ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hodek** Jméno: **Jakub** Osobní číslo: **476084**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídící techniky**

Studijní program: **Automatizační a přístrojová technika**

Specializace: **Automatizace a průmyslová informatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Detekce plovoucích objektů na moři pomocí metod strojového učení

Název diplomové práce anglicky:

Floating object detection in maritime environment using machine learning algorithms

Pokyny pro vypracování:

1. Proveďte rešerši algoritmů strojového vidění s důrazem na detekce objektů na moři
2. Proveďte rešerši existujících řešení na téma detekce objektů na moři, včetně dostupných datasetů
3. Vytvořte vlastní dataset cca 300 obrázků pro validaci implementovaných algoritmů
4. Na základě rešerše vyberte vhodný algoritmus pro detekci objektů na moři, tento algoritmus implementujte
5. Implementovaný algoritmus otestujte a zhodnoťte výsledky

Seznam doporučené literatury:

[1] SONKA, Milan; HLAVAC, Vaclav; BOYLE, Roger. Image processing, analysis, and machine vision. Cengage Learning, 2014.
[2] CORKE, Peter I.; KHATIB, Oussama. Robotics, vision and control: fundamental algorithms in MATLAB. Berlin: Springer, 2011.
[3] BORJI, Ali, et al. Salient object detection: A survey. Computational visual media, 2019, 5: 117-150.
[4] Szegedy, Christian, Alexander Toshev, and Dumitru Erhan. "Deep neural networks for object detection." Advances in neural information processing systems 26 (2013).

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Adam Peichl U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **28.04.0223** Termín odevzdání diplomové práce: **08.06.2023**

Platnost zadání diplomové práce: _____

Ing. Adam Peichl podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	Doplňte jméno děkana k datu 28.04.0223. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Diplomová práce

DETEKCE PLOVOUCÍCH OBJEKTŮ NA MOŘI POMOCÍ METOD STROJOVÉHO UČENÍ

Bc. Jakub Hodek

Fakulta strojní
Ústav přístrojové a řídicí techniky
Vedoucí: Ing. Adam Peichl
15. srpna 2023

České vysoké učení technické v Praze
Fakulta strojní
© 2023 Bc. Jakub Hodek. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě strojní. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení, je nezbytný souhlas autora.

Odkaz na tuto práci: Hodek Jakub. *Detekce plovoucích objektů na moři pomocí metod strojového učení.* Diplomová práce. České vysoké učení technické v Praze, Fakulta strojní, 2023.

Obsah

Poděkování vii

Prohlášení viii

Abstrakt ix

Seznam zkratek x

Úvod 1

1 Algoritmy strojového vidění pro detekci (plovoucích) objektů 3
- 1.1 Algoritmy strojového učení založené na CNN . . . 3
  - 1.1.1 Two-stage detektory . . . 3
    - 1.1.1.1 R-CNN . . . 4
    - 1.1.1.2 Fast R-CNN . . . 4
    - 1.1.1.3 Faster R-CNN . . . 5
  - 1.1.2 One-stage detektory . . . 5
    - 1.1.2.1 YOLO . . . 5
    - 1.1.2.2 SSD . . . 6
- 1.2 Detekce objektů za použití Transformer-ů . . . 7
  - 1.2.1 DEtection TRansformer (DETR) . . . 7
    - 1.2.1.1 Real-Time DEtection TRansformer (RT-DETR) . . . 8

2 Návrh řešení 9
- 2.1 Existující řešení . . . 9
- 2.2 Dataset . . . 10
  - 2.2.1 Existující datasety . . . 12
  - 2.2.2 Příprava datasetů . . . 13
    - 2.2.2.1 Trénovací (a validační) dataset . . . 13
    - 2.2.2.2 Testovací dataset . . . 13
  - 2.2.3 Anotace snímků . . . 14
    - 2.2.3.1 LabelUI - Nástroj pro anotaci s využitím SAM . . . 15
- 2.3 Metriky pro evaluaci algoritmů detekce objektů . . . 16
  - 2.3.1 Pomocné metriky . . . 16
  - 2.3.2 Average Precision (AP) . . . 17
  - 2.3.3 Mean Average Precision (mAP) . . . 18
- 2.4 Implementace . . . 19
  - 2.4.1 Detekce objektů . . . 19
    - 2.4.1.1 YOLOv8-n . . . 19
  - 2.4.2 YOLOv8-m . . . 21
  - 2.4.3 RT-DETR . . . 21
  - 2.4.4 Porovnání modelů . . . 22
    - 2.4.4.1 Možnosti zlepšení . . . 24

3 Závěr **29**

A Přiložené médium - microSD karta **31**

Obsah přiloženého média **37**

Seznam obrázků

1.1 Schéma R-CNN, Zdroj obrázku [4] 4
1.2 Schéma Region Proposal Network (RPN), Zdroj obrázku [21] 6
1.3 Architektura SSD. Zdroj [5] 7
1.4 Feature mapy s default boxy. (a) Obrázek s anotacemi. Zdroj [5] 7
1.5 Architektura DETR. Zdroj: [6] 8
1.6 Architektura RT-DETR. Zdroj: [3] 8

2.1 Ukázka možné podoby řešení typu SEA.AI 10
2.2 Ukázky jednotlivých tříd datasetu 11
2.3 Ukázky jednotlivých tříd datasetu 12
2.4 Ukázky fotografií s anotacemi z testovacího datasetu 14
2.5 LabelUI - Návrh anotace za pomoci SAM podle promptu (aktuální polohy kurzoru myši) 15
2.6 LabelUI - Výsledná anotace 16
2.7 Ilustrace metriky IoU. Zdroj obrázku: [35] 17
2.8 Porovnání variant modelu *YOLOv8*. Zdroj: [39] 19
2.9 Confusion matrix pro natrénovaný model *YOLOv8-n* 20
2.10 Průběh trénování YOLOv8-n 21
2.11 Průběh trénování YOLOv8-m 22
2.12 Průběh trénování RT-DETR 23
2.13 Zvětšené oblasti z anotovaného obrázku a obrázku s detekovanými objekty pomocí natrénovaného modelu RT-DETR 24
2.14 Průběh trénování YOLOv8-m 24
2.15 Porovnání výsledků detekce na testovacím datasetu 25
2.16 Porovnání výsledků detekce na testovacím datasetu 25
2.17 Porovnání výsledků detekce na testovacím datasetu 26
2.18 Porovnání výsledků detekce na testovacím datasetu 26
2.19 Porovnání výsledků detekce na testovacím datasetu 26
2.20 Porovnání výsledků detekce na testovacím datasetu 26
2.21 Porovnání výsledků detekce na testovacím datasetu 26

Seznam tabulek

2.1 Počet anotací jednotlivých tříd trénovacího a validačního datasetu 13
2.2 Počet anotací jednotlivých tříd testovacího datasetu 14
2.3 Porovnání trénovaných modelů - na datasetech se všemi třídami objektů 24

2.4 Porovnání trénovaných modelů - na datasetech s jednou obecnou třídou `obstacle` (překážka) 25

Seznam výpisů kódu

Chtěl bych poděkovat vedoucímu mé diplomové práce Ing. Adamovi Peichlovi za cenné připomínky. Dále bych rád poděkoval mé rodině za podporu v průběhu mého studia.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s použitím informačních zdrojů, které cituji a uvádím v seznamu použité literatury a zdrojů informací.

V Praze dne 15. srpna 2023 .....

Abstrakt

Tato diplomová práce se zabývá detekcí plovoucích objektů na hladině. V první části je provedena rešerše možných algoritmů pro detekci objektů, zejména pomocí metod strojového učení. V druhé části práce je vytvořen trénovací a validační dataset 964 fotografií s 2 864 anotovanými objekty ve 14 třídách, testovací dataset 344 fotografií s 795 anotovanými objekty. Následně jsou, s důrazem na přesnost a možnost provádět detekci v reálném čase bez potřeby vysokého výpočetního výkonu, zvoleny dva modely - YOLOv8 ve dvou variantách a RT-DETR, které jsou natrénovány. Výsledky modelů jsou následně porovnány dle Mean Average Precision (mAP) a rychlosti detekce. Na základě vyhodnocení výsledků byl zvolen RT-DETR pro svoji schopnost detekovat i velmi malé objekty při zachování možnosti detekce v reálném čase.

Klíčová slova detekce objektů, strojové učení, detekce malých objektů, detekce plovoucích objektů, YOLOv8, RT-DETR

Abstract

The subject of this thesis is detection of floating objects on the water surface. The first part provides a review of possible algorithms for object detection, especially using machine learning methods. In the second part, a training and validation dataset of 964 photos with 2 864 annotated object in 14 classes, test dataset of 344 with 795 annotated objects. Then, with emphasis on accuracy and the ability to detect objects real-time without a need for high computing power, two models were selected - YOLOv8 in two variants and RT-DETR, which were trained. The results were then evaluated according to Mean Average Precision (mAP) and detection speed. Based on the evaluation, RT-DETR was selected for its ability to detect really small objects while maintaining performance for a real-time detection.

Keywords object detection, machine learning, small objects detection, floating objects detection, YOLOv8, RT-DETR

Seznam zkratek

SAM	Segment Anything Model [1]
RGB	Red, Green, Blue
YOLO	You Look Only Once [2]
RT-DETR	Real Time DEtection TRansformer [3]
CNN	Convolutional Neural Network [4]
R-CNN	Region-based Convolutional Neural Network [4]
RPN	Region Proposal Network [**"cite –FasterRCNN"**]
SSD	Single Shot Detector [5]
DETR	Detection Transformer [6]
FFN	Feed Forward Network [6]
AIFI	Intrascale feature intraction[3]
CCFM	Cross-scale feature-fusion module[3]
LiDAR	Light Detection and Ranging [7]
Radar	Radio Detection and Ranging [8]
NIR	Near Infrared [9]
LWIR	Long Wave Infrared [9]
NMEA	National Marine Electronics Association [10]
SUP	Stand-up paddleboard
UI	User Interface
TP	True Positive
FP	False Positive
FN	False Negative
P	Precision
R	Recall
ČJ	Český jazyk
IoU	Intersection over Union
BB	Bounding Box
AP	Average Precision [11]
mAP	Mean Average Precision [11]
CPU	Central Processing Unit
RAM	Random Access Memory
GPU	Graphics Processing Unit
GB	Giga Byte
AIS	Automatic Identification System

Úvod

Detekce plovoucích objektů je užitečná zejména pro předcházení kolizím na moři. Využít ji lze k prevenci kolize mezi plavidly nebo kolize plavidla s neidentifikovaným plovoucím objektem - například ztraceným lodním kontejnerem nebo vyplaveným kmenem stromu. To může, zejména u menších plavidel, způsobit vážné následky. Dále nalezne uplatnění například v rámci záchranných operací při vyhledávání osob. [12, 13]

Vzhledem k variabilní velikosti detekovaných objektů – od velmi malých jako je bóje nebo osoba ve vodě až po velikost nákladní lodi, proměnlivým světelným podmínkám, které jsou závislé například na počasí a denní době, a vlnám, které mohou způsobit falešné detekce, je tak spolehlivé rozpoznávání objektů netriviálním úkolem.

Cílem této práce je nalézt, implementovat a otestovat vhodný algoritmus strojového učení pro detekci plovoucích objektů, zejména v kontextu použití na menších plavidlech – tedy je třeba uvažovat o omezeném výpočetním výkonu a omezené spotřebě energie. Zároveň by algoritmus měl být schopen pracovat v reálném čase. Předpokládá se použití za denního světla, což umožňuje k akvizici dat využívat RGB kameru, která je běžnou součástí mobilního telefonu. Je tak možné snáze pořídit vhodná data pro testovací dataset a také nalézt vhodné, volně dostupné fotografie, pro trénovací a validační dataset.

Práce je rozdělena na dvě hlavní části. V první kapitole je provedena rešerše základních algoritmů pro detekci objektů založených zejména na strojovém učení. Algoritmy jsou rozděleny na dvě skupiny, a to na algoritmy založené primárně na konvolučních neuronových sítích a na algoritmy využívající v počítačovém vidění relativně nový (počátky v r.2020) koncept *transformerů* [14, 15].

Na začátku druhé části je provedena rešerše již exitujících řešení a vhodných datasetů následovaná vytvořením vlastních datasetů. Dále jsou zvoleny algoritmy *YOLOv8* ([16]) a *Real Time DEtection TRansformer (RT-DETR)* ([3]), které jsou implementovány a otestovány na vytvořených datasetech.

Kapitola 1

Algoritmy strojového vidění pro detekci (plovoucích) objektů

1.1 Algoritmy strojového učení založené na CNN

Konvoluční neuronová sít (CNN)

Konvoluční neuronová síť je speciálním případem neuronové sítě, která vyniká ve zpracování dat organizovaných do struktury mřížky – zejména tedy obrazových dat (2-D) nebo časových řad (1-D) [17].

Konvoluční vrstva

Od plně propojených vrstev neuronových sítí, kde je každý neuron ve vrstvě propojen se všemi neurony v následující vrstvě, se konvoluční vrstvy liší tím, že na vstupní data (propojení s předchozí vrstvou) aplikují filtry, které je možné trénovat. Aplikací filtrů se snižuje počet parametrů neuronové sítě. Zároveň se CNN učí lokální příznaky, které extrahuje do svého výstupu, kterým je tzv. *mapa příznaků (feature map).*[17]

Pooling vrstva

Pooling vrstva je vrstva, která sumarizuje oblasti vstupních dat do jejích výstupních elementů. Obvykle následuje za konvoluční vrstvou a zmenšuje rozměr mapy příznaků. [17]

Kombinací konvolučních vrstev a pooling vrstev, vznikne CNN, která je schopná detekovat komplexní vzorce [17]. Je tak vhodným základem pro algoritmy dektekce objektů. Můžeme je rozdělit do dvou hlavních skupin. Konkrétně poté na *one-stage* (jednostupňové) a *two-stage* (dvoustupňové) detektory. One-stage detektory jsou obecně rychlejší, nicméně však mohou dosahovat horších výsledků v porovnání s two-stage detektory.[18]

1.1.1 Two-stage detektory

Two-stage detektory, jak je již z jejich názvu patrné, se obecně skládají ze dvou hlavních částí: [18]

1. Nalezení oblastí, kde se pravděpodobně nachází nějaký objekt (*region proposals* nebo *object proposals*[19])[18] a jejich případné předzpracování. [4]
2. Detekce přítomnosti objektu v dané oblasti, respektive jeho klasifikace. [18]

1.1.1.1 R-CNN

R-CNN neboli *Region-based Convolutional Neural Network* je prvním z two-stage detektorů[18], publikovaný v r. 2014.[4]

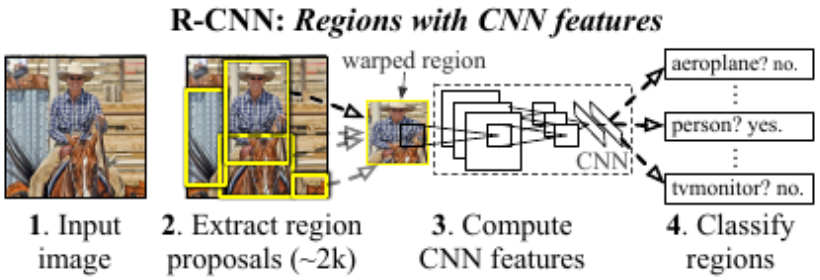
V rámci two-stage detektorů je nejpomalejším a jeho mAP je obdobné jako u následujících detektorů.[20, s. 2]

Princip

Prvním krokem R-CNN je nalezení tzv. ***region proposals*** - tedy oblastí, ve kterých se pravděpodobně nachází nějaký objekt. Samotná R-CNN je na volbě algoritmu pro nalezení region proposals nezávislá. V původním článku je použit algoritmus *Selective Search*.

Druhým krokem je **extrakce příznaků** (feature extraction) pro každý z nalezených region proposal. Každý region proposal je pro zachování alespoň části "kontextu"zvětšen o p pixelů ve všech směrech ([4] používá $p = 16$), převeden na rozměr $227 \times 227px$ (může tedy být zdeformován) a normalizován odečtením průměrné hodnoty pixelů. Následně je použit jako vstup pro konvoluční neuronovou síť (CNN), jejíž výstupem je vektor příznaků (anglicky *feature vector*) o rozměru 1×4096. [4] Tento proces se opakuje pro všechny region proposals, což dělá R-CNN velmi pomalým.[19] Struktura použité CNN je následující:[4]

- 5 konvolučních vrstev (convolutional layers)
- 2 plně propojené vrstvy (fully-connected layers)



Obrázek 1.1 Schéma R-CNN, Zdroj obrázku [4]

Posledním krokem je klasifikace nalezených region proposals, kterou autoři [4] řeší pomocí *Support Vector Machine* (SVM) klasifikátoru.[4]

1.1.1.2 Fast R-CNN

Fast R-CNN je vylepšením R-CNN. Zlepšení - primárně zrychlení trénování a inference (dle [19] přibližně 10× oproti R-CNN, dle [20] až 25×) - je dosaženo tím, že

- Extrakce příznaků (feature extraction) je prováděna pouze jednou pro celý obrázek, nikoliv pro každý region proposal zvlášť.[19]
- Nepotřebuje využívat pevný disk pro ukládání výstupů z CNN pro každý region proposal zvlášť.[19]

Také díky použití *multi-task loss* funkce je možné Fast R-CNN trénovat v jednom kroku a ne každou komponentu zvlášť, jako u R-CNN.[19]

Princip

Vstupem do Fast R-CNN je obrázek a jeho region proposals. Vstupní obrázek je předzpracován několika konvolučními vrstvami (convolutional layers) a max-pooling vrstvou (max-pooling layer), čímž je získána tzv. *feature map* (mapa příznaků). [19]

Následně je pro každý region proposal pomocí *RoI pooling* vrstvy (Region of Interest pooling layer) vytvořen vektor příznaků (*feature vector*) o fixním rozměru.[19]

Každý vektor příznaků (feature vector) je následně předán plně propojeným vrstvám (fully-connected layers), které se větví do dvou výstupních větví - jedna pro klasifikaci a druhá pro regresi bounding boxu.[19]

- Větev pro klasifikaci využívá soft-max aktivační funkce a jejíchž výstupem je vektor pravděpodobností výskytu jednotlivých tříd a "pozadí"(tj. obrázek neobsahuje žádný objekt k detekování).[19]
- Větev pro regresi bounding boxu pro každou třídu objektů vypočítává 4 hodnoty, které určují pozici a velikost bounding boxu.[19]

1.1.1.3 Faster R-CNN

Faster R-CNN je opět vylepšením Fast R-CNN. Zlepšení je dosaženo nahrazením algoritmu pro nalezení region proposals (v R-CNN a Fast R-CNN je použit algoritmus *Selective Search*) tzv. *Region Proposal Network* (RPN). [21]

Region Proposal Network

Region Proposal Network (RPN) je konvoluční neuronová síť, která využívá několik konvolučních vrstev s detekční sítí Fast R-CNN. Vstupem do RPN je mapa příznaků (feature map) získaná extrakcí příznaků (feature extraction) pomocí konvolučních vrstev sdílených s Fast R-CNN. [21]

Následně je mapa příznaků zpracována metodou *sliding window* (posuvné okno) o rozměru $n \times n$ (autoři [21] používají $n = 3$), kdy je pro každou pozici posuvného okna snížena dimenze vektoru "vybrané"oblasti (na 512 v případě použití VGG nebo 256 v případě ZF). Ten je předán $n \times n$ konvoluční vrstvě následovanou ReLU aktivační funkcí a rozdělením na dvě větve, které jsou fully connected vrstvami s velikostí výstupního vektoru závislého na počtu *anchor boxů* k:[21]

- **Box-classification layer** - vrstva, jejíž výstupem je vektor pravděpodobností, že v daném anchor boxu se nachází objekt nebo pozadí (tj. neobsahuje žádný objekt k detekování). Výstupní rozměr této vrstvy je $2 \cdot k$[21]
- **Box-regression layer** - vrstva, jejíž výstupem jsou 4 hodnoty, které určují pozici a velikost detekovaného bounding boxu vzhledem k *k-tému* anchor boxu na aktuální pozici posuvného okna. Výstupní rozměr této vrstvy je tedy $4 \cdot k$.[21]

Autoři [21] používají $k = 9$ anchor boxů - 3 různé velikosti a 3 různé poměry stran.[21]

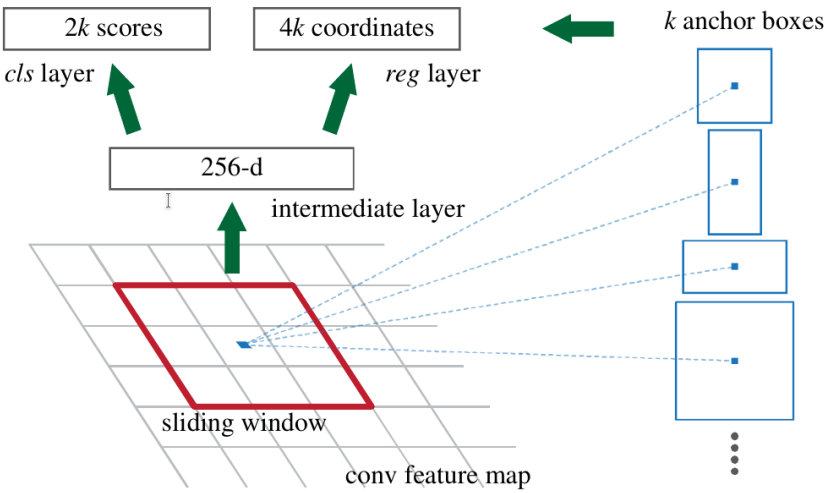
Princip

Princip fungování Faster R-CNN je obdobný jako u (Fast) R-CNN, avšak místo použití algoritmu *Selective Search* pro nalezení region proposals je použita RPN.

1.1.2 One-stage detektory

1.1.2.1 YOLO

YOLO (You Only Look Once) je konvoluční neuronovou sítí jejíž vstupem je celý obrázek a výstupem jsou tzv. *bounding boxy* s příslušnými pravděpodobnostmi, že se v nich nachází nějaký objekt. Výhodou YOLO je jeho rychlost a fakt, že tím, že vstupem je vždy celý obrázek, je zachován kontext. Méně tedy zaměňuje pozadí s objektem v porovnání s Fast R-CNN.[2]



Obrázek 1.2 Schéma Region Proposal Network (RPN), Zdroj obrázku [21]

Princip (YOLOv1)

Vstupem YOLO je obrázek, který je v zápětí rozdělen na $S \times S$ čtvercových buněk. Pro každou buňku je predikováno B bounding boxů. Pro každý bounding box je poté predikováno pět hodnot: P_c, x, y, w, h, kde Pc pravděpodobnost, že se v něm nachází nějaký objekt a případně, jak moc se bounding box pravděpodobně překrývá s objektem. Dále (x, y) jsou souřadnice středu bounding boxu relativní vzhledem k aktuální buňce, dále jeho šířka w a výška h relativní k celému obrázku. Nezávisle na počtu bounding boxů jsou pro každou buňku predikováno C pravděpodobností, že se v ní nachází objekt z C tříd. [2, 22]. Výstupem je tedy vektor o rozměru $S \times S \ \times (5 \cdot B + C)$. [2]

YOLOv8

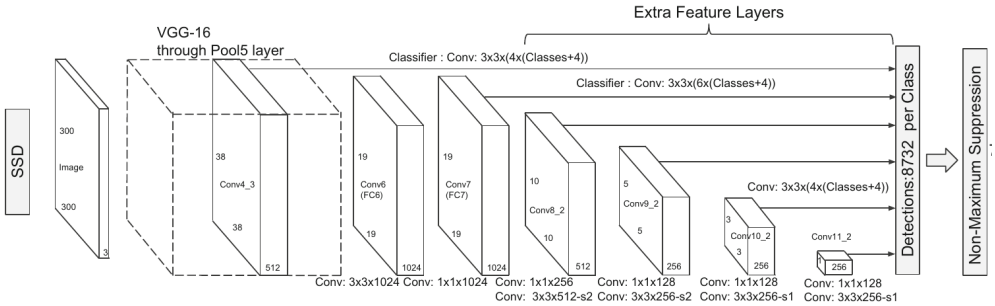
Osmá verze YOLO [16] přináší několik vylepšení oproti předchozím verzím. Jedná se zejména o *anchor-free model* s oddělenými výstupními vrstvami (*head*) pro nezávislou predikci přítomnosti objektu (*objectness*), klasifikaci a regresi bounding boxu. Dále je přidán *C2f (cross-stage partial bottleneck s 2 konvolučními vrstvami)* blok, který pomáhá kombinovat *high-level* příznaky s informacemi o kontextu. [22]

1.1.2.2 SSD

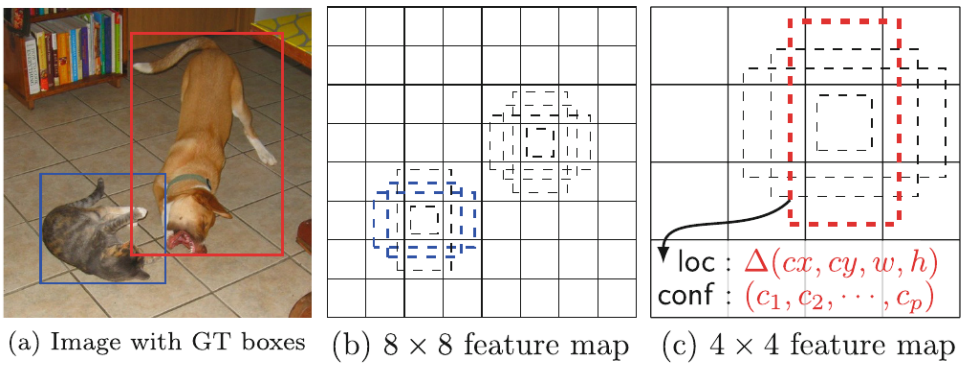
SSD (Single Shot Detector) je konvoluční neuronovou sítí, která využívá *multi-scale feature maps* pro detekci objektů různých velikostí. SSD může využívat pro extrakci příznaků libovolnou konvoluční neuronovou síť - tzv. *base network*, autoři [5] používají VGG-16.[5]

Princip

Vstupem je obrázek konstantního rozměru, který je předán tzv. *base networku* (VGG-16) jejíž výstupem je *feature mapa*. Pro každou pozici ve feature mapě je vygenerována množina výchozích bounding boxů (viz Obrázek 1.4 (b) a (c)), tzv. *default boxů*, pro které je následně predikován offset (upravení pozice tak, aby co nejlépe odpovídal detekovanému objektu) a pravděpodobnosti pro každou jednotlivou třídu (viz Obrázek 1.4 (c)) Dále následují další konvoluční vrstvy a pro některé z nich je proces opakován.[5]



Obrázek 1.3 Architektura SSD. Zdroj [5]



Obrázek 1.4 Feature mapy s default boxy. (a) Obrázek s anotacemi. Zdroj [5]

Nakonec jsou všechny default boxy předány *non-maximum suppression* algoritmu, který odstraní překrývající se bounding boxy s nižší pravděpodobností. [5]

1.2 Detekce objektů za použití Transformer-ů

Transformer

Transformer původně pochází ze strojového zpracování textu. Jeho úkol je transformace vstupní sekvence do výstupní, tak aby extrahoval relevantní informace. [14] Je založen na tzv. *self-attention* mechanismu, který umožňuje v případě zpracování textu "seskupit" slova, která spolu významově souvisí. Obdobný proces je možné provést i v případě obrazových dat (převedených do formy vektoru), kde je tak možné seskupit pixely, které přísluší stejnému objektu. [14, 15]

Skládá se z tzv. *encoderu* a *decoderu*. Encoder se skládá z self-attention mechanismu následovaného neuronovou sítí. Decoder se skládá opět z bloku self-attention mechanismu následovaného neuronovou sítí. [14] Nicméně transformer ze svého principu nemá žádný kontext, proto se přidává tzv. *positional encoding*, který zajišťuje, že model bude znát pozici daného *embeddingu* (vektor číselné reprezentace slova / části obrázku atd.) v rámci sekvence. [14]

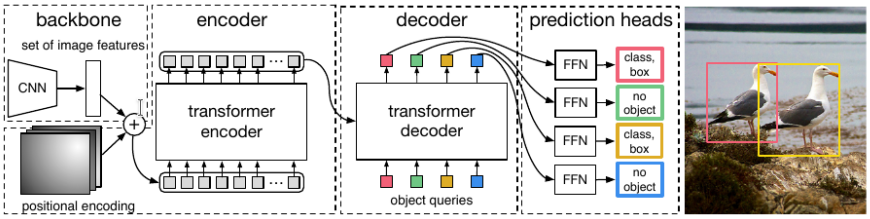
1.2.1 DEtection TRansformer (DETR)

DEtection TRansformer (DETR) je architekturou založenou na transformerech pro detekci objektů. Oproti ostatním algoritmům pro detekci objektů je jeho princip výrazně jednodušší -

nezávisí na žádném algoritmu pro nalezení region proposals, anchor boxů nebo žádné jiné heuristice. [6]

Princip

Vstupem je obrázek, ze kterého je získána *mapa příznaků* (*feature map*) pomocí tzv. *backbone* (konvoluční neuronová síť). Ta je následně spolu s *position embedding-em* předána transformeru, respektive jeho encoderu. Jeho výstup (tzv. *object queries*) je následně předán do dekodéru, jehož výstup je předán neuronové síti (FFN - Feed Forward Network), která zajišťuje pomocí softmax funkce klasifikaci třídy a predikci bounding boxů (normalizované souřadnice jeho středu, výšku a šířku). [6]



Obrázek 1.5 Architektura DETR. Zdroj: [6]

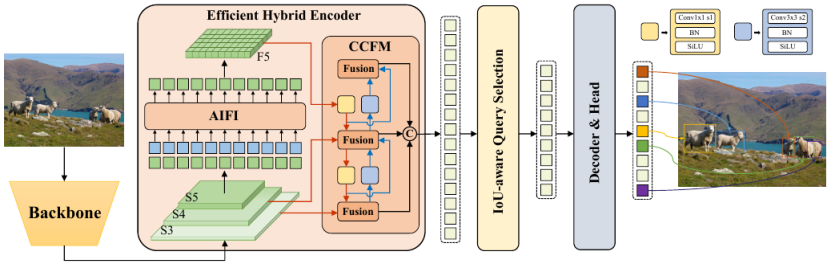
1.2.1.1 Real-Time DEtection TRansformer (RT-DETR)

RT-DETR je vylepšenou verzí DETR, která je optimalizovaná pro detekci objektů v reálném čase.

Dle [3] má DETR dvě nevýhody:

- Pomalá konvergence při trénování
- Obtížné trénování/optimalizace *object queries*

Autoři [23] uvádí, že enkodér samotný poté z celkového výpočetního času zabírá 49%. Algoritmus RT-DETR tedy přichází s efektivnějším enkodérem, kteří autoři [3] nazývají jako *hybridní encoder*. Jako svůj vstup hybridní encoder využívá poslední tři vrstvy z backbone, jejichž výstupy následně transformuje pomocí *AIFI* a *CCFM* do sekvence příznaků, které jsou poté předány *IoU aware (object) Query Selector-u*, který zajišťuje výběr fixního počtu object queries, které jsou dále předány dekodéru, jako u DETR. [3]



Obrázek 1.6 Architektura RT-DETR. Zdroj: [3]

Podle testování autorů [3] dosahuje RT-DETR lepších výsledků než YOLOv8, přičemž vykazuje i vyšší rychlost inference. [3]

Kapitola 2
Návrh řešení

2.1 Existující řešení

Detekci plovoucích objektů na moři je možná provést na základě dat získaných z různých senzorů, jakými jsou například:

- LiDAR (Light Detection and Ranging) [7]
- Radar (Radio Detection and Ranging) [8]
- RGB kamera [8]
- Near Infrared (NIR) kamera [9]
- Long Wave Infrared (LWIR) kamera [9]
- Případně kombinací výše uvedených [7, 24]

Kompletní, již existující řešení, které řeší detekci plovoucích objektů a následně i odhad vzdálenosti objektu, bylo nalezeno pouze jedno, a to uzavřené komerční - SEA.AI [24].

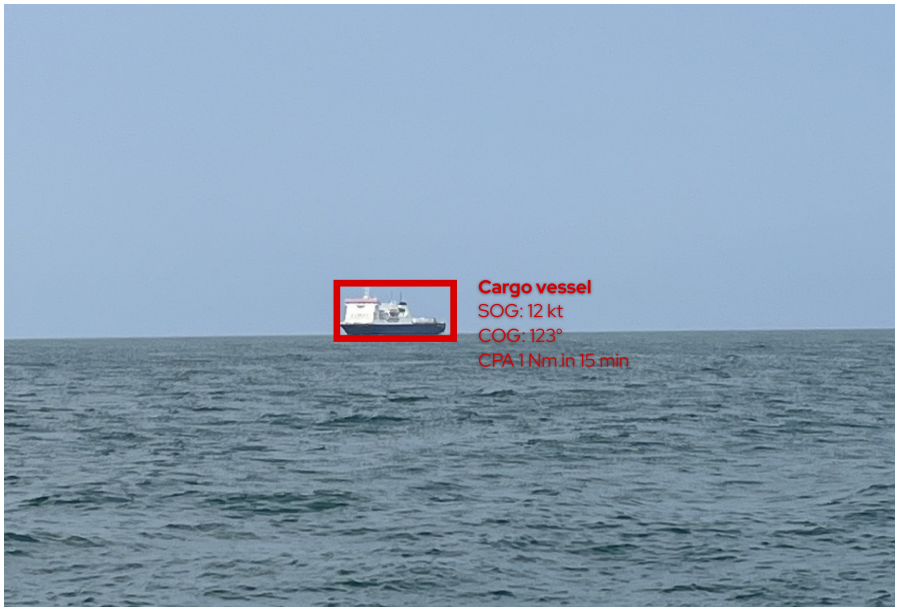
SEA.AI

Společnost SEA.AI (dříve OSCAR) nabízí komerční řešení pro detekci a sledování plovoucích objektů na moři určené k montáži na plachetnice (jak rekreační, tak regatové - závodní), motorové i nákladní lodě. Vzhledem k uzavřenému řešení nebylo možné zjistit přiliš technických detailů. Je však známo, že pro akvizici snímků používájí "klasickou"RGB kameru v kombinaci s termokamerou. Jejich rozlišení se liší v závislosti na zvoleném modelu. Pro samotnou detekci objektů využívají blíže nespecifikovaných algoritmů strojového vidění a hlubokého učení.[24]

Firma uvádí, že jejich systém je schopen detekovat objekty jako *ostatní plavidla, obecně plovoucí překážky, (ztracené) kontejnery, bóje, nafukovací plavidla, kajaky a dokonce i osoby na hladině.*[24]

Nejvyšší řada jejich produktu umožňuje i sledování detekovaných objektů a monitoring celého okolí plavidla,[24] což může být vhodné například pro nákladní lodě plující v oblastech s rizikem pirátství nebo pro zvýšení bezpečnosti zakotvených plavidel.

Největší výhodou jejich řešení je, že díky využití termokamery funguje i v noci a za zhoršené viditelnosti.[24] Dále poskytují konektivitu v rámci interní sítě plavidla za použití komunikačního standardu *NMEA 2000*[10], který umožňuje integraci s ostatními zařízeními na palubě - jak mobilními zařízeními, tak chart-plottery od výrobců jako B&G, Garmin, Raymarine a další.[25]



Obrázek 2.1 Ukázka možné podoby řešení typu SEA.AI

2.2 Dataset

Pro detekci bylo identifikováno *14 kategorií (tříd)* objektů, které je cílem detekovat:

1. *Plachetnice* (`sailboat`)
2. *Motorová loď* (`motorboat`)
3. *Bóje* (`buoy`)
4. *Nafukovací motorový člun* (`inflatable_boat`)
5. *Nákladní loď* (`cargo_vessel`)
6. *Plovoucí překážka* (`floating_obstacle`) - například kmen stromu, který pluje na hladině
7. *Trajekt* (`ferry_boat`)
8. *Cruise ship* (`cruise_ship`)
9. *Kajak* (`kayak`)
10. *Těžební plošina* (`oil_rig`)
11. *Ostatní plavidla* (`other_vessel`)
12. *Stand-up paddleboard (SUP)* (`sup`)
13. *Plavec / osoba na hladině* (`swimmer`)
14. *Ostatní nezařaditelné (nafukovací) plovoucí objekty* (`inflatable_whatever`)

cargo_vessel

ferry_boat

cruise_ship

kayak

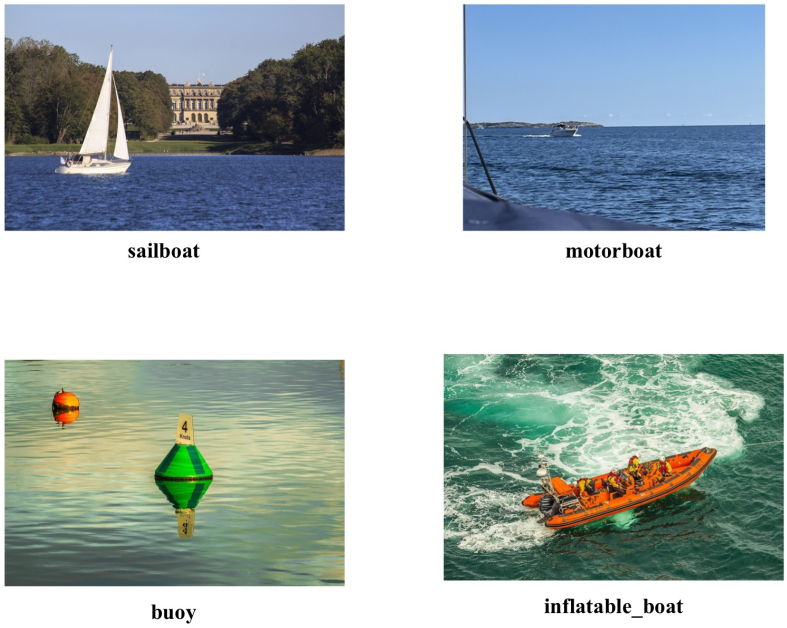
oil_rig

other_vessel

sup

swimmer

Obrázek 2.2 Ukázky jednotlivých tříd datasetu



Obrázek 2.3 Ukázky jednotlivých tříd datasetu

2.2.1 Existující datasety

V rámci rešeršní části bylo nalezeno několik volně dostupných datasetů obsahujících fotografie plovoucích objektů.

- Kaggle - Boat types regonition[26]

  Tento dataset obsahuje přibližně 1500 obrázků. Jedná se zejména o lodě, které jsou roztříděné celkem do 9 tříd. Bohužel 2 třídy (`gondola` a `paper_boat`) jsou pro účel této diplomové práce nepotřebné. Dataset dále obsahuje fotografie bójí, nákladních lodí, nafukovacích motorových člunů, plachetnic, kajaků, trajektů a cruise ships, které je z části možné použít.

- Kaggle - Vessel identification[27]

  Tento dataset o přibližně 1200 obrázcích má třídy stejné jako [26] - tedy celkem 9 tříd, z nichž je použitelných 7. Opět se jedná o dataset určený pro klasifikaci a je potřeba jednotlivé objekty anotovat.

- SeaShips [28]

  Tento dataset obsahuje přibližně 7000 obrázků různých druhů nákladních lodí získaných z kamery umístěné na břehu u blíže nespecifikovaného průplavu. Z tohoto konkrétního datasetu je možné vybrat některé snímky.

- Singapore Maritime Dataset [29]

  Jedná se o videa z pobřeží a z lodí v okolí Singapuru. Obsahuje i snímky pořízené NIR (Near IR) kamerou.

2.2.2 Příprava datasetů

Pro účely této diplomové práce jsou vytvořeny dva datasety. První, určený k trénování a validaci v průběhu trénování a druhý – testovací dataset, určený výhradně pro vyhodnocení a porovnání natrénovaných modelů.

2.2.2.1 Trénovací (a validační) dataset

Dataset použitý pro učení je sestaven z volně dostupných fotografiích:

- Kaggle - Boat types regonition[26] - dataset pro klasifikaci (CC0)
- Kaggle - Vessel identification[27] - dataset pro klasifikaci (CC BY 4.0)
- Fotobanka *unsplash.com*[30] (Unsplash License)
- z části Singapore Maritime Dataset [29]

Fotografie byly následně anotovány. Jejich anotace (manuální) byla provedena za pomoci anotačního nástroje, který je popsán v sekci 2.2.3.1. Zmíněný nástroj byl vytvořen pro účely této diplomové práce. Zastoupení jednotlivých tříd v trénovacím datasetu je uvedeno v Tabulce 2.1.

Třída	**Počet anotací**
Plachetnice	860
Motorová loď	290
Bóje	480
Nafukovací motorový člun	82
Nákladní loď	135
Plovoucí překážka	77
Trajekt	110
Cruise ship	226
Kajak	434
Těžební plošina	33
Ostatní plavidla	91
Stand-up paddleboard (SUP)	15
Plavec / osoba na hladině	20
Ostatní nezařaditelné (nafukovací) plovoucí objekty	11
Celkem	**2 864**

Tabulka 2.1 Počet anotací jednotlivých tříd trénovacího a validačního datasetu

Tento dataset byl náhodně v poměru 80:20 rozdělen na trénovací a validační.

2.2.2.2 Testovací dataset

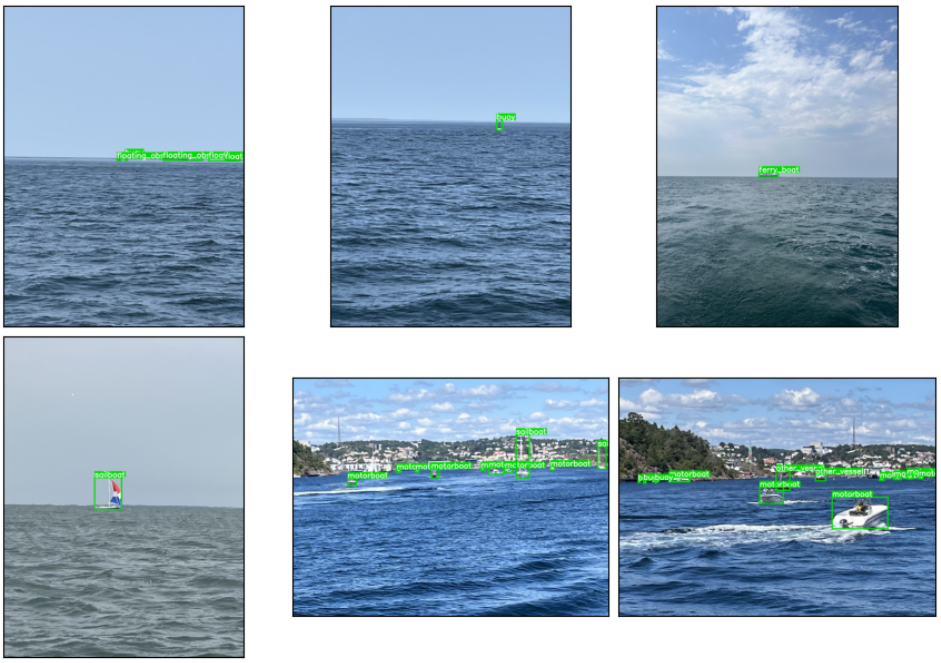
Fotografie pro validační dataset byly pořízeny v reálném prostředí, pomocí mobilního telefonu.

Celkem bylo pořízeno **344 fotografií**, které byly následně manuálně anotovány pomocí LabelUI (viz sekce 2.2.3.1).

Bohužel vzhledem k omezeným možnostem pořízení fotografií v reálném prostředí, nebylo možné pořídit fotografie všech tříd. Zastoupení jednotlivých tříd v datasetu je uvedeno v Tabulce 2.2. Dataset obsahuje fotografie jak z prostředí příbřežních vod, tak volného moře.

Třída	Počet anotací
Plachetnice	115
Motorová loď	329
Bóje	94
Nafukovací motorový člun	36
Nákladní loď	138
Plovoucí překážka	5
Trajekt	10
Cruise ship	8
Kajak	-
Těžební plošina	5
Ostatní plavidla	55
Stand-up paddleboard (SUP)	-
Plavec / osoba na hladině	-
Ostatní nezařaditelné (nafukovací) plovoucí objekty	-
Celkem	**795**

Tabulka 2.2 Počet anotací jednotlivých tříd testovacího datasetu



Obrázek 2.4 Ukázky fotografií s anotacemi z testovacího datasetu

2.2.3 Anotace snímků

V době počátku anotování fotografií byl společností Meta zveřejněn *Segment Anything model (SAM)*[1], který umožňuje bez předchozího trénování (tzv. *zero-shot*) segmentovat objekty na základě:

1. tzv. *promptů* - bodů v obrázku, které uživatel označí jako příslušející k danému objektu [1]
2. *bounding boxů* - ohraničujících obdélníků, které uživatel označí kolem objektu [1]

Jeho výstupem je vysegmentovaný objekt v podobě masky (případně více masek) příslušející k danému objektu. [1] Jelikož SAM byl uvolněn teprve nedávno a v době anotování datasetu nebyl ještě dostupný žádný nástroj, který by ho využíval a tím zyrchlil a zvýšil přesnost anotací, bylo vytvořeno vlastní řešení - viz sekce 2.2.3.1.

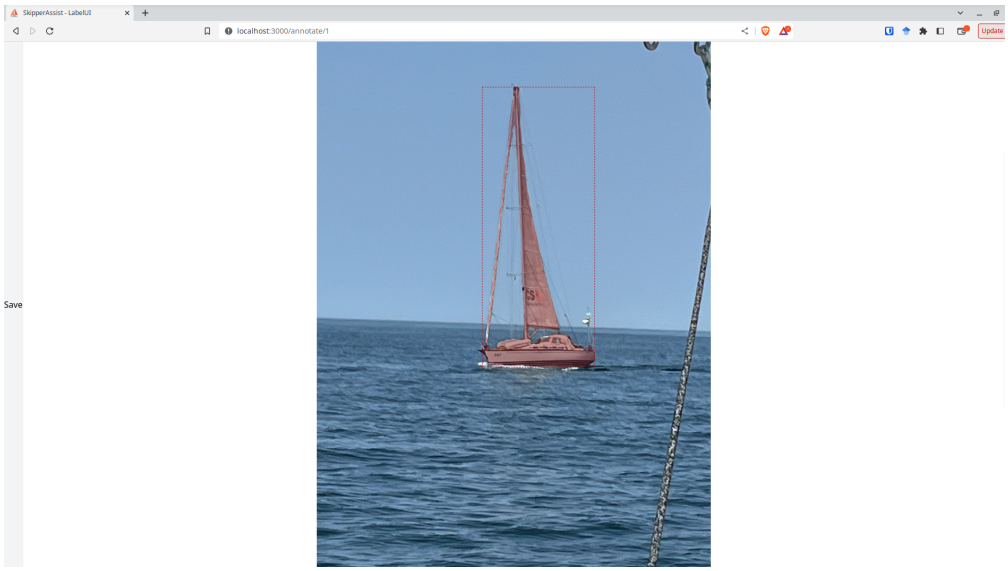
2.2.3.1 LabelUI - Nástroj pro anotaci s využitím SAM

Pro anotaci fotografií v datasetech byl vytvořen nástroj *LabelUI*, který umožňuje anotaci fotografií za využití Segment Anything modelu (SAM). Jedná se o webovou aplikaci napsanou v TypeScriptu[31] za použití React frameworku Next.js[32]. Data jsou ukládána do SQLite databáze.

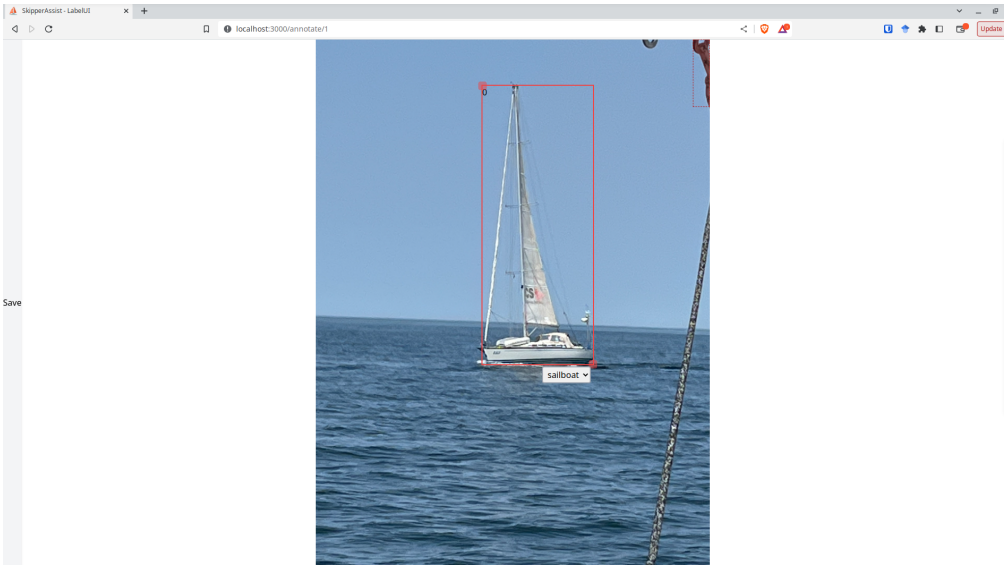
Inference SAM modelu je prováděna přímo v prohlížeči a v reálném čase za pomocí knihovny *onnx-runtime-web*[33]. Pro samotné obrázky datasetu je nutné provést jejich předzpracování - vygenerování tzv. *embeddingů* pomocí Python skriptu (respektive Jupyter notebooku) `generate_embeddings.ipynb`, který je na přiloženém médiu.

Kurzorem myši je možné označit bod v obrázku, který přísluší objektu, který chceme anotovat. Poloha kurzoru je předávána SAM modelu a ten vrací masku (zobrazena jako transparentní červená plocha), která by měla odpovídat anotovanému objektu (viz obrázek 2.5). Na základě masky je vytvořen návrh bounding boxu, který je možné potvrdit stiskem klávesy `"a"` a následně, pokud je to potřeba, upravit. Následně již stačí jen vybrat odpovídající třídu, které objekt přísluší a anotace uložit (viz obrázek 2.6).

Po dokončení anotace všech fotografií je možné vyexportovat anotace v YOLO formátu pomocí Python skriptu `labelui_yolo_export.py`, který se nachází na přiloženém médiu.



Obrázek 2.5 LabelUI - Návrh anotace za pomoci SAM podle promptu (aktuální polohy kurzoru myši)



Obrázek 2.6 LabelUI - Výsledná anotace

2.3 Metriky pro evaluaci algoritmů detekce objektů

2.3.1 Pomocné metriky

Pro vysvětlení jednotlivých metrik je nutné nejdříve definovat několik pojmů.

Kategorie detekovaných objektů, resp. jejich bounding boxů:

- *True Positive (TP)* - správně detekovaný objekt[11]
- *False Positive (FP)* - objekt, který byl detekován, ale ve skutečnosti se v obrázku nenachází, případně se objekt ve skutečnosti v obrázku nachází, ale byl detekován jinde[11]
- *False Negative (FN)* - objekt se ve skutečnosti v obrázku nachází, ale nebyl detekován[11]

Precision

Česky *Precision* může být přeloženo jako *přesnost* či *preciznost*[34], ale, z důvodu nekonzistentních překladů, pro účely této práce je používán anglický název.

Jedná se o metriku, která udává poměr správně detekovaných objektů (TP) oproti všem **detekovaným** objektům ($TP + FP$).[11, 35]

Předpokládejme, že máme dataset, který obsahuje G anotovaných objektů a model detekuje N objektů. Z nich je správně detekováno S objektů ($S < G$). Potom je možné vypočítat Precision jako:[11]

$$Precision = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{N-S} FP_n} \tag{2.1}$$

Recall

Česky *Recall* může být přeloženo jako *úplnost*, ale opět, z důvodu nekonzistentních překladů do ČJ, je používán anglický název.

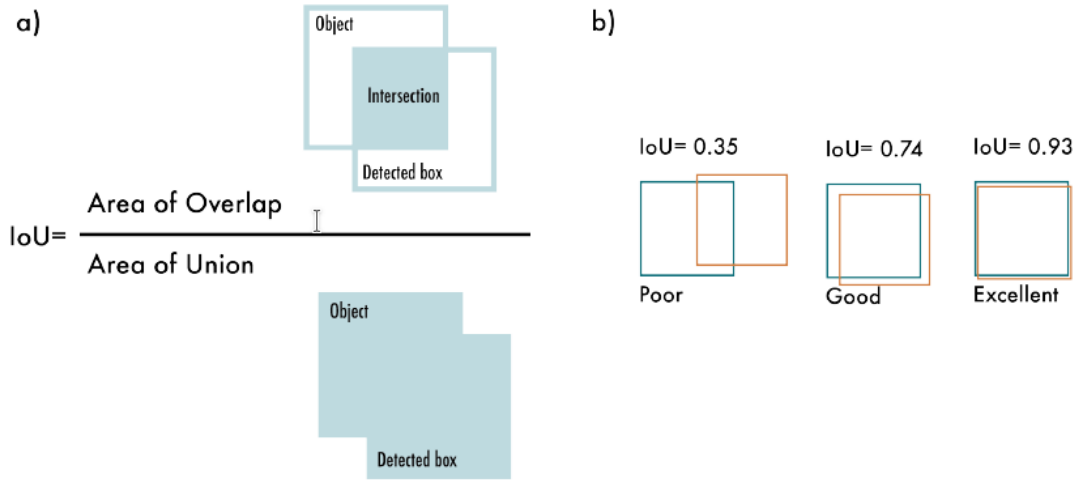
Jedná se o metriku, která udává poměr správně detekovaných objektů (TP) oproti všem **anotovaným** objektům v datasetu ($TP + FN$).[11, 35]

Obdobně jako u Precision (sekce 2.3.1) - za předpokladu, že máme dataset, který obsahuje G anotovaných objektů a model detekuje N objektů. Z nich je správně detekováno S objektů ($S < G$). Potom je možné vypočítat Recall jako:[11]

$$Recall = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{G-S} FN_n} \tag{2.2}$$

Intersection over Union (IoU)

Intersection over Union je metrika, která nám říká míru "správnosti"detekovaného bounding boxu oproti správnému bounding boxu označující (detekovaný) objekt.[11, 35] V případě detekce objektů je IoU definováno jako poměr plochy průniku dvou objektů a plochy sjednocení dvou objektů.[11, 35]



Obrázek 2.7 Ilustrace metriky IoU. Zdroj obrázku: [35]

IoU je možné vypočítat pomocí vzorce 2.3[11, p. 8]:

$$IoU = \frac{plocha(BB_{pred} \cap BB_{gt})}{plocha(BB_{pred} \cup BB_{gt})} \tag{2.3}$$

kde BB_{pred} je predikovaný bounding box a BB_{gt} je tzv. *ground truth* (správný) bounding box.[11]

Jak je možné ze vztahu 2.3 vyčíst, IoU je vždy v intervalu $[0, 1]$, kde 0 značí, že se bounding boxy nepřekrývají vůbec a 1 značí, že se bounding boxy překrývají zcela.[11]

V případě, že je IoU vyšší než zvolená mez (*threshold*), je detekce považována za správnou. V opačném případě je detekce považována za chybnou.[11, 35]

2.3.2 Average Precision (AP)

Average Precision, do češtiny může být přeloženo jako *průměrná přesnost*, je metrika, která je využívána pro vyhodnocení "kvality"detekce objektů v rámci jedné třídy objektů.[11]

Average Precision je v grafické reprezentaci obsah plochy pod tzv. *precision-recall křivkou*. Precision-recall křivka se předzpracovává pomocí například *N-bodové interpolace* (*N-point interpolation*, [11, s. 11]) nebo *All-point interpolace*([11, s. 11]), aby bylo eliminováno její "zig-zag"chování.[11]

Pro účely zjištění Average Precision, se může pro výpočet Precision a Recall metrik uvažovat mez (threshold) τ, tzv. *confidence level*-u (míra jistoty, s jakou si je model jistý, že v daném bounding boxu se nachází objekt dané třídy)

Vztahy pro výpočet Precision a Recall budou tedy následující:

$$Precision(\tau) = \frac{\sum_{n=1}^{S} TP_n(\tau)}{\sum_{n=1}^{S} TP_n(\tau) + \sum_{n=1}^{N-S} FP_n(\tau)} \tag{2.4}$$

$$Recall(\tau) = \frac{\sum_{n=1}^{S} TP_n(\tau)}{\sum_{n=1}^{S} TP_n(\tau) + \sum_{n=1}^{G-S} FN_n(\tau)} \tag{2.5}$$

[11]

Nejprve je nutné seřadit všechny K detekovaných objektů dané třídy podle jejich confidence level-u τ sestupně:

$$\tau(k), k \ = 1, 2, ..., K \qquad \text{takové, že} \qquad \tau(i) \geq \tau(j), \ pro\ i > j \tag{2.6}$$

Máme tedy množinu hodnot ($Precision(\tau(k))$, $Recall(\tau(k))$), kde $k = 1, 2, ..., K$.

Pro výpočet Average Precision, se využívá tzv. *interpolovaná precision-recall křivka*, která je vytvořena interpolací hodnot Precision metriky v závislosti na Recall a je definována funkcí 2.7. [11]

$$Precision_{interp}(r) = \max_{k|Recall(\tau(k))\geq r} Precision(\tau(k)) \tag{2.7}$$

kde r leží v intervalu $[0, 1]$.[11]

Následně je možné vypočítat Average Precision jako obsah plochy pod interpolovanou precision-recall křivkou například pomocí N-bodové interpolace podle vztahu 2.8.[11]

$$AP = \frac{1}{N} \sum_{n=1}^{N} Precision_{interp}(R_r(n)) \tag{2.8}$$

, kde $R_r(n)$ je definováno vztahem 2.9.[11]

$$R_r(n) = \frac{N - \ n}{N - \ 1}, n = 1, 2, ..., N \tag{2.9}$$

[11]

2.3.3 Mean Average Precision (mAP)

Pro porovnání různých modelů napříč všemi detekovanými třídami se používá metrika *Mean Average Precision*.[11] Do češtiny by se dala přeložit jako *průměrná střední hodnota přesnosti*.[36]

Je možné ji vypočítat jako průměr AP napříč všemi třídami - tedy pomocí následujícího vztahu:[11]

$$mAP = \frac{1}{C} \sum_{i=1}^{C} AP_i \tag{2.10}$$

kde C je celkový počet všech tříd objektů a AP_i je Average Precision i-té třídy.[11]

2.4 Implementace

2.4.1 Detekce objektů

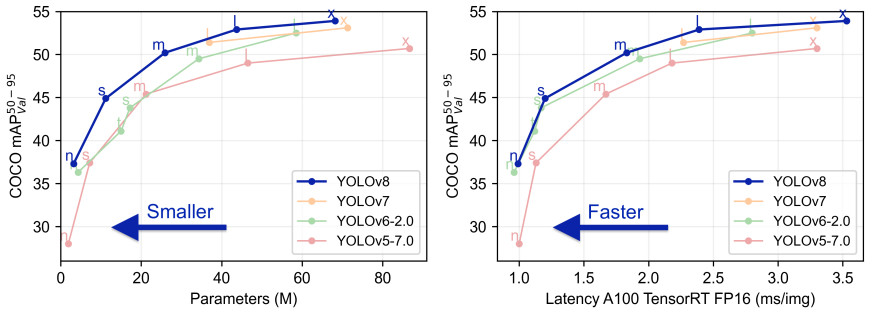
Na základě rešerše byl jako detekční algoritmus identifikován *YOLO*, *Faster R-CNN* a *RT-DETR*. Následně, na základě například [37, 38] bylo zvoleno **YOLOv8**[39] oproti Faster R-CNN. Důvodem je dle [37, s. 382] přibližně obdobná přesnost detekce (mAP@0.5 až 90, 57% pro Faster R-CNN a 89, 45% pro YOLOv8-n; na jejich datasetu[37, s. 382]), ale výrazně vyšší rychlost (nejméně 0, 0281 s/obrázek u Faster R-CNN, 0, 0011 s/obrázek u YOLOv8-n - tj. 25× rychlejší [37, s. 382]), což umožňuje pro detekci objektů v reálném čase využít méně výkonný hardware a tím snížit i spotřebu elektrické energie, která bývá na menších plavidlech značně omezeným zdrojem.

Druhým identifikovaným, potenciálně vhodným modelem je *RT-DETR (Real Time Detection Transformer)*, který na COCO datasetu dosahuje (RT-DETR-L) mAP@0.5 71, 6 oproti 69, 8 u YOLOv8-L. [3, s. 7] Navíc je RT-DETR při inferenci přibližně **1,6**× rychlejší než YOLOv8-L.

2.4.1.1 YOLOv8-n

Pro první pokus byla zvolena jedna z dostupných variant modelu YOLOv8. Konkrétně se jednalo o jeho nejmenší možnou variantu - *YOLOv8-n* [16], která je za použití i jen CPU pro inferenci velmi rychlá (80.4 ms[39]). Je potřeba zmínit, že je však z variant YOLOv8 modelů nejméně přesná - viz Obrázek 2.8.

Výchozí šířka vstupních obrázků pro YOLOv8 je *640px* [39], což pro detekci malých objektů nemusí být dostatečné. Z tohoto důvodu byly použity vstupní obrázky o šířce *1280px*, což je největší rozměr obrázku, který splňují všechny fotky v trénovacím datasetu.

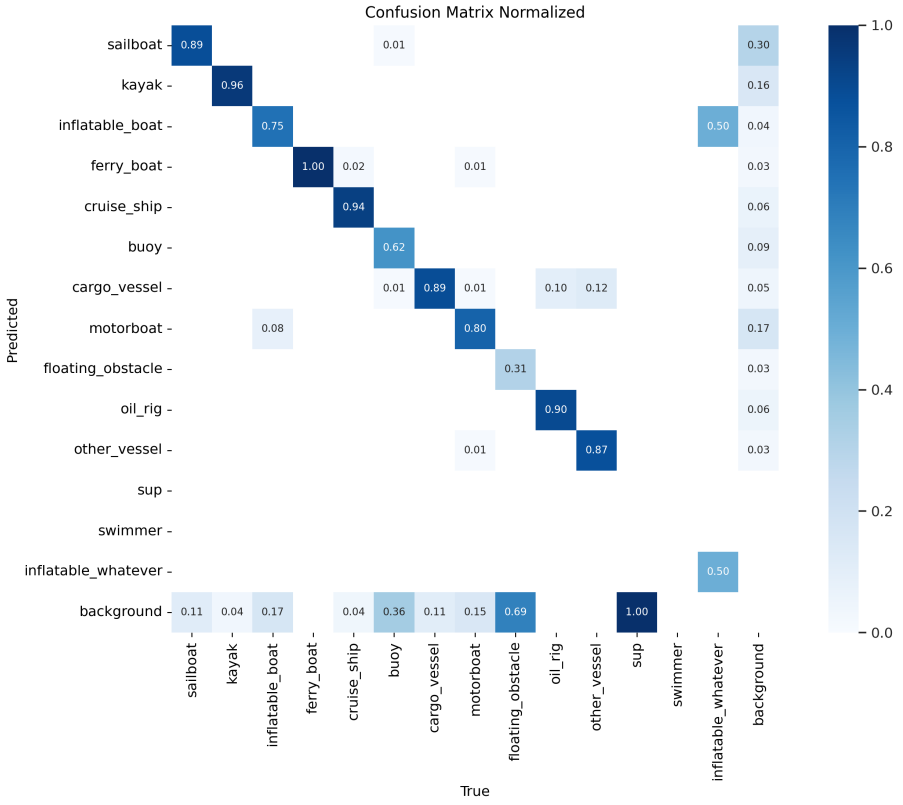


Obrázek 2.8 Porovnání variant modelu *YOLOv8*. Zdroj: [39]

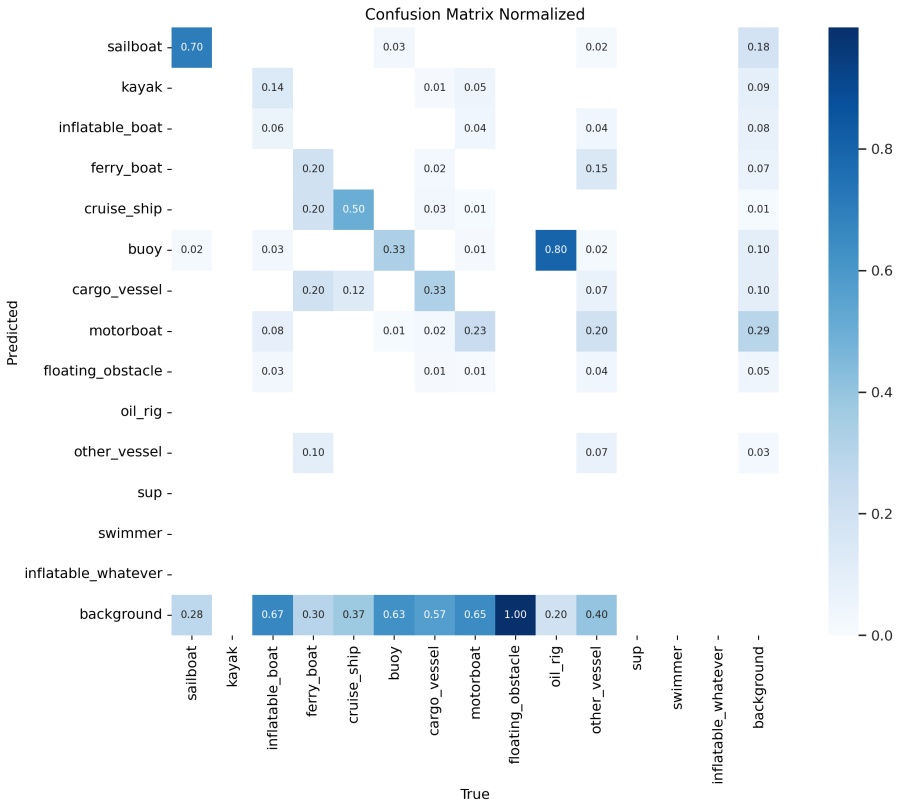
Výsledky

Natrénovaný model *YOLOv8-n* dosáhl na validačním datasetu přesnosti detekce mAP@50 0, 481, zatímco na testovacím datasetu pouze 0, 373. Jak je možné vidět z matic záměn (Confusion matrix) na Obrázku 2.9, model měl potíže se správnou klasifikací objektů - testovací dataset je výrazně náročnější než trénovací a validační, neboť obsahuje větší podíl malých objektů, velmi variabilní světelné podmínky a některé fotografie jsou přiblíženy na vzdálené objekty - tedy nejsou vždy ostré.

Vzhledem k tomu, že primární cíl je detekovat objekty, se kterými může hrozit kolize a jejich klasifikace je sekundární, byl vytvořen dataset, který obsahuje všechny anotované objekty, ale pouze jednu obecnou třídu `obstacle` (překážka). Model natrénovaný na tomto datasetu již dosáhl na validačním datasetu mAP@50 0, 677 a na testovacím datasetu 0, 69.

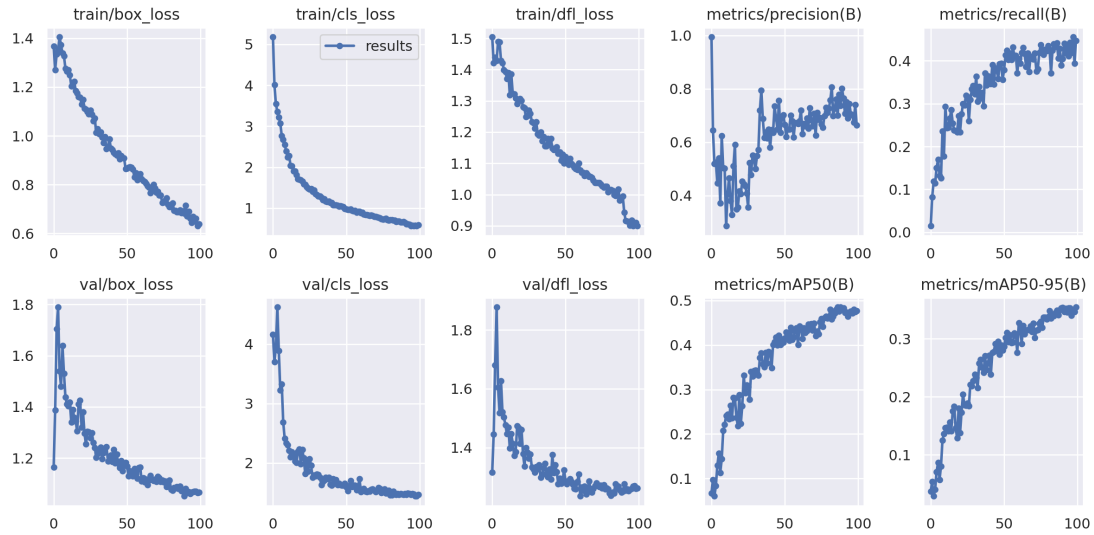


(a) validační dataset



(b) testovací dataset

Obrázek 2.9 Confusion matrix pro natrénovaný model *YOLOv8-n*



Obrázek 2.10 Průběh trénování YOLOv8-n

2.4.2 YOLOv8-m

Druhým trénovaným modelem byla varianta *YOLOv8-m*, která by měla dosahovat lepších výsledků než YOLOv8-n, ale zároveň být dostatečně rychlá pro detekci v reálném čase (viz Obrázek 2.8 a [39]).

Obdobně jako u YOLOv8-n (Sekce 2.4.1.1), byly použity vstupní obrázky o šířce *1280px* a model byl trénován na dvou variantách datasetu - s jednou obecnou třídou `obstacle` a s původními třídami.

Výsledky

Na testovacím datasetu dosáhl model se všemi detekovanými třídami mAP@50 $0,439$ při rychlosti detekce $74ms$ na obrázek. Celkově tedy dosáhl přibližně o 17% lepších výsledků než YOLOv8-n. Model s jednou obecnou třídou `obstacle` dosáhl na validačním datasetu mAP@50 $0,732$ a na testovacím datasetu $0,719$ při rychlosti detekce $48,3ms$ na obrázek.

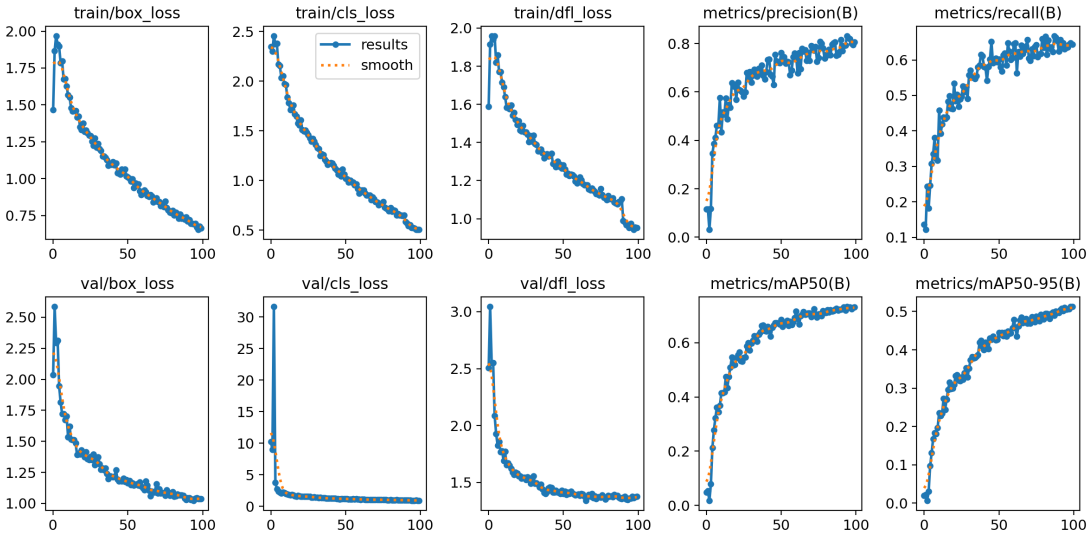
2.4.3 RT-DETR

Třetím trénovaným modelem byl RT-DETR, popsaný v Sekci 1.2.1. Tento model by měl dle [3] dosahovat lepších výsledků než YOLOv8 a zároveň být i rychlejší.

Narozdíl od trénovaných YOLOv8 modelů, byl RT-DETR trénován na obrázcích o rozměru *640px* z důvodu vyšší náročnosti na výpočetní výkon, zejména paměť grafické karty při trénování. Trénování probíhalo na pronajatém stroji pomocí platformy Paperspace s parametry:

- CPU: *8 vCPU*
- RAM: *45 GB*
- GPU: *NVIDIA RTX A4000* s *16 GB* paměti

[40]



Obrázek 2.11 Průběh trénování YOLOv8-m

Při prvním běhu model po několika iteracích vyčerpal paměť GPU a trénování se ukončilo. Následně bylo v trénování modelu pokračováno.

Výsledky

Na validačním datasetu dosáhl model mAP@50 **0,813** a na testovacím pouhých 0,293 při rychlosti detekce 59ms na obrázek. Výsledky jsou tedy na validačním datasetu výrazně lepší než u YOLOv8, ale na testovacím jsou dle mAP horší.

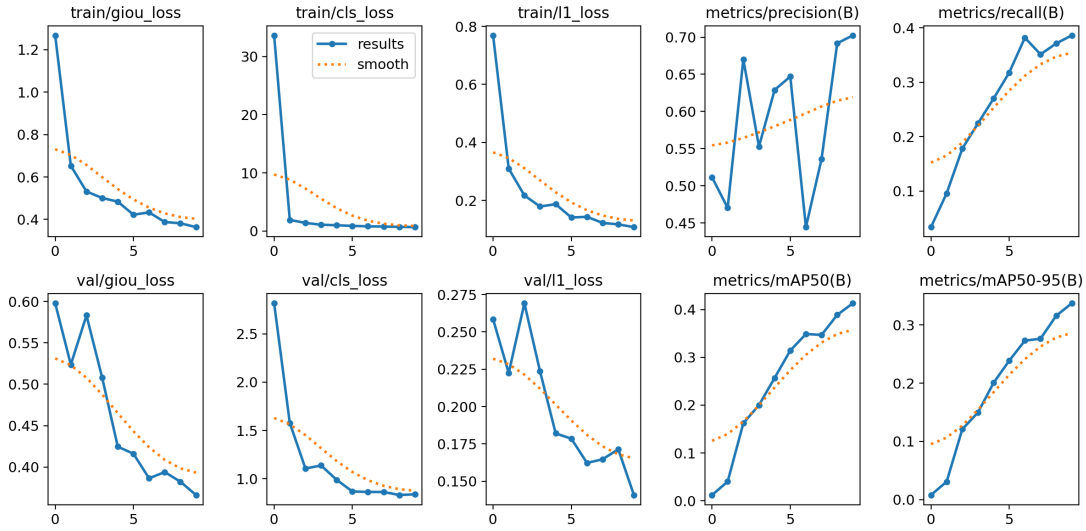
Když se blíže podíváme na Obrázek 2.13, je možné vidět, že model RT-DETR zvládnul detekovat velmi malé a pouhým okem velmi špatně viditelné objekty, které ani nebyly nalezeny při anotaci a tedy nejsou anotovány. V metrikách jsou tedy ale započítány jako falešně pozitivní detekce. Autoři [3] uvádějí, že model RT-DETR je nejlépe detekuje velké objekty, což se dle vyhodnocení jednotlivých detekcí na testovacím datasetu nepotvrdilo.

Nicméně, pokud se podíváme na Obrázky 2.15 nebo 2.21, je možné vidět, že model občas detekuje některé objekty vícekrát nebo se objevují falešně pozitivní detekce, spíše než falešně negativní, což je pro zamýšlenou aplikaci menší problém, než kdyby to bylo opačně.

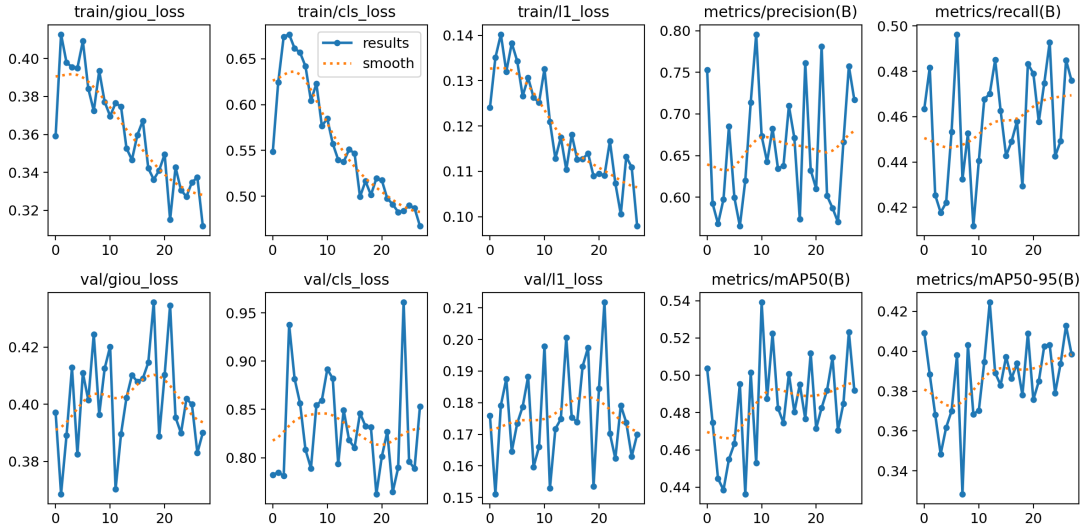
2.4.4 Porovnání modelů

Testování všech modelů probíhalo na totožném hardwaru a v maximálně izolovaném prostředí. Využit byl virtuální stroj od Paperspace s GPU NVIDIA Quadro P5000. [40]

Výsledky jsou uvedeny v Tabulkách 2.3 a 2.4. Obecně u všech trénovaných modelů na datasetech se všemi třídami objektů (Tabulka 2.3) je možné pozorovat snížení mAP mezi validačním a testovacím datasetem. Tento jev může být z části způsoben tím, že testovací dataset je "náročnější" pro detekování objektů, protože obsahuje obrázky s objekty, které jsou hůře viditelné a obsahuje větší podíl malých objektů, a je tedy trochu odlišný od trénovacího a validačního datasetu, kde jsou objekty většinou dobře viditelné a rozeznatelné. Největší změna byla u modelu RT-DETR Large, kde se hodnota mAP@50 snížila z 0,813 na 0,293. Zároveň ale na validačním datasetu RT-DETR Large dosáhl nejlepších výsledků z testovaných modelů, konkrétně mAP@50 0,813 a mAP@50-95 0,672 při vyšší rychlosti inference než YOLOv8-m. Nicméně, jak je uvedeno v sekci 2.4.3, při kontrole jednotlivých detekcí na testovacím datasetu, bylo zjištěno, že



(a) první běh



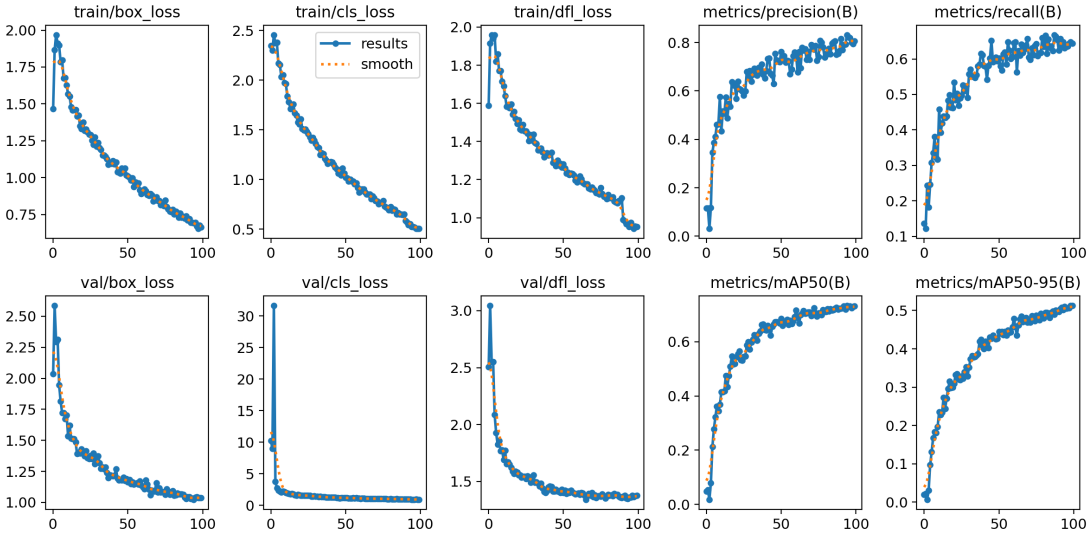
(b) druhý běh

Obrázek 2.12 Průběh trénování RT-DETR

model RT-DETR detekoval i objekty, které nebyly při anotaci pouhým okem, bez zvětšení nalezené. Tato neúplnost anotací je pravděpodobně také důvodem, proč se dle metrik RT-DETR na testovacím datasetu jeví jako horší než YOLOv8.

Při porovnání výsledků napříč datasety se všemi třídami a jednou třídou, je možné pozorovat, že hodnoty mAP i Precision (P) a Recall (R) jsou vyšší u datasetu s jednou třídou objektů. Z toho je možné usuzovat,že by bylo třeba rozšířit trénovací dataset. Dále také je možné z Tabulky 2.4 vidět, že naopak oproti datasetu se všemi třídami objekt (Tabulka 2.3) je možné u obou variant YOLOv8 modelu pozorovat z validačního na testovací dataset zachování hodnot mAP@50 a mAP@50-95, případně jejich mírné zvýšení. To je možné opět přisuzovat, že dataset pro detekci jedné třídy je již dostatečně velký a různorodý.

Obrázek 2.13 Zvětšené oblasti z anotovaného obrázku a obrázku s detekovanými objekty pomocí natrénovaného modelu RT-DETR



Obrázek 2.14 Průběh trénování YOLOv8-m

Model	P	R	mAP@50	mAP@50-95	Rychlost inference
YOLOv8-n	0,757	0,401	0,481	0,351	19,7 ms
YOLOv8-m	0,792	0,458	0,525	0,398	74,0 ms
RT-DETR Large	0,807	0,786	***0,813***	***0,672***	59,0 ms

(a) na validačním datasetu

Model	P	R	mAP@50	mAP@50-95	Rychlost inference
YOLOv8-n	0,373	0,412	0,373	0,271	19,7 ms
YOLOv8-m	0,49	0,392	0,439	0,322	74,0 ms
RT-DETR Large	0,569	0,283	0,298	0,225	59,0 ms

(b) na testovacím datasetu

Tabulka 2.3 Porovnání trénovaných modelů - na datasetech se všemi třídami objektů

2.4.4.1 Možnosti zlepšení

Z matic záměn (confusion matrices) na Obrázku 2.9 je zřejmé, že na testovacím datasetu, vzhledem k jejich podobnosti, byly často zaměňovány třídy *cruise ship* a *trajektu*, dále *motorová*

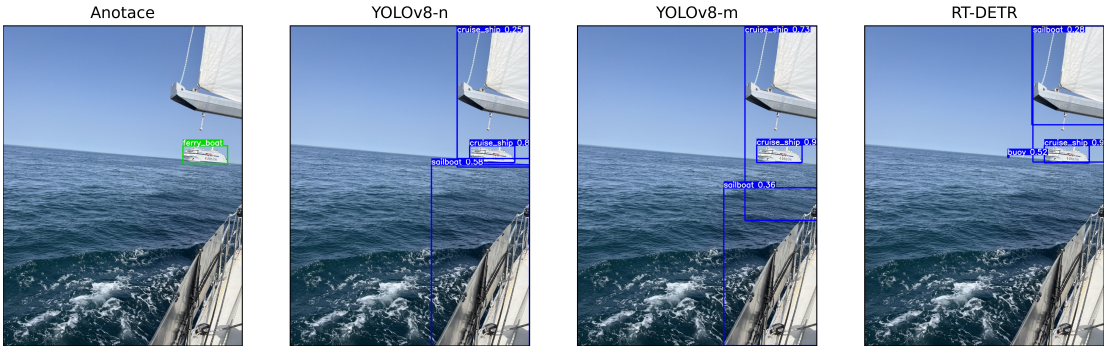
Model	P	R	mAP@50	mAP@50-95	Rychlost inference
YOLOv8-n	0,785	0,595	0,677	0,456	42,7 ms
YOLOv8-m	0,802	0,647	0,732	0,512	48,3 ms

(a) na validačním datasetu

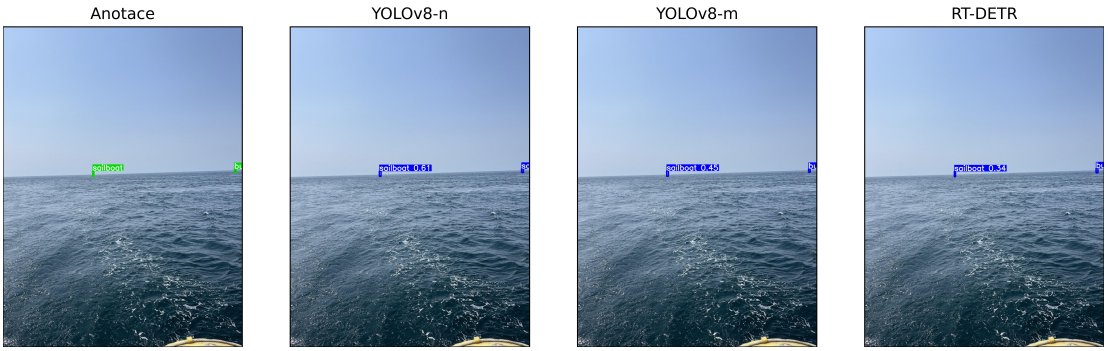
Model	P	R	mAP@50	mAP@50-95	Rychlost inference
YOLOv8-n	0,772	0,597	0,69	0,482	42,7 ms
YOLOv8-m	0,761	0,619	0,719	0,531	48,3 ms

(b) na testovacím datasetu

Tabulka 2.4 Porovnání trénovaných modelů - na datasetech s jednou obecnou třídou `obstacle` (překážka)



Obrázek 2.15 Porovnání výsledků detekce na testovacím datasetu

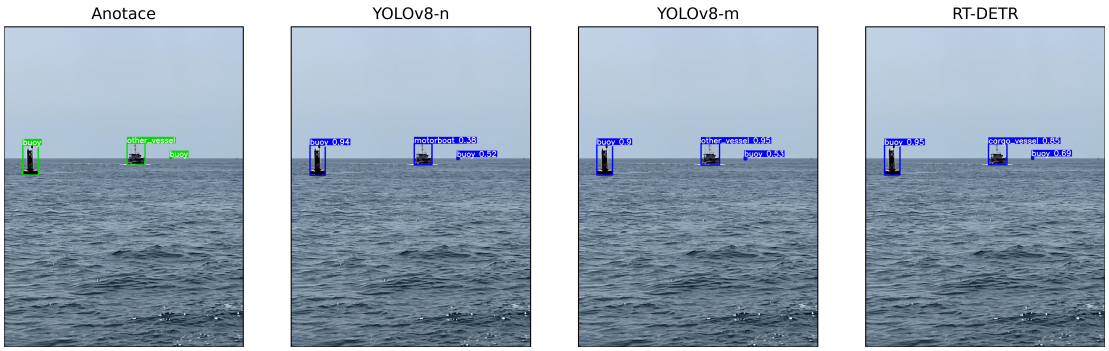


Obrázek 2.16 Porovnání výsledků detekce na testovacím datasetu

loď a *ostatní plavidla* případně *motorová loď* a *nákladní loď*. Vzhledem k hlavní motivaci nasazení těchto algoritmů, tedy detekovat nebezpečné objekty a předejít tak srážce s lodí, by bylo možné některé třídy sloučit (například *cruise ship* a *trajekt* sjednotit do společné třídy `passenger_vessel`).

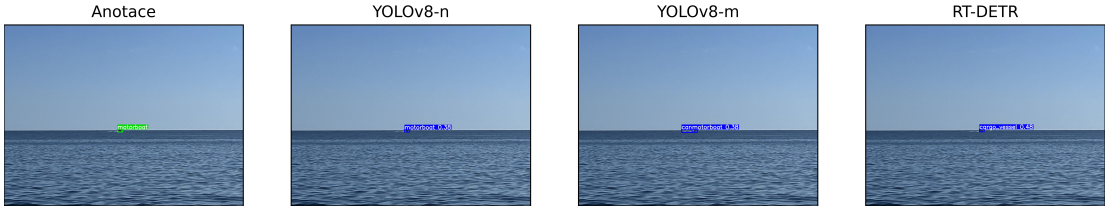
Dále by bylo vhodné rozšířit trénovací dataset, zejména o třídy, které jsou zastoupené nejméně, abychom v ideálním případě dosáhly rovnoměrného rozložení. Pokud nebude možné (a nebo to bude obtížné) získat sadu dodatečných fotografií přirozeně, může se trénovací dataset rozšířit tzv. *augmentací* (proces umělého rozšíření datasetu pomocí modifikovaných kopií stávající obrázků)

Pro zvýšení robustnosti by bylo možné objekty detekovat pomocí několika modelů zároveň a zvyšovat tak šanci, že potenciálně nebezpečný plovoucí objekt bude detekován. Bylo by také vhodné implementovat algoritmus pro sledování objektů v čase. Vzhledem k tomu, že rychlost algoritmů (tabulky 2.3 a 2.4) není nijak omezující, bylo by možné implementovat algoritmus,

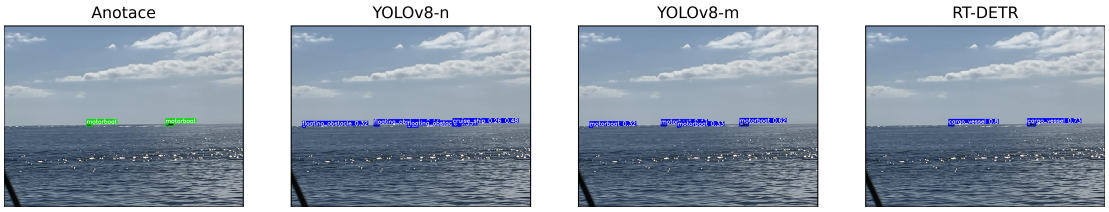


Obrázek 2.17 Porovnání výsledků detekce na testovacím datasetu

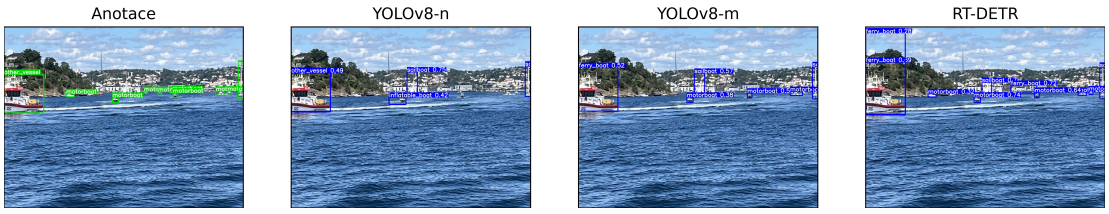
Obrázek 2.18 Porovnání výsledků detekce na testovacím datasetu



Obrázek 2.19 Porovnání výsledků detekce na testovacím datasetu



Obrázek 2.20 Porovnání výsledků detekce na testovacím datasetu



Obrázek 2.21 Porovnání výsledků detekce na testovacím datasetu

který bude plovoucí objekt sledovat v čase. Předejde se tak například situaci, kdy objekt náhle zmízí apod.

V neposlední řadě by také bylo možné využít v kombinaci se standardní kamerou jiné senzory, jako je například NIR kamera, AIS nebo radar. Detekce objektů založená pouze na výstupech kamery budou mít zcela jistě potíže pří špatném počasí (jako déšť a mlha) a budou naprosto velmi obtížně použitelné v noci.

Kapitola 3
Závěr

V rámci této diplomové práce byly prozkoumány možné algoritmy pro detekci plovoucích objektů založených na strojovém učení. Pro detekci objektů byl zvolen algoritmus, který může být nasazen na menších plavidlech, kde jsou omezené možnosti výpočetního výkonu z důvodu spotřeby elektrické energie a který je schopen detekovat objekty v reálném čase. Výzvou pro detekci je velmi různorodá velikost objektů, zejména velmi malé a vzdálené objekty společně s variabilními světelnými podmínkami prostředí.

Byly prozkoumány stávající state-of-the-art algoritmy pro detekci objektů a vybrány dva algoritmy – YOLOv8 a RT-DETR, které byly následně natrénovány pomocí vlastních datasetů, které byly za tímto účelem vytvořeny a anotovány. Pro zpřesnění a zrychlení procesu anotace byla vytvořena aplikace, která implementuje *Segment Anything model (SAM)* umožňující segmentovat objekty v obrázku například zadáním bodu, který leží v oblasti objektu. Tím bylo možné získat přesnější anotace, než kdyby byly anotovány zcela ručně – stačilo na objekt najet myší a aplikace automaticky navrhla anotaci. Dataset pro trénování a validaci (ověřování výsledků v průběhu učení) obsahuje 2 864 anotací. Dataset pro testování, který byl celý pořízen v reálném prostředí na moři, obsahuje celkem 795 anotací.

Testování dvou variant (nano a medium) YOLOv8 modelu a modelu RT-DETR Large ukázalo, že nejlepších výsledků dosáhl model RT-DETR Large s mAP 0,813 na validačním datasetu. Na testovacím datasetu se hodnota mAP snížila – model byl schopný nalézt i velmi malé objekty, které při anotaci nebyly bez zvětšení pouhým okem viditelné. Experiment se sjednocením všech tříd v datasetech do jedné obecné třídy `obstacle` (překážka) ukázal důležitost dostatečně rozsáhlého a různorodého datasetu. V tomto případě se u YOLOv8 modelů změnila hodnota mAP mezi validačním a testovacím datasetem pouze minimálně.

Pro další zlepšení přesnosti detekce a zvýšení robustnosti je možné využít například sledování objektů, neboť jednou detekovaný objekt by neměl náhle zmizet. Další možností je rozšířit dataset.

Celkem je tedy zřejmé, že by mělo být možné nasadit algoritmus pro detekci i malých plovoucích objektů v reálném čase na zařízeních s omezeným výpočetním výkonem. Tento systém tedy může pomoci zvýšit bezpečnost plavby. Práce zároveň přináší srovnání YOLOv8 a RT-DETR v oblasti detekce plovoucích objektů.

Příloha A

Přiložené médium - microSD karta

Přiložená microSD karta se zdrojovými kódy, natrénovanými modely a datasety.

Bibliografie

1. KIRILLOV, Alexander; MINTUN, Eric; RAVI, Nikhila; MAO, Hanzi; ROLLAND, Chloe; GUSTAFSON, Laura; XIAO, Tete; WHITEHEAD, Spencer; BERG, Alexander C.; LO, Wan-Yen; DOLLÁR, Piotr; GIRSHICK, Ross. *Segment Anything*. 2023. Dostupné z arXiv: 2304.02643 [cs.CV].
2. REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. You Only Look Once: Unified, Real-Time Object Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
3. LV, Wenyu; XU, Shangliang; ZHAO, Yian; WANG, Guanzhong; WEI, Jinman; CUI, Cheng; DU, Yuning; DANG, Qingqing; LIU, Yi. *DETRs Beat YOLOs on Real-time Object Detection*. 2023. Dostupné z arXiv: 2304.08069 [cs.CV].
4. GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor; MALIK, Jitendra. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.
5. LIU, Wei; ANGUELOV, Dragomir; ERHAN, Dumitru; SZEGEDY, Christian; REED, Scott; FU, Cheng-Yang; BERG, Alexander C. Ssd: Single shot multibox detector. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, s. 21–37.
6. CARION, Nicolas; MASSA, Francisco; SYNNAEVE, Gabriel; USUNIER, Nicolas; KIRILLOV, Alexander; ZAGORUYKO, Sergey. End-to-end object detection with transformers. In: *European conference on computer vision*. Springer, 2020, s. 213–229.
7. LEE, Seong Ju; MOON, Yong Seon; KO, Nak Yong; CHOI, Hyun-Taek; LEE, Jong-Moo. A method for object detection using point cloud measurement in the sea environment. In: *2017 IEEE Underwater Technology (UT)*. IEEE, 2017, s. 1–4.
8. LYU, Hongguang; SHAO, Zeyuan; CHENG, Tao; YIN, Yong; GAO, Xiaowei. Sea-Surface Object Detection Based on Electro-Optical Sensors: A Review. *IEEE Intelligent Transportation Systems Magazine*. 2022, s. 2–27.
9. STETS, Jonathan D; SCHÖLLER, Frederik ET; PLENGE-FEIDENHANS, Martin K; ANDERSEN, Rasmus H; HANSEN, Søren; BLANKE, Mogens. Comparing spectral bands for object detection at sea using convolutional neural networks. In: *Journal of Physics: Conference Series*. IOP Publishing, 2019, sv. 1357, s. 012036. Č. 1.
10. *NMEA 2000 Standard - National Marine Electronics Association. [online]. [cit. 2023-05-16]*. c2023. Dostupné také z: https://www.nmea.org/nmea-2000.html.

11. PADILLA, Rafael; PASSOS, Wesley L.; DIAS, Thadeu L. B.; NETTO, Sergio L.; SILVA, Eduardo A. B. da. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*. 2021, roč. 10, č. 3. ISSN 2079-9292. Dostupné z DOI: 10.3390/electronics10030279.
12. MUSIĆ, Josip; MARASOVIĆ, Tea; PAPIĆ, Vladan; OROVIĆ, Irena; STANKOVIĆ, Srdjan. Performance of Compressive Sensing Image Reconstruction for Search and Rescue. *IEEE Geoscience and Remote Sensing Letters*. 2016, roč. 13, č. 11, s. 1739–1743. Dostupné z DOI: 10.1109/LGRS.2016.2606767.
13. LEE, Jangwon; WANG, Jingya; CRANDALL, David; ŠABANOVIĆ, Selma; FOX, Geoffrey. Real-Time, Cloud-Based Object Detection for Unmanned Aerial Vehicles. In: *2017 First IEEE International Conference on Robotic Computing (IRC)*. 2017, s. 36–43. Dostupné z DOI: 10.1109/IRC.2017.77.
14. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is all you need. *Advances in neural information processing systems*. 2017, roč. 30.
15. DOSOVITSKIY, Alexey; BEYER, Lucas; KOLESNIKOV, Alexander; WEISSENBORN, Dirk; ZHAI, Xiaohua; UNTERTHINER, Thomas; DEHGHANI, Mostafa; MINDERER, Matthias; HEIGOLD, Georg; GELLY, Sylvain et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 2020.
16. JOCHER, Glenn; CHAURASIA, Ayush; QIU, Jing. *Ultralytics YOLOv8*. 2023. Ver. 8.0.0. Dostupné také z: https://github.com/ultralytics/ultralytics.
17. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
18. ZOU, Zhengxia; CHEN, Keyan; SHI, Zhenwei; GUO, Yuhong; YE, Jieping. Object detection in 20 years: A survey. *Proceedings of the IEEE*. 2023.
19. GIRSHICK, Ross. Fast R-CNN. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015.
20. KIM, Jeong-ah; SUNG, Ju-Yeong; PARK, Se-ho. Comparison of Faster-RCNN, YOLO, and SSD for real-time vehicle type recognition. In: *2020 IEEE international conference on consumer electronics-Asia (ICCE-Asia)*. IEEE, 2020, s. 1–4.
21. REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*. 2015, roč. 28.
22. TERVEN, Juan; CORDOVA-ESPARZA, Diana. A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. *arXiv preprint arXiv:2304.00501*. 2023.
23. LIN, Junyu; MAO, Xiaofeng; CHEN, Yuefeng; XU, Lei; HE, Yuan; XUE, Hui. D^ 2ETR: Decoder-Only DETR with Computationally Efficient Cross-Scale Attention. *arXiv preprint arXiv:2203.00860*. 2022.
24. *How it works ¿ SEA.AI. [online]. [cit. 2023-05-14]*. c2023. Dostupné také z: https://sea.ai/how-it-works/.
25. *Offshore ¿ SEA.AI. [online]. [cit. 2023-05-14]*. c2023. Dostupné také z: https://sea.ai/category/offshore/.
26. *Boat types recognition — Kaggle. [online]. [cit. 2023-03-22]*. c2018. Dostupné také z: https://www.kaggle.com/datasets/clorichel/boat-types-recognition.
27. *Vessel identification — Kaggle. [online]. [cit. 2023-03-22]*. c2022. Dostupné také z: https://www.kaggle.com/datasets/gauravduttakiit/vessel-identification.

28. YOLOV5SEASHIPS. *SeaShips Dataset* [https://universe.roboflow.com/yolov5seaships/seaships-mcvwt]. Roboflow, 2022. Dostupné také z: https://universe.roboflow.com/yolov5seaships/seaships-mcvwt. visited on 2023-04-12.
29. PRASAD, Dilip K; RAJAN, Deepu; RACHMAWATI, Lily; RAJABALLY, Eshan; QUEK, Chai. Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey. *IEEE Transactions on Intelligent Transportation Systems*. 2017, roč. 18, č. 8, s. 1993–2016.
30. *License — Unsplash. [online]. [cit. 2023-03-23]*. c2023. Dostupné také z: https://unsplash.com/license.
31. *TypeSript: JavaScript With Syntax For Types. [online]. [cit. 2023-05-02]*. c2023. Dostupné také z: https://www.typescriptlang.org/.
32. *Next.js by Vercel - The React Framework. [online]. [cit. 2023-05-02]*. c2023. Dostupné také z: https://nextjs.org/.
33. *Deploy on web — onnxruntime. [online]. [cit. 2023-05-04]*. c2023. Dostupné také z: https://onnxruntime.ai/docs/tutorials/web/.
34. VERNEROVÁ, Anna. *Strojové učení: klasifikace (6. přednáška). [online]. [cit. 2023-05-11]*. c2016. Dostupné také z: https://ufal.mff.cuni.cz/~vernerova/2016/docs/prednaska_06_klasifikace.pdf.
35. TERVEN, Juan; CORDOVA-ESPARZA, Diana. *A Comprehensive Review of YOLO: From YOLOv1 and Beyond*. 2023. Dostupné z arXiv: 2304.00501 [cs.CV].
36. *Mean average - Český překlad. [online]. [cit. 2023-05-11]*. c2016. Dostupné také z: https://www.linguee.cz/%C4%8De%C5%A1tina-angli%C4%8Dtina/search?source=auto&query=mean+average.
37. YANDOUZI, Mimoun; GRARI, Mounir; BERRAHAL, Mohammed; IDRISSI, Idriss; MOUSSAOUI, Omar; AZIZI, Mostafa; GHOUMID, Kamal; ELMIAD, Aissa KERKOUR. Investigation of Combining Deep Learning Object Recognition with Drones for Forest Fire Detection and Monitoring. *International Journal of Advanced Computer Science and Applications*. 2023, roč. 14, č. 3.
38. REIS, Dillon; KUPEC, Jordan; HONG, Jacqueline; DAOUDI, Ahmad. Real-Time Flying Object Detection with YOLOv8. *arXiv preprint arXiv:2305.09972*. 2023.
39. *YOLOv8 - Ultralytics YOLOv8 Docs. [online]. [cit. 2023-05-16]*. c2023. Dostupné také z: https://docs.ultralytics.com/models/yolov8/#overview.
40. *Machines — Paperspace. [online]. [cit. 2023-07-16]*. c2023. Dostupné také z: https://docs.paperspace.com/gradient/machines/#gpu-machines.

Obsah přiloženého média

- `readme.md` ..... stručný popis obsahu média
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `models` ..... natrénované modely
  - `datasets` ..... datasety
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF