anglicky root mean square propagation) v_t v čase t jako vážený průměr druhého momentu gradientu ztrátové funkce a předchozích druhých momentů vážených parametrem β_2, který ovlivňuje vliv předchozích druhých momentů na nově vypočtený (rovnice 3.2).

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.2}$$

V čase $t = 0$ je hodnota m_t i v_t inicializována v 0. Počáteční odhady jsou tedy směrem k nule zkresleny, proto se používá následující korekce (viz rovnice 3.3 a 3.4) potlačující závislost na inicializační hodnotě a dále se pracuje s korigovanými hodnotami prvního a druhého momentu gradientu ztrátové funkce $\widehat{m_t}$ a $\widehat{v_t}$.

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{3.3}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_1^t} \tag{3.4}$$

Z těchto hodnot jsou je vypočten nový vektor optimalizovaných vah modelu θ_{t+1} (rovnice 3.5). Výpočet se skládá ze tří členu, kdy vektor vah z minulé iterace θ_t je pomocí druhého členu posunut k novému optimálnějšímu výsledku (kde ϵ je malá konstanta zajišťující stabilitu dělení). Třetí člen je specifický pro optimalizátor AdamW, kde koeficient λ představuje hodnotu hyperparametru weight decay, který penalizuje příliš vysoké hodnoty vah modelu. Konstanta η značí nastavenou rychlost učení modelu [33].

$$\theta_{t+1} = \theta_t - \eta \frac{\widehat{m_t}}{\sqrt{\widehat{v_t} + \epsilon}} - \eta \lambda \theta_t \tag{3.5}$$

3.2.9 Testování hyperparametrů

Pro dosažení co nejlepší přesnosti modelu bylo testováno velké množství hyperparametrů. Každý z nich byl testován trénováním několika totožných modelů s rozdílem pouze v hodnotě daného parametru. Všechny testy byly prováděny na velikosti modelu YOLO11s a modely byly porovnány na základě nejvyšších dosažených hodnot metrik $@mAP50$ a $@mAP50{:}95$ v průběhu tréninku. Dále jsou uvedeny výsledky testování vybraných hyperparametrů.

Okomentoval(a): [IJ25]: Tady bylo „Výše uvedených“, ale já testoval i výše neuvedené, jen jsem je tam nenapsal aby to nebylo dlouhé.

3.2.9.1 Testování rychlosti učení

Rychlost učení byla testována na hodnotách v rozmezí od 10^{-6} do 10^{-1} viz (**Obr. 3.6a**~~obrázek~~). Závislost rychlosti učení na výkonnosti modelu vyšla podle předpokladu, kdy pro velmi nízké hodnoty trénink probíhal stabilně, ale v průběhu epoch nedocházelo k dostatečnému zlepšování. U vysokých hodnot docházelo k nestabilitě tréninku, což značně projevilo i na metrikách modelů. Nejlepší hodnoty bylo dosaženo při rychlosti učení 0,005.

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné

3.2.9.2 Testování batch size

Hyperparametr batch size byl testován v hodnotách {8; 16; 32; 64}. Používání hodnot, které jsou mocninou dvou, je při trénování modelů strojového pro zpracování obrazu učení běžnou praxí, jelikož umožňuje nejefektivnějšímu využití výpočetních prostředků, zejména GPU. Při trénování modelu je tedy vhodné používat tyto hodnoty pro ideální využití výkonu hardwaru při nastavení co možná nejvyšší hodnoty batch size. Vyšší hodnoty hyperparametru než 64 nebyly testovány kvůli přílišné paměťové náročnosti. Z výsledků (**Obr. 3.6b**~~obrázek~~) je patrné, že nejlepších metrik $mAP@50$ a $mAP@[50:95]$ dosáhnul model trénovaný při hodnotě batch size rovné 16. Očekávaná pozitivní korelace mezi velikostí batch size a kvalitou výsledku se tak nepotvrdila a závislost metrik modelu na hyperparametru batch size se v případě této úlohy neprojevila.

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné

3.2.9.3 Testování rozlišení obrazů

Pro účel trénování modelu upraví YOLO algoritmus všechny snímky z datasetu na stejné rozlišení. Toto rozlišení lze pro jednotlivé modely nastavit pomocí hyperparametru *imgsz*, a obvykle se nastavuje na hodnoty násobků 32, v tomto případě byly testovány hodnoty {320; 640; 960}, které udávají velikost snímku v px v obou rozměrech. Např hodnota 320 tedy značí, že obrazy budou převedeny na rozlišení 320 × 320 px. Při nastavení vyšší hodnoty rozlišení zachytí model jemnější detaily snímku a dochází k vyšší přesnosti detekce. S vyšším rozlišení ale výrazně stoupá paměťová náročnost tréninku. Při testování různých velikostí obrázků dosáhl nejlepších výsledků model s nastavením *imgsz* rovnému 640 (**Obr. 3.6c**). Tento výsledek se neshoduje s předpokladem, že vyšší rozlišení vstupujících trénovacích dat zajišťuje vyšší přesnost modelu. Důvodem výsledku může být fakt, že YOLO modely jsou předtrénovány na výchozí hodnotě rozlišení, kterou je velikost 640 × 640 px. Zároveň část snímků v datasetu sama nedisponuje ani rozlišení 960 × 960 px a pře tréninkem musí být uměle nadvzorkovány.

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné

3.2.9.4 Testování hyperparametru scale

Pro ukázku vlivu augmentačních technik na výsledky modelu byl vybrán hyperparametr scale. Tento parametr simuluje změnu vzdálenosti objektů od kamery tak, že snímek přiblíží či oddálí, přičemž velikosti bounding boxů jsou úměrně přepočteny. Například při nastavení scale na hodnotu 0,5 je měřítko každého vzorku před průchodem skrze CNN vynásobeno náhodně vybranou hodnotou v rozmezí $\langle 0{,}5; 1{,}5\rangle$. Rozměry snímku nastavené parametrem imgsz přitom zůstávají zachovány, takže augmentovaný obraz je buď ořezán nebo doplněn monochromatickým pozadím v závislosti na velikosti násobícího faktoru ~~(~~[34, 35]~~citace dokumentace hyperparametrů i augmentace)~~. V rámci testování byla pokryta celá povolená škála hyperparametru scale, který je v YOLOv11 definován pro hodnoty v intervalu $\langle 0; 1\rangle$. Konkrétně byly zkoušeny hodnoty $\{0{,}00;\ 0{,}25;\ 0{,}50;\ 0{,}75;\ 1{,}00\}$ (~~(~~viz **Obr. 3.6d**~~(obrázek)~~). Nejlepších výsledků dosáhnul model s použitou hodnotou scale ronvou 0,75.

nastavil formátování: Písmo: Tučné
nastavil formátování: Písmo: Tučné

3.2.9.5 Testování hodnoty momentum

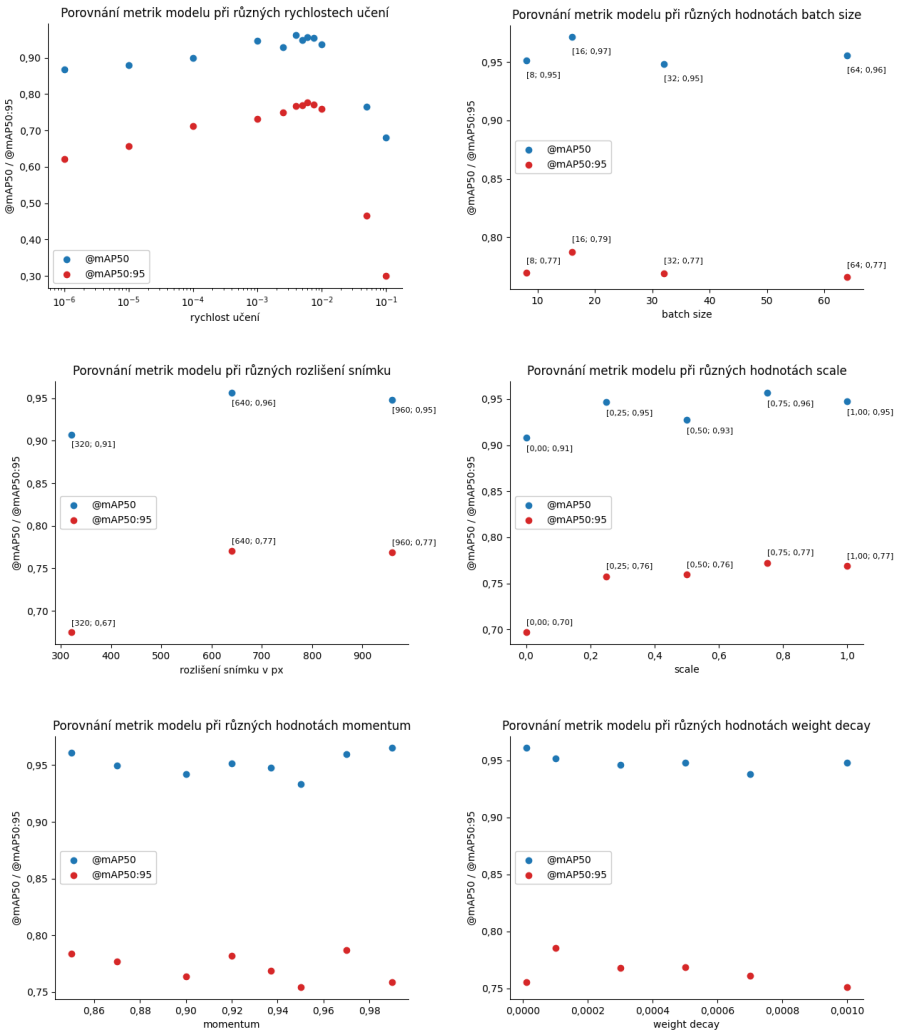
Optimalizace vah modelu při zpětné propagaci je závislá na několika nastavitelných hyperparametrech. Jedním z nich je hodnota mometum, která se při použití optimalizačního algoritmu AdamW vyskytuje v rovnici 3.1 jako koeficient β_1 [36]. Jedná se tedy o váhový koeficient měnící vliv předchozích gradientů ztrátové funkce na nově vypočítané váhy. YOLO modely nastavují momentum na výchozí hodnotu 0,937, z čehož vyplívá, že vliv předchozích grafientů ztrátové funkce na nově vypočítaný není ani desetinový. Testované hodnoty se pohybovaly v rozmezí od 0,87 do 0,99 (**Obr. 3.6e**~~viz obrázek~~). Nejvyšší hodnoty $mAP@[50{:}95]$ bylo dosaženo při momentum rovném 0,99, v případě metriky $mAP@50$ měl nejpřesnější detekční schopnosti model s momentum 0,97.

nastavil formátování: Písmo: Tučné
nastavil formátování: Písmo: Tučné

3.2.9.6 Testování hodnoty weight decay

Weight decay slouží jako hyperparametr při použití optimalizátoru AdamW. Vyskytuje se v posledním členu rovnice 3.5 pro optimalizaci nově vypočítaných vah jako parametr λ. Jeho úlohou je penalizace příliš vysokých vah modelu, čímž se předchází přeučení. Výchozí hodnota tohoto hyperparametru je rovna 0,0005. V podobných velikostech se pohybovaly také testované hodnoty (od 10^{-5} ~~(číslo)~~ do 10^{-3}~~(číslo)~~). Podle výsledků modelů (**Obr. 3.6f**~~obrázek~~) ani několikanásobné změny parametru weight decay nemají výrazný vliv na přesnosti detekce. Za nejvhodnější hodnotu tohoto hyperparametru pro použitý dataset se zdá být 0,0003~~(hodnota)~~, která dosáhla nejlepší metriky $mAP@[50{:}95]$ a nevedla si špatně ani v případě $mAP@50$.

nastavil formátování: Písmo: Tučné
nastavil formátování: Písmo: Tučné



Obr. 3.6: Vliv testovaných hyperparametrů na metriky mAP@50 a mAP@[50:95]. V každém dílčím grafu je měněn jeden hyperparametr. A – rychlost učení, B – batch size, C – rozlišení snímku, D – scale, E – momentum, F – weight decay

nastavil formátování: Písmo: Tučné

Naformátováno: Titulek, Nesvazovat s následujícím

4 VÝSLEDKY A DISKUSE

Experimentální část ~~této~~ práce se skládá ze 2 hlavních částí.

4.1 Příprava datasetu

Pro účely této práce, ale také pro budoucí výzkumné projekty, byl připraven dataset sestavený ze snímků hlasivek pacientů, kteří podstoupili laryngoskopické vyšetření ve Fakultní nemocnici královské Vinohrady.

Dataset zahrnuje záznamy od 20 různých pacientů, a to jak zdravých, tak pacientů s blíže nespecifikovaným postižením v oblasti hlasivkového ústrojí. Délky použitých záznamů se značně liší, po rozdělení videí na jednotlivé obrazy obsahovaly skupiny snímků jednotlivých pacientů od nižších stovek do 2000. Tyto snímky byly anotovány nejprve pomocí nástroje LabelStudio, přičemž byly ve snímcích vyznačeny pravá hlasivka, glotická štěrbina a levá hlasivka (**Obr. 3.1**~~odkaz na obr. 3.1~~). Po dosažení datasetu o přibližně 3500 ručně anotovaných snímcích bylo použito semi-automatických anotací. Pro tento účel byl na dosavadním datasetu natrénován model, který byl použit pro anotování dalších snímků. Ty byly manuálně kontrolovány a rozřazovány na úspěšné a neúspěšné anotace (**Obr. 3.2**). Neúspěšné anotace byly ručně opraveny a společně s ostatními správně anotovanými snímky byly zařazeny do datasetu. Výsledný dataset obsahuje 12991 snímků.

Po rozdělení datasetu na trénovací, validační a testovací data sadu byla zachována příslušnost snímků ke konkrétním pacientům. V případě této úlohy je rozdělení po pacientech variantou, která nejlépe odpovídá reálnému využití datasetu v praxi. Zároveň zabraňuje riziku přeučení modelu, způsobené přidáním snímků stejných hlasivek jednoho pacienta do trénovací i validační části datasetu, k čemuž by mohlo dojít například při náhodném rozdělení snímků mezi skupiny datasetu.

4.2 Trénink modelu

Jako nejvhodnější algoritmus pro detekci hlasivek byl zvolen one-stage algoritmus YOLOv11. Tato aktuálně nejnovější verze modelu YOLO vyniká zejména schopností provádět detekci objektů v reálném čase, přičemž si zachovává vysokou přesnost. Díky těmto vlastnostem patří YOLO mezi nejpoužívanější algoritmy ve zpracování obrazu.

Na stejném datasetu byly otestovány různé velikosti modelu YOLOv11, přičemž všechny dosáhly srovnatelných výsledků (**Obr. 3.5**~~odkaz na obr 3.5~~). Nejvyšších hodnot metrik dosáhl

nastavil formátování: Písmo: Tučné

velmi těsně model YOLO11m. Při výběru velikosti modelu je třeba zohlednit i další faktory, například rychlost vyhodnocení neznámých snímků nebo množství parametrů modelu. Modely YOLO11x či YOLO11l by také mohly nabídnout uspokojivou přesnost detekce, při použití běžně dostupného hardwaru by ovšem model nedosahoval rychlosti použitelné pro vyhodnocování v reálném čase. Nejmenší velikost YOLO11n dosahovala zdaleka nejvyšší detekční rychlosti, ale její nižší počet parametrů se projevil horšími výsledky. Tato velikost je vhodnější pro jednodušší úlohy s menším počtem dat. Dataset o přibližně 13 000 vzorcích již tento model nedokázal adekvátně popsat. Pro výsledný model tedy byl vybrán nejlepší YOLO11m a menší a rychlejší YOLO11s.

Dále byly testovány nejrůznější hyperparametry ovlivňující jak průběh tréninku, tak výslednou přesnost detekce modelu. Podle testování popsaného v 3.2.9 byly jako nejvhodnější zvoleny hodnoty ~~parametrů~~ uvedené v **Tabulka 4.1**~~(Tabulka 4odkaz na tabulku)~~. ~~V t~~Tabul~~ce~~ka ~~jsou~~ zároveň ~~uvedeny~~ obsahuje i další testované či použité hodnoty hyperparametrů, které v kapitole 3.2.9 nebyly uvedeny.~~Tyto~~

~~Tabulka 4: Hodnoty hyperparametrů výsledného modelu~~

Tabulka 4.1: Hodnoty hyperparametrů výsledného modelu

hyperparametr	**hodnota**	**hyperparametr**	**hodnota**
počet epoch	300	optimalizátor	AdamW
patience	50	momentum	0,99
batch size	16	weight decay	0,0003
velikost obrazu	640	mozaiková augmentace	1,0
rychlost učení	0,006	horizontální převrácení	0,25
scale	0,75	vertikální převrácení	0,00

~~…Tabulka parametrů…~~

Testování bylo provedeno na velikosti modelu YOLO11s. Z~~z~~ časových a výpočetních důvodů probíhal trénink modelu na konkrétním rozložení datasetu bez využití metody křížové validace. Optimalizované hodnoty hyperparametrů byly použity ~~pro~~ při finálním trénování modelu jako pro velikost YOLO11s, tak pro YOLO11m.

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: 12 b.

Naformátováno: Titulek, Mezera Za: 10 b., Řádkování: 1,5 řádku, Svázat s následujícím

nastavil formátování: Písmo: Tučné

Naformátováno: Titulek, Mezera Před: 8 b., Svázat s následujícím

nastavil formátování: Písmo: Tučné

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátovaná tabulka

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Text v tabulce

Naformátováno: Text v tabulce, zarovnání na střed

Naformátováno: Mezera Před: 8 b.

4.3 Výsledný model

K vytvoření výsledného modelu pro detekci hlasivek a jeho vyhodnocení bylo využito metody křížové validace, která je popsána v kapitole 3.1.4. ~~Dataset byl rozdělen na 10 podobně objemných skupin. Bylo tedy provedeno 9 tréninků…~~Pro finální výpočet metrik vyhodnocujících přesnost detekce modelu bylo natrénováno 9 modelů, se stejně objemnými datasety, ale rozdílným rozložením vzorků mezi trénovací a validační sadou. Křížová validace byla provedena na velikostech modelu s (**Obr. 4.1**) a m (**Obr. 4.2**).

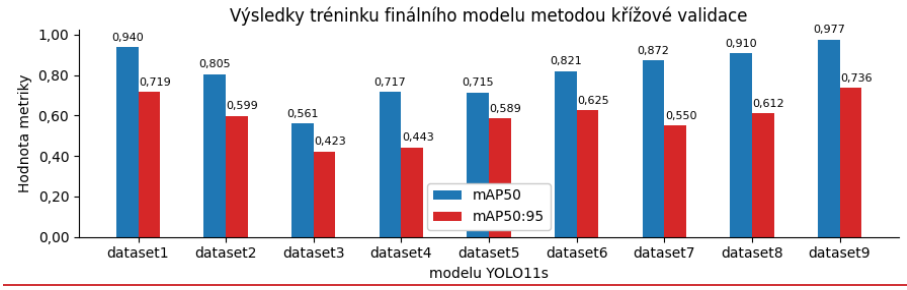
V obou případech jsou výsledky modelů trénovaných na dílčích datasetech velmi odlišné. Tato odlišnost je pravděpodobně způsobena malým množstvím dat a jejich specifickou strukturou. Celý dataset obsahuje pouze 20 pacientů a každý dílčí validační soubor při křížové validaci je tvořen jedním až třemi pacienty. Vzhledem k vysoké variabilitě mezi pacienty (výskytu různých patologických jevů, kvalitě záznamu atd.) má přítomnost určitého pacienta ve validační sadě výrazný vliv na výsledné metriky modelu.

Na ~~grafu~~ **Obr. 4.1** jsou znázorněny výsledné metriky $mAP@50$ a $mAP@[50:95]$ pro dílčí modely YOLO11s natrénované v rámci procesu křížové validace. Hodnoty $mAP@50$ se pohybují v ~~rozpětí~~ rozmezí od 0,561 do 0,977. ~~Přísnější~~, zatímco přísnější metrika $mAP@[50:95]$ ~~vykazuje hodnoty~~dosahuje hodnot mezi 0,423 a 0,736. ~~Výsledkem křížové validace jsou průměrné naměřených metrik získaných~~Průměrné hodnoty těchto metrik byly vypočteny dle ~~podle~~ rovnice 4.1

$$\bar{E} = \frac{1}{9}\sum_{i=1}^{9} E_i \qquad (4.1)$$

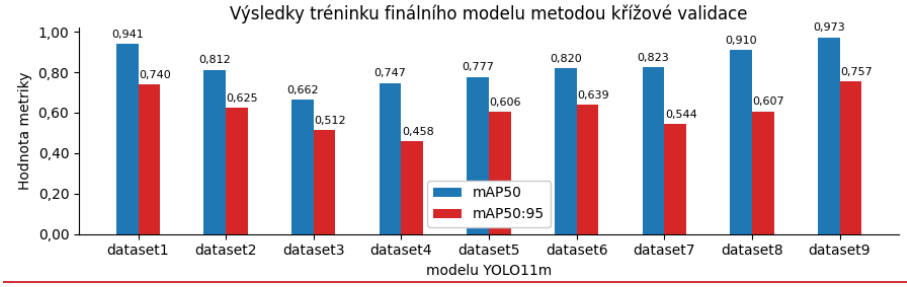
kde $\bar{E}$ značí průměrnou hodnotu metriky a E_i reprezentuje hodnotu metriky pro dílčí model. Výsledné průměrné hodnoty pro model YOLO11s dosáhly $\overline{mAP@50} = 0{,}813$ a $\overline{mAP@[50:95]} = 0{,}588$.

nastavil formátování: Písmo: Tučné
nastavil formátování: Písmo: Tučné
nastavil formátování: Písmo: Tučné
nastavil formátování: Písmo: Tučné
Naformátováno: zarovnání na střed, Mezera Za: 0 b.
Naformátovaná tabulka
Naformátováno: zarovnání na střed
nastavil formátování: Písmo: Kurzíva
nastavil formátování: Písmo: Cambria Math, Kurzíva
nastavil formátování: Písmo: Times New Roman, není Kurzíva
nastavil formátování: Písmo: Cambria Math, Kurzíva
nastavil formátování: Písmo: Cambria Math, Kurzíva
nastavil formátování: Písmo: Cambria Math



Obr. 4.1: Výsledné metriky dílčích modelů YOLO11s získaných pomocí křížové validace

Na **Obr. 4.2** jsou obdobným způsobem zobrazeny výsledky křížové validace modelu YOLO11m. Hodnoty $mAP@50$ se zde pohybují mezi hodnotami 0,662 a 0,973. Průměrná hodnota počítaná dle rovnice 4.1 vychází $\overline{mAP@50} = 0{,}829$. Hodnoty $mAP@[50{:}95]$ se pohybují mezi 0,458 a 0,757, přičemž průměrná hodnota činí $\overline{mAP@[50{:}95]} = 0{,}609$.



Obr. 4.2: Výsledné metriky dílčích modelů YOLO11m získaných pomocí křížové validace

Z výsledků je patrné, že YOLO11m dosahuje lepších výsledků než YOLO11s. Na základě těchto výsledků byl zvolen model YOLO11m s nastavením hyperparametrů uvedeným v **Tabulka 4.1** jako nejvhodnější řešení pro úlohu detekce hlasivek v záznamech z laryngoskopických vyšetření.

Naformátováno: Svázat s následujícím

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Kurzíva

nastavil formátování: Písmo: Cambria Math, Kurzíva

nastavil formátování: Písmo: Cambria Math

Naformátováno: Normální

Naformátováno: Svázat s následujícím

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné

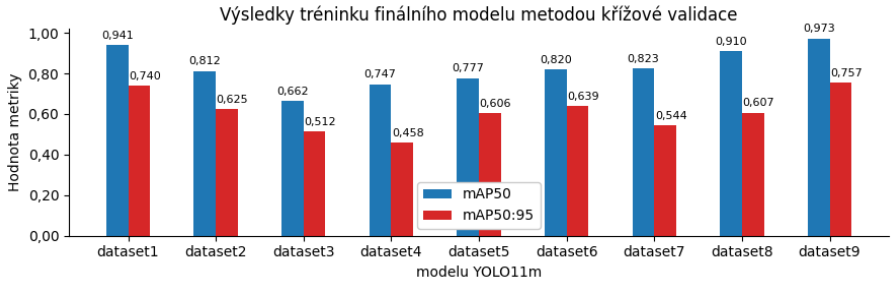
nastavil formátování: Písmo: Tučné

Naformátováno: Normální

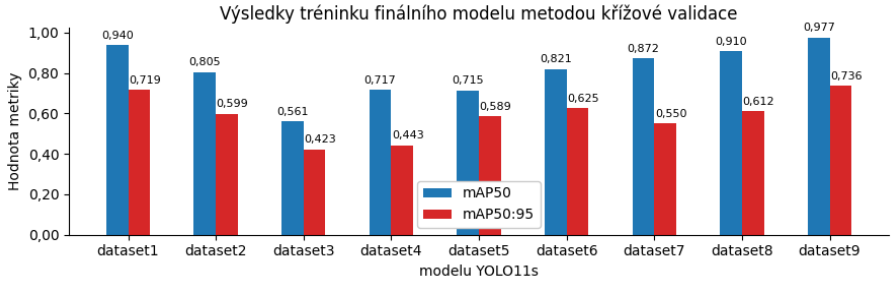
nastavil formátování: Písmo: Tučné

nastavil formátování: Písmo: Tučné

Naformátováno: Normální



Průměr: mAP@50 - 0.8294, mAP@50:95 - 0.6098



Průměr: mAP@50 - 0.8131, mAP@50:95 - 0.5884

5 ZÁVĚR

Cílem práce bylo vytvoření modelu AI schopného rozpoznávat části hlasivek v laryngoskopických záznamech a přesně detekovat jejich polohu pomocí algoritmů využívající CNN.

~~5~~ Pro účel této práce, ale i dalších výzkumných projektů byl připraven dataset složený ze snímků hlasivek pacientů podstupujících laryngoskopické vyšetření. Do tohoto byly zahrnuty záznamy 20 různých pacientů, jak zdravých, tak pacientů trpících určitým postižením v oblasti hlasivkového ústrojí. Délky použitých záznamů se značně liší, po rozdělení videí na jednotlivé obrazy obsahovaly

Naformátováno: Normální, Přístupy klávesou tabulátor: není na 0,76 cm

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) Brownlee, J. *Deep Learning for Computer Vision*

Image Classification, Object Detection, and Face Recognition in Python; Machine Learning Mastery, 2019.
(7) Le, K. *An overview of VGG16 and NiN models*. 2021. https://lekhuyen.medium.com/an-overview-of-vgg16-and-nin-models-96e4bf398484 (accessed.
(8) 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.
(9) 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.
(10) 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.
(11) Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition: 2009.
(12) 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.
(13) 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.
(14) Uijlings, J. R. R.; van de Sande, K. E. A.; Gevers, T.; Smeulders, A. W. M. Selective Search for Object Recognition. *INTERNATIONAL JOURNAL OF COMPUTER VISION* **2012**, *104*, 154-171. DOI: 10.1007/s11263-013-0620-5.
(15) 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.
(16) 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.
(17) 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.
(18) 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.
(19) 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.
(20) 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.
(21) 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.
(22) Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA; 2016.
(23) Zhao, Z.-Q.; Zheng, P.; Xu, S.-t.; Wu, X. Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems* **2019**, 1-21.
(24) 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.
(25) 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.
(26) 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.
(27) Lin, T.-Y.; Maire, M.; Belongie, S. J.; Bourdev, L. D.; Girshick, R. B.; Hays, J.; Perona, P.; Ramanan, D.; Dollar, P.; Zitnick, C. L. Microsoft COCO: Common Objects in Context. *CoRR* **2014**, *abs/1405.0312*.
(28) Shetty, A. K.; Saha, I.; Sanghvi, R. M.; Save, S. A.; Patel, Y. J. A Review: Object Detection Models. In 2021 6th International Conference for Convergence in Technology (I2CT), Maharashtra, India; 2021.
(29) 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.
(30) Padilla, R.; Netto, S. L.; Silva, E. A. B. d. A Survey on Performance Metrics forObject-Detection Algorithms. In 27th International Conference on Systems, Signals and Image Processing (IWSSIP), ELECTR NETWORK; 2020.
(31) *Label Studio*; 2020-2025. https://github.com/HumanSignal/label-studio (accessed 10.4.2025).
(32) Paris, G.; Robilliard, D.; Fonlupt, C. Exploring Overfitting in Genetic Programming. In *Artificial Evolution*, Liardet, P., Collet, P., Fonlupt, C., Lutton, E., Schoenauer, M. Eds.; Springer Berlin Heidelberg, 2004; pp 267--277.
(33) Tian, Y.; Zhang, Y.; Zhang, H. Recent Advances in Stochastic Gradient Descent in Deep Learning. *Mathematics* **2023**, *11*. DOI: 10.3390/math11030682.
(34) Ultralytics. *Data Augmentation using Ultralytics YOLO*. Ultralytics, 2025. https://docs.ultralytics.com/guides/yolo-data-augmentation/ (accessed 2025 21. 5.).

(35) Ultralytics. *Configuration*. Ultralytics, 2025. https://docs.ultralytics.com/usage/cfg/ (accessed 2025 21. 5.).
(36) Loshchilov, I.; Hutter, F. DECOUPLED WEIGHT DECAY REGULARIZATION. In 7th International Conference on Learning Representations, New Orleans; 2019.