

## Hvordan bruke en to-trinns objekt deteksjons-pipeline med YOLOWorld og YOLOv8

Denne to-trinns objekt deteksjons-pipelinen er en metode som kombinerer zeroshot deteksjonsmodell (YOLOWorld) med en valideringsmodell (YOLOv8) for å filtrere ut falske positiver. Dette gjøres ved at YOLOv8 bekrefter eller avkrefter resultatene fra YOLOWorld. I dette prosjektet har pipelinen blitt testet på flyfoto for å detektere svømmebasseng, som et eksempel på hvordan teknikken kan brukes til spesifikke oppgaver innen bildedeteksjon.

Dette dokumentet inkluderer:

- Hvordan å sette opp pipelinen
- Resultater og benchmark-tester
- Forslag til fremtidig arbeid
- Fremgangsmåte på diverse verktøy (GQIS, labellmg)
- Prompting

### **Steg:**

#### **Steg 1: Forberedelse av Miljø og Avhengigheter**

#### **Steg 2: Definer Datasettene og Konfigurasjonsfiler**

- **Datasett**
- **Prompts Fil:** Last inn beskrivelser av objektene som skal finnes fra `prompts.yml`. Rediger `prompts.yml` slik at den inneholder ønskede beskrivelser for objektene som skal detekteres.

#### **Steg 3: Initialisering av Modellene**

- **YOLOWorld for Zeroshot Deteksjon**
- **Valideringsmodell med YOLOv8**

#### **Steg 4: Kjør Pipelinen - Kjøre Objekt Deteksjonsprosessen**

## Steg 5: Manuell Gjennomgang av Deteksjoner (hvis ønskelig)

- **Manuell Validert Gjennomgang:** Etter automatisk deteksjon kan en manuell gjennomgang gjøres:  
`user_input = input("Do you want to start the manual review of detections? (Yes/No):").strip().lower()`  
 Ved valg av "Yes" kan hvert detektert objekt vurderes og enten godkjennes eller avvises.
- **Lagre Resultatene:** metoden lagrer resultatene i de tilhørende mappene for "accepted" og "rejected" deteksjoner.

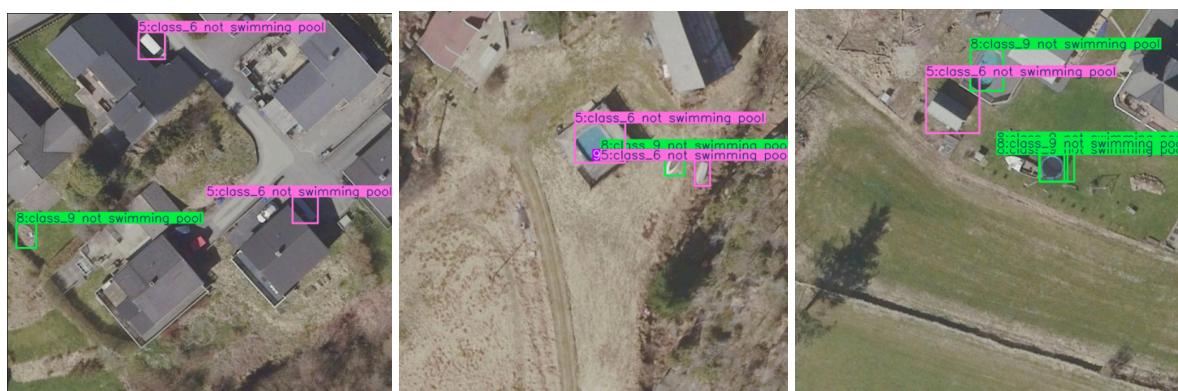
## Steg 6: Forberedelse av Datasett for Videre Trening

- **Organisering av Falske Positiver for Trening:** Etter manuell gjennomgang kan de avviste deteksjonene brukes til å forberede et datasett for videre trening:  
`user_input = input("Do you want to prepare the false positive dataset for training? (Yes/No)")`  
 Hvis svaret er "Yes", kopieres de avviste bildene til en egen mappe for videre modelltrenings sammen med tilhørende labels og `.yaml`-fil. De blir lagret i en mappe som heter `false_positives`, som opprettes automatisk første gang koden kjøres.
- 

## Resultater og Benchmark

Dette datasettet besto av 62 bilder, hvorav 11 inneholdt svømmebasseng. Totalt ble 33 av de 62 bildene (53%) identifisert med deteksjoner ved bruk av zero-shot YOLOWorld-modellen. Vedlagt er resultatene fra modellene. Bildene fra "Zero-shot resultater" viser flere eksempler på at YOLOWorld-modellen både klarer å identifisere svømmebasseng, men også feiler ved å klassifisere enkelte objekter feilaktig som svømmebasseng. Valideringsmodellen forbedrer resultatene ved å fjerne en del av de feilaktige deteksjonene, som fremgår av "Valideringsmodellen"-bildene. Til tross for forbedringen, er det fortsatt noen feilaktige deteksjoner som ikke filtreres bort.

### Zero-shot resultater:



### Validerings modellen:



**Benchmark resultater:** Tabellen under viser en oversikt over de samlede resultatene. Dette gir en indikasjon på modellens ytelse, og viser at det er rom for videre forbedring av både deteksjon og validering.

True Positives (TP)	7
False Positives (FP)	4
False Negatives (FN)	4
Precision	0.64
Recall	0.64
F1 Score	0.64



### Resultater og benchmark fra hele datasettet

Benchmark-resultater for hele datasettet er foreløpig ikke tilgjengelige.

## Fremtidig Arbeid

Fremtidig arbeid for å optimalisere modellen:

1. Forbedre prompts.
2. Bruke avviste deteksjoner fra den manuelle gjennomgangen iterativt for å trenere valideringsmodellen.
3. Utvide treningsdataene med et som inneholder flere svømmebasseng.
4. Utforske alternativer for å bytte ut zero-shot-modellen med en annen som kan gi bedre ytelse, for eksempel Grounding Dino.

## Oppsummering

Pipen har blitt testet på flyfoto for å finne svømmebasseng. Denne tilnærmingen med å kombinere zeroshot deteksjon med en valideringsmodell gir mer pålitelige resultater ved å fjerne falske positiver fra den første modellen. Pipelen gir også mulighet for manuell gjennomgang av resultatene, slik at avviste deteksjoner kan brukes til å forbedre modellen videre. Målet er å gjøre prosessen så automatisk og effektiv som mulig.

---

## Fremgangsmåte på diverse verktøy

Dette er en kort beskrivelse av programmene som ble brukt for å forstå dataen bedre: QGIS for geodata og LabelImg for bildeannotering. Roboflow ble ikke brukt ettersom den er open source.

### QGIS og Ortofoto (Flyfoto)

Dette er et program som brukes i geomatikk for geodata til å visualisere .tif filer. Ble brukt i sammenheng med FKB-data (Felles Kartbase) som inneholder geografiske data om ulike typer objekter i Norge, som f.eks. bygninger, veier, elver, linjer og annen infrastruktur. Disse dataene er ofte representert i forskjellige geometriyper: punkter, linjer, plan. I dette tilfellet er FKB-data begrenset til kun Farsund-området.

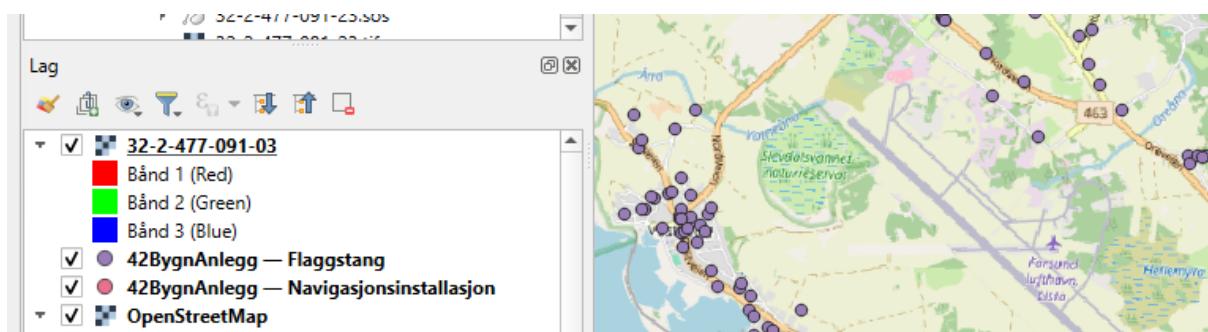
Ortofotoene er i .tif-format (GeoTIFF), som er et rasterbasert format for georefererte bilder. De tilhørende .tfw- og .aux.xml-filene inneholder georeferanseinformasjon (som posisjon, rotasjon og skalering) slik at bildet kan plasseres korrekt i geografiske koordinatsystemer.

- **.tif:** Dette er selve ortofotoet (bildet).
- **.tfw:** Denne filen inneholder "world file"-informasjon, som hjelper til med å plassere bildet geografisk.
- **.aux.xml:** En tilleggsfil for metadata som kan inneholde mer informasjon om bildet, som projeksjon.

Bruk av QGIS:

Når programmet er lastet ned og åpnet, legges både FKB-dataene og ortofotoene inn. For å vise dem over et ønsket bakgrunnskart i QGIS, kan man gå til 'Layer' -> 'Add Layer' -> 'Add

WMS...'. Her kan Kartverkets WMS legges inn (link på GeoNorge) for å få Norgeskart som bakgrunn.



Ved å legge til flere filer, vil flyfotoene dekke hele området. Når mange slike filer er tilgjengelige, kan merge-funksjonen brukes til å slå sammen alle de mindre bildene til ett stort. Velg alle bildefilene som skal slås sammen, høyreklikk, og gå deretter til Raster -> Miscellaneous -> Merge. Da vil det bli fremvist i satellittbildet.

## labelImg

Forberedelse for bruk av labelImg:

1. Last ned som zip-fil på: <https://github.com/HumanSignal/labelImg> og pakk ut filen.
2. Anaconda prompt, ny env
3. Installer nødvendige biblioteker: Conda install pyqt=5
4. Naviger til labelImg-mappen: cd C:\Users\berna\Downloads\labelImg-master\labelImg
5. Kjør kommandoen: pyrcc5 -o libs/resources.py resources.qrc
6. Hvis error:
  1. pip install sip
  2. pip install lxml
7. Start labelImg med kommandoen: python labelImg.py

## labelImg Workflow

Få opp bounding boxes av detekterte bilder på labelImg for å korrekt annotere dem/markere falske positiver:

**Lag .txt-filer** for bildene som er kjørt gjennom deteksjonsmodellen:

- .txt-filene skal inneholde koordinater og annotasjoner (klassens indeks skal være et heltall, ikke desimaltall), for eksempel: erstatt 8.0 med 8 og 5.0 med 5.

**Strukturering av filer:**

- Lag en mappe med det originale bildet og tilhørende .txt-fil med samme navn.
  - /train

- `image1.jpg` – Det originale bildet som ble kjørt gjennom deteksjonsmodellen.
- `image1.txt` – YOLO .txt-fil med koordinater for bboxes.
- `/val` – Kan settes opp tilsvarende for valideringsbilder.

### Definer klasser i `classes.txt`:

For å annotere falske positiver som ble detektert av modellen, må datastrukturen settes opp på riktig måte. Resultatene organiseres i mapper med en klar struktur: Under `/results` opprettes undermappene `/train` og `/val`. Hver av disse mappene inneholder to undermapper, `images` for de originale bildene og `labels` for YOLO .txt-filer som beskriver bokskoordinater og klassenumre. I tillegg må det ligge en `classes.txt`-fil i `labels`-mappen som definerer alle klassene i prosjektet.

I `classes.txt` spesifiseres alle klassene som ble brukt, inkludert feil deteksjoner. Hvis modellen for eksempel har flere klasser for samme objekt (som ulike klasser for svømmebassenger), må alle disse klassene defineres i `classes.txt`. I dette tilfellet så det se slik ut:

```
0 not swimming pool
1 not swimming pool
.
.
33 not swimming pool
```

For å åpne annoteringene i LabelImg, naviger til `/results/train/`. Når du jobber i LabelImg, behold kun de deteksjonene som er feil (siden det var falske positiver val modellen skal trenes på). Korrekte bokser, som for eksempel riktig identifiserte svømmebassenger, fjernes. På feil deteksjonene, endres klassen til en ny klasse, som for eksempel `not swimming pool`. Det er viktig at posisjonene i `classes.txt` stemmer overens med de originale klassene i YOLO .txt-filene. For eksempel, hvis en feil deteksjon hadde klassenummer 8, må `not swimming pool` stå på posisjon 8 i `classes.txt`.

### Oppdater `predifined_classes.txt`:

- Innholdet i `predifined_classes.txt` fra LabelImg-master-mappen må samsvare med `classes.txt` i prosjektmappen.

### Kjør valideringsmodellen:

- Opprett en YAML-fil for trenings- og valideringsoppsettet og kjør gjennom valideringsmodellen.

Når dette er gjort, må treningsmodellen konfigureres med en oppdatert `data.yaml`-fil som reflekterer de nye klassene. En typisk `data.yaml` vil inneholde antall klasser (`nc`) og en liste over klassene:

```
nc: 33
0: '0'
1: '1'
.
.
33: '33'
```

## Prompt engineering

**Prompt-optimalisering** handler om å formulere spørsmålet på en måte som gjør at AI-en best mulig kan forstå det du ber om, slik at den gir et resultat som samsvarer med forespørselen din [1](#).

Denne prosessen innebærer [2](#):

1. Å velge ord nøyne.
2. Strukturere forespørslene dine.
3. Gi kontekst som veileder AI-ens forståelse.

### Scenario 1: Utydelige Instruksjoner

Du: "Kim, kan du skaffe meg noe å drikke?"

Kim kan komme tilbake med alt fra et glass vann til en flaske hot sauce, siden "noe å drikke" kan tolkes på mange måter.

### Scenario 2: Optimaliserte Instruksjoner

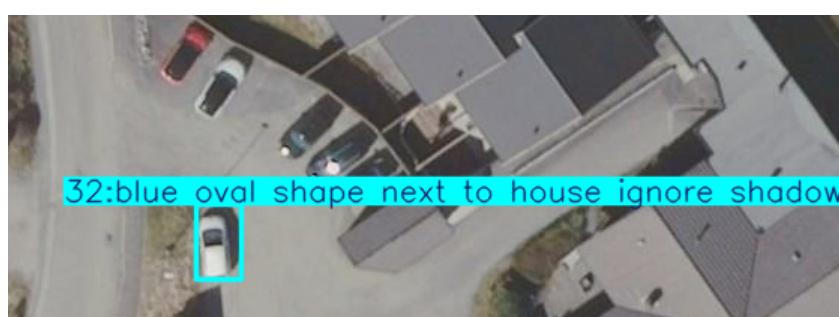
Du: "Kim, kan du gå til kjøkkenet, åpne kjøleskapet og ta med en kald boks cola? Hvis det ikke er cola, er et glass kaldt vann fint."

I dette tilfellet er det mye mer sannsynlig at Kim gir deg akkurat det du ønsker.

### Zero-Shot Prompting

Dette er en teknikk som utnytter en stor språkmodells generaliseringsevner til å utføre nye oppgaver uten spesifikk opplæring eller eksempler [3](#).

Teknikken bruker modellens forhåndstrening på forskjellige datasett på nye oppgaver basert på klare og konsise instruksjoner. Saksessen avhenger av oppgavens kompleksitet og promptens kvalitet.



Kommer objektet tydelig frem?



6:rectangular blue pool beside house in a yard